EKHARA 2.1 is a Monte Carlo event generator which simulates the processes $e^+e^-\rightarrow e^+e^-\pi^0$, $e^+e^-\rightarrow e^+e^-\eta$, $e^+e^-\rightarrow e^+e^-\eta'$, and $e^+e^-\rightarrow e^+e^-\pi^+\pi^-$ at the leading order (LO) accuracy.
1 Program summary

Programming language  FORTRAN 77 with quadruple precision

Computer            PC, main frame

Operating system    Linux, Unix, MS Windows

RAM                  up to 10 Megabytes for operation of the compiled program

Number of processors used  one

Keywords            Monte Carlo generator, Event simulation, Pion transition form factor, Pair production, Two-photon process, e+ e- collision

Classification      11.2 Phase Space and Event Simulation, 11.6 Phenomenological and Empirical Models and Theories

Nature of problem   The first version of EKHARA [A,B] was developed to simulate background for the pion form factor measurement at meson factories coming from the process \( e^+e^- \rightarrow e^+e^-\pi^+\pi^- \). This channel is not yet suitable for \( \gamma\gamma \) physics studies.

The channel \( e^+e^- \rightarrow e^+e^-\pi^0 \) added in version 2.0 is being used for feasibility studies of the \( \pi^0 \rightarrow \gamma\gamma \) decay width measurement and the \( \gamma^*\gamma \rightarrow \pi^0 \) transition form factor measurement with the KLOE-2 experiment [C].

The version 2.1 is an improvement of 2.0: we added the possibility of “two-photon” production of \( \eta \) and \( \eta' \) mesons, namely, the channels \( e^+e^- \rightarrow e^+e^-\eta \) and \( e^+e^- \rightarrow e^+e^-\eta' \). We also included new improved two-photon form factors of the \( \pi^0 \), \( \eta \) and \( \eta' \) mesons [D], which are tuned to reproduce the experimental space-like form factor data.

Solution method     Events consisting of the momenta of the outgoing particles are generated by Monte Carlo methods. The generated events are distributed accordingly to the theoretical cross section. For the \( e^+e^- \rightarrow e^+e^-\pi^0 \) mode the Monte Carlo sampling developed in [E] was adopted. A detailed description of the \( e^+e^- \rightarrow e^+e^-\pi^0 \) channel implementation is given in [F] (EKHARA 2.0). The sampling for the \( e^+e^- \rightarrow e^+e^-\eta^{(*)} \) channel (EKHARA 2.1) is similar [G].

Restrictions        In order to compile the code, the FORTRAN 77 compiler should support quadruple precision numbers.

Unusual features    Calculations are carried in quadruple precision, in order to avoid numerical cancellations in \( e^+e^- \rightarrow e^+e^-\pi^+\pi^- \) mode.
Main references


[C] D. Babusci and others, On the possibility to measure the $\pi^0 \rightarrow \gamma\gamma$ decay width and the $\gamma^*\gamma \rightarrow \pi^0$ transition form factor with the KLOE-2 experiment, Eur. Phys. J. C72 (2012) 1917


[F] H. Czyż, S. Ivashyn, EKHARA Monte Carlo generator for $e^+e^- \rightarrow e^+e^-\pi^0$ and $e^+e^- \rightarrow e^+e^-\pi^+\pi^-$ processes, Comput. Phys. Commun. 182 (2011) 1338-1349

2 Introduction

2.1 Terms and conditions

All EKHARA versions are available from the authors’ web page. Version 2.0 is also available from the CPC program library. Redistribution of the source code in unchanged or changed form by third parties is prohibited.

EKHARA is distributed “as is” and the authors are not responsible for any misuse of EKHARA and for any damage, direct or indirect, caused by EKHARA. You are using EKHARA at your own risk and by using it you take all the responsibility for any consequences, including the interpretation of the numerical results.

We try our best to perform thorough tests of the generator, but some bugs may elude our control. If you find an error in EKHARA, please do not hesitate to let us know. We are always glad to receive a feedback from users.

In any publication, where the results are obtained with the use of EKHARA, you are kindly asked to cite the EKHARA papers. The references for $e^+e^- \rightarrow e^+e^-\pi^+\pi^-$ physics case are [1, 2]; the program structure was described in [3]. The reference for the first appearance of $e^+e^- \rightarrow e^+e^-\pi^0$ channel in EKHARA and the program structure is [3]. The references for the first appearance of $e^+e^- \rightarrow e^+e^-\eta$ and $e^+e^- \rightarrow e^+e^-\eta'$ channels in EKHARA are [4, 5]. The improved transition form factors included in version 2.1 are described in detail in the reference [5].

You are free to modify the source code at your own risk. In the case you publish results obtained with the use of a modified version, you should explicitly state, that EKHARA has been modified by you for your research.

EKHARA uses the unpublished double precision version of RANLUX random number generator [6, 7] written by M. Luscher. All rights for RANLUX are reserved to the legal author of RANLUX.

2.2 Basic functionality

The big picture of the program functionality is the following.

$e^+e^- \rightarrow e^+e^-\pi^+\pi^-$

- generates weighted events;
- fills the histograms;
- gives the integrated cross section within cuts.

$e^+e^- \rightarrow e^+e^-\pi^0, \eta, \eta'$

- generates and stores unweighted events;
- fills the histograms;
- gives the integrated cross section within cuts.

The program is supplemented with Gnuplot scripts for visualization of the histograms, produced by EKHARA.

2.3 Features of $e^+e^- \rightarrow e^+e^-\pi^0, \eta, \eta'$ channels

- exact formulae and exact kinematics (in contrast to Equivalent Photon Approximation),
- includes both s- and t-channel amplitudes and their interference,
- allows user-defined form factors,
- implements specific kinematic cuts,
• accounts for the peaking behaviour of the cross section, in order to have a good Monte Carlo efficiency.

3 Overview of the software structure

The directory structure of the distribution is the following. The code of the Monte Carlo generator is located in the directory `ekhara-routines`. The main source file of EKHARA is `ekhara.for`. There are other source files in the directory `ekhara-routines`, which are automatically included:

- the $e^+e^- \rightarrow e^+e^0\pi^0$, $\eta, \eta'$ mode is implemented in `routines_1pi.inc.for` and its supplementary histogramming routines are given in `routines-histograms_1pi.inc.for`;
- the $e^+e^- \rightarrow e^+e^-\pi^+\pi^-$ mode is coded in `routines_2pi.inc.for`, its supplementary histogramming routines are in `routines-histograms_2pi.inc.for` and helicity-amplitude routines are given in `routines-helicity-aux.inc.for`;
- the routines for the matrix and vector manipulations are located in `routines-math.inc`;
- in `routines-user.inc.for` several routines are collected, which can be changed by a user in order to customise the operation of EKHARA; these include the data-card reading, reporting of events, form factor formulae, filling the histograms, additional phase space cuts, etc.;
- all common blocks are included from the file `common.ekhara.inc.for`. This file contains the detailed comments on the explicit purpose of the most important common variables.

The operation of the EKHARA generator requires the following steps:

1. initialization,
2. event generation,
3. finalization.

This structure allows us to build a stand-alone generator, as well as an interface to a separate program (e.g., a detector simulation) when EKHARA is called on an event-by-event basis. In the main directory of the distribution there are examples of both uses of EKHARA. A stand-alone example is given in `ekhara-standalone.for`. An example of an EKHARA interface on an event-by-event basis is given in `ekhara-call-example.for`. The main directory of the distributed version contains a `readme.txt` file with a short description how to compile, run and test the program in the regimes, described above. It is suggested to use the `Makefile`, which is placed in the main directory. An example of the full set of input files and the plotting environment is supplied in the `Env` subdirectory. If one uses the distributed `Makefile`, the content of the `Env` subdirectory will be put into the `EXE` subdirectory together with an executable `ekhara.exe` (for details, see Section 4 and `readme.txt`). In the following we assume that `ekhara.exe` is located for execution together with the input files in the `EXE` subdirectory.

A source code of RANLUX random number generator\(^1\) written in C (`ranlxd.c`), together with its C–FORTRAN wrap (`ranlux_fort.c` for standard build and `ranlux_fort_vs.c` for build with `cl` in MS Windows and `xcl` in IBM AIX) are supplied in the directory `ranlux-routines`.

\(^1\)The unpublished double precision version of fast RANLUX [6, 7] written by M. Lüscher (F. James, private communication).
3.1 I/O scheme

All the input files of EKHARA are supposed to be located in the same directory as the main executable, `ekhara.exe`. There are the following types of the input files: random seeds, parameter input, data-cards and histogram settings. An example of the full set of input files can be found in the `Env` directory.

All the output files of EKHARA are written into `./output` subdirectory. There are the following types of the output files: logs of execution, histograms and events.

3.1.1 Input files

The main input file is called `input.dat`. It contains particle masses and constants and all global settings.

The channel-dependent parameters, which are supposed to be often changed by a user (e.g., the phase space cuts) are collected in “data-cards” `card_1pi.dat` and `card_2pi.dat`. A detailed description can be found in comments within these files.

The channel-dependent histogramming settings are given in the files `histo-settings_1pi.dat` and `histo-settings_2pi.dat`.

3.1.1.1 Global settings: input.dat

\[
\text{n\text{ges}}=(\text{real}\*16)
\]

Number of generated events

\[
\text{channel\_id}=1, 2, 3, 4
\]

Mode selection

1 two pions in the final state: \(e^+e^- \rightarrow e^+e^-\pi^+\pi^-
\)

2 one \(\pi^0\) meson in the final state: \(e^+e^- \rightarrow e^+e^-\pi^0\)

3 one \(\eta\) meson in the final state: \(e^+e^- \rightarrow e^+e^-\eta\)

4 one \(\eta'\) meson in the final state: \(e^+e^- \rightarrow e^+e^-\eta'\)

\[
\text{NeedHisto}=1, 0
\]

Histogram writing switch

1 write histograms

0 ignore histogramming

\[
\text{WriteEvents}=1, 0
\]

Events writing switch

1 write events to the file

0 do not write events to the file

\[
\text{SeedMode}=1, 2
\]

Chooses the way the seeds for the random number generator are handled. In the constant seed mode, the file `seed.dat` is used for every execution and it is never modified. In the variable seed mode, the file `seed-v.dat` is used and on successful completion of each run, a new random seed is written into `seed-v.dat`. The latter mode is convenient for a subsequent production of statistically independent samples.

1 use constant seed

2 use consequently variable seed
### 3.1.1.2 Datacard: card_1pi.dat

\[ ss = \langle \text{real} \rangle \]

Total energy in the center-of-mass frame, \( \sqrt{s} \) [GeV].

\[ sw_1pi = 1, 2, 3 \]

Types of included amplitudes, see discussion in [3].

1. s-channel
2. t-channel (“signal”)
3. s- and t- channel, with interference

\[ piggFFsw = 1, 2, 3, 4, 5, 7, 8 \] for channel \( e^+e^- \rightarrow e^+e^-\pi^0 \)

Selects the model for the two photon pion transition form factor.

1. (WZWconst) constant form factor; this is not physical and should only be used only on a specific purpose;
2, 3, 4. (“ rho pole”, “LMD” and “LMD+V”) the form factors given in [8];
5. (LMD+V new) the form factor of the lowest meson dominance model with two vector resonances fitted [9] to the BaBar data [10].
7. “1 octet” version — chiral effective theory with resonances [5];

\[ piggFFsw = 1, 3, 7, 8 \] for channel \( e^+e^- \rightarrow e^+e^-\eta(\prime) \)

Selects the model for the two photon \( \eta \) and \( \eta' \) transition form factor.

1. (WZWconst) constant form factor; this is not physical and should only be used only on a specific purpose;
7. “1 octet” version — chiral effective theory with resonances [5];

Please see the remark about the transition form factors in the page 17.

### 3.1.1.3 Datacard: card_2pi.dat

\[ ss = \langle \text{real} \rangle \]

Total energy in the center-of-mass frame, \( \sqrt{s} \) [GeV].

\[ sw_2pi = 1, 2, 3 \]

Types of included amplitudes, see discussion in [1, 2].

1. s-channel
2. s+t-channel
3. s+t-channel + 2gamma
Output: logging

The main execution log file is the output/runflow.log. It contains the main information about the operation mode and status of EKHARA, this information is also partly written into the standard output (i.e., the console). At the end of a successful execution, the total cross section is reported to output/runflow.log and also to the standard output.

A non-standard behaviour of the MC generator is reported into output/warnings.log, while the critical problems in the event generator operation are reported into the file output/errors.log.

In the case of a correct operation, the output/errors.log and output/warnings.log should remain blank. We strongly recommend to keep track on this issue and report to the authors any warnings or errors.

Output: histograms. Plotting scripts

When histogramming is allowed through settings in the input.dat, the plain text files with the histogram data are saved at the end of the generator execution.

\[ e^+e^- \rightarrow e^+e^-\pi^+\pi^- \]

The file histograms_2pi.out contains the data for \(d\sigma/dQ^2\) histogram. One may use the plotting script doplots.sh from directory histo-plotting_2pi in order to plot this histogram (an installed Gnuplot is required).

\[ e^+e^- \rightarrow e^+e^-\pi^0, \eta, \eta' \]

There is a wide set of histograms stored in the files histo<Number>.<variable>.dat, where <Number> stands for the histogram number and <variable> is the histogramming variable acronym.

One may use the plotting script do-everything.sh in the directory histo-plotting_1pi in order to plot all the histograms and collect them into a single postscript file. An installed \LaTeX{} system is required for the latter.

One may use the plotting script doplots.sh in the directory t1-t2-bars_1pi in order to plot the 3D-bar graph, which shows the distribution in two variables: \(t_1\) and \(t_2\).

As the histograms are stored as plain text files the user can use also her/his favourite plotting programs to visualize the histograms.

Output: events

The generated four-momenta of the particles are stored in the following variables, which can be accessed through common blocks:

\[
\begin{align*}
  p_1 & \quad \text{initial positron}, \\
p_2 & \quad \text{initial electron}, \\
q_1 & \quad \text{final positron}, \\
q_2 & \quad \text{final electron}, \\
q_{\text{pion}} & \quad \text{final pseudoscalar \((e^+e^- \rightarrow e^+e^-\pi^0, \eta, \eta'\) modes)}, \\
p_{11},p_{12} & \quad \text{final pseudoscalars \((e^+e^- \rightarrow e^+e^-\pi^+\pi^-\) mode)}. \\
\end{align*}
\]

In the file ekhara-call-example.for we give an example how the generated momenta can be used, when EKHARA works in the event-by-event regime. In the standalone regime we suggest to use the routine reportevent_1pi defined in the file routines-user.inc.for, which is called automatically for every accepted unweighted event \((e^+e^- \rightarrow e^+e^-\pi^0, \eta, \eta'\) modes only). In the distributed version this routine writes the events to the file output/events.out when WriteEvents flag is on. One can modify this routine in order to accommodate the way how the generated events are collected.
3.2 Selected procedures

The top-level interface to the Monte Carlo generator is provided by the routine

\texttt{EKHARA(i)}
- \(i=-1\): initialize,
- \(i=0\): generate event(s),
- \(i=1\): finalize.

Only this routine should be called from an external program, when you use EKHARA in the event-by-event regime, see example \texttt{ekhara-call-example.for}.

In order to describe briefly the “internal” structure of EKHARA, we list several important routines.

- \texttt{EKHARA\_INIT\_read}: reading the input files and datacards,
- \texttt{EKHARA\_INIT\_set}: initialization of the MC loop and mappings,
- \texttt{EKHARA\_RUN}: MC loop execution,
- \texttt{EKHARA\_FIN}: MC finalization, saving the results.

\(e^+e^- \rightarrow e^+e^-\pi^0, \eta, \eta'\) procedures

- \texttt{mc\_loop\_lpi}: Monte Carlo loop (see the flowchart in Figure 1),
- \texttt{EvalUpperBound}: evaluation of the upper bound for the Monte Carlo integrand,
- \texttt{phasespace\_lpi}: a wrap for the phase space generation routines,
- \texttt{eventselection\_lpi}: kinematic cuts,
- \texttt{m\_el\_lpi}: calculation of the matrix element for \(e^+e^- \rightarrow e^+e^-\pi^0, \eta, \eta'\),
- \texttt{tellSIGMA\_lpi}: reports the total cross section.

\(e^+e^- \rightarrow e^+e^-\pi^+\pi^-\) procedures

- \texttt{mc\_loop\_2pi}: Monte Carlo loop (see the flowchart in Figure 2),
- \texttt{eventselection\_2pi}: kinematic cuts,
- \texttt{phsp1}: phase space generation routine (branch 1),
- \texttt{phsp2}: phase space generation routine (branch 2),
- \texttt{phsp3}: phase space generation routine (branch 3),
- \texttt{matrixelmt}: calculation of the matrix element for \(e^+e^- \rightarrow e^+e^-\pi^+\pi^-\),
- \texttt{tellSIGMA\_2pi}: reports the total cross section.

For details see the comments in the source code.

4 Compilation instructions

Being distributed as a source code the program does not require installation, but a compilation and a linking is needed. EKHARA does not need any specific external libraries. In order to compile the program, a user may run any OS (UNIX, Linux, MS Windows, etc) with correctly installed

- FORTRAN 77 compiler with support of a quadruple precision,

and

- C compiler.

The default and recommended compilers are Intel Fortran and GCC. We suggest to use by default the \texttt{-O2} optimization. The program was tested mainly on the following platforms:
while Loop_Iterator > 0
  generate random numbers uniformly on [0..1]
  construct the particle momenta;
  calculate volume factor and Jacobian
  Numerically stable ?
    Yes
    consistent with cuts ?
      Yes
      calculate matrix element
      construct the "integrand" (differential cross section)
      do accept/reject (using an extra random number)
      accepted ?
        Yes
        add event to histograms, write event to file, account in Loop_Iterator
        No
        No
        more than 10 unstable cases
        Yes
        Abort
    No
  No
  Yes
  No

Figure 1: Flowchart for the $e^+e^- \rightarrow e^+e^-\pi^0$, $\eta$, $\eta'$ event generation.
while Loop_Iterator > 0
  generate random numbers uniformly on [0..1]
  construct the particle momenta;
  calculate volume factor and Jacobian
  consistent with cuts ?
    Yes
    calculate pion form factor,
    calculate momentum scalar products,
    prepare spinors
    calculate matrix element
    construct the “integrand”
    (differential cross section)
    add event to histograms
  No
  exit loop
iterate loop

Figure 2: Flowchart for $e^+e^- \rightarrow e^+e^-\pi^+\pi^-$ event generation.
<table>
<thead>
<tr>
<th>Description</th>
<th>Linux</th>
<th>Windows</th>
<th>IBM AIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default: Standalone MC generator, sample input files and histogramming</td>
<td>default</td>
<td>default-vs</td>
<td>default-ibm</td>
</tr>
<tr>
<td>routines. Everything put into EXE directory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compile everything: default, ranlux-testing program and seed-production</td>
<td>all</td>
<td>all-vs</td>
<td>all-ibm</td>
</tr>
<tr>
<td>programs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testrun: Compile everything and execute the testrun scripts</td>
<td>test</td>
<td>test-vs</td>
<td>test-ibm</td>
</tr>
<tr>
<td>Remove the redundant and temporary files</td>
<td>clean</td>
<td>clean-vs</td>
<td>clean-ibm</td>
</tr>
</tbody>
</table>

Table 7: The main Makefile targets

- Linux, 64-bit (Ubuntu 11.04)
  GNU C Compiler (gcc) 4.5.2
  Intel(R) FORTRAN Compiler (ifort) 12.0.4
- Linux, 32-bit (Gentoo)
  GNU C Compiler (gcc) 4.1.2
  Intel(R) FORTRAN Compiler (ifort) 10.0
- MS Windows (XP SP3)
  MS Visual Studio 2008 (nmake, cl)
  Intel(R) FORTRAN Compiler (ifort) 11.

The program distribution contains the Makefile, with targets for Linux, IBM AIX and Windows environments. The Makefile is annotated, in order to help a user to tune it up for her/his own requirements. The main Makefile targets are listed in Table 7. In the 64-bit environment on Linux you should use the make targets default64, all64 and test64 instead.

For example, in Linux, a simple way to compile a program is to issue

```
$ make default
```

or (64bit):

```
$ make default64
```

being in the directory where the Makefile is located. This will produce ekhara.exe (main program executable) and copy it into the sub-directory EXE, together with the contents of Env sub-directory. The latter contains the set of sample input files and histogram plotting scripts. We provide a full set of necessary input files in the distribution package. It is recommended to execute ekhara.exe in the directory EXE, where it is placed by default. Every time one does make default, the input files in EXE are replaced with the sample ones from Env.

In order to produce only an object file with the EKHARA MC generator, one can use, for example

```
$ ifort -c ekhara-routines/ekhara.for -o ./ekhara.o
```
EKHARA needs a random seed for operation. Different random seeds can be obtained by using a Makefile target seed_prod-ifort. It produces an executable program seed_prod.exe, which generates a set of random seeds.

5 Test run description

It is recommended to test the random number generator on a given machine, before using EKHARA. It is also important to check whether EKHARA can function properly on a given operational system and that there are no critical bugs due to the compiler. We provide a testrun package for these purposes.

It is suggested to use the Makefile targets test, test-vs or test-ibm depending on your environment (Linux, Windows and IBM AIX, correspondingly), e.g:

```bash
$ make test
```

or (64bit):

```bash
$ make test64
```

This will automatically prepare and execute the following two test steps.

The first step of the testrun is the random number generator control. The source file test1xf.for contains the ranlux test routines.

The second step is the verification if the user-compiled EKHARA can reproduce the set of results, created by a well-tested copy of EKHARA in various modes. The testrun environment contains directories test and test-vs with pre-calculated data for a comparison, along with the random seed and input files for each mode. The script test.sh is responsible for the execution of a user-compiled ekhara.exe in all the control modes and for the comparison of the output.

Please read carefully the output of the testrun execution in your console and be sure there are no warnings and/or error messages.

6 Customization of the source code by a user

We leave for a user an option to customise the generator to her/his needs by editing the source code file ekhara-routines/routines-user.inc.for. Notice, we always use explicit declaration of identifiers and the implicit none statement is written down in each routine.

In the file ekhara-routines/routines-user.inc.for one can change

- the data-card reading (routines read_card_1pi and read_card_2pi),
- the form-factor formula (routine piggFF),
- the events reporting (routine reportevent_1pi),
- histogramming (routines histo_event_1pi and histo_event_2pi),
- additional kinematic cuts (routines ExtraCuts_1pi and ExtraCuts_2pi).

7 Validation of the generator

We simulate the cross sections \( d\sigma/dQ^2 \) for the process \( e^+e^- \to e^+e^-\pi^0 \), \( \eta, \eta' \) using the “2-octet” model of Ref. [5] and compare it with existing “single-tag” data from the CELLO [12], CLEO [13] and BaBar [14, 15] experiments. The experimental \( d\sigma/dQ^2 \) is given within the cuts,
Figure 3: The cross section $d\sigma/dQ^2$ for the process $e^+e^- \rightarrow e^+e^-\pi^0$ compared to BaBar [14] (left) and CLEO [13] (right).

and, therefore, the simulated $d\sigma/dQ^2$ is computed with similar event selection\(^2\). As expected, a good agreement between the generator predictions and the data [15, 13] is observed, see Figs. 3, 4. The tension between the $\pi^0$ BaBar data [14] and the results of $d\sigma/dQ^2$ simulations is similar to the one observed for the form factor, see discussion in [5]. From the verification illustrated in Figs. 3, 4 we conclude that the matrix element calculation is well under control.

The procedure of the phase space generation was validated by means of the high statistics phase space volume calculation in EKHARA and comparison of the result with that from the independent dedicated numerical calculation. The volume was also compared to that obtained by GALUGA generator [16]. We have also verified that our numerical results for the three-body phase space volume in the limit of massless $\pi^0$ reproduce well those of the analytic expression.

The $e^+e^- \rightarrow e^+e^-\pi^+\pi^-$ mode is validated by means of the reproduction of the results from the previous version of EKHARA. In references [1, 2, 17] the tests of this part are presented in detail.

8 Frequent questions

We strongly advice to read the user manual of your compiler in order to setup and use it properly. Below are some hints from our experience, provided as examples.

Intel FORTRAN in Linux Please be sure, before executing ekhara.exe, that the Intel FORTRAN runtime environment is set properly: e.g. put into ~/.bashrc or ~/.bash_profile the following lines:

```bash
source /export/pc/compiler/intel/intel_fc_80/bin/ifortvars.sh
NLSPATH=/export/pc/compiler/intel/intel_fc_80/lib/ifcore_msg.cat
export NLSPATH
```

or, depending on you version of FORTRAN,

```bash
source /opt/intel/Compiler/11.0/069/bin/ifortvars.sh ia32
NLSPATH=/opt/intel/Compiler/11.0/069/lib/ifcore_msg.cat
```

\(^2\)In a single-tag experiment, the “tagged” lepton fixes the value of $Q^2 = -t_1$ and the 4-momentum squared of the “untagged” lepton $t_2 = -q_2^2$ is kinematically restricted nearby zero. For example, in the BaBar experiment, the actual thresholds for $q_2^2$ are 0.18 GeV\(^2\) for pions [14] and 0.38 GeV\(^2\) for $\eta$ and $\eta'$ [15] due to the imposed event selection.
Figure 4: The cross section $d\sigma/dQ^2$ for the process $e^+e^- \rightarrow e^+e^-\eta$ (left) and $e^+e^- \rightarrow e^+e^-\eta'$ (right) compared to CLEO [13] and BaBar [15] data.

export NLSPATH

Use the correct paths, specific to your environment.

In order to use Intel debugger idb (GUI) make sure that Java Runtime Environment 5.0 is installed (JRE 1.5) and accessible from PATH:

```bash
export PATH=/usr/lib/jvm/java-1.5.0-sun-1.5.0.16/jre/bin:$PATH
```

or just run idbc from command line.

Please, remember to compile with -g option, if you want to enable debugging.

Intel Fortran under MS Windows You may want to use the supplemented batch file make_vs.bat in order to pre-load the environment settings required for compilers cl and ifort. Example:

```bash
$ make_vs.bat all-vs
```

You are advised to accommodate make_vs.bat to your version of ifort and Visual Studio.

Problem with an option for C compiler The following problem is known for gcc version 3.4.6 in Red Hat Linux.

```bash
... 
gcc -c -m32 ./ranlux-routines/ranlxd.c -o ranlxd.o 
cc1: error: invalid option ‘32’ 
... 
```

In order to fix this problem, one needs to modify the Makefile, namely to remove the -m32 option.

Non-compatible FORTRAN compilers The following problem is known for FORTRAN compilers, which do not support the F77 convention on include files.

```bash
... 
"common.ekhara.inc.for": No such file or directory 
... 
```
The following problem is known for FORTRAN compilers, which do not support the quadruple precision in F77 code.

```
Error: Old-style type declaration REAL*16 not supported at (1)
```

or

```
common.inc:13:
  real*16 pi
  1 2
Invalid kind at (2) for type at (1) -- unsupported or not permitted
```

At the moment, EKHARA can not be compiled with the following compilers: gfortran, g77, f95. We recommend to use ifort. Alternatively, you may consider IBM AIX xlf or a recent version of nagfor or sunf95, which also support the F77 code with quadruple precision. However, in order to do so, you will need to carefully modify the Makefile. We have not tested these options, but we are aware of successful cases of the xlf and sunf95 use with EKHARA.

**Problem with testrun**  The following problem is known for cp version 5.2.1 in Red Hat Linux.

```
$ make test
... 
  cp: invalid option -- t
Try ‘cp --help’ for more information.
  cp: invalid option -- T
Try ‘cp --help’ for more information.
... 
```

Some versions of cp does not support the -t and -T option. A possible workaround is implemented in the Windows-oriented testrun script. Try to substitute the file ./test/test.sh by the alternative ./test-vs/test.sh.

**32-bit and 64-bit architecture**  It is possible to use EKHARA both in 64-bit or 32-bit environment. If you see an error similar to

```
ld: i386 architecture of input file ‘ranlux_fort.o’ is incompatible with i386:x86-64 output
ld: i386 architecture of input file ‘ranlxd.o’ is incompatible with i386:x86-64 output
... 
```

please make sure that you are the C and FORTRAN source code is compiled for the same architecture. Use make default, make test for 32-bit and make default64, make test64 for 64-bit. If this does not fix the problem, please change the compilation flags in Makefile accordingly to the specification of your compilers (e.g., -m32 and -m64 flags for gcc).

**Reference frame, polar angles**  EKHARA by default produces the events in the center of mass frame of the initial electron and positron. The z axis direction is defined as the direction of the initial positron. The polar angles are defined with respect to z axis.
Generator efficiency for $e^+e^- \rightarrow e^+e^-\pi^0$, $\eta$, $\eta'$: The channels $e^+e^- \rightarrow e^+e^-\pi^0$, $\eta$, $\eta'$ are optimized for the $\gamma\gamma$ physics studies. However, if a user applies some specific event selection, the generator efficiency can get reduced. If it is the case, we would like to bring your attention to the possibility of restricting kinematics range for the generated events at the level of generator (i.e., before actual application of the event selection), which usually helps to improve the efficiency. This is controlled by parameters \texttt{CUT\_t1max}, \texttt{CUT\_t1min}, \texttt{CUT\_t2max}, \texttt{CUT\_t2min} (cuts on the photons’ virtualities) in the file \texttt{input.dat} and \texttt{CUT\_th1min}, \texttt{CUT\_th1max}, \texttt{CUT\_th2min}, \texttt{CUT\_th2max} (polar angle cuts on the final lepton directions) in the file \texttt{card\_1pi.dat}. In this way the generator efficiency can be considerably improved, but the user should make sure that the actual event selection is less inclusive than the one imposed by the cut parameters in the input files.

Convention on the transition form factor normalization: The transition form factors in EKHARA are normalized as $F(0,0)=1$. In this convention, for example, the “WZW” form factor choice corresponds to a constant $F(t_1,t_2)=1$ for all values of $t_1$ and $t_2$. The common way to write down the amplitude is given by equation (3) in the paper \cite{5} and the coupling strength of the pseudoscalar to two real photons is not contained in the form factor but given by an independent parameters $f_P = f_\pi$, $f_\eta$, $f_\eta'$. The numerical value of $f_P$ is controlled by the two-photon width of the particle $\Gamma_{P \rightarrow \gamma\gamma}$:

$$f_P^2 = \frac{\alpha^2 m_P^3}{96 \pi^3 \Gamma_{P \rightarrow \gamma\gamma}},$$

where $\alpha$ is the electromagnetic fine structure constant and $m_P$ is the mass of the particle. The numerical value of $f_P$ in EKHARA is defined in the subroutine \texttt{init\_1pi}, \texttt{init\_leta} and \texttt{init\_letaP}.

Please notice that in \cite{5} the normalization $F(0,0)=1$ was not used. This does not affect the normalization used in the program.

Transition form factor switch \texttt{piggFFsw} and \texttt{channel\_id}: It is important to stress that the difference between the generator modes $e^+e^- \rightarrow e^+e^-\pi^0$, $\eta$ and $\eta'$ is not only in the mass of the pseudoscalar meson. Since the flavor content of the $\pi^0$, $\eta$ and $\eta'$ is different, the different form factor formulae have to be used in each case for a realistic simulation. While the $e^+e^- \rightarrow e^+e^-\pi^0$, $\eta$ or $\eta'$ mode is selected by the parameter \texttt{channel\_id}, the form factor model is selected by the parameter \texttt{piggFFsw}. For example, the formula used with \texttt{channel\_id=3} $\oplus$ \texttt{piggFFsw=8} is different from the one used with \texttt{channel\_id=4} $\oplus$ \texttt{piggFFsw=8} and \texttt{channel\_id=2} $\oplus$ \texttt{piggFFsw=8}.

We recommend the form factor modeling with \texttt{piggFFsw=8}, for which the most realistic simulation is anticipated. For details see \cite{5}.

9 Acknowledgements

This work was partially supported by by Sonderforschungsbereich SFB1044 of the Deutsche Forschungsgemeinschaft, MRTN-CT-2006-035482 “FLAVIAnet” under the Sixth Framework Program of EU, Polish Ministry of Science and High Education from budget for science for years 2010-2013: grant number N N202 102638 and National Academy of Science of Ukraine under contract 50/53–2012. We acknowledge the partial support of the European Community-Research Infrastructure Integrating Activity “Study of Strongly Interacting Matter” (acronym HadronPhysics3, Grant Agreement n. 283286) under the Seventh Framework Programme of EU.

Additional bibliography