WHIZARD 1.0

A generic Monte-Carlo integration and event generation package for multi-particle processes

MANUAL

W. Kilian[1]

Institut für Theoretische Teilchenphysik, Universität Karlsruhe, D–76128 Karlsruhe, Germany

ABSTRACT

WHIZARD is a program system designed for the efficient calculation of multi-particle scattering cross sections and simulated event samples. The events can be written to file in STDHEP or ASCII format. Tree-level matrix elements are generated automatically for arbitrary partonic processes by calling external programs (O'Mega, MadGraph and CompHEP). Matrix elements obtained by alternative methods (e.g., including loop corrections) may be interfaced as well. The program is able to calculate numerically stable signal and background cross sections and generate unweighted event samples with reasonable efficiency for processes with up to six final-state particles. Polarization is treated exactly for both the initial and final states. Final-state quark or lepton flavors can be summed over automatically where needed. For Linear Collider physics, beamstrahlung (CIRCE) and ISR spectra are included for electrons and photons. Currently, WHIZARD supports the Standard Model, optionally with anomalous couplings, but model extensions or completely different models can be added.

[1] kilian@particle.uni-karlsruhe.de

Contents

1 General remarks **2**

2 Installation **3**
2.1 Sources . . . 3
2.2 Prerequisites . . . 4
2.2.1 Necessary . . . 4
2.2.2 Optional . . . 4
2.2.3 Included . . . 5
2.3 Configuration . . . 5
2.3.1 Configure options . . . 6

3 Running `WHIZARD` **7**
3.1 Process selection . . . 7
3.1.1 Model selection . . . 7
3.1.2 Process list . . . 9
3.2 Compilation and installation . . . 11
3.3 Input data . . . 12
3.3.1 The `process_input` block . . . 12
3.3.2 The `integration_input` block . . . 14
3.3.3 The `simulation_input` block . . . 16
3.3.4 The `diagnostics_input` block . . . 17
3.3.5 The `parameter_input` block . . . 18
3.3.6 The `beam_input` blocks . . . 19
3.4 Cuts . . . 21
3.5 Integration . . . 23
3.5.1 Running `WHIZARD` . . . 23
3.5.2 The phase space configuration . . . 25
3.5.3 What if it does not converge? . . . 26
3.5.4 Output files . . . 27
3.6 Event generation . . . 28
3.6.1 Built-in analysis . . . 29

4 Examples **33**
4.1 Higgs production at LEP . . . 33
4.1.1 The on-shell process . . . 34
4.1.2 The missing-energy channel . . . 37
4.1.3 The four-jet channel . . . 41
4.1.4 The lepton channels . . . 43
4.1.5 QCD contribution . . . 45
4.2 6-fermion production: Higgs pairs . . . 46
4.3 Vector boson scattering: polarization and beamstrahlung . . . 51

1 General remarks

The increase of complexity in the scattering processes that are of interest at the next generation of colliders calls for new tools for automatic computation and event generation. Many different problems have to be tackled simultaneously: The program has to be versatile and necessarily involves a large degree of automatization since the list of multi-particle processes to be considered is much longer than can be included in a hard-coded process library like `PYTHIA` [1]. The precision required for the predictions makes the traditional distinction between signal and background processes obsolete, since interference effects often cannot be neglected. Thus, massive multi-particle phase space has to be handled in the presence of many resonances and nearby singularities. Since detector effects need to be studied, the program has to have a convenient user interface, and it must be able to generate unweighted events with reasonable efficiency.

During the workshops of the ECFA/DESY study for a future Linear Collider it became obvious that no single existing package was able to meet all these needs. Therefore, the initial idea of `WHIZARD` was to combine known packages for generating matrix elements with a program which is able to treat generic phase space, integrate and generate events. The matrix element packages included are `CompHEP` [2], `MadGraph` [3], and `O'Mega` [4], which together cover the whole set of processes that currently can be handled automatically (at tree level). The latter problem could be solved with the help of the new `VAMP` [5] integration program, which extends the `VEGAS` algorithm to multi-channel parameterizations and thus makes it possible to handle the complex singularity patterns of multi-particle phase space in a uniform way.

The problem for `WHIZARD` was to provide the actual phase space parameterizations, Jacobians and transformations, provide a consistent environment and to make the programs communicate with each other by common interfaces. The user had to be given a simple setup with common configuration and parameter definition files, commands to run all programs consistently without the need for manual intervention (a simple `make` should suffice), and an analysis system which allows for rapid inspection of the results as well as for interfacing hadronization and detector simulation programs. The program had to keep full track of beam polarization and include beamstrahlung and initial-state radiation. Finally, it should allow for flavor summation in the final state: usually, many different processes contribute to a single final state that cannot be distinguished experimentally, and thus should be covered in a single run.

These goals have to a large extent been achieved by the current release. However, there is still room for improvement. Most prominent, quark structure functions are still missing and the treatment of QCD effects is incomplete. Once this has been added, `WHIZARD` will be usable for processes at hadron colliders as well. Furthermore, the program is best suited for genuine electroweak processes, which typically have resonant signals (cascade decays) and a non-resonant background which is of the same order of magnitude. It is not tested as well for processes which involve the emission of many soft particles without a dominating resonant part, which may require improvements in the phase space parameterization. Another restriction is due to the matrix element programs involved and their interfaces, which currently limit `WHIZARD` to the Standard Model. Future versions will likely permit the user to define arbitrary models, and the model library will include SM extensions such as the minimal supersymmetric Standard Model (MSSM).

2 Installation

2.1 Sources

WHIZARD is Free Software and the sources can be obtained from

http://www-ttp.physik.uni-karlsruhe.de/Progdata/whizard/

The command

```
gzip -cd whizard.tgz | tar xf -
```

will unpack the sources in the current working directory. After unpacking, the following subdirectories will exist:

doc contains the manual

conf contains the configuration file and default input files

kinds-src contains the definition of the Fortran REAL kinds

chep-src contains the CompHEP sources and, in the subdirectory modellib, the model definitions

mad-src contains the MadGraph sources

helas-src contains the HELAS library which is used by MadGraph

omega-src contains files needed for running O'Mega together with WHIZARD; the actual O'Mega distribution is kept separate

circe-src contains the CIRCE beamstrahlung library

vamp-src contains the VAMP integration library

gml-src contains the gamelan sources which allow a graphical representation of WHIZARD results

whizard-src contains the actual WHIZARD sources

bin contains the scripts needed for running WHIZARD and, after compilation, the WHIZARD executable

processes is created during compilation and holds the source code for the generated matrix elements and corresponding Feynman graph pictures

results contains the WHIZARD executable and input files after installation

2.2 Prerequisites

2.2.1 Necessary

The following programs must be available on the host system in order to successfully compile and run `WHIZARD`:

- GNU Make
- Perl 5
- Fortran 90/95
- C: The C compiler is needed by `CompHEP`.[1]
- `O'Mega`: This program for automatic matrix element generation is available from

  `http://heplix.ikp.physik.tu-darmstadt.de/~ohl/omega/`

  or from

  `ftp://heplix.ikp.physik.tu-darmstadt.de/pub/ohl/omega`

  For the compilation of `O'Mega` you need Objective Caml (a.k.a. `O'Caml`), version 3.00 or higher. You can get it from

  `http://pauillac.inria.fr/ocaml/`

  When installing `O'Mega`, you have to make either one (or both) of

  `make bin` or `make opt`

  for the executables, and

  `make f95`

  for the run-time library.

2.2.2 Optional

The following programs are not needed for compiling and running `WHIZARD`, but only for some additional features:

- The STDHEP library for writing events in STDHEP format. This library is contained in the CERN library in pre-compiled version only. Using it may require the corresponding Fortran77 runtime library, which is determined by `configure` by envoking the Fortran 77 compiler.

[1] The compiler and compiler flags are set in `chep-src/unix_com/CC`.

- LaTeX for typesetting the documentation (including this manual)
- HeVeA for making the HTML version of the manual
- MetaPost for on-line generation of histogram and data plots
- noweb for weaving the program sources. This is also needed for generating the `MetaPost` macro set for plotting.
- autoconf for creating a configure script in the restricted bundle.

2.2.3 Included

The following programs are included in the `WHIZARD` distribution and need not be obtained separately. The URLs below should be inspected for the documentation, references, and license conditions:

- `CompHEP`:

  `http://theory.npi.msu.su/~pukhov/comphep_html/comphep.html`

  which is mirrored at

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

  The included version is somewhat out-of-date (version 3.2.18), but has an additional Fortran 90 interface.

- `MadGraph`:

  `http://www-pheno.physics.wisc.edu/Software/MadGraph`

  with a modified Fortran 90 interface.

- `CIRCE`:

  `http://heplix.ikp.physik.tu-darmstadt.de/lc/beam.html`

  with an additional `WHIZARD` interface.

- `VAMP` by Thorsten Ohl

2.3 Configuration

Running the configure script

```
./configure
```

will check the availability of programs and utilities and generate the Makefiles appropriate for the host system.

2.3.1 Configure options

The following environment variables are recognized by `configure`. They need to be set only if `configure` cannot find a program or library, or if a choice different from the default one is desired. Path and program names should be specified absolute, not relative to the working directory.

GMAKE: GNU Make.

F90: The Fortran 90 compiler.

F90FLAGS: Flags to be passed to the Fortran 90 compiler.

PERL: The PERL executable.

OMEGADIR: The `O'Mega` directory.

TMPDIR: Directory where temporary files are stored. By default, `/var/tmp` is used.

LD_LIBRARY_PATH: Default path to search for libraries.

STDHEPDIR: The directory where the precompiled STDHEP library resides (optional).

NOWEBDIR: Directory where the `notangle`, `noweave`, `cpif` executables reside (optional).

GTAR: GNU tar (optional).

MPOST: Metapost (optional).

LATEX: LaTeX (optional).

DVIPS: `dvips` (optional).

PRECISION: (*temporarily not supported, will be re-enabled in a future release:*) Set this to `quadruple` if quadruple precision code is desired. This will be applied to all parts of the program and may result in a considerable increase in execution time. If no quadruple complex type is allowed by the processor, it will be emulated

F90_MAXDIAG: Maximum number of diagrams contained in one `CompHEP`-generated file. If there are more diagrams, files will be split. A value too large may lead to memory overflow for particular compilers. Default value: 100.

3 Running WHIZARD

3.1 Process selection

Before compilation, a file `whizard.prc` has to be put in the subdirectory `conf` which contains the list of processes to be considered. In principle, any process allowed by the physics model can be included there. If no file is provided by the user, the file `whizard.prc.default` (Fig. 1) will be used. The user may take this file as a template, insert or delete processes as desired, and save the new version as `whizard.prc`.

3.1.1 Model selection

WHIZARD is not restricted to the Standard Model. The physics models for CompHEP and O'Mega are selected in two lines in the header of `whizard.prc`, respectively.

However, a generic interface for defining models is still under development. For this reason, only certain combinations of models can presently be used without modifications of the program source code.

The following combinations should work as expected:

- QED with three lepton generations (doesn't work with MadGraph):

```
model sm 1
model QED
```

- The Standard Model in unitarity gauge, with e, $\sin\theta_W$ and M_Z as independent parameters in the electroweak sector:

```
model sm 3
model SM
```

- The Standard Model as before, CompHEP using Feynman gauge for evaluation:

```
model sm 4
model SM
```

- The Standard Model in unitarity gauge, with G_F, M_W and M_Z as independent parameters in the electroweak sector:

```
model sm-GF 3
model SM
```

- **Recommended:** The Standard Model as before, CompHEP using Feynman gauge for evaluation:

```
# WHIZARD configuration file

# The selected model (CompHEP)
model   sm-GF   4

# The selected model (O'Mega)
model   SM

# Processes
# (Methods: chep=CompHEP, mad=MadGraph, omega=O'Mega)
# (Options: s=selected diagrams, number=QCD order [Madgraph])
#           f=fudged width [O'Mega]
#
# Tag           In      Out             Method  Option
#=====================================================
ee              e1,E1   e1,E1           chep
ee_m            e1,E1   e1,E1           mad
ee_o            e1,E1   e1,E1           omega
ww              e1,E1   W+,W-           chep
ww_m            e1,E1   W+,W-           mad
ww_o            e1,E1   W+,W-           omega
zh              e1,E1   Z,H             chep
zh_m            e1,E1   Z,H             mad
zh_o            e1,E1   Z,H             omega
nnh             e1,E1   n1,N1,H         chep
nnh_m           e1,E1   n1,N1,H         mad
nnh_o           e1,E1   n1,N1,H         omega
nnbb            e1,E1   n1,N1,b,B       mad
nnbb_o          e1,E1   n1,N1,b,B       omega
```

Figure 1: *Default process configuration file*

```
model sm-GF 4
model SM
```

- The Standard Model with anomalous couplings. In this combination CompHEP defines the Standard Model in Feynman gauge in the limit of infinite Higgs mass and includes the quartic anomalous couplings $\alpha_{4,5,6,7,10}$, while O'Mega defines the usual Standard Model (so M_H must be set to a large value if desired) and includes the anomalous triple gauge couplings $g_{1\gamma,Z}$, $\kappa_{\gamma,Z}$, $\lambda_{\gamma,Z}$ as well as the anomalous quartic couplings $\alpha_{4,5,6,7,10}$.

```
model sm-GF 5
model SM_ac
```

In future versions of WHIZARD, this will be made more consistent, more models will be added (in particular, the MSSM), and the possibility for the user to define arbitrary models will be supported.

3.1.2 Process list

Each process is defined by a single line in whizard.prc (see Fig. 1). The meaning of the entries within a line is as follows:

The first entry is a unique alphanumeric tag which will be used to identify the process. It can be arbitrarily chosen, but should be a valid Fortran 90 token, lower case only.

The second entry defines the initial state. It consists of two particle names, separated by a comma, with no space in between. The particle names are those used by CompHEP (by convention) and are found in the appropriate model definition file, e.g.

```
chep-src/modellib/sm-GF/prtcls4.mdl
```

(see Fig. 2).

The third entry defines the final state. It consists of from two to six particle names[2], separated by a comma, with no space in between. Flavor sums are defined by particle names separated by colons. For instance, the line

```
eeqq  e1,E1   u:d:s,U:D:S     omega
```

will create a sum of all processes $e^-e^+ \to q\bar{q}$, with q being any of the light quarks. (Flavor sums are available only for O'Mega matrix elements.)

The particles to be summed over must have identical spin and color representation; otherwise the summation will not work.

The fourth entry selects one of the three programs CompHEP, MadGraph, or O'Mega for matrix element generation. Many processes can be treated by more than one package, allowing for cross-checks, but there are limitations:

[2]More are possible, but WHIZARD has not been well tested for more than six final-state particles.

```
St.Model(Feyn.gauge)
 Particles
 Full  name   | P | aP|2*spin| mass |width |color|aux|
photon        |A  |A  |2     |0     |0     |1    |G
gluon         |G  |G  |2     |0     |0     |8    |G
electron      |e1 |E1 |1     |0     |0     |1    |
e-neutrino    |n1 |N1 |1     |0     |0     |1    |L
muon          |e2 |E2 |1     |Mm    |0     |1    |
m-neutrino    |n2 |N2 |1     |0     |0     |1    |L
tau-lepton    |e3 |E3 |1     |Mt    |0     |1    |
t-neutrino    |n3 |N3 |1     |0     |0     |1    |L
u-quark       |u  |U  |1     |0     |0     |3    |
d-quark       |d  |D  |1     |0     |0     |3    |
c-quark       |c  |C  |1     |Mc    |0     |3    |
s-quark       |s  |S  |1     |Ms    |0     |3    |
t-quark       |t  |T  |1     |Mtop  |wtop  |3    |
b-quark       |b  |B  |1     |Mb    |0     |3    |
Higgs         |H  |H  |0     |MH    |wH    |1    |
W-boson       |W+ |W- |2     |MW    |wW    |1    |G
Z-boson       |Z  |Z  |2     |MZ    |wZ    |1    |G
```

Figure 2: *Particle names and parameters as defined by* `CompHEP`.

- CompHEP can only generate matrix elements for up to four particles in the final state. No polarization or flavor summation is possible[3].

- MadGraph has the Standard Model implemented with the exception of the trilinear or quartic Higgs couplings. Color is treated exactly. The CKM matrix is set to unity. The implementation included in WHIZARD can generate matrix elements with up to six final-state particles. No flavor summation is possible.

- O'Mega has the Standard Model implemented, optionally with anomalous couplings. The color treatment is still incomplete, such that for a given process with colored particles only the leading electroweak contribution is taken into account. The CKM matrix is set to unity. It can generate matrix elements with an arbitrary number of final-state particles, limited only by the host resources. (WHIZARD has extensively been tested only for processes up to $2 \to 6$.)

At least one process must be calculated by CompHEP; otherwise the parameter definition code will not be written[4]. One may use a simple dummy process like $e^+e^- \to e^+e^-$ for this purpose.

3.2 Compilation and installation

After the processes have been selected, the matrix elements and executable code are compiled and linked by the simple command

```
make prg
```

This will generate an executable named `whizard` in the `bin` subdirectory.

An alternative is

```
make bundle
```

which will create a *restricted bundle* containing the f90 source code of the selected matrix elements and all necessary libraries. This bundle will be put as a gzipped tar file in the main directory under the name `whizard-bundle-XXX.tgz`, where `XXX` specifies the current date and time. The bundle can be unpacked on a different platform. There, it can be configured, compiled and run without the need for a working O'Mega installation. The restricted bundle has all capabilities of the WHIZARD system except for the possibility to re-generate the matrix elements.

A run of the Monte Carlo generator requires, apart from the executable, the following input and configuration files. In their initial form they are found in the `conf` subdirectory:

- `whizard.in` containing all input data and parameters.

[3] The current official CompHEP version allows for polarization and has an interface for flavor sums. These features are not yet supported by the WHIZARD–CompHEP interface.

[4] This restriction will be removed in a future version.

- whizard.cut1 containing a-priori cut definitions. This file may be empty.
- whizard.cut5 containing cut and histogram definitions for the data analysis step (the 5th pass of the program). This file may be empty.

When the command

```
make install
```

is issued, all necessary configuration files, together with the whizard executable, are copied into the results subdirectory. If no user-defined files are found, default ones will be used. The contents of the results directory may be copied anywhere without affecting the functionality of the whizard executable within, leaving room for the simultaneous use of multiple WHIZARD copies with different process content.

3.3 Input data

When the whizard executable is started, it will use the parameters from the copy of the file whizard.in in the working directory. In this file, the physical input parameters and runtime switches are set. There are very many parameters; however, the only mandatory entry is the process tag which determines the process to be considered. For all the rest, WHIZARD will try to insert sensible default values.

The file whizard.in is written using the Fortran 90 NAMELIST conventions. It consists of several blocks marked by a keyword beginning with the & character and closed by a slash /. Each block contains a list of assignments of the form *variable* = *value*. Variables left out are assigned their default value. The assignments are separated by commas or newlines. Whitespace can be inserted anywhere. Comments beginning with ! are allowed if the compiler supports the Fortran 95 standard.

An example for the input file is shown in Fig. 3. In many cases, it suffices to fill the process_input block and leave all others empty. If polarization, beamstrahlung etc. are intended, the beam_input blocks must also be considered.

3.3.1 The process_input block

These parameters should always be inspected.

Parameter	Value	Default	Description
process_id	*string*		Process tag as defined in whizard.prc. Mandatory.
sqrts	*number*	1000	C.M. energy of the initial state in GeV.
luminosity	*number*	0	Integrated luminosity in fb^{-1}. A nonzero value will activate event generation.

```
&process_input
process_id = "nnh"
sqrts = 500
luminosity = 100
/

&integration_input
calls =
  1  10000
  5  10000
  2  20000
/

&simulation_input
write_events = T
/

&diagnostics_input /

&parameter_input
Me = 0
Ms = 0
Mc = 0
MH = 115
wH = 0.3228E-02
/

&beam_input
/

&beam_input
/
```

Figure 3: *Sample input file.*

Parameter	Value	Default	Description
`polarized_beams`	T/F	F	If true, the helicity content of the beams must be defined below in the blocks `beam_input`.

3.3.2 The `integration_input` block

These parameters will not affect the cross section value (up to statistical fluctuations, and except for the default cuts) but can improve or disprove the running time, the stability and accuracy of the result, and the efficiency of event generation.

Parameter	Value	Default	Description
`calls`	*6 numbers*	*(yes)*	Array describing the number of iterations and number of calls per integration pass. Default values depend on the selected process. See below in subsection 3.5 for details.
`seed`	*integer*	*undefined*	Random number generator seed (integer). When omitted, the time counter will be used, resulting in a different value each run.
`stratified`	T/F	T	Use stratified (T) / importance (F) sampling.
`use_efficiency`	T/F	F	Use efficiency (T) / accuracy (F) as the criterion for adapting the channel weights.
`weights_power`	*number*	0.25	Power used for adapting the channel weights. Lower value means slower adaptation (to suppress fluctuations).
`write_grids`	T/F	T	Write grids to files `whizard.grb` (best grid) and `whizard.grc` (current grid), to be reused later.
`write_all_grids`	T/F	F	Write all grids to files `whizard.grXXX`, where `XXX` is the iteration number.
`read_grids`	T/F	F	Read existing grids `whizard.grb` and `whizard.grc` if they have been written by a previous run. This avoids the time-consuming adaptation step. Makes sense only if no physical parameters have been changed.

Parameter	Value	Default	Description
`read_grids_force`	T/F	F	Set this to T if you want to read the grids from file even if some parameters have changed. Use with care! This may result in a program crash if the grid structures are incompatible.
`initial_weights`	*numbers*	*equal*	Initial weights of the integration channels.
`generate_phase_space`	T/F	T	Generate a phase space configuration appropriate for the current process and write it to `whizard.phs`.
`read_phase_space`	T/F	T	Read phase space configuration from `whizard.phs` if possible.
`overwrite_phase_space`	T/F	T	Overwrite an existing file `whizard.phs`.
`phase_space_only`	T/F	F	Stop the program after phase space generation.
`off_shell_lines`	*integer*	2	Maximum number of off-shell-lines in Feynman graphs relevant for the phase space configuration.
`exchange_lines`	*integer*	1	This number n_e specifies additional Feynman graphs as relevant for the phase space configuration. They may have one off-shell line in addition, and at most n_e less exchange lines than the maximum.
`extra_off_shell_lines`	*integer*	0	Write extra configurations to file but don't use them.
`s_channel_resonance`	T/F	F	If the total energy is variable (structure functions) and an s-channel resonance is within the kinematical range, setting this to true may improve the phase space configuration.
`single_off_shell_decays`	T/F	T	Whether single-off-shell decays are relevant for the phase space configuration.
`double_off_shell_decays`	T/F	F	Whether double-off-shell decays are relevant for the phase space configuration.

Parameter	Value	Default	Description
`single_off_shell_branchings`	T/F	T	Whether single-off-shell branchings are relevant for the phase space configuration.
`double_off_shell_branchings`	T/F	T	Whether double-off-shell branchings are relevant for the phase space configuration.
`massive_fsr`	T/F	T	Whether the radiation of a massive particle in the final state is relevant for the phase space configuration.
`default_jet_cut`	*number*	10	The default invariant mass cut in GeV applied to pairs of massless colored particles.
`default_mass_cut`	*number*	10	The default invariant mass cut in GeV applied to pair production of massless colorless charged particles and to photon emission.
`default_energy_cut`	*number*	10	The default energy cut in GeV applied to photon and gluon emission.
`default_q_cut`	*number*	10	The default Q cut in GeV applied to photon and gluon exchange.

3.3.3 The `simulation_input` block

These parameters control event generation and output. If the luminosity (above) is left undefined or zero, one may still generate a fixed number of events by setting `n_events` nonzero. By default, the events are written to file in raw format and can be reused in that way. Writing events in another format (which uses up more disk space) must be requested explicitly. However, if this has not been done, one may use `WHIZARD` afterwards as a translator to any file format by reading pre-generated events and immediately writing them in another format:

```
&integration_input  read_grids = T /
&simulation_input   read_events_raw = T  write_events = T /
```

Parameter	Value	Default	Description
`n_events`	*integer*	0	Number of events to generate at least, independent of the luminosity.
`n_events_warmup`	*integer*	0	Number of extra warmup events (see below, Sec. 3.6).
`unweighted`	T/F	T	Reweight events to generate an unweighted event sample.
`write_events`	T/F	F	Write generated events to file `whizard.evt` to be used by an external analysis package.

Parameter	Value	Default	Description
`write_events_format`	*integer*	1	The format to be used for writing `whizard.evt` (see Sec. 3.6).
`write_events_raw`	T/F	T	Write generated events to file `whizard.evx` (raw format) to be reused in a later run.
`read_events_raw`	T/F	F	Read events from file `whizard.evx` (raw format) instead of generating them.
`read_events_force`	T/F	F	Read events from file even if some parameters have changed. Use with care!

3.3.4 The `diagnostics_input` block

These parameters do not affect the result, but the information displayed on screen and stored in files.

Parameter	Value	Default	Description
`screen_results`	T/F	T	Whether to show results online on screen.
`screen_histograms`	T/F	F	Whether to show histograms on screen.
`screen_diagnostics`	T/F	F	Whether to repeat the input parameters on screen.
`write_logfile`	T/F	T	Whether to write the output file `whizard.out`.
`show_input`	T/F	T	Whether to repeat the input parameters in the logfile.
`show_phase_space`	T/F	F	Whether to show the phase space configuration in the logfile.
`show_cuts`	T/F	T	Whether to show the cut configuration in the logfile.
`show_histories`	T/F	T	Whether to show the individual `VAMP` channel histories in the logfile.
`show_history`	T/F	T	Whether to show the overall `VAMP` history in the logfile.
`show_weights`	T/F	T	Whether to show the weight adaptation in the logfile.
`show_event`	T/F	F	Whether to show the last event in the logfile.
`show_histograms`	T/F	F	Whether to show histograms in the logfile.

Parameter	Value	Default	Description
`show_overflow`	T/F	F	Whether to show events beyond the first or last bin in histogram listings.
`show_excess`	T/F	T	Whether to show a summary of events with weight exceeding one.
`plot_width`	*number*	130	The width in mm of the plots if online analysis is enabled.
`plot_height`	*number*	90	The height in mm of the plots if online analysis is enabled.

3.3.5 The `parameter_input` block

These are the physical constants used in evaluating the matrix elements. If this block is left empty, default values (see below) will be inserted.

Which constants are actually present, and which ones are dependent or derived, depends on the physical model (see the files in the corresponding subdirectory of `chep-src/modellib`). The constants relevant for the Standard Model in the G_F parameterization are shown below:

Parameter	Value	Default	Description
`GF`	*number*	1.16639×10^{-5}	Fermi constant
`MZ`	*number*	91.187	Z-boson mass
`MW`	*number*	80.41	W-boson mass
`MH`	*number*	130	Higgs mass
`GG`	*number*	1.12	Strong coupling constant $\alpha_s(M_Z)$
`s12`	*number*	0.221	CKM matrix parameter (PDG)
`s23`	*number*	0.040	CKM matrix parameter (PDG)
`s13`	*number*	0.0035	CKM matrix parameter (PDG)
`Me`	*number*	0.000511	electron mass
`Mm`	*number*	0.1057	muon mass
`Mt`	*number*	1.777	τ-lepton mass
`Mc`	*number*	1.300	c-quark mass
`Ms`	*number*	0.200	s-quark mass
`Mb`	*number*	4.300	b-quark mass
`Mtop`	*number*	170	t-quark mass
`wtop`	*number*	1.442	t-quark width
`wZ`	*number*	2.502	Z-boson width
`wW`	*number*	2.094	W-boson width
`wH`	*number*	1.461	Higgs width

The dependent parameters (such as e, $\sin\theta_W$, V_{cb} etc.) will be shown in the output file `whizard.out`, but one should not try to reset them in the input file.

Setting fermion masses to zero will considerably speed up the matrix element evaluation since certain helicity combinations vanish identically (with `MadGraph` and `O'Mega`). In that case, cuts may be needed for a finite answer. However, if necessary, finite masses can be kept everywhere.

Concerning quark masses in particular, they depend in fact on the chosen scale. The default values for the parameters include the strong coupling constant defined at the scale $\mu = M_Z$, but the quark masses evaluated at the low scale $\mu = 2$ GeV. These default values are taken from the PDG 2000 compilation. For a consistent scheme, running quark masses should be inserted instead wherever this is of importance. For instance, the b mass is only about 2.9 GeV at $\mu = M_Z$, thus affecting the Higgs coupling to b quarks.

The default values for the particle widths are given by the sum of the tree-level $1 \to 2$ decay channels, using the default masses and couplings as input. This choice is questionable as well. Since `O'Mega`, `MadGraph` and `CompHEP` can only calculate cross sections at tree level, this is at least consistent; however, for each particular problem one should re-investigate the underlying assumption that 2-particle decays are dominant, and recalculate the widths accordingly. This cannot (yet) be done by `WHIZARD`.

Finally, even if a parameter is defined and appears in the output, this does not necessarily mean that its value is used by all three programs `CompHEP`, `MadGraph` and `O'Mega`. For instance, the `chpt-GF` class of models defines the Higgs mass in order to be compatible with `O'Mega`'s `SM_ac` model, but the matrix elements are calculated by `CompHEP` effectively using an infinite Higgs mass. However, if a matrix element does depend on a particular parameter, a value specified in the input file will be taken.

There are two switches within the `parameter_input` block:

Parameter	Value	Default	Description
`gwidth`	T/F	F	Use the gauge-invariant width prescription for unstable particles in `CompHEP` matrix elements.
`rwidth`	T/F	F	Use a running-width prescription for unstable particles in `CompHEP` matrix elements.

In `O'Mega` matrix elements, the gauge-invariant width prescription ("fudge-factor" or "overall" scheme) is implemented by setting the `f` option in the last column of the process definition (in `whizard.prc`).

3.3.6 The `beam_input` blocks

The input file is finished by two blocks which describe the properties of the first and second incoming particle beam, respectively. These blocks may be left empty, in which case unpolarized beams with fixed energy will be assumed.

Parameter	Value	Default	Description
`polarization`	*two (three) numbers*	0 0 0	Fraction of left/right polarization (fermions, photons, gluons), resp. left/longitudinal/right polarization (massive vector bosons).

Parameter	Value	Default	Description
fixed_energy	T/F	T	If false, look for a structure function to apply to this beam.

The polarization is applied only if polarized_beams is true in the process_input block above. The remainder is filled with unpolarized particles, so .5 0 and .75 .25 would both stand for 50 % left-handed polarized electrons.

If fixed_energy is false, WHIZARD will check if any of the structure functions defined below is applicable for this beam. In case of beamstrahlung and ISR, two structure functions may be applied consecutively.

For beamstrahlung, the accelerator type should always be defined, and it is not recommended to use an energy different from the values for which valid CIRCE [6] parameterizations exist.

Parameter	Value	Default	Description
CIRCE_on	T/F	F	Whether to apply beamstrahlung (electron, positron or photon beam).
CIRCE_map	T/F	T	Whether to apply a mapping to improve convergence.
CIRCE_sqrts	*number*	$\sqrt{s}$	The reference energy. Note that CIRCE spectra are defined only for discrete energy values.
CIRCE_ver	*integer*	0	The CIRCE version number required.
CIRCE_rev	*integer*	0	The CIRCE revision number required.
CIRCE_acc	*integer*	0	The accelerator type as needed by CIRCE.
CIRCE_chat	*integer*	0	The level of CIRCE diagnostics.

The ISR spectrum for electrons or positrons accounts for the leading-logarithmic effect of multiple photon emission. Since $\alpha_{\rm QED}$ is usually calculated from G_F as far as the hard scattering is concerned, it makes sense to reset it to $\frac{1}{137}$ here. The incoming mass needs only be defined here if m_e has been set to zero in the parameter input block (which speeds up the calculation).

Parameter	Value	Default	Description
ISR_on	T/F	F	Whether to apply ISR (electron or positron beam).
ISR_alpha	*number*	$e^2/4\pi$	The value of $\alpha_{\rm QED}$ to be used for the spectrum.
ISR_m_in	*number*	m	The mass of the incoming particle.
ISR_sqrts	*number*	$\sqrt{s}$	The hard scale which cuts off photon radiation.

The EPA spectrum is the Weizsäcker-Williams approximation for a real photon radiated off the incoming beam. Here WHIZARD does not know the nature of the incoming particle, so

```
! e-  e+  ->  e-  e+  gamma
! 16   8       1   2    4
process eeg
  cut Q of 10 within -99999 -1
  cut Q of 17 within -99999 -1
  cut M of  3 within 10 99999
  cut E of  4 within  5 99999
  cut PT of 4 within 19 99999
  cut THETA(DEG) of  4 1 within 5 180
  cut THETA(DEG) of  4 2 within 5 180
```

Figure 4: *Cut configuration file*

its PDG code must be specified here. In addition, $Q_{\max}$ and m_X must be supplied. (If the produced system is massive, m_X should be set equal or less to the mass sum of all produced particles.)

Parameter	Value	Default	Description
`EPA_on`	T/F	F	Whether to use the EPA spectrum (photon beam).
`EPA_map`	T/F	T	Whether to apply a mapping to improve convergence.
`EPA_in_prt`	*integer*	0	PDG code of the particle emitting the photon.
`EPA_alpha`	*number*	$e^2/4\pi$	The value of α_{QED} to be used for the spectrum.
`EPA_mX`	*number*	0	The lower cutoff for the produced invariant mass.
`EPA_Q_max`	*number*	0	The upper cutoff on the virtuality of the photon ($Q_{\max} > 0$).
`EPA_x0`	*number*	0	The lower cutoff on the energy fraction of the incoming photon
`EPA_x1`	*number*	1	The upper cutoff on the energy fraction of the incoming photon

3.4 Cuts

`WHIZARD` allows for cuts in some standard kinematical variables. Cuts may be necessary in case the matrix element is infinite without them (which happens if massless particles are either exchanged or produced). In this case, `WHIZARD` will apply default cuts on the invariant mass of colored or charged particle pairs, on the energy of emitted photons or gluons, and on the momentum transfer to exchanged photons or gluons. The values of those cuts are controlled by parameters in the input file (see above).

Code	Alternative code	Args	Description
`-`		$0-2$	No cut
`M`	`Q`	1	(Signed) invariant mass $M = \mathrm{sgn}(p^2)\sqrt{\lvert p^2 \rvert}$
`LM`	`LQ`	1	$\log_{10} \lvert M \rvert$
`MSQ`	`QSQ S T U`	1	Squared invariant mass $M^2 = p^2$
`E`		1	Energy in the lab frame
`LE`		1	$\log_{10} E$
`PT`		1	Transverse momentum $p_\perp$
`PL`		1	Longitudinal momentum p_L
`P`		1	Absolute value of momentum $\lvert\vec{p}\rvert$
`Y`	`RAP RAPIDITY`	1	Rapidity y
`ETA`		1	Pseudorapidity η
`DETA`	`DELTA-ETA`	2	Pseudorapidity distance $\Delta\eta$
`A`	`ANGLE TH THETA`	2	Polar angle θ (lab frame) in radians
`AD`	`ANGLE(DEG)`		
	`TH(DEG) THETA(DEG)`	2	Polar angle θ (lab frame) in degrees
`CT`	`COS(TH) COS(THETA)`	2	$\cos\theta$
`PH`	`PHI`	2	Azimuthal distance $\Delta\phi$ in radians
`PHD`	`PHID PHI(DEG)`	2	Azimuthal distance $\Delta\phi$ in degrees

Table 1: *Cut and histogram code letters. Masses, energies and momenta are measured in GeV.*

Alternatively, the user may specify his own set of cuts by supplying an entry in the file `whizard.cut1`. If such an entry exists, the program will ignore the default cuts, and the user is responsible for setting up his cuts in such a way that the cross section is still finite. The format of such an entry for the process $e^-e^+ \to e^-e^+\gamma$ is shown in Fig. 4. The contents of `whizard.cut0`, where the default cuts are logged by `WHIZARD` (if any), may be used as a template.

The entry is begun by the keyword `process` followed by the process tag (`eeg` in this case). If the same set of cuts should be applied to several processes, one may specify more than one tag per entry, e.g.

```
process uudd uuss uucc
    cut ...
```

The cut configuration file may contain any number of process entries. Comments are marked by the letter `!` or `#`.

For a given process, each cut is indicated by the keyword `cut`, followed by a code letter, the keyword `of`, one or two binary codes, and cut window specifications. The code letters are listed in Table 1. The binary codes indicate the combination of particle momenta to which the cut should be applied. As the comment lines in Fig. 4 suggest, the outgoing particles are given codes $1, 2, 4, \ldots$. Two additional codes are assigned to the incoming particles, where the first incoming particle has the highest number. Momentum sums (particle combinations) are defined by adding binary codes, such that the code 5, for instance, stands for the sum of momenta of particles 1 and 4. The initial momenta are always counted negative, such that (in Fig. 4) the number 17 stands for the *difference* of the momenta of particle 1 (outgoing) and 16 (incoming). The same applies to angles: The polar angle between 1 and 16 is not `A of 1 16` but `A of 1 8` since the initial particle 16 goes into the direction of $-p(8)$.

Cut window specifications consist of the keyword `within` and two real numbers, optionally followed by additional specifications like

```
cut PT of  4 within 0 150
cut M  of  3 within 80 100 or 180 200 or 500 99999
```

The absence of an upper bound should be marked by a number outside the kinematical range.

3.5 Integration

3.5.1 Running WHIZARD

After the input parameters and cuts have been specified, the integration can be started by

```
make run
```

or, if we are in the working directory (`results`),

```
./whizard
```

After each iteration, the result will be displayed on screen and logged in the file `whizard.out`. An example is shown in Fig. 5. Clearly, the integration converges towards a stable result, and at the same time the estimated reweighting efficiency increases (allowing for some fluctuations).

The number of iterations and the number of calls per iterations are as given in the input file `whizard.in` (see Fig. 3). Adaptation proceeds in three levels which are separated by a horizontal line in the output:

In the first integration level, the grids are adapted after each iteration, but the relative weight of the integration channels is kept fixed. In the example of Fig. 5, this level consists of two iterations with $10,000$ calls each. This level is useful in particular for many-particle processes, where it takes some time before VAMP has found the physical region in each channel.

In the second level, the grids within each phase space channel and the relative weights of the phase space channels are adapted after each iteration. The example of Fig. 5 shows how the integral converges, the error estimate improves, and the estimated reweighting efficiency increases, until some saturation is reached. Here, the sixth column is of main interest, which is given by the estimator for the relative error multiplied by the square root of the number of calls. One expects a number of the order of unity (or smaller) here for a well-adapted grid, independent of the number of calls. Each time this number improves, it is marked by an asterisk. The best grid will be used later for the final integration and event generation steps.

In the third level, the relative weights are fixed again, and the remaining iterations are averaged for the final cross-section result.

```
! WHIZARD 0.40 (Sep 11 2000)
! Reading process data from file whizard.in
! Process nnbb_m:
!   e a-e  -> nu_e a-nu_e   b a-b
!  32  16  ->    1      2   4   8
! Generating phase space channels for process nnbb_m ...
! 40 channels generated.
!
! WHIZARD run for process nnbb_m:            ( checksum =  131465255 )
!----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!----------------------------------------------------------------------
! Adapting (fixed weights):  Generating 2 samples of 10000 events ...
  2      20000  5.7019717E+01  1.58E+00    2.76    3.91*  2.31    0.31
!----------------------------------------------------------------------
! Adapting (var. weights):  Generating 8 samples of 10000 events ...
  3      10000  5.5642224E+01  1.23E+00    2.21    2.21*  7.58
  4      10000  5.9028368E+01  1.06E+00    1.80    1.80*  7.51
  5      10000  5.8586436E+01  8.34E-01    1.42    1.42*  9.82
  6      10000  5.8997829E+01  6.89E-01    1.17    1.17* 12.18
  7      10000  5.8626448E+01  1.04E+00    1.78    1.78  10.78
  8      10000  5.7737567E+01  5.12E-01    0.89    0.89* 17.50
  9      10000  5.7693393E+01  4.75E-01    0.82    0.82* 19.50
 10      10000  5.8216141E+01  5.42E-01    0.93    0.93  14.60
!----------------------------------------------------------------------
! Integrating (fixed w.):  Generating 2 samples of 10000 events ...
 12      20000  5.8910540E+01  4.25E-01    0.72    1.02  11.64    0.05
!----------------------------------------------------------------------
```

Figure 5: *Sample output of running **WHIZARD**.*

3.5.2 The phase space configuration

The critical point for the integration is the phase space configuration. This is set up automatically by `WHIZARD` before starting the integration. `WHIZARD` phase space consists of a set of *channels* which formally correspond to Feynman diagrams. It should be kept in mind that the choice of phase space channels does not influence the integration results (*all* Feynman graphs are included in the cross section calculation and event generation). However, the accuracy and the speed of convergence depend on this choice. In many cases it will not be necessary to modify anything, but if you are not satisfied with the results of the default configuration, the following remarks may help.

The default phase space configuration includes essentially all structures up to a certain number of internal off-shell lines (2 by default), together with additional t-channel topologies (t-channel photons/gluons are regarded as on-shell). This yields satisfactory results for many processes, in particular those with a clear resonance structure (cascade decays).

The number of channels in the default configuration tends to be large. This is reasonable since in principle all Feynman diagrams contribute, but it is a disadvantage for various reasons: First, the necessary CPU time increases with the number of channels since the fraction of time spent in calculating the mappings between different channels is not negligible. Second, fluctuations have a strong effect if there are too many degrees of freedom, possibly spoiling the convergence of the adaptive procedure. Third, in order to have a sufficient number of calls within each integration channel, the total number of calls per iteration should also increased with the number of channels, resulting again in more CPU time. As a rule, one should avoid to have more than about 100-200 phase space channels or much less than $O(1000)$ phase space points per channel.

On the other hand, it may happen that important channels are missed by the default configuration. In some cases the program is unable to generate any phase space configuration because no graphs pass all criteria. In this case the restrictions should be weakened.

Therefore, various switches are available to control the phase space configuration. First of all, to reduce the number of irrelevant channels, one may set `double_off_shell_branchings=F` or even `single_off_shell_branchings=F`. This tells `WHIZARD` not to consider cases where an *off-shell* particle decays into two particles, one or both of them being off-shell as well. The total number of off-shell lines allowed is controlled by the parameter `off_shell_lines`.

Considering on-shell particles, `WHIZARD` by default will take two- and three-body decays into account. The latter can be removed from the phase space setup by setting `single_off_shell_decays=F`. On the other hand, one may account for certain four-body decays by `double_off_shell_decays=T`.

Whether exchange (t-channel) graphs are important or not depends on the dynamics. For instance, Higgs production at low energies is due to e^+e^- annihilation at low energies, while vector boson fusion dominates at high energies. In the phase space configuration, this is controlled by the parameter `exchange_lines`, which is the number of massive exchange lines that should be counted as on-shell. The default (1) has proven to be adequate for most cases. Finally, to reduce the number of channels one can set `massive_FSR=F`. This will, e.g., remove graphs where a massive boson is radiated off the spectator fermion after boson exchange.

To summarize, while for simple processes ($2 \to 3$, $2 \to 4$) the default configuration can usually be kept if one is not too much concerned about execution speed, in more complicated cases a good starting point is a setup like

```
&integration_input
...
double_off_shell_branchings = F
single_off_shell_branchings = F
single_off_shell_decays = F
massive_FSR = F
/
```

which will result in a low number of channels. (Set `phase_space_only = T` and look at the generated configuration in `whizard.phs` if you want to inspect the configuration before starting the integration.) One should check that no important subprocesses are missed; ultimately, the test for that is the convergence and stability of the integration. If it fails, some parameters should be modified in order to increase the number of channels.

3.5.3 What if it does not converge?

Although `WHIZARD` is able to produce stable results with the default settings for a large variety of scattering processes, it is not possible to guarantee fast convergence in all circumstances. There is no simple rule for the achievable accuracy and event generation efficiency for a given process; the examples in Sec. 4 should provide some guideline. *In any case the screen output of the adaptation procedure should be inspected, and questionable results should not be used for simulation.*

Experience shows that in well-behaved $2 \to 4$ processes an accuracy (`Err/Exp` in the output) below 1 and an estimated reweighting efficiency above 10 % is reasonable. Due to the increased dimensionality of phase space, for $2 \to 6$ processes the numbers are typically worse by a factor 2 to 4. Activating ISR and beamstrahlung also reduces the efficiency.

If convergence is apparently not reached, this may be due to one of the following causes:

- The integral is infinite. One should check the cuts if they are sufficient for a finite result.
- The number of iterations is insufficient. For simple $2 \to 3$ processes, one reaches a stable result after $2 - 5$ iterations, but in complicated cases (many particles, photon emission with weak cuts, etc.) $20 - 30$ iterations may be needed. `WHIZARD` will make a guess, but this may be not sufficient in some cases.
- The number of calls per iteration is insufficient. With increasing dimension and complexity of the phase space, this number should be increased as well.
- The phase space parameterization is inadequate (see above). One may try to include or remove channels by modifying parameters such as the number of off-shell lines allowed in generating parameterizations. The examples in Sec. 4 illustrate some of the possible problems and solutions.

- The matrix element is numerically unstable. This has not (yet) been observed for `O'Mega` matrix elements, but cannot be excluded. In a future version of `WHIZARD` and `O'Mega`, quadruple precision will be available.

One should be careful to avoid the effects of gauge invariance violation, which can be large if unstable particles and photons are around. With `CompHEP` matrix elements, the `gwidth` switch should be set in this case, while `O'Mega` matrix elements should be generated with the '`f`' option.

3.5.4 Output files

All results are logged in the file `whizard.out`, which is rewritten after each iteration. Depending on the settings of the diagnostics parameters, it will contain additional information: The contents of the input file are repeated, including also the default values which have been inserted by the program. Furthermore, it shows the detailed history of the individual integration channels and of their relative weights.

After each iteration, the current `VAMP` grids are written to file as well: `whizard.grc` for the current grid and `whizard.grb` for the best grid. These may be reused in another run.

3.6 Event generation

Event generation is activated in the input file, either by specifying a nonzero luminosity

```
&process_input
...
luminosity = 100
/
```

or by a nonzero number of events

```
&simulation_input
n_events = 10000
/
```

In the former case, the actual number of generated events depends on the calculated cross section. If both numbers are given, the one leading to the larger event sample will be used.

The simulation makes sense only if `WHIZARD` could find a stable result for the cross section. This should be judged from inspecting the results displayed on screen and in the log file `whizard.out`.

With event generation activated, `WHIZARD` will not stop after the integration step, but start event generation. By default, `WHIZARD` will generate unweighted events, which result from reweighting the events produced by the `VAMP` sampling algorithm. In the event generation step, importance sampling is used, such that the events are random and have the proper distribution as required for a realistic simulation.

`WHIZARD` is able to write events to file in various formats:

- The file `whizard.evx` contains the events in a binary machine-dependent format. By setting `read_events_raw=T` they can be re-read, bypassing the time-consuming event generation.

- The file `whizard.evt` contains the events in a format controlled by the parameter `write_events_format`. The supported formats are:

  0 Verbose format (ASCII), useful mainly for debugging.

  1 HEPEVT format (ASCII). The format corresponds to the HEPEVT standard, such as if the contents of a HEPEVT common block were written in their natural order (Fig. 6, 7). As an extension, the helicity (-1, 0, or 1) is written for each particle in the same line after `JDAHEP`.

  2 SHORT format (ASCII). This is similar to HEPEVT, but unnecessary entries are suppressed (Fig. 8).

  3 STDHEP format (binary, machine-independent)[5]

[5]This works only if the `STDHEP` library is present.

```
integer, parameter :: nmxhep = 4000
integer :: nevhep, nhep
integer, dimension(nmxhep) :: isthep, idhep
integer, dimension(2, nmxhep) :: jmohep, jdahep
real(kind=double), dimension(5, nmxhep) :: phep
real(kind=double), dimension(4, nmxhep) :: vhep
common /HEPEVT/ nevhep, nhep, isthep, idhep, &
     & jmohep, jdahep, phep, vhep
```

Figure 6: *Definition of the HEPEVT common block in Fortran 90 format*

Within each event, the particles will appear in the order determined by the process definition (for `MadGraph` and `O'Mega`; `CompHEP` may reorder them). Since hadronization packages such as `JETSET` require a particular order of colored particles to properly assign showers, it be may necessary for the user to rearrange them before feeding them into the hadronization step.[6]

Reweighting Monte-Carlo events requires the knowledge of the highest weight. This can never be known with certainty in advance, so there may be some fraction of events with weight greater than one. By default, `WHIZARD` will sum up the excess weight of those events and show the induced error in the total cross section, which should be small compared to the statistical error of event generation. If desired, the excess events can be further analyzed by `WHIZARD`'s own analysis capabilities (see below), to make sure that they have no significant impact on final results.

`WHIZARD` will use the adaptation step to determine the highest weight beforehand. This requires sufficient statistics in the adaptation procedure. If many events have to be generated, the default choice for the number of calls per iteration may be too low, and too many excess events are produced. As a rule of thumb, the number of calls per adaptation iteration should be of the order of the number of unweighted events divided by the estimated efficiency.

Alternatively, a "test" run can be made by specifying a nonzero value of `n_events_warmup` in the input file. For instance, one may generate $200,000$ (weighted) events to determine the highest weight before generating the actual sample to be used for the reweighting procedure:

```
&simulation_input
n_events = 10000
n_events_warmup = 200000
/
```

Fortunately, this is not necessary in most cases since the pre-determined highest weight is usually quite accurate.

3.6.1 Built-in analysis

`WHIZARD` has capabilities to analyze the events without reference to external packages. To this end, a configuration file `whizard.cut5` should be set up which has a similar format as

[6] It is planned to implement a direct `PYTHIA` interface in `WHIZARD`.

```
      WRITE(22, 11, IOSTAT=ICHEK) NHEP
      DO I=1,NHEP
         WRITE (22, 13, IOSTAT=ICHECK) ISTHEP(I),IDHEP(I), &
       &              (JMOHEP(J,I),J=1,2), (JDAHEP(L,I),L=1,2)
         WRITE (22, 12, IOSTAT=ICHECK) (PHEP(J,I),J=1,5)
         WRITE (22, 12, IOSTAT=ICHECK) (VHEP(L,I),L=1,5)
      END DO

 11   FORMAT(1I5)
 12   FORMAT(10F17.10)
 13   FORMAT(I9,I9,4I5)
```

Figure 7: *Equivalent FORTRAN77 code which would produce the event file* ***whizard.evt*** *from the contents of a HEPEVT common block.*

```
write(u,11) evt%n_out
do i=1, evt%n_out
   p = array(particle_momentum(evt%prt(i)))
   write (u,13) particle_code(evt%prt(i)), evt%hel_out(i)
   write (u,12) p(1:3), p(0), particle_mass(evt%prt(i))
end do
11 format(1I5)
12 format(10F17.10)
13 format(I9,I9,5I5)
```

Figure 8: *Output routine for the SHORT event file format.*

```
! e-  e+  ->  e-  e+  gamma
! 16   8       1   2    4
process eeg
  cut M of 3 within  80 100
and
  cut M  of 3 within 180 200
  cut PT of 4 within 100 99999
and
  cut E of 4 within 0 100
  histogram PT of 1 within 0 500
```

Figure 9: *Analysis configuration file.*

`whizard.cut1`. For instance, in Fig. 9 a configuration file is shown which tells `WHIZARD` to analyze the generated event sample in three different ways: to calculate the cross section within a window $80 < M(e^+e^-) < 100$, then within $180 < M(e^+e^-) < 200$ with an additional cut on $p_\perp(\gamma)$, and finally to generate a histogram of $p_\perp(e^-)$ with a cut on the photon energy.

The histogram specification may take either one of the form

```
histogram PT of 1 within 0 500
histogram PT of 1 within 0 500 nbin 50
histogram PT of 1 within 0 500 step 10
```

Instead of the number of bins (20 by default), the bin width can be specified by the keyword `step`.

The cross section results are displayed on screen. In addition, they are written together with the histogram data to the file `whizard-plots.dat`. If `show_excess=T` has been set in the input file, the summary and the histograms will contain also information about any events with weight greater than one, such that their properties can be examined.

Clearly, for a quick analysis it is not necessary to simulate proper unweighted events. Therefore, one may set `unweighted=F` in order to analyze a sample of weighted events, which saves a lot of execution time.

The contents of `whizard-plots.dat` can be visualized directly by means of the gamelan graphical analysis package which is contained in the distribution. This requires the existence of the `MetaPost` program, which is part of many LaTeX distributions. Plots can then be made by typing

```
make plots
```

This will result in a Postscript file `whizard-plots.ps` which can be viewed and printed.

The event sample can further be analyzed in terms of helicity and flavor content. Example

```
! e-  e+  ->  q  qbar  b  bbar
! 16   8      1   2    4   8
```

```
process qqbb
  cut M of 3 within 85 95
  helicity  1 -1 -1 1
  helicity -1  1 -1 1
  flavor    1 -1 5 -5
  histogram M of 12 within 0 500
```

This would plot a histogram of the invariant mass of the $b\bar{b}$ pair with the additional requirement that the other quark pair must be $u\bar{u}$, and that the b (anti)quarks must be left- resp. right-handed.

The order should be as above: first cut specifications (the event must pass all cuts), then flavor and helicity specifications (the event must match any of the given helicity combinations and any of the given flavor combinations), finally histogram specifications. Any of these may be absent; an empty entry will just display the total cross section again. More analysis configurations can be added for the same process, they must be separated by the keyword `and`.

4 Examples

In the following sections three realistic applications of WHIZARD are presented: Higgs production in the four-fermion channels, Higgs pair production (six fermions), and strong WW scattering (six fermions). With certain refinements and modifications, they could be used for actual physical applications. However, here they are intended as a form of tutorial, showing typical setups and common problems. Some of the applications shown take a considerable amount of CPU time to run completely; for a first try, one may reduce this by removing flavor summation, require a smaller number of iterations, or consider simpler processes for test purposes.

4.1 Higgs production at LEP

Let us simulate the production of a 115 GeV Standard Model Higgs in e^+e^- collisions at 209 GeV. With this mass, the Higgs boson decays mainly into $\bar{b}b$ with a small fraction of $\tau^+\tau^-$ decays. Here, we only consider the signal together with its irreducible background, contributing to the processes

$$\begin{aligned}
e^-e^+ &\to \nu\bar{\nu}b\bar{b} \\
&\to q\bar{q}b\bar{b} \\
&\to b\bar{b}b\bar{b} \\
&\to \mu^-\mu^+b\bar{b} \\
&\to e^-e^+b\bar{b} \\
&\to q\bar{q}\tau^-\tau^+
\end{aligned}$$

where $q = u, d, s, c$ and $\nu = \nu_e, \nu_\mu, \nu_\tau$. This list of processes directly translates into the configuration file `whizard.prc`:

```
# WHIZARD configuration file
# The selected models (CompHEP/O'Mega):
model   sm-GF   4
model   SM

# Processes
# (Methods: chep=CompHEP, mad=MadGraph, omega=O'Mega)
# (Options: number=QCD order [Madgraph])
#
# Tag           In      Out                     Method  Option
#=============================================================
# On-shell process:
zh              e1,E1   Z,H                      chep

# Full four-fermion matrix elements (no QCD):
nnbb            e1,E1   n1:n2:n3,N1:N2:N3,b,B    omega
```

```
qqbb            e1,E1   u:d:s:c,U:D:S:C,b,B        omega
bbbb            e1,E1   b,B,b,B                    omega
eebb            e1,E1   e1,E1,b,B                  omega
mmbb            e1,E1   e2,E2,b,B                  omega
qqtt            e1,E1   u:d:s:c,U:D:S:C,e3,E3      omega
bbtt            e1,E1   b,B,e3,E3                  omega

# QCD contribution (gluon splitting):
uubb_qcd        e1,E1   u,U,b,B                    mad      2
ddbb_qcd        e1,E1   d,D,b,B                    mad      2
ssbb_qcd        e1,E1   s,S,b,B                    mad      2
ccbb_qcd        e1,E1   c,C,b,B                    mad      2
bbbb_qcd        e1,E1   b,B,b,B                    mad      2
```

Here, we use flavor summation for the quark and missing-energy channels. Since `O'Mega` can't yet handle QCD corrections where one quark pair originates from gluon fragmentation, those are implemented using `MadGraph` for the matrix elements. (We could also use `CompHEP` for that purpose. For $2 \to 4$ processes, however, `CompHEP` matrix elements typically need considerable more CPU time both in compilation and execution.)

After this file has been saved in the `conf` subdirectory, we should run `configure` in the directory where we had unpacked the distribution, if not already done,

```
> ./configure
```

and make and install the executable

```
> make install
```

After this step is completed, we are ready for running the program.

4.1.1 The on-shell process

All necessary files have now been installed in the `results` subdirectory:

```
> cd results
> ls
Makefile      whizard       whizard.cut5  whizard.mdl
Makefile.in   whizard.cut1  whizard.in    whizard.prc
```

The file `whizard.prc` is a copy of the one we prepared above. `whizard.mdl` is a list of Feynman rules to be considered for phase space generation; usually we don't need to modify it. The two cut configuration files are empty. The file which is of interest for us now is `whizard.in` which we have to edit:

```
 &process_input
 process_id = "zh"
```

```
sqrts = 209
/

&integration_input
stratified = F
/

&simulation_input /

&diagnostics_input /

&parameter_input
MH = 115
wH = 0.3228E-02
Mb = 2.9
Me = 0
Ms = 0
Mc = 0
/

&beam_input /

&beam_input /
```

As a warm-up, we examine the on-shell process $e^- e^+ \to ZH$, labeled `zh`. We insert the c.m. energy 209 GeV and the Higgs mass 115 GeV. The next two parameters are not relevant here, but will be needed for the other processes: The Higgs width value `wH` is the one returned by HDECAY [7]. The b mass `Mb` is the running mass evaluated at a scale around m_H. This is important since the Hbb coupling is proportional to m_b.[7] The setting `stratified=F` calls for importance sampling instead of stratified sampling (see below). All other fields are left empty.

Having saved `whizard.in`, the program can be started:

```
> ./whizard
```

(if you like, you can also type `make run`). Since this process is trivial, the results are there immediately:

```
! WHIZARD 1.00beta (Nov 15 2000)
! Reading process data from file whizard.in
! Process zh:
!   e a-e  ->   h   Z
!   8   4  ->   1   2
```

[7] For a genuine analysis, one should inspect the underlying assumptions and try to find a consistent parameter set. Since the `WHIZARD` matrix elements are evaluated at tree level, some higher-order corrections should not be included in the Higgs width.

```
! Reading phase space information from file whizard.phs ...
!            2  phase space channels found for process zh
! Reading cut configuration data from file whizard.cut1
! No cut data found for process zh
!
! WHIZARD run for process zh:            ( checksum =   2042343512 )
!-------------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!-------------------------------------------------------------------------
! Adapting (fixed weights):  Generating          1 sample of       5000
events ...
   1       5000  1.6112998E+02  1.16E-02    0.01    0.01* 99.43    0.00
!-------------------------------------------------------------------------
! Adapting (var. weights):  Generating          3 samples of       5000
events ...
   2       5000  1.6117984E+02  6.88E-02    0.04    0.03  91.43
   3       5000  1.6119047E+02  6.83E-02    0.04    0.03  90.85
   4       5000  1.6121313E+02  5.75E-02    0.04    0.03  92.52
!-------------------------------------------------------------------------
! Integrating (fixed w.):  Generating          1 sample of       5000
events ...
   5       5000  1.6116885E+02  6.74E-02    0.04    0.03  90.99    0.00
!-------------------------------------------------------------------------
!
! Time estimate for generating 10000 unweighted events:        0:00:00 hours
! WHIZARD run finished.
```

We read off the total cross section

$$\sigma = 161.17 \pm 0.07 \text{ fb}$$

Clearly the grid adaptation was of no use at all, since the final error is larger than the initial one. This would have been even worse if we had not inserted `stratified = F`. However, for less trivial processes stratified sampling usually gives better results, and in the examples presented below we keep `stratified = T`.

If we were interested in a more accurate result, we could increase the statistics like this

```
&integration_input
calls = 1 100000 1 100000 3 500000
...
```

(one initial run of 100,000 events, one run of 100,000 events for adaptation, 3 runs of 500,000 events for integration) and would get something like

$$\sigma = 161.1417 \pm 0.0006 \text{ fb}.$$

This serves merely as an illustration: for this process the cross section is easily obtained analytically, and the accuracy is limited by the input parameters anyway.

4.1.2 The missing-energy channel

The actual virtues of WHIZARD lie in calculating complete multi-fermion processes. Requesting the process

$$e^- e^+ \to \nu\bar{\nu}b\bar{b} \quad \text{where } \nu = \nu_e, \nu_\mu, \nu_t au$$

in the input file, which we have called nnbb,

```
&process_input
process_id = "nnbb"
sqrts = 209
/
```

and returning to the default integration setting

```
 &integration_input /
```

we arrive at a result like this:

```
! WHIZARD 1.00beta (Nov 15 2000)
! Reading process data from file whizard.in
! Process nnbb:
!    e a-e  -> nu_e a-nu_e   b a-b
!  32  16  ->    1      2   4   8
! Reading phase space information from file whizard.phs ...
! No entry found for process nnbb.
! Generating phase space channels for process nnbb ...
!          40 channels generated.
! Wrote new phase space configuration file whizard.phs
! Reading cut configuration data from file whizard.cut1
! No cut data found for process nnbb
!
! WHIZARD run for process nnbb:            ( checksum =  -1141000641 )
!-----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!-----------------------------------------------------------------------
! Adapting (fixed weights):  Generating          1 sample of       20000
events ...
   1      20000  1.0469953E+02  1.53E+00    1.46    2.06*  5.21    0.00
!-----------------------------------------------------------------------
! Adapting (var. weights):  Generating         10 samples of       20000
events ...
   2      20000  1.0644204E+02  9.77E-01    0.92    1.30*  9.42
   3      20000  1.0679193E+02  6.47E-01    0.61    0.86* 17.78
   4      20000  1.0716129E+02  6.67E-01    0.62    0.88  16.57
   5      20000  1.0645896E+02  5.69E-01    0.53    0.76* 18.23
   6      20000  1.0673084E+02  6.07E-01    0.57    0.80  16.95
   7      20000  1.0562857E+02  5.07E-01    0.48    0.68* 20.64
   8      20000  1.0894981E+02  2.17E+00    1.99    2.81   7.35
   9      20000  1.0587926E+02  5.20E-01    0.49    0.69  19.53
```

```
  10      20000  1.0699021E+02  5.05E-01    0.47    0.67* 19.41
  11      20000  1.0661609E+02  4.90E-01    0.46    0.65* 22.29
!----------------------------------------------------------------------
! Integrating (fixed w.):  Generating          1 sample of       20000
events ...
  12      20000  1.0655991E+02  6.05E-01    0.57    0.80  18.03    0.00
!----------------------------------------------------------------------
!
! Time estimate for generating 10000 unweighted events:        0:01:06 hours
! WHIZARD run finished.
```

The cross section is[8]

$$\sigma = 106.6 \pm 0.6 \text{ fb}$$

This calculation takes about 5 minutes on a state-of-the-art Alpha processor (Nov. 2000), and about 15 minutes on a standard PC. Of course, if we invest more CPU time, we can improve the accuracy, as explained in the previous subsection.

In inspecting the output, the column tagged `Err/Exp` is of main interest (read: relative error over expected relative error). This is the estimated relative error multiplied by the square root of the number of calls, a number which should be of the order one. Each time this error improves, the entry is marked with a star. The best grid so far (the last one with a star) will be used for integration, and later for event generation. As evident from the column `Eff[%]`, the estimated reweighting efficiency improves as well. In the last column the χ^2 value divided by the number of iterations is shown. This value is meaningful only if there is more than one iteration in the integration step. This is not the case here, so it is zero in our example.

Let us now generate events. We define a nonzero luminosity (100 fb^{-1}, which is more than 100 times the LEP integrated luminosity per year):

```
&process_input
process_id = "nnbb"
sqrts = 209
luminosity = 100
/
```

We do not want to repeat the adaptation and integration steps, therefore we read in the previously adapted grids

```
&integration_input
read_grids = T
/
```

Now running `WHIZARD` will result in a repetition of the previous result, before event generation is started:

[8]This number should be taken with a grain of salt. See footnote 7.

```
! Reading analysis configuration data from file whizard.cut5
! No analysis data found for process nnbb
!
! Event sample corresponds to luminosity [fb-1] =   100.0
!
! Generating      10656 unweighted events ...
!=========================================================================
! Analysis results for the generated event sample:
!-------------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!-------------------------------------------------------------------------
  13      10656  1.0655991E+02  1.03E+00    0.97    1.00 100.00
!-------------------------------------------------------------------------
! Excess events:   14.1       (Error[%]:    0.13 )
! WHIZARD run finished.
```

The generation of 10,656 events takes one more minute in this case (on an Alpha processor). No further analysis has been requested, so we just get the total cross section again. By definition, this is equal to the previously calculated cross section, but the error now corresponds to the actual event sample.

There are some excess events (weight greater than one), but the effect of this excess being dropped is considerably less than the statistical error of the event sample. If necessary, the excess could be reduced by a higher-statistics adaptation or by a warmup run with large statistics:

```
&simulation_input
n_events_warmup = 500000
/
```

The event sample can be further analyzed: Let us plot the missing invariant mass distribution and the dijet invariant mass distribution, which should exhibit peaks at the Z and Higgs mass. To this end, we tell `WHIZARD` to reread the generated event sample

```
&simulation_input
read_events_raw = T
/
```

and make up an analysis configuration file `whizard.cut5`:

```
# cut/histogram configuration file
# e- e+ -> nu nubar b bbar
# 32 16    1    2   4  8
process nnbb, qqbb, bbbb, eebb, mmbb, qqtt, bbtt,
        uubb_qcd, ddbb_qcd, ssbb_qcd, ccbb_qcd, bbbb_qcd
  histogram M of 3 within 0 209 nbin 40
and
  histogram M of 12 within 0 209 nbin 40
and
```

```
cut M of 12 within 114 116
histogram M of 3 within 0 209 nbin 40
```

This type of analysis can serve for all processes, therefore we have added the other tags. The last histogram counts only those events which have a $b\bar{b}$ invariant mass close to the Higgs mass. The numbers result from adding up the binary codes of the external particles, which are shown as a the comment in the file header.

Running WHIZARD this time takes almost no time since only the previously generated files have to be read:

```
! Reading analysis configuration data from file whizard.cut5
! Found           3 analysis configuration datasets
!
! Event sample corresponds to luminosity [fb-1] =   100.0
!
! Looking for raw event file whizard.evx ...
! Reading      10656 unweighted events ...
!=======================================================================
! Analysis results for the generated event sample:
!-----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!-----------------------------------------------------------------------
  13      10656  1.0655991E+02  1.03E+00    0.97    1.00 100.00
!=======================================================================
! Analysis results for the generated event sample:
!-----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!-----------------------------------------------------------------------
  13      10656  1.0655991E+02  1.03E+00    0.97    1.00 100.00
!=======================================================================
! Analysis results for the generated event sample:
!
! Additional cuts:
! integration level            5
cut M of  12  12 within  1.14000E+02  1.16000E+02
!-----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!-----------------------------------------------------------------------
  13       2276  2.2759980E+01  4.77E-01    2.10    1.00  21.36
! WHIZARD run finished.
```

From the last line we read off that the actual contribution of Higgs production in this channel is 2276 events (or 21.36 %). Here the efficiency is not the reweighting efficiency as before, but the fraction of events remaining after cuts. The histograms can now be found in the output file `whizard-plots.dat`:

```
!=======================================================================
! WHIZARD 1.00beta (Nov 15 2000)
! Process nnbb:
!   e a-e  -> nu_e a-nu_e   b a-b
!  32  16  ->    1      2   4   8
```

```
! Analysis results for the generated event sample:
!
! Histograms:
!
histogram M of   3     within  0.00000E+00  2.09000E+02  nbin   40
  2.61250000       1.00000000       1.00000000       0.00000000
  7.83750000       2.00000000       1.41421356       0.00000000
  13.0625000       3.00000000       1.73205081       0.00000000
  18.2875000       4.00000000       2.00000000       0.00000000
  23.5125000       4.00000000       2.00000000       0.00000000
  28.7375000       8.00000000       2.82842712       0.00000000
  33.9625000       12.0000000       3.46410162       0.00000000
  39.1875000       23.0000000       4.79583152       0.00000000
  44.4125000       12.0000000       3.46410162       0.00000000
  49.6375000       35.0000000       5.91607978       0.00000000
  54.8625000       38.0000000       6.16441400       0.00000000
  60.0875000       47.0000000       6.85565460       0.00000000
  65.3125000       73.0000000       8.54400375       0.00000000
  70.5375000       93.0000000       9.64365076       0.00000000
  75.7625000       123.000000       11.0905365       0.00000000
  80.9875000       277.000000       16.6433170       0.00000000
  86.2125000       1106.00000       33.2565783       0.00000000
...
```

In these histograms, the first column is the bin midpoint. The second column is the number of entries, the third column the statistical error on this number ($\sqrt{N}$). The fourth column would show the excess events (since we have read the events from file, they are not shown here).

The actual histograms are shown in Fig. 10. In the dijet invariant mass distribution the Z and Higgs peaks are clearly visible. At low dijet invariant mass there is a tail of continuum $b\bar{b}$ production. In the missing mass distribution the peak is at the Z mass. If the dijet mass is required to be close to the Higgs mass, the background with $M_{\rm miss} > \sqrt{s} - M_H$ is rejected, including continuum $b\bar{b}$ production from WW fusion which peaks at high missing mass. The Higgs signal, which includes Higgs-strahlung, WW fusion and interference, is retained.

4.1.3 The four-jet channel

There are two four-jet channels

$$e^-e^+ \to q\bar{q}b\bar{b}, \quad b\bar{b}b\bar{b}.$$

If the light quark masses are set to zero (see the input file above), all light quark channels can be treated in a single run. For unequal masses, flavor summation is not possible, therefore the b channel has to be treated separately. (We can't set the b mass to zero since this would remove the b Yukawa coupling.)

Running the program as before for the process `qqbb` with the given input and analysis configuration files takes about 10 minutes in total on an Alpha processor. The cross section result is

$$\sigma = 284.0 \pm 1.2 \text{ fb}$$

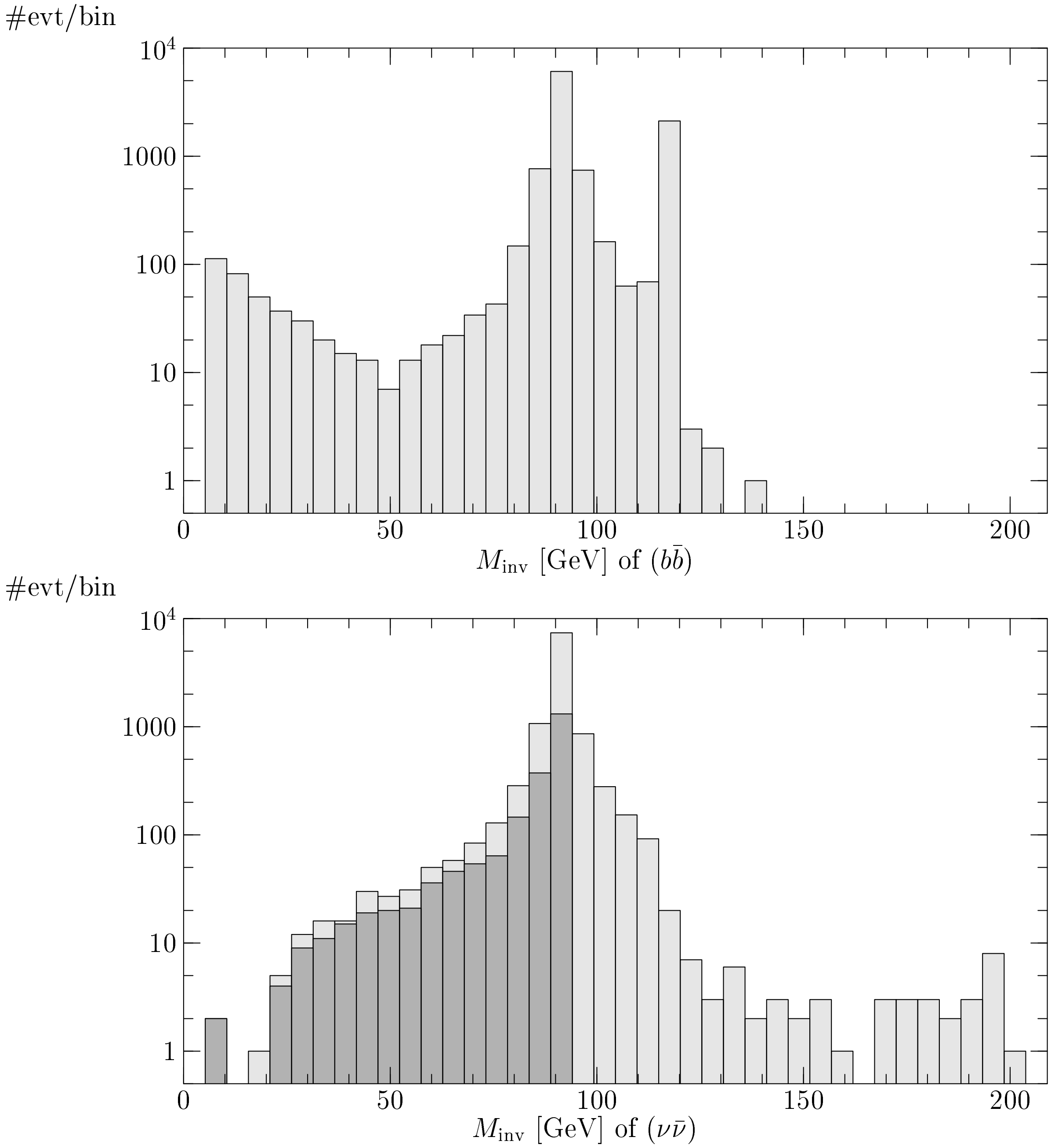

Figure 10: *Histograms for the process* $e^-e^+ \to \nu\bar{\nu}b\bar{b}$ *at* $\sqrt{s} = 209$ *GeV with* $m_H = 115$ *GeV. Top: Dijet invariant mass distribution. Bottom: Missing invariant mass distribution; light: all events, dark: only those events with* 114 *GeV* $< M(b\bar{b}) <$ 116 *GeV.*

and 28399 events are generated in this run, 4704 of them within the Higgs mass window.

To obtain a finite result, WHIZARD had to insert a cut of 10 GeV on the dijet invariant mass of the light quarks. This cut is written into the file whizard.cut0:

```
! Automatically generated set of cuts
! Process qqbb:
!   e a-e  ->   u a-u   b a-b
!  32  16  ->   1   2   4   8
process qqbb
cut M of   3     within  1.00000E+01  1.00000E+99
```

The cut could be modified either by changing the default value in whizard.in, e.g.

```
&integration_input
...
default_jet_cut = 5
/
```

or by providing a non-empty file whizard.cut1 of the same format, with a different set of cuts. If an appropriate process entry is found in this file, any default cuts are ignored.

The $b\bar{b}b\bar{b}$ channel is similar, although the presence of identical particles in the final state makes phase space more complicated. The adaptation again works well and we get

$$\sigma = 44.5 \pm 0.2 \text{ fb} \tag{1}$$

The τ lepton channels pose no additional problem.

4.1.4 The lepton channels

In the $\mu^+\mu^-$ channel adaptation is slightly worse due to the small muon mass, if no cuts are provided.

$$\sigma = 52.5 \pm 0.4 \text{ fb}$$

If there are cuts, we can set the muon mass to zero, which as a side effect speeds up the calculation by a factor of two.

A more difficult task is to get to a stable result for the cross section of the process

$$e^-e^+ \to e^-e^+\bar{b}b$$

without cuts, but with nonzero electron mass. The default choice for the number of calls and iterations does not suffice here:

```
! WHIZARD run for process eebb:            ( checksum =    539229792 )
!----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!----------------------------------------------------------------------
```

```
! Adapting (fixed weights):  Generating          1 sample of       20000
events ...
   1      20000  2.0951810E+02  1.23E+02   58.52   82.76*  0.27    0.00
!----------------------------------------------------------------------
! Adapting (var. weights):  Generating         10 samples of       20000
events ...
   2      20000  1.5504546E+02  3.19E+01   20.60   29.13*  0.37
   3      20000  2.2879370E+02  4.00E+01   17.49   24.74*  0.34
   4      20000  2.8014176E+02  7.74E+01   27.63   39.07   0.31
   5      20000  4.4875009E+02  9.73E+01   21.69   30.67   0.20
   6      20000  9.5489073E+02  3.00E+02   31.38   44.37   0.10
   7      20000  6.5788137E+02  8.21E+01   12.49   17.66*  0.19
   8      20000  1.8098498E+03  5.94E+02   32.79   46.38   0.09
   9      20000  5.4673479E+03  2.71E+03   49.65   70.21   0.05
...
```

Apparently, convergence is not reached. We reuse these grids, but do further iterations with higher statistics:

```
&integration_input
calls = 1 20000 30 100000 1 100000
read_grids = T
/
```

For the first iterations the old results are copied. The additional high-statistics iterations now find convergence, although the final accuracy and efficiency are not as good as for the other processes:

```
...
  26     100000  7.6569064E+03  9.51E+01    1.24    3.93*  0.53
  27     100000  7.5403570E+03  6.52E+01    0.86    2.73*  1.37
  28     100000  7.6645772E+03  5.83E+01    0.76    2.40*  1.40
  29     100000  7.7191224E+03  5.63E+01    0.73    2.31*  1.09
  30     100000  7.6527526E+03  6.15E+01    0.80    2.54   0.99
  31     100000  7.5842005E+03  5.07E+01    0.67    2.11*  2.32
!----------------------------------------------------------------------
! Integrating (fixed w.):  Generating          1 sample of      100000
events ...
  32     100000  7.6335267E+03  4.39E+01    0.58    1.82*  2.76    0.00
!----------------------------------------------------------------------
!
! Time estimate for generating 10000 unweighted events:        0:11:40 hours
```

Of course, the CPU time needed is much longer than before, scaling linearly with the total number of calls. With a cross section as large as this and a comparatively low efficiency (which could still be somewhat improved by further adaptation iterations), the adaptation and integration takes one hour.

With suitable cuts a stable result would have been reached as fast as in the other processes. The extra effort is needed only if we are particularly interested in the events with electron and positron going in the very forward and backward regions, respectively.

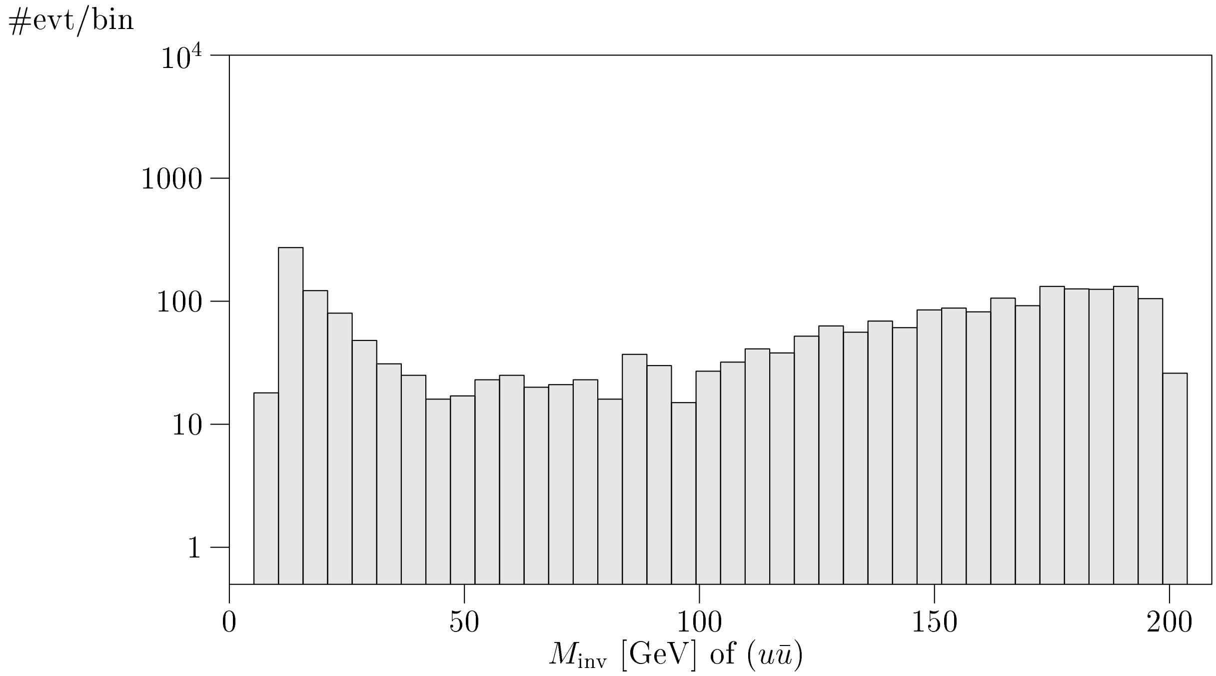

Figure 11: *Invariant mass distribution of the light quark pair for the process $e^- e^+ \to u\bar{u}b\bar{b}$, taking into account only the QCD contribution.*

4.1.5 QCD contribution

The QCD background where one quark pair results from gluon splitting is not included in the above. It has been listed separately in our process set. Here, no flavor summation is (yet) possible, and we have to calculate the individual quark flavors separately. Taking, e.g., the channel `uubb_qcd`

$$e^- e^+ \to u\bar{u}b\bar{b} \qquad \text{(QCD contribution)}$$

with the default cut $M(u\bar{u}) > 10$ GeV, we obtain

$$\sigma = 23.78 \pm 0.16 \text{ fb}$$

with the default settings. The invariant mass distribution of the jet pair can be analyzed as before (Fig. 11). Although the total cross section is quite large, the amount contributing in the Higgs mass region is tiny (0.34 %), as the analysis shows:

```
! Additional cuts:
! integration level          5
cut M of  12  12 within  1.14000E+02  1.16000E+02
!----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!----------------------------------------------------------------------
  13         8  7.9985184E-02  2.83E-02   35.36    1.00   0.34
```

4.2 6-fermion production: Higgs pairs

A process which is of interest at a future Linear Collider is Higgs pair production, which is sensitive to the Higgs trilinear coupling. A process definition file `whizard.prc` for this process could look like

```
# WHIZARD configuration file
# The selected models (CompHEP/O'Mega):
model   sm-GF   4
model   SM

# Processes
# (Methods: chep=CompHEP, mad=MadGraph, omega=O'Mega)
# (Options: number=QCD order [Madgraph])
#
# Tag           In      Out                     Method  Option
#============================================================
# On-shell process:
zhh             e1,E1   Z,H,H                    chep

# Full six-fermion matrix elements (no QCD):
qqbbbb          e1,E1   u:d:s:c,U:D:S:C,b,B,b,B   omega

# QCD contribution (gluon splitting):
uubbbb_qcd1     e1,E1   u,U,b,B,b,B              mad     2
uubbbb_qcd2     e1,E1   u,U,b,B,b,B              mad     4
```

For simplicity, we consider only six-quark production where four quarks are b quarks (the main decay channel of the Higgs pair), and the remaining quark pair is light. This is supplemented by second- and fourth-order QCD contributions calculated by `MadGraph`.

The input file `whizard.in` is prepared for the six-fermion process at $\sqrt{s} = 500$ GeV with default settings. We would like to generate an event sample corresponding to 10 ab^{-1}.

```
&process_input
process_id = "qqbbbb"
sqrts = 500
luminosity = 1000
/

&integration_input /

&simulation_input /

&diagnostics_input /
```

```
&parameter_input
MH = 115
wH = 0.3228E-02
Mb = 2.9
Me = 0
Ms = 0
Mc = 0
/

&beam_input /

&beam_input /
```

Without further considerations, we start integration and event generation

```
> ./whizard
```

After the initial message

```
! WHIZARD 1.00beta (Nov 15 2000)
! Reading process data from file whizard.in
! Process qqbbbb:
!   e a-e  ->   u a-u   b a-b   b a-b
! 128  64  ->   1   2   4   8  16  32
! Phase space configuration file whizard.phs not found
! Generating phase space channels for process qqbbbb ...
```

the program needs quite some time for generating the phase space configuration (this would be faster without flavor summation). Fortunately, the configuration is written to the file `whizard.phs` and `WHIZARD` will reuse it when possible.

After this step is finished, a default cut for the light quark pair is inserted and integration is started.

```
!         104 channels generated.
! Wrote new phase space configuration file whizard.phs
! *** Warning: The cross section may be infinite without cuts.
! Wrote default cut configuration file whizard.cut0
! User cut configuration file whizard.cut1 not found.
! Using default cuts.
cut M of   3     within  1.00000E+01  1.00000E+99
!
! WHIZARD run for process qqbbbb:            ( checksum =   -214172314 )
!-----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!-----------------------------------------------------------------------
! Adapting (fixed weights):  Generating          1 sample of      100000
events ...
```

Since the physical region is small in some of the phase space channels and the number of 1000 events per channel is not very large, messages like

```
! Warning: Function identically zero in channel          4
```

may appear, which usually can be ignored. The whole adaptation and integration run takes a considerable amount of CPU time (about one day on an Alpha processor). If only one quark flavor were considered, this could be reduced by a factor of four (currently, `WHIZARD/O'Mega` does not take advantage of the fact that some matrix elements with different flavor content are in fact identical).

A usable result is already reached after about 10 iterations, with considerable fluctuation around the optimal grid[9]:

```
   1     100000  1.7245154E-01  9.02E-03    5.23   16.54*  0.73    0.00
!----------------------------------------------------------------------
! Adapting (var. weights):  Generating         20 samples of      100000
events ...
   2     100000  1.4965342E-01  2.15E-03    1.43    4.54*  1.97
   3     100000  1.4929719E-01  1.29E-03    0.86    2.73*  2.16
   4     100000  1.5426710E-01  5.29E-03    3.43   10.84   0.99
   5     100000  1.5014838E-01  1.48E-03    0.99    3.12   1.93
   6     100000  1.4741355E-01  1.11E-03    0.75    2.37*  3.02
   7     100000  1.5119871E-01  1.33E-03    0.88    2.78   2.20
   8     100000  1.4969993E-01  8.36E-04    0.56    1.77*  3.26
   9     100000  1.5623658E-01  5.13E-03    3.28   10.38   1.17
  10     100000  1.5145665E-01  7.81E-04    0.52    1.63*  3.55
  11     100000  1.5170008E-01  1.05E-03    0.69    2.19   3.22
  12     100000  1.5207701E-01  9.18E-04    0.60    1.91   3.35
  13     100000  1.6122997E-01  1.03E-02    6.36   20.13   0.88
  14     100000  1.5167050E-01  7.18E-04    0.47    1.50*  4.22
  15     100000  1.5346776E-01  1.60E-03    1.04    3.30   2.59
  16     100000  1.5190641E-01  9.46E-04    0.62    1.97   3.72
  17     100000  3.9266569E-01  2.40E-01   61.19  193.49   0.11
  18     100000  1.5427618E-01  1.94E-03    1.26    3.98   2.23
  19     100000  1.5282396E-01  1.90E-03    1.24    3.94   2.57
  20     100000  1.5299694E-01  1.11E-03    0.73    2.30   3.15
  21     100000  1.6406596E-01  1.30E-02    7.90   24.98   0.68
!----------------------------------------------------------------------
! Integrating (fixed w.):  Generating          1 sample of      100000
events ...
```

Nevertheless, the best grid obtained so far can safely be used for event generation. The final estimate for the integral is

```
  22     100000  1.5188597E-01  7.48E-04    0.49    1.56   4.19    0.00
!----------------------------------------------------------------------
!
! Time estimate for generating 10000 unweighted events:        2:24:24 hours
```

However, we don't need that many events: simulating 10 ab^{-1} now takes only half an hour:

```
! Event sample corresponds to luminosity [fb-1] =  0.1000E+05
```

[9]By reducing the parameter `weights_power` in the input file, these fluctuations can be somewhat dampened.

```
!
! Generating       1519 unweighted events ...
!=======================================================================
! Analysis results for the generated event sample:
!-----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!-----------------------------------------------------------------------
  23       1519  1.5188597E-01  3.90E-03    2.57    1.00 100.00
!-----------------------------------------------------------------------
! Excess events:    6.8       (Error[%]:    0.45 )
! WHIZARD run finished.
```

If we wish to know how many of those events are originating from HH pairs, we should set up an analysis configuration file `whizard.cut5` like this

```
! e- e+ -> q qbar b bbar b bbar
!128 64    1  2   4  8  16  32
process qqbbbb
  cut M of 12 within 114 116
  cut M of 48 within 114 116
and
  cut M of 36 within 114 116
  cut M of 24 within 114 116
```

and rerun the program, setting `read_grids=T` and `read_events_raw=T`. The result is

```
! Analysis results for the generated event sample:
!
! Additional cuts:
! integration level           5
cut M of  12  12 within  1.14000E+02  1.16000E+02
cut M of  48  48 within  1.14000E+02  1.16000E+02
!-----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!-----------------------------------------------------------------------
  23        150  1.4998614E-02  1.22E-03    8.16    1.00   9.87
!=======================================================================
! Analysis results for the generated event sample:
!
! Additional cuts:
! integration level           5
cut M of  36  36 within  1.14000E+02  1.16000E+02
cut M of  24  24 within  1.14000E+02  1.16000E+02
!-----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!-----------------------------------------------------------------------
  23        199  1.9898162E-02  1.41E-03    7.09    1.00  13.10
! WHIZARD run finished.
```

The two event samples may be added (assuming that no events pass both cuts simultaneously), to yield 349 “signal” events.

The stability of the result and the computing time can be improved by reducing the number of phase space channels (see Sec. 3.5.2).

4.3 Vector boson scattering: polarization and beamstrahlung

In case no light Higgs boson exists, one will try to measure vector boson scattering at high-energy colliders, e.g.

$$W^+W^- \to W^+W^-,\ ZZ$$

At an e^+e^- collider these processes occur as a subprocess of

$$e^-e^+ \to \nu_e\bar{\nu}_e q\bar{q}q\bar{q}.$$

This can be simulated in a single run, if we set up the process configuration file `whizard.prc` as follows:

```
# WHIZARD configuration file
# The selected models (CompHEP/O'Mega):
model   chpt-GF   4
model   SM_ac

# Processes
# (Methods: chep=CompHEP, mad=MadGraph, omega=O'Mega)
# (Options: number=QCD order [Madgraph])
#
# Tag    In      Out                                 Method Option
#=================================================================
# On-shell process:
ww       W+,W-   W+,W-                               chep
zz       W+,W-   Z,Z                                 chep

# Full six-fermion matrix elements (no QCD):
nnqqqq   e1,E1   n1:n2:n3,N1:N2:N3,u:d,U:D,u:d,U:D   omega
enqqqq   e1,E1   e1,N1,u:d,U:D,u:d,U:D               omega   f
neqqqq   e1,E1   n1,E1,u:d,U:D,u:d,U:D               omega   f
eeqqqq   e1,E1   e1,E1,u:d,U:D,u:d,U:D               omega   f

# First neutrino generation only:
# WW and ZZ
nnuudd   e1,E1   n1,N1,u,U,d,D                       omega
# WW only
nnucsd   e1,E1   n1,N1,u,C,s,D                       omega
eeucsd   e1,E1   e1,E1,u,C,s,D                       omega   f
# ZZ only
nnuuss   e1,E1   n1,N1,u,U,s,S                       omega
# WZ
enudss   e1,E1   e1,N1,u,D,s,S                       omega   f
```

```
# Second neutrino generation only:
# WW and ZZ
nnuudd2  e1,E1   n2,N2,u,U,d,D                                omega
# WW only
nnucsd2  e1,E1   n2,N2,u,C,s,D                                omega
# ZZ only
nnuuss2  e1,E1   n2,N2,u,U,s,S                                omega
```

The models we have chosen correspond to the no-Higgs model with anomalous quartic vector boson couplings. The Higgs mass is set to a very large value (resp. infinity) by default. Apart from the signal process nnqqqq we have included some important background processes, which must be considered if the final-state electron is not observed. For these processes, gauge invariance is an issue, and we must set the f (fudged-width) option to obtain a consistent result.

The summation over flavors in the processes nnqqqq etc. is convenient, but it is a luxury in terms of computing time, even if we include only the first quark generation, as indicated above. It is more economical to generate definite flavor states, where particular states like $u\bar{c}s\bar{d}$ project either on WW or on ZZ pairs (the interference of both is negligible, and will not be taken into account in the hadronization anyway). Therefore, in the following we investigate the processes nnuudd etc. with a definite quark content, assuming that the event rates are multiplied by appropriate factor in order to take the missing flavor combinations into account.

The setup below is for the planned TESLA collider. We may have polarization and have to account for ISR and beamstrahlung. The input file whizard.in specifies 80 % left-handed electron polarization and 40 % right-handed positron polarization:

```
&process_input
process_id = "nnqqqq"
sqrts = 800
luminosity = 1000
polarized_beams = T
/

&integration_input
double_off_shell_branchings = F
single_off_shell_branchings = F
single_off_shell_decays = F
massive_FSR = F
exchange_lines = 2
/

&simulation_input /

&diagnostics_input /
```

```
&parameter_input
a4 = 0
a5 = 0
Me = 0
Ms = 0
Mc = 0
/

&beam_input
polarization = 0.80 0
fixed_energy = F
CIRCE_on = T
CIRCE_acc = 2
ISR_on = T
ISR_alpha = 0.0072993
ISR_m_in = 0.000511
/

&beam_input
polarization = 0 0.40
fixed_energy = F
CIRCE_on = T
CIRCE_acc = 2
ISR_on = T
ISR_alpha = 0.0072993
ISR_m_in = 0.000511
/
```

The beamstrahlung settings (`CIRCE`) are for the accelerator type 2 (TESLA), default parameterization version and revision numbers. Concerning initial-state radiation (ISR), we should set the electromagnetic coupling constant equal to the low-energy value of 1/137 since on-shell photons are radiated. The incoming mass must be reset equal to 511 keV, since the physical electron mass `Me` has been set to zero.

In the parameter section, the two anomalous couplings α_4 and α_5 are included. Here, we set them to zero which is also the default value.

The phase space setup for processes like this turns out to be critical. In general, the default setup could result in a too large number of channels. The settings in `&integration_input` here result in a manageable number (see Sec. 3.5.2). Although irrelevant channels will be removed during the adaptation procedure, the convergence is much faster if they are absent from the beginning.

The setting `exchange_lines = 2` is important. The meaning of this parameter is somewhat obfuscated: For each graph, on may determine the *difference* of the multiplicity and the number of t-channel (exchange) lines. The multiplicity is the number of produced external or resonant particles, not counting cascade decays. If this number is less than the value of `exchange_lines`, the channel will be included even if it has one off-shell line more than allowed.

With the default value (1), the signal graphs we are interested in (WW scattering) would not be included in the phase space setup. In the absence of a Higgs boson, the multiplicity of these graphs is 4, since 2 fermions and 2 vector bosons are produced on-shell resp. resonant. The s-channel WW scattering graph (containing $WW \to \gamma^* \to WW$) has 2 t-channel lines, so the difference is 2. This is the value of the parameter `exchange_lines` we need. If there were a resonant Higgs boson, this problem would be absent since the multiplicity of the same graph would be 3.

Unfortunately, this is not sufficient to yield stable results for all six-fermion channels in the present case. The problem is that the effective Zee coupling is accidentally suppressed. This fact is not taken into account by the `WHIZARD` phase space configuration. Although the corresponding channels die out during adaptation, convergence is bad in the ZZ production processes. This can be observed by inspecting the `whizard.out` log file, where the history of the relative channel weights is shown. A straightforward strategy is therefore to start a run where the irrelevant channels have been removed by hand (commented out) from the configuration file `whizard.phs`. If `read_phase_space = T` (default), a pre-generated file will be read again. In the present case, the rule is to remove all channels which do not have at least two W bosons or one neutrino as internal lines.

With these caveats, `WHIZARD` is able to integrate the corresponding cross sections and generate event samples. The output shown below is for the complete process, including all possible flavor combinations in the process labeled `nnqqqq`

$$e^-e^+ \to \nu_e\bar{\nu}_e q\bar{q}q\bar{q}; \qquad \nu = \nu_e, \nu_\mu, \nu_\tau; \qquad q = u, d$$

```
! WHIZARD 1.00beta (Nov 15 2000)
! Reading process data from file whizard.in
! Process nnqqqq:
!   e a-e  -> nu_e a-nu_e   u a-u   u a-u
! 128  64  ->    1      2   4   8  16  32
circe:warning: *************************************
circe:warning: * This release is not official yet, *
circe:warning: * do not use it in publications!    *
circe:warning: *************************************
! Reading phase space information from file whizard.phs ...
!          168  phase space channels found for process nnqqqq
! *** Warning: The cross section may be infinite without cuts.
! Wrote default cut configuration file whizard.cut0
! User cut configuration file whizard.cut1 not found.
! Using default cuts.
cut M of  12     within  1.00000E+01  1.00000E+99
cut M of  20     within  1.00000E+01  1.00000E+99
cut M of  36     within  1.00000E+01  1.00000E+99
cut M of  24     within  1.00000E+01  1.00000E+99
cut M of  40     within  1.00000E+01  1.00000E+99
cut M of  48     within  1.00000E+01  1.00000E+99
!
! WHIZARD run for process nnqqqq:            ( checksum =   1659754136 )
```

```
!----------------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!----------------------------------------------------------------------------
! Adapting (fixed weights):  Generating          1 sample of      200000
events ...
   1     200000  6.0040794E+00  6.18E-01   10.29   46.00*  0.19    0.00
!----------------------------------------------------------------------------
! Adapting (var. weights):  Generating         30 samples of      300000
events ...
   2     300000  5.7430567E+00  2.57E-01    4.47   24.48*  0.20
   3     300000  6.2731287E+00  2.93E-01    4.67   25.59   0.19
   4     300000  6.3060272E+00  2.25E-01    3.56   19.52*  0.21
   5     300000  5.8487971E+00  1.84E-01    3.15   17.25*  0.28
   6     300000  6.3231405E+00  2.66E-01    4.21   23.08   0.24
   7     300000  6.2152705E+00  1.57E-01    2.52   13.81*  0.30
   8     300000  6.0017351E+00  1.15E-01    1.91   10.48*  0.38
   9     300000  6.0617537E+00  1.22E-01    2.01   11.00   0.38
  10     300000  6.3693499E+00  1.71E-01    2.68   14.69   0.34
  11     300000  6.2081047E+00  1.09E-01    1.76    9.64*  0.41
  12     300000  6.1271515E+00  7.48E-02    1.22    6.68*  0.50
  13     300000  6.2637971E+00  1.42E-01    2.27   12.42   0.52
  14     300000  6.2333353E+00  4.80E-02    0.77    4.22*  0.88
  15     300000  6.1428110E+00  4.24E-02    0.69    3.78*  1.00
  16     300000  6.1264061E+00  7.16E-02    1.17    6.40   0.74
  17     300000  6.1872816E+00  5.17E-02    0.84    4.58   0.94
  18     300000  6.1253950E+00  3.02E-02    0.49    2.70*  1.65
  19     300000  6.1726745E+00  3.29E-02    0.53    2.92   1.38
  20     300000  6.1184702E+00  2.59E-02    0.42    2.32*  2.01
  21     300000  6.1153004E+00  2.35E-02    0.38    2.11*  2.35
  22     300000  6.1568659E+00  2.41E-02    0.39    2.14   2.07
  23     300000  6.0982476E+00  2.18E-02    0.36    1.96*  2.49
  24     300000  6.1644998E+00  2.58E-02    0.42    2.29   2.32
  25     300000  6.1176604E+00  2.00E-02    0.33    1.79*  2.75
  26     300000  6.1114816E+00  2.17E-02    0.35    1.94   2.49
  27     300000  6.1406600E+00  3.39E-02    0.55    3.03   1.73
  28     300000  6.1303492E+00  2.50E-02    0.41    2.24   1.87
  29     300000  6.1456257E+00  2.03E-02    0.33    1.81   2.78
  30     300000  6.1410563E+00  2.26E-02    0.37    2.02   2.47
  31     300000  6.1278282E+00  2.69E-02    0.44    2.40   2.31
!----------------------------------------------------------------------------
! Integrating (fixed w.):  Generating          2 samples of      300000
events ...
  33     600000  6.1017075E+00  1.36E-02    0.22    1.73*  2.40    0.68
!----------------------------------------------------------------------------
!
! Time estimate for generating 10000 unweighted events:        5:17:12 hours
! Analysis configuration file not found
!
! Event sample corresponds to luminosity [fb-1] =   1000.
!
! Generating       6102 unweighted events ...
!============================================================================
```

```
! Analysis results for the generated event sample:
!----------------------------------------------------------------------
! It     Calls   Integral[fb]  Error[fb]  Err[%] Err/Exp Eff[%]   Chi2
!----------------------------------------------------------------------
  34      6102  6.1017075E+00  7.81E-02    1.28    1.00 100.00
!----------------------------------------------------------------------
! Excess events:   19.2       (Error[%]:    0.31 )
! WHIZARD run finished.
```


Acknowledgements

I am most grateful to Thorsten Ohl, who provided me with pre-release versions of VAMP and O'Mega and had many valuable suggestions and criticisms during the development of WHIZARD. I would also like to thank R. Chierici, K. Desch, N. Meyer and S. Rosati, who used preliminary versions of the program for real-life applications and thus helped a lot in debugging and improving the code.

References

[1] T. Sjöstrand, Comput. Phys. Commun. **82** (1994) 74.

[2] A. Pukhov, *et al.*, Preprint INP MSU 98-41/542, hep-ph/9908288.

[3] T. Stelzer and W.F. Long, Comput. Phys. Commun. **81** (1994) 357.

[4] T. Ohl, to appear in: *Proceedings of the Seventh International Workshop on Advanced Computing and Analysis Technics in Physics Research*, ACAT 2000, Fermilab, October 2000, IKDA-2000-30, hep-ph/0011243; M. Moretti, Th. Ohl, and J. Reuter, to be published.

[5] T. Ohl, Comput. Phys. Commun. **120** (1999) 13.

[6] T. Ohl, Comput. Phys. Commun. **101** (1997) 269.

[7] A. Djouadi, J. Kalinowski, M. Spira, Comput. Phys. Commun. **108** (1998) 56-74.