POLARES – an event generator for polarized electron-proton scattering

Razvan-Daniel Bucoveanu[1] and Hubert Spiesberger[2]

[1] PRISMA$^+$ Cluster of Excellence, Institut für Kernphysik,
Johannes-Gutenberg-Universität, Staudinger Weg 7, D-55099 Mainz, Germany,

[2] PRISMA$^+$ Cluster of Excellence, Institut für Physik,
Johannes-Gutenberg-Universität, Staudinger Weg 7, D-55099 Mainz, Germany,

June 10, 2021

Abstract

`POLARES` is a `C++` program for the simulation of elastic scattering of polarized leptons off nuclei. It is optimized for applications at low energy electron proton scattering, covering the kinematic conditions of the P2 and QWeak experiments, but can also be used for muon scattering and for scattering off other nuclei. It allows one to calculate cross sections including radiative corrections at first and second order in the fine-structure constant, i.e. including radiation of one or two photons. If used as an event simulation program, it generates weighted events with fully reconstructed kinematics. We describe in detail how the program can be used.

1 Introduction

Low energy precision experiments have become an important tool in the search for new physics beyond the Standard Model (SM). These experiments can exclude new physics at mass scales extending well into the TeV range and are complementary to searches at the Large Hadron Collider (LHC). In particular lepton nucleon scattering at low energies is an important example. With polarized elastic electron proton scattering, one can determine the weak charge of the proton, which is related to the weak mixing angle in the SM. Deviations from the SM prediction for the weak mixing angle can provide important tests of models beyond the SM. Results of the QWeak experiment at the Jefferson Laboratory have been already published [1] and the Mainz P2 experiment at the MESA accelerator being under construction is expected to see first events in mid 2023 [2]. Moreover, there has been a continued interest in the determination the proton's form factors, notably the electric and magnetic ones, G_E and G_M, contributing to the unpolarized scattering cross section. Their precise knowledge is a vital ingredient in a determination of the proton radius from electron proton scattering. With polarized scattering also axial and strangeness form factors contribute. Accurate data for them are needed for a complete understanding of how matter is formed from quarks and gluons.
In order to match the precision of present day's lepton nucleon scattering experiments it is important to include the full set of radiative corrections at first and second order in perturbation theory. Higher-order corrections, in particular QED radiative effects, can often not be taken from the classical work of Mo and Tsai [3] (see also [4, 5]) without carefully revisiting the underlying assumptions and improving approximations which had been acceptable in previous experiments. In addition, corrections due to hard photon radiation depend on details of the detector setup. Their calculation therefore requires a full Monte Carlo event simulation.
For this purpose we have developed a modern and versatile new `C++` program which we call `POLARES`. It is a program for elastic lepton-nucleon scattering with longitudinally polarized electron beams. It includes QED radiative corrections at the one and two-loop level for unpolarized, as well as for polarized incident leptons. It can be used as an integrator to calculate cross sections and asymmetries for given kinematic conditions. It can also be used as an event generator. The design of the program `POLARES` was developed in such a way that it can be easily combined with the detector simulation software of the experiment. It contains an option to generate events with varying energy of the incoming lepton. In addition, we have included the possibility to switch between electron and muon as the incident particle and between proton and carbon-12 as the target particle. Note, however, that in the present version the two-loop corrections are based on an calculation which assumes the lepton mass to be negligibly small compared with the momentum transfer. This approximation may not be valid for muon scattering at low energies (see [6] for a study of this approximation).

The theory background is described in the next section and in more details in Ref. [6]. This document may be consulted for more details, e.g. concerning tests which have been performed to check the performance of the program. At first order in perturbation theory there are other programs available that include radiative corrections (see for example [7]) and partially second order corrections (see, e.g., [8]). Our program agrees very well with these other calculations.
`POLARES` is build as a library with `C++` code. For the Monte Carlo integration it uses the `Cuba` library [9]. The `Cuba` library is contained in the `POLARES` package and doesn't need to be separately installed. A reference manual *refman* created by `doxygen`, that includes a complete list of all classes, functions and variables can also be found in the distribution, both in *pdf* and *html* format.

In the next section we give a short overview of the theoretical background of the required calculations. The focus of this article is, however, on technical aspects. It is written in the style of a manual, for users who want to use our program. Section 3 contains information about the installation of the program. Subsequent Sections 4 to 8 contain the detailed description of functions, input and output, and the list of files contained in the `POLARES` package. We end with presenting a few examples in Section 9.
The present write-up refers to version V1.1 of `POLARES`. The program can be obtained from `https://github.com/razvanbucoveanu/POLARES.git`. Please consult this URL for possible corrections and updates which may appear in the future.

2 Physics background

We denote the 4-momenta of the incoming and scattered lepton (nucleon) by l^μ and l'^μ (p^μ and p'^μ). According to the applications considered in this work we choose a coordinate frame where the target nucleon is at rest and the z axis is directed along the momentum of the incident lepton. Symbols for energies and angles of the particles involved in the scattering process can be found in Fig. 1.
At leading order, lepton nucleon scattering is described by the exchange of a virtual photon or a virtual Z^0 boson. For a spin-1/2 nucleon, like the proton, the matrix elements for the electromagnetic and weak neutral currents, $g_{e,Z}\,\bar{u}(p')\Gamma^\mu_{\gamma,Z}u(p)$, where $g_e = e$ is the electromagnetic charge and $g_Z = \sqrt{G_F M_Z^2/2\sqrt{2}}$, with G_F the Fermi constant and M_Z the mass of the Z_0 boson, can be decomposed into Dirac and Pauli form factors, $F_1^{\gamma,Z}(Q^2)$ and $F_2^{\gamma,Z}(Q^2)$, plus an additional axial form factor $G_A^Z(Q^2)$ for Z^0 exchange. In the case of γ exchange the proton vertex is given explicitly by

$$\Gamma^\mu_\gamma = \gamma^\mu F_1^\gamma + \frac{i\sigma^{\mu\nu}q_\nu}{2M}F_2^\gamma \tag{1}$$

where M is the proton mass, while in the case of Z^0 exchange it is determined by

$$\Gamma^\mu_Z = \gamma^\mu F_1^Z + \frac{i\sigma^{\mu\nu}q_\nu}{2M}F_2^Z + \gamma^\mu\gamma_5 G_A^Z. \tag{2}$$

The vector form factors $F_1^{\gamma,Z}$ and $F_2^{\gamma,Z}$ can be related to the Sachs form factors, defined as $G_E^{\gamma,Z} = F_1^{\gamma,Z} - \tau F_2^{\gamma,Z}$ and $G_M^{\gamma,Z} = F_1^{\gamma,Z} + F_2^{\gamma,Z}$ with $\tau = Q^2/(4M^2)$.
Assuming isospin symmetry one can write the Z^0 Sachs form factors in terms of the electromagnetic Sachs form factors of the proton and neutron and a contribution from the

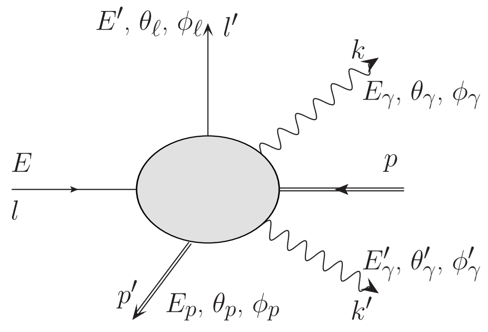
$E', \theta_\ell, \phi_\ell$
l'
k
$E_\gamma, \theta_\gamma, \phi_\gamma$
E
l
p
$E'_\gamma, \theta'_\gamma, \phi'_\gamma$
p'
E_p, θ_p, ϕ_p
k'

Figure 1: Schematic definition of kinematic variables for lepton nucleon scattering. Angles are defined with respect to the direction of the incoming lepton with 4-momentum l^μ and in our application the proton is at rest, i.e. with 4-momentum $p^\mu = (M, 0, 0, 0)$.

strange quarks ([10]) as

$$G^Z_{E,\,M} = Q^p_W G^{\gamma,\,p}_{E,\,M} - G^{\gamma,\,n}_{E,\,M} - G^s_{E,\,M}, \tag{3}$$

where Q^p_W is the weak charge of the proton.
For a spin-0 nucleus, like ^{12}C, the vertex for γ and Z^0 exchange is described by $(p+p')^\mu F^{\gamma,\,Z}_C(Q^2)$, where $F^{\gamma,\,Z}_C$ is the form factor for γ and Z^0 exchange respectively. In the present version we simply assume that the carbon form factors for γ and Z^0 exchange are equal up to the normalization to the total electric and weak charge (see [11] for how this assumption is motivated).
Higher-order corrections are described by loop diagrams with virtual photons or by bremsstrahlung diagrams with real photons (for more details about the treatment of higher order corrections to the unpolarized cross section see [6]). Both parts contain infrared (IR) divergent parts which exactly cancel when combined. In our approach we use the phase-space slicing method to separate soft-photon radiation from hard-photon contributions [3]. The separation is implemented by using a cut-off Δ for the energy of a radiated photon. Δ is chosen small, below the detection threshold for the observation of a photon in the detector. The soft-photon part (designated by a lower index "$1s\gamma$" in the following) combined with loop diagrams is called non-radiative. First-order corrections, at order $\mathcal{O}(\alpha)$ relative to the Born cross section, are written as

$$\sigma^{(1)} = \sigma^{(1)}_{\text{non-rad}} + \sigma^{(1)}_{1h\gamma}\,, \tag{4}$$

and

$$\sigma^{(1)}_{\text{non-rad}} = \sigma^{(1)}_{1-\text{loop}} + \sigma^{(1)}_{1s\gamma}\,. \tag{5}$$

At second relative order, one has to include contributions with both one or two radiated photons and one has to distinguish the cases where only one or both photons are either soft or hard. The second-order contribution to the cross section, $\sigma^{(2)}$, is therefore split into three parts:

$$\sigma^{(2)} = \sigma^{(2)}_{\text{non-rad}} + \sigma^{(2)}_{1h\gamma} + \sigma^{(2)}_{2h\gamma}, \tag{6}$$

where

$$\sigma^{(2)}_{\text{non-rad}} = \sigma^{(2)}_{2-\text{loop}} + \sigma^{(2)}_{1-\text{loop}+1s\gamma} + \sigma^{(2)}_{2s\gamma}\,, \qquad \sigma^{(2)}_{1h\gamma} = \sigma^{(2)}_{1-\text{loop}+1h\gamma} + \sigma^{(2)}_{1s\gamma+1h\gamma}\,. \tag{7}$$

The non-radiative parts are rendered IR-finite by including loop diagrams: $\sigma^{(2)}_{\text{non-rad}}$ contains two-loop contributions and mixed soft-photon + one-loop parts, while $\sigma^{(2)}_{1h\gamma}$ contains one-loop corrections to the radiative process with one hard photon.
The correction factors can be defined relative to the differential Born-level cross section $\mathrm{d}\sigma^{(0)}$ as

$$\sigma^{(1)}_{\text{non-rad}} = \int \mathrm{d}\sigma^{(0)} \left[\delta^{(1)}_{1-\text{loop}} + \delta^{(1)}_{1s\gamma}(\Delta)\right], \tag{8}$$

$$\sigma^{(2)}_{\text{non-rad}} = \int \mathrm{d}\sigma^{(0)} \left[\delta^{(2)}_{2-\text{loop}} + \delta^{(2)}_{1-\text{loop}+1s\gamma}(\Delta) + \delta^{(2)}_{2s\gamma}(\Delta)\right], \tag{9}$$

where each δ is labeled with indices as described above for the total cross sections. We also show explicitly the dependence of the soft-photon parts on the IR cut-off Δ. The soft-photon part can be calculated analytically, integrating up to the cut-off Δ, by using a soft-photon approximation as described in [6]. Contributions with a hard photon, i.e. with energy above the cut-off Δ, are infrared finite and the phase space integration can be performed numerically. For one hard photon at tree level, we can write

$$\sigma^{(1)}_{1h\gamma} = \int_{E_\gamma > \Delta} \mathrm{d}^4\sigma^{(1)}_{1\gamma}\,, \tag{10}$$

while at second order we define relative correction factors for the one-loop and soft photon contributions by writing

$$\sigma^{(2)}_{1h\gamma} = \int_{E_\gamma > \Delta} \mathrm{d}^4\sigma^{(1)}_{1\gamma} \left[\delta^{(1)}_{1-\mathrm{loop}+1h\gamma} + \delta^{(1)}_{1s\gamma+1h\gamma}(\Delta)\right] , \tag{11}$$

where $\mathrm{d}^4\sigma^{(1)}_{1\gamma}$ is the differential cross section for one radiated hard photon at the tree-level. Finally, the cross section for two hard photons is given by

$$\sigma^{(2)}_{2h\gamma} = \int_{E_\gamma,\, E'_\gamma > \Delta} \mathrm{d}^7\sigma^{(2)}_{2\gamma} . \tag{12}$$

It is free of any infra-red singularities and can be calculated numerically. We emphasize that the cut-off parameter Δ is introduced only for a technical reason: it allows us to separate the IR singularities. Only separate parts contributing to the cross section carry a Δ-dependence as shown in the formulas given above. The sum of non-radiative and hard-photon contributions has to be independent of Δ. Our choice to use the phase-space slicing technique is dictated by our goal to develop a full Monte Carlo event simulation program where individual non-radiative and one- or two-photon radiative events can be generated. Independence of the IR cut-off parameter has been tested carefully and we found that Δ can be varied over a large range of values without changing the final total result (see [6]).

Due to parity violation originating from weak interactions there is a polarization-dependent contribution to the cross section proportional to the degree of polarization of the lepton beam, P. We consider longitudinal polarization and write $\sigma = \sigma_{\mathrm{unpol}} + P\sigma_{\mathrm{pol}}$. The polarization-dependent part of the cross section is given by the difference between cross sections for electrons with positive and negative helicities, $\sigma_{\mathrm{pol}} = (\sigma_+ - \sigma_-)/2$, while the unpolarized part is defined as $\sigma_{\mathrm{unpol}} = (\sigma_+ + \sigma_-)/2$.
At small energies, σ_{pol} is very small and can often safely be neglected in the cross section calculation, e.g. for the P2 and QWeak experiments. However, the parity-violating asymmetry

$$A_{\mathrm{PV}} = P\frac{\sigma_+ - \sigma_-}{\sigma_+ + \sigma_-} = P\frac{\sigma_{\mathrm{pol}}}{\sigma_{\mathrm{unpol}}} \tag{13}$$

is an important observable accessible in these experiments. `POLARES` provides options to include σ_{pol} in the calculation of cross sections and in event simulation.
The focus of our newly developped event generator was put on a precise implementation of QED corrections of first and second order. For the time being, we do not include the full set of purely weak higher-order corrections, described by Feynman diagrams with heavy weak gauge bosons. Their inclusion will be relatively trivial since they introduce only small corrections of the non-radiative cross section and do not affect the kinematics of the events. Their effect on the parity-violating asymmetry, however, may be large. The value of A_{PV} obtained from `POLARES` should be corrected as described for example in Ref. [12]. Note that `POLARES` assumes an input value for the weak mixing angle defined at the Z-boson pole in the $\overline{\mathrm{MS}}$ scheme. We calculate an effective weak mixing angle appropriate for the observable of interest at small Q^2 by using

$$\sin^2\theta_W(Q^2) = \kappa(Q^2)\sin^2\theta_W(M_Z^2). \tag{14}$$

The calculation of $\kappa(Q^2)$ was taken from [13] via F. Jegerlehner's program `alphaQED` (see [?]).

Radiation of photons from the proton, combined with corresponding loop diagrams at the proton vertex, are known to be suppressed due to the large proton mass. On the other side, bremsstrahlung from the proton enters through its interference with radiation from the lepton which has to be combined with two-photon exchange diagrams to obtain an infrared-finite result. These correction terms have been discussed in the literature as a possible source to resolve the observed disagreement in the separation of the electric and magnetic proton form factors from different measurements. Their calculation requires model assumptions to describe the off-shell proton vertex, also including intermediate states other than the proton. We follow the prescription of Ref. [7] (see also [8]) and include in our program an option to take into account these corrections at first order and assuming that the proton form factors can be evaluated as for on-shell protons. This approximation should work well for soft-photon radiation.

We emphasize that the matrix elements implemented in `POLARES` are always obtained from explicit simple, though lengthy, analytic expressions. All required scalar integrals are available as a function of kinematic invariants [16, 17]. Therefore the evaluation of loop correction factors is fast and not limiting the statistics of the Monte Carlo simulation.

3 Installation

The prerequisites for installing the `POLARES` library are a GNU C++ compiler that supports the C++11 standard (`g++` version 1.6 or higher) and the `GSL`[1] library. The present version of `POLARES` was written and tested with `GSL` version 2.1 on Linux systems. The `POLARES` distribution comes in a compressed tar archive *POLARES-x.y.tar.gz*, where *x.y* is the version number. Execute the usual sequence of commands to unpack and install the library:

```
gunzip -c POLARES-x.y.tar.gz | tar xvf -
cd POLARES-x.y
./configure
make
make install
make examples
make clean
```

This list of commands will install the library in the default path (*/usr/local*) which requires root permission. The `prefix`-option can be used to choose a different path:

```
./configure --prefix=/user/defined/path
```

In case the `GSL` library is not installed in the default location (*/usr/local/bin*) it is possible to specify a different path by:

```
./configure GSL_CONFIG_PATH=/path/to/folder/containing/gsl-config
```

or set the environment variable

```
export GSL_CONFIG_PATH=/path/to/folder/containing/gsl-config
```

[1] http://www.gnu.org/software/gsl/

If the POLARES library is not installed in the default path, one needs to set the environment variable

```
export LD_LIBRARY_PATH=/user/defined/path/lib
```

so that the linker can find the shared library of the POLARES package. The command `make examples` compiles all the example programs that are found in sub-folder *examples*. After running this command, if the compilation was successful, the user should be able to run any of the programs that were build in this folder. This command can be omitted if the compilation of the example programs is not required.

4 General description

After installation the user can include *POLARES.h* into his or her own program and use the **class** PES, which is part of the **namespace** POLARES and contains all the necessary functions. An example of how this class can be used can be found in the file *examples/main_example.cpp*. More examples with the functions from this class are found in the sub-folder *examples/*. In order to use the program, the user has to be careful specifying the paths to the source code and to the POLARES library. A sample makefile, called *Makefile_example*, can be found in the folder *examples*. Additionally, for the program to run, the working directory has to contain the folder named *share* which keeps additional input files, for example for the calculation of the hadronic part of the vacuum polarisation (see below).
The user can provide input to the program in a file, for which the default name is *POLARES.in*. The program looks for this file in the working directory. The name of the input file can be changed from the input class. The program will create also an output file with the same name as the input file. A sample version of the input file comes with the distribution and contains all the possible input combinations.
In many cases, a complete specification of input data is not needed and default values defined in the library can be used. For this case, a basic input file, called *POLARES_basic.in*, is also provided and contains only the important input that a user requires to run the program. In addition to using a file to specify input options, one can also use the **class** Input in the main program, as described in the next section. However, input from the input file will overwrite input from the input class.
For complete functionality the data files *vp_Ignatov.dat*, *vp_Jeger.dat* and *vp_KNT18.dat* are also required to run the program. They contain tables needed for the calculation of the hadronic contribution to the vacuum polarization. The files are shipped with the distribution and can be found in the directory */user/defined/path/share/*.
Unless specified, all the energies and momenta are given in natural units of 1 GeV in the laboratory frame (see Sec. 2 for the definition of this reference frame). For convenience, the angles are given in degrees in the input, but in radians in the output.

5 How to use the library: description of the main functions

After installation the user can include *POLARES.h* into his or her own program and use the **class** PES, which is part of the **namespace** POLARES and contains all the necessary functions. An example of how this class can be used can be found in the file *examples/main_example.cpp*. More examples with the functions from this class are found in the sub-folder *examples*. The user has to provide input to the program in an input file, for which the default name is *POLARES.in*. The name of the input file can be changed from the input class (see below). A sample version of the input file, called *POLARES.in*, comes also with the distribution and contains all the possible input combinations. A basic input file, called *POLARES_basic.in*, is also provided and contains only the important input that a user requires to run the program. In addition to using a file to specify input options, the user can use the **class** `Input` in the main program, as described below. One should keep in mind that one may have to set the environment variable

```
export LD_LIBRARY_PATH=/user/defined/path/lib
```

so that the linker can find the shared library of the `POLARES` package.
The program expects to find the input file in the working directory from where the program is called. The data files `vp_Ignatov.dat`, `vp_Jeger.dat` and `vp_knt18.dat` are also required to run the program and contain tables needed for the calculation of the hadronic vacuum polarization. The files are shipped with the distribution and can be found in in the folder called *share*.
The POLARES **class** `PES` can be constructed explicitly by using

```
PES();
```

If values for parameters are defined in the input file they will overwrite the ones from **class** `Input`. A convenient way to start the calculations is by using the code

```
Input input;
// ...
// define input values as described below
// ...
PES my_pes_class;
my_pes_class.set_input(input);
my_pes_class.initialization();
```

and using functions of **class** `PES` to start event generation and analyze results. The most important public functions that can be found in **class** `PES` are:

- **void** `set_input(`**const** `Input& input);`

  This function is used to transfer values for input parameters defined in the main program to the **class** `PES`.

- **int** `initialization();`

  This function can be used to calculate total cross sections or asymmetries integrated over the phase space defined in the input. It is also needed to generate grids which are required for event generation. The output of this function is stored in two instances of the **class** `Output` (see below for a detailed description of the output):

  ```
  Output output;
  ```

  contains the actual results for total and partial cross sections, and

  ```
  Output errors;
  ```

  contains estimates of the numerical uncertainties of the results.

- **int** `sigma_diff_Omega_l(`**const double** `thl_deg);`

  This function calculates the differential cross section $\mathrm{d}\sigma/\mathrm{d}\Omega = \mathrm{d}\sigma/(2\pi \sin\theta \mathrm{d}\theta)$ and the corresponding asymmetries for a given scattering angle `thl_deg` (input value in degrees). The output is stored again in the objects `output` and `errors`. At next-to-leading order, this function performs an integration over the phase space for bremsstrahlung photons.

- **int** `shiftQ2(`**const double** `thl_deg);`

  This function calculates the average shift in Q^2 due to hard photon bremsstrahlung for a given scattering angle `thl_deg` (in degrees). The output is stored as in the case of the initialization in the objects `output` and `errors`.

- **int** `events();`

  This function generates events with a corresponding weight, which is stored in the final state class (see below). The function `events()` can be used after the grid initialization was completed successfully. Each call of this function generates one event and the user is free to choose the number of events to be generated. For each event the output is given in the object `FS` of the **class** `Final_State` (see below for a detailed description of its members).

- **bool** `change_energy_initialization(`**const double** `E);`

  With this function the user can change the energy of the incoming lepton beam after a first initialization. The input value for E is expected in units of GeV.

- **bool** `change_energy_events(`**const double** `E);`

  This function is useful in case the user wants to generate events for a range of energies. A sample program is provided in *examples/multiple_random_E_test.cpp.*

  `change_energy_events` returns **true** if the given energy is valid and the grid initialization was successful, **false** otherwise.

- **bool** `set_child_process(`**const int** `child_process);`

  This function must be used in case the user wants to generate events on multiple cores for creating different seeds for each child process. This version allows a maximum number of 100 child processes.

All required input can be defined in the input file. A sample version with the name *POLARES.in* is contained in the distributed package. Each input item consists of a key word (possibly including spaces) and a value, separated by an equal sign, `=`. The order of input items in this file is arbitrary. Lines starting with `#` are comments. Passing the name of the input file to the program is done via **class** `Input` and giving this object as an argument to the function `set_input`. The variable in which the user can define the name of the input file is `string input_file`. For example, if the user wants to use the file *POLARES.in* as input the following code has to be used

```
Input input;
input.set_input("POLARES");
PES pes;
pes.set_input(input);
```

All the input can be inserted directly in **class** `Input`, which contains also all the default values, without the need of an input file. However, by providing the required values in the input file, the input from **class** `Input` is overwritten. Below we show a list of the variables contained in **class** `Input` with their corresponding names from the input file. All the flags can be accessed in **class** `Input` as `input.flag[input.name_of_flag]`, where `name_of_flag` is given below in square brackets after the corresponding name from the input file. For better readability the input file is structured in four sections as

1. `[General Input]`

   contains input required for the calculation of cross sections and asymmetries. Input given in this section is used for both the elastic process (including soft-photon and virtual corrections) and the radiative process with a hard photon in the final state.

2. `[E_gamma < Delta]`

   contains input required for the calculation of the non-radiative part of the cross section, including higher-order corrections.

3. `[E_gamma > Delta]`

   contains input required only for the radiative part with hard-photon emission.

4. `[Event Generator]`

   contains input required for the event generator.

The items contained in each section of the input file are the following (unless specified the default value for the flags is 0):

1. `[General Input]`

   - `Incident Lepton` [`lepton`] — a flag that specifies the type of lepton in the initial state. The options that are implemented in the current version are electron (0), positron (1), muon (2), anti-muon (3).
   - `Target Particle` [`target`] — a flag that specifies the type of target particle. The only options that are implemented in the current version are proton (0), carbon-12 (1) and electron (2). For carbon-12 only first order corrections are implemented.
   - `Incident Lepton Energy` [**`double`** `E`] — in units of GeV. The default value is 0.155 GeV.
   - `Polarization` [**`double`** `polarization`] — degree of the longitudinal polarization of the incident lepton beam ($-1 \leq P \leq 0$ for left-handed, $0 \leq P \leq 1$ for right-handed polarization). Default value is 1.
   - `Type of Cuts` [`cuts_born`] — a flag with which the user can choose to use cuts on the scattering angle θ_ℓ (0) or on Q^2 (1).
   - **`theta_l min`** [**`double`** `thl_min`] and **`theta_l max`** [**`double`** `thl_max`] — minimum and maximum values of the scattering angle in degrees. Default values are $25°$ and $45°$.
   - **`Q^2 min`** [**`double`** `Q2min`] and **`Q^2 max`** [**`double`** `Q2max`] — minimum and maximum values of Q^2 in GeV^2. Default values are 0.0044 GeV^2 and 0.0134 GeV^2.
   - `Delta` [**`double`** `Delta`] — value of the photon energy cut-off in GeV to separate soft from hard photon radiation. Default value is 0.01 GeV.
   - `Asymmetry` [`asymmetry`] — a flag which tells the initialization function whether to calculate (0) or not (1) the polarization-dependent part of the cross section and the resulting asymmetries. Default value is 1.
   - `sin2thetaW` [**`double`** `sw2`] — value of the weak mixing angle, $\sin\theta_W^2(M_Z)$ in the $\overline{\text{MS}}$ scheme, where M_Z is the mass of the Z^0 boson. The default value is $\sin\theta_W^2(M_Z) = 0.23122$ (see the latest PDG review by J. Erler and A. Freitas [?]).

- `LO` [`LO`] — flag to suppress the calculation of the leading order (Born level) contribution to the cross section (for `LO=0`). The default, `LO=1`, is to include leading order.
- `Form Factors` [`form_factors`] — flag for the form factor parametrization of the proton. In the present version one can choose among the following options:
  - `Form Factors=0` — Simple dipole form factor with $M_D^2 = 0.71$ GeV^2 and the proton magnetic moment $\mu_p = 2.7928473$;
  - `Form Factors=1` — Dipole times polynomial taken from Bernauer's PhD thesis [19], pp. 181 (see also [20]);
  - `Form Factors=2` — Friedrich-Walcher parametrization [21];
  - `Form Factors=3` — Static limit, $G_E = 1$ and $G_M = \mu_p$;
  - `Form Factors=4` — User defined;
  - `Form Factors=5` — Symmetrized Fermi Form Factor for ^{12}C; (see Ref. [22]).
  - `Form Factors=6` — User defined form factor for ^{12}C.
- `Integration method` [`int_method`] — flag that specifies with witch algorithm is the integration over phase space performed. If the value is set to 0, the program uses Vegas Monte Carlo routine. This is required for event generation after initialization. Total cross sections can also be calculated with Suave (1) or with Cuhre (2). See the Cuba documentation [9] for details.
- `Maximum Number of Evaluations LO` [**`int`** `no_eval_LO`] — maximum number of evaluations of the integrand during the initialization for the leading order and elastic cross sections (default value is 10^7).
- `Maximum Number of Evaluations 1st` [**`int`** `no_eval_1st`] — maximum number of evaluations of the integrand during the initialization for the first order hard-photon bremsstrahlung (default value is 10^8).
- `Maximum Number of Evaluations gamma_loop` [**`int`** `no_eval_gamma_loop`] — maximum number of evaluations of the integrand during the initialization for one hard-photon bremsstrahlung combined with one-loop (default value is 10^8).
- `Maximum Number of Evaluations 2nd` [**`int`** `no_eval_2nd`] — maximum number of evaluations of the integrand during the initialization for the second order hard-photon bremsstrahlung (default value is 10^9).
- `Maximum Number of Evaluations 2nd sg finite` [**`int`** `no_eval_2nd_add`] — maximum number of evaluations of the integrand during the initialization for the finite part of one hard-photon and one soft-photon correction (default value is 10^8).
- `Minimum Number of Evaluations` [**`int`** `no_min_eval`] — minimum number of evaluations of the integrand during the initialization (default value is 10^5).
- `Relative Accuracy` [**`double`** `epsrel`] — required relative accuracy of the numerical integration (default value is 0).
- `Number of cores` [**`int`** `no_cores`] — number of cores to be used by the integration routines (default value is 4).
- `Echo input` [`echo_input`] — a flag to tell the initialization function whether to print the given input.
- `Output` [**`char`** `output_file`] — if any name is chosen here, a summary of input parameters and results of the numerical integration is written to the file *name-given.out*. If 0 is inserted the program doesn't create an output file.

- `Integration Output level` [`int_output`] — flag which tells the Cuba library to print details about the integration (see the `Cuba` documentation [9] for details, a copy of which is found in *doc/cuba.pdf*).
- `NSTART` [**`int`** `nstart`] — Vegas parameter: the number of points per iteration in the first iteration (see *cuba.doc*). Default value is 1000.
- `NINCREASE` [**`int`** `nincrease`] — Vegas parameter: increment for the number of points in subsequent iterations (see *cuba.doc*). Default value is 500.
- `NBATCH` [**`int`** `nbatch`] — Vegas parameter: the batch size for sampling (see *cuba.doc*). The default value is 1000.
- `NNEW` [**`int`** `nnew`] — Suave parameter: the number of new integrand evaluations in each subdivision (see *cuba.doc*). Default value is 100000.
- `NMIN` [**`int`** `nmin`] — Suave parameter: the minimum number of samples for subregions (see *cuba.doc*). Default value is 200.
- `FLATNESS` [**`double`** `flattness`] — Suave parameter to compute the fluctuation of a sample (see *cuba.doc*). Default value is 5.
- `Seed` [**`double`** `seed`] — seed for the pseudo-random-number generator. See *cuba.doc* for more details.

2. `[E_gamma < Delta]`

- `Order SP_loop` [`order`] — a flag to control inclusion of higher-order loop and soft-photon bremsstrahlung corrections. For hard-photon bremsstrahlung corrections see `Bremsstrahlung Type`. 0: include only leading order, i.e. Born-level cross section; 1: include first-order corrections, i.e. one-loop and one soft-photon bremsstrahlung corrections; 2: include second-order corrections, i.e. two-loop, two soft-photon bremsstrahlung and one-loop + one soft-photon bremsstrahlung. If `Bremsstrahlung Type` 1 or 2 is selected, this option adds also one hard-photon bremsstrahlung + one-loop and one hard-photon bremsstrahlung + one soft-photon. Default value is 2.
- `Vacuum Polarization` [`vac_pol`] — flag for choosing the contribution from the vacuum polarization correction (running α).
  - `Vacuum Polarization=0` — vacuum polarization is not included;
  - `Vacuum Polarization=1` — only electron one-loop contribution;
  - `Vacuum Polarization=2` — full leptonic contribution;
  - `Vacuum Polarization=3` — including leptonic and hadronic contributions. The hadronic part is taken from [23];
  - `Vacuum Polarization=4` — including leptonic and hadronic contributions. The hadronic part is taken from [15];
  - `Vacuum Polarization=5` — including leptonic and hadronic contributions. The hadronic part is taken from [24].

  The default value is 3.
- `Hadronic corrections` [`hadr_corr`] — flag for choosing the contribution from corrections to the proton lines. 0: not included; 1: include only the interference between leptonic and hadronic corrections, i.e. the two-photon exchange and the interference between leptonic and hadronic radiation; 2: only purely hadronic terms, i.e. only corrections to the proton line; 3: total contribution, i.e. includes both the interference and the purely hadronic contributions. Option 1 and 2 have to be combined with `Two-photon exchange`¿0 in order for the calculation to be consistent.

- `Two-photon exchange` [tpe] — flag for choosing the contribution of the two-photon exchange correction. The present version includes the Feshbach term (1), i.e. the calculation in which the proton is treated as a point-like particle and a calculation that includes both elastic and inelastic contributions (2) (see Ref. [25]). The latter is valid however only for forward angles and for energy of the incoming lepton of 155 MeV. The default value is 0, for which only the IR terms are included (according to Maximon and Tjon see Ref. [14]).
- `Kappa Form Factor` [kappa_weak] — flag that specifies if the correction factor responsible for the running of $\sin^2\theta_W$ is included (1) or not(1). If 1 is inserted the full contribution is included as described in Ref. [15].

3. `[E_gamma > Delta]`

- `Bremsstrahlung Type` [brems] — flag for choosing the type and the order of the bremstrahlung calculation. The default value is 2 for which first-order and second order hard-photon bremsstrahlung is included. If 1 is inserted only first order-bremsstrahlung is calculated and for 0 the program doesn't include at all hard-photon bremsstrahlung. For testing there are additional options that are provided, including only second order (3) and various differential cross section (4-8). Use with care for testing! Wrong input can produce incomplete results.
- `Bremsstrahlung Add` [brems_add] — flag for specifying whether the remainder finite contribution to one hard photon and one soft photon, $\sigma^R_{1h\gamma+1s\gamma}$ is included (1), or not (0). This calculation is available only in combination with `Order SP_loop`=2 and with `Bremsstrahlung Type`=1 or 2.
- `Gamma Loop` [GL] — flag for specifying whether the exact calculation (0) or the approximation (1) is used for one hard photon and one-loop correction. This calculation is available only in combination with `Order SP_loop`=2 and with `Bremsstrahlung Type`=1 or 2.
- `Hadronic Radiation` [brems_hadr] — flag for specifying the type of contribution coming from proton radiation. 0: the contribution is not included; 1: only the interference terms between leptonic and proton radiation are included; 2: only the squared terms from proton radiation are include; 3: the total contribution, i.e. the interference and squared terms, are included. In order for the calculation to be consistent this option has to be combined with the corresponding option from `Hadronic corrections`. This correction is available only at first order.
- `E_gamma max` [**double** E_gamma_max] — maximum value of the photon energy in GeV.
- `E' min` [**double** E_prime_min] and `E' max` [**double** E_prime_max] — minimum and maximum values of the scattered electron energy in GeV.
- `theta_gamma min` [**double** thg_min] and `theta_gamma max` [**double** thg_max] — minimum and maxium values of the photon polar angle in degrees.
- `Phase Space Parametrization` [ps] — flag for choosing the type of phase space parametrization. It is valid only for the one hard-photon correction and can be useful depending on the experimental cuts that are needed. If 0 is inserted, the azimuthal angle ϕ_γ is expressed in terms of the remaining variables E', θ_ℓ, E_γ, θ_γ. If 1 is inserted, then the energy of the scattered lepton E' is expressed of the remaining variables θ_ℓ, E_γ, θ_γ ϕ_γ.

4. `[Event Generator]`

- `E min` [**double** `E_min`] and `E max` [**double** `E_max`] — minimum and maximum values for the energy of the incoming lepton. Maximum allowed range that is implemented in this version is between 0.01 GeV and 10 GeV.
- `Delta E` [**double** `Delta_E`] — step size for event generation with variable initial-state energy. Valid steps in the current version are 10^{-4}, 10^{-3} and 10^{-2} GeV.

6 Description of the output

For the functions `initialization()`, `sigma_diff_Omega_l(`**const double** `thl)` and `shiftQ2(`**const double** `thl)` the output is given in the objects `Output output` and `Output errors`. The object `output` contains the results of the numerical integration and the object `errors` the estimated uncertainties of these results. The output of the function **int** `events()` is given in the object `Final_State FS`. The detailed output for each function is:

1. **int** `initialization()`

   The members of `output` contain the values of partial cross sections, their uncertainties and asymmetries. Their names are structured as follows:

$$\texttt{output.sigma\_}\begin{bmatrix}\texttt{unpol}\\\texttt{pol}\end{bmatrix}\texttt{\_}\begin{bmatrix}\texttt{elastic}\\\texttt{inelastic}\end{bmatrix}\texttt{\_}\begin{bmatrix}\texttt{born}\\\texttt{1st}\\\texttt{2nd}\\\texttt{loop}\end{bmatrix}$$

   and similarly for `errors`. `unpol` and `pol` distinguish between unpolarized and polarized parts of the cross section while `inelastic` and `elastic` separate contributions to the cross section which have or have not a hard photon in the final state. `born`, `1st` and `2nd` denote contributions at leading, next-to-leading and next-to-next-to-leading order, i.e. the Born level cross section (Rosenbluth cross section, the contribution with one virtual or real photon, and contributions with two virtual or real photons. The one-loop correction to one-photon bremsstrahlung is denoted by the key word `loop` at the end of the name. All cross sections are given in units of nanobarns. In addition, the left-right asymmetries at Born level, as well as at first and second order are calculated and stored in the variables `output.asymm_born`, `output.asymm_1st` and `output.asymm_2nd`. Finally, there are vectors to store results for multiple energies:

   - `output.sigma_born_vect` — vector for storing the Rosenbluth cross section in case the initialization is done for multiple energies (nb).
   - `output.sigma_unpol_vect` — vector for storing the unpolarized total cross section in case the initialization is done for multiple energies (nb).
   - `output.sigma_pol_vect` — vector for storing the polarized total cross section in case the initialization is done for multiple energies (nb).
   - `output.ev_brems` — vector for storing the ratio between the inelastic and unpolarized total cross sections in case the initialization is done for multiple energies (nb).

2. **int** `shiftQ2()`

   The results for the shift in Q^2 are given in `output.shiftQ2`, while the uncertainties for this result in `errors.shiftQ2`, both in GeV^2.

3. **int** sigma_diff_Omega_l(**const double** thl_deg)

   The results of this function are stored in members of the output class as described above for total cross sections, but now for the differential cross section $\mathrm{d}\sigma/\mathrm{d}\Omega$ in units of nanobarns times sterad. Details can be found in the sample program *examples/sigma_Omega_l_test.cpp.*

4. **int** events()

   The results of the events function are stored in **class** FS as:

   - FS.E_l — energy of the scattered lepton E' (MeV).
   - FS.theta_l — lepton scattering (polar) angle θ_ℓ (rad).
   - FS.phi_l — lepton azimuthal angle ϕ_ℓ (rad).
   - FS.E_p — energy of the final proton (MeV).
   - FS.theta_p — final proton polar angle (rad).
   - FS.phi_p — final proton azimuthal angle (rad).
   - FS.E_gamma — emitted photon energy (MeV).
   - FS.theta_gamma — photon polar angle (rad).
   - FS.phi_gamma — photon azimuthal angle (rad).
   - FS.Q2 — leptonic momentum transfer squared (GeV^2).
   - FS.l_1[4] — momentum 4-vector of the incoming lepton.
   - FS.p_1[4] — momentum 4-vector of the incoming proton.
   - FS.l_2[4] — momentum 4-vector of the scattered lepton.
   - FS.p_2[4] — momentum 4-vector of the recoil proton.
   - FS.k[4] — momentum 4-vector of the bremsstrahlung photon.
   - weight — Event weight.
   - avg_weight — Averaged event weight.
   - FS.event_no — event number.
   - FS.event_type — 0 for elastic (non-radiative) and 1 for radiative events.

The function **int** events() will generate two types of events:

1. Elastic events characterized by the scattered lepton and the recoil proton in the final state. Event simulation follows the one-fold differential unpolarized and polarized cross sections in the scattering angle (in units of nb/rad),

$$\frac{\mathrm{d}\sigma_{\mathrm{unpol}}}{\mathrm{d}\theta_\ell} \quad \text{and} \quad \frac{\mathrm{d}\sigma_{\mathrm{pol}}}{\mathrm{d}\theta_\ell}.$$

   At first order, the cross sections include one-loop corrections and a soft-photon correction which was obtained analytically by integrating over photon momenta with energies $E_\gamma < \Delta$. The cut-off Δ must be chosen small enough so that the photon is soft and cannot be detected. Elastic events can be recognized by the fact that the photon energy and angles in the final-state listing, Final_State FS, are exactly 0. At second order the cross sections include two-loop, one-loop and one soft photon and two soft photon corrections. The soft photon energies are taken to be separately smaller than the cut-off value, E_γ, $E'_\gamma < \Delta$.

2. Radiative events which contain an additional bremsstrahlung photon with energy $E_\gamma > \Delta$, or two bremsstrahlung photons with energies $E_\gamma > \Delta$ and $E'_\gamma > \Delta$. The photon is hard and can be detected, provided its energy is above a threshold and its angle within the acceptance range of the detector. Event simulation is performed according to the four-fold differential unpolarized and polarized cross sections (units nb/(GeV^{2}rad^2)) in case of one radiated hard photon as

$$\frac{\mathrm{d}^4\sigma_{\mathrm{unpol}}}{\mathrm{d}E'\mathrm{d}\theta_\ell\mathrm{d}E_\gamma\mathrm{d}\theta_\gamma} \quad \text{and} \quad \frac{\mathrm{d}^4\sigma_{\mathrm{pol}}}{\mathrm{d}E'\mathrm{d}\theta_\ell\mathrm{d}E_\gamma\mathrm{d}\theta_\gamma}.$$

or according to the seven-fold differential unpolarized and polarized cross sections (units nb/(GeV^{2}rad^2)) in case of two radiated hard photons as

$$\frac{\mathrm{d}^7\sigma_{\mathrm{unpol}}}{\mathrm{d}E'\mathrm{d}\theta_\ell\mathrm{d}E_\gamma\mathrm{d}E'_\gamma\mathrm{d}\theta_\gamma\mathrm{d}\theta'_\gamma\mathrm{d}\phi'_\gamma} \quad \text{and} \quad \frac{\mathrm{d}^7\sigma_{\mathrm{pol}}}{\mathrm{d}E'\mathrm{d}\theta_\ell\mathrm{d}E_\gamma\mathrm{d}E'_\gamma\mathrm{d}\theta_\gamma\mathrm{d}\theta'_\gamma\mathrm{d}\phi'_\gamma}.$$

7 How to use the program

Typical running times can vary considerably, depending on the provided input options. The initialization of unpolarized and polarized cross sections at second order, on a 2.3 GHz Intel Core i7 processor with default values, takes approximately 1h, while at first order it takes approximately 8 seconds. The numerical uncertainties in this case are of order 10^{-4}. By default the numerical integration is done using Vegas integration method, except for the finite part of one hard photon and one soft photon correction (see [6], sec. 4.3), which is done using Suave integration method. The reason for this is that we found that Suave performs substantially better in this particular case. The memory usage of Vegas is negligible, however Suave uses significant memory to perform the integration (see [9] for details). The memory usage increases with the number of evaluations. For the default value of 10^7 evaluations, the memory required is approximately 0.7 GB.
Note that the evaluation of the polarization asymmetry requires the separate evaluation of the unpolarized and polarized cross sections resulting in a factor of two increase in run-time.
The program was designed for low-energy high-precision experiments and has to be used with care for higher energies. It has not been tested for lepton energies above 10 GeV. Also event generation was tested for energies between 0.1 and 10 GeV only. The initialization will work with energies outside this range, but not event generation.
As was mentioned earlier, the program always checks if the input given is correct and throws a warning, that specifies the problem, otherwise. At the same time it changes the value that is considered wrong to the default value. A possibility of wrong input are for example the values of the flags, which can be chosen outside the range they were defined in the program. It may also happen that an input value is not correct from a physical point of view. For example, the input of an angle or of an energy can be outside the range allowed by energy-momentum conservation. In this case the program either sets the value to the default one, or calculates the maximum or the minimum allowed value.
An input object has always to be defined, even if it is empty, and given as argument to the function `set_input` as

```
Input my_input;
my_pes_class.set_input(my_input);
```

The program will stop and it will show an warning in case the input object is not defined. When generating events the user has to make sure that the initialization is done using Vegas. The other integration methods are just for the calculation of the cross sections and do not work for event generation.

7.1 Event simulation for variable initial-state energy

Event simulation always requires information about the fully differential cross section which is stored in grids. This information is evaluated as a function of the energy of the incoming lepton. The program allows to administer grids for a list of beam energies such that event generation can be performed with variable initial-state energy. For technical reasons, the current version of the event generator works only with energies of the incoming lepton between 0.01 and 10 GeV with increments of 10^{-4}, 10^{-3}, or 10^{-2}. The integration will still work for energies outside these values, but not the event generator.
Before event generation, the user has to call the initialization for a grid of energy values:

```
for (E = input.E_min; E <= input.E_max; E += input.Delta_E) {
if (my_pes_class.change_energy_initialization(E))
my_pes_class.initialization();
}
```

Subsequent event generation can be performed with the following function:

```
for (int i = 0; i < no_events; i++) {
double Ei=//take value from somewhere//;
if (my_pes_class.change_energy_events(Ei))
my_pes_class.events();
//write or analyze event listing//
}
```

The energy value `Ei` has to be inside the range `[E_min, E_max]`. If `Ei` is not equal to one of the values in the list of energies for which the initialization was performed, event generation will be called for the closest value found in the grid. `my_pes_class.events()` will create an event listing and provide energies and scattering angles for all final-state particles in a class called `FS`. The members of this class have been described above in Sec. **??**. A sample program is contained in *examples/multiple_random_E_test.cpp*.

8 File list

Fig. 2 contains a map of all class dependencies of the main class `PES`. Each class is defined in a separate file, except for the smaller input/output classes, which are defined in one single file. Below we provide a list of these files and a short description of each class and how they depend on the other classes:

- *Polares.h* — the main header file containing **class** `PES`. A description of this class can be found in Sec. 5. It defines all the main functions and depends on all other classes.

- *IO_classes.h* — header file containing the definitions of the input/output classes. The description of these classes can be found in Sec. **??** and Sec. **??**.

- *parameters.h* — header file containing the **class** `Parameters` which reads the input from the input file and sets all the parameters required by the program. It also makes sure the correct input is given and gives warnings and/or modifies it accordingly, otherwise. Most other classes depend on this class.

- *cuba_param.h* — header file containing **class** `Cuba_parameters`, which defines all the parameters required by the Cuba library for numerical integration (see Ref. [9]). Depends on the **class** `Parameters`.

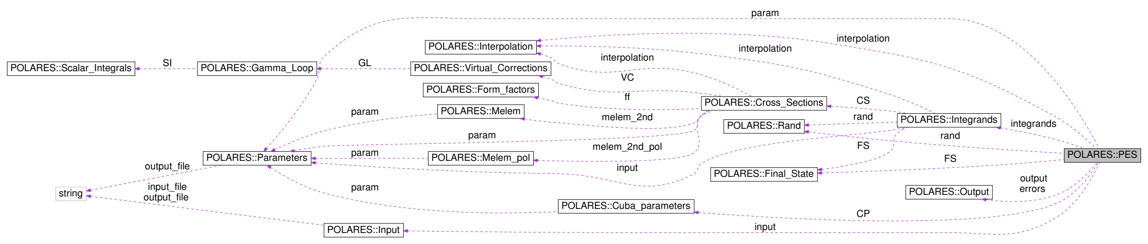

Figure 2: Graph that shows the class dependencies of POLARES.

- *gsl_rand.h* — header file that defines the **class** Rand which generates random numbers using the GSL library. Class Integrands and **class** PES depend on this class.

- *integrands.h* — header file containing **class** Integrands which defines all the integrands for numerical evaluation. **class** PES depends on this class.

- *cross_sections.h* — header file containing **class** Cross_Sections which defines all the expressions for the cross sections. Class Integrands depends on this class.

- *virtual_corrections.h* — header file containing **class** Virtual_Corrections which defines all the expressions for the non-radiative corrections.
Class Cross_Sections depends on this class.

- *form_factors.h* — header file containing **class** Form_Factors which defines all form factors parametrizations. Class Cross_Sections depends on this class.

- *melem.h* — header file containing **class** Melem which defines second order unpolarized matrix elements. Class Cross_Sections depends on this class.

- *melem_pol.h* — header file containing **class** Melem_pol which defines second order polarized matrix elements. Class Cross_Sections depends on this class.

- *interpolation.h* — header file containing **class** Interpolation which defines the interpolation functions for hadronic vacuum polarization and two photon exchange corrections. Class Cross_Sections, **class** Integrands and **class** PES depend on this class.

- *gamma_loop.h* — header file containing **class** Gamma_Loop which defines the corrections for one loop and one hard-photon. Class Virtual_Corrections depends on this class.

- *scalar_integrals.h* — header file containing the class Scalar_Integrals which defines the expressions for the required scalar integrals. Class Gamma_Loop depends on this class.

- *const.h* — header file that defines all the required constants. It is used by almost all other files.

Some examples of how the library can be used can be found in the folder *examples/*. These files are

- *main_example.cpp* — A sample program that shows how to calculate cross sections and perform event generation for a fixed energy of the incoming lepton beam, using the functions initialization and events. An example with the output of `initialization` can be found in Sec. 9. The output of `events` is printed in a file called *events.txt*. The order in which the variables are printed in this file is the following: event number, E, E', θ_ℓ, ϕ_ℓ, E_γ, θ_γ, ϕ_γ, E'_γ, θ'_γ, ϕ'_γ and the event weight. The notation was defined above in Fig. 1.

- *multiple_random_E_test.cpp* — A program file that shows how to use the functions `initialization` and `events` for a range of energies of the incoming lepton beam. The output of `events` is printed, as in the previous example in a file called *events.txt*. The order in which the variables are printed in this case is the following: event number, $E_{\rm rand}$, E, the components of the four-vector l_2, the components of the four-vector p_2, the components of the four-vector k_1, the components of the four vector k_2 and the event weight, where the notations are defined in Sec. 2. Notice that the random energy that is provided as input $E_{\rm min} < E_{\rm rand} < E_{\rm max}$ is not identical with the energy for which the event is generated because the value of $E_{\rm rand}$ is rounded to the closest value defined in the grid.
- *sigma_diff_Omega_l_test.cpp* — A program file that shows how to use the function `sigma_diff_Omega_l` to calculate the unpolarized and polarized cross sections, differential with respect to the solid angle of the scattered lepton.

9 Examples

Here we reproduce the output generated from the program *main_example.cpp* using input from the file *POLARES.in*. It is contained in the file named *POLARES.out* and can be found in the package. This output is helpful for a user to test his/her new installation of POLARES.

```
###########################################################################
##                               POLARES 1.4
##
##       Radiative Corrections for Polarized Electron-Proton Scattering
##
##                              R.-D. Bucoveanu
##
##                        Tue Sep 10 11:36:02 2019
###########################################################################
##                                  Input
##
## [General Input]
## Type of incident lepton = electron
## Type of target particle = proton
## Incident lepton energy = 0.155 GeV
## Lower cut-off value for the photon energy (Delta) = 10 MeV
## Type of cuts for elastic scattering = Scattering angle (theta_l) cuts
## theta_l min = 25 degrees
## theta_l max = 45 degrees
## Form factor parametrization = Simple Dipole
## Calculate the asymmetry = yes
## Degree of Polarization = 100%
## sin2thetaW = 0.231
## kappa form factor = 1 - full contribution for the running of sin2thetaW
## Maximum number of evaluations for 1st order bremsstrahlung = 20000000
```

```
##
## [E_gamma < Delta]
## Vacuum Polarization = Hadronic contributions
## Two-photon exchange correction (TPE) = no contribution
##
## [E_gamma > Delta]
## Type of hard-photon bremsstrahlung = 1st order
## Hadronic Radiation = no hadronic radiation contribution
## E_gamma max = 0.11 GeV
## E' min = 0.045 GeV
## E' max = 0.145 GeV
## theta_gamma min = 0 degrees
## theta_gamma max = 180 degrees
##########################################################################
##                    Numerical integration results
##
## Sigma unpol Born = 34539.26 +- 0.1009354 nb
## Sigma unpol soft-photon 1st order = 32030.48 +- 0.09361499 nb
## Sigma unpol hard-photon 1st order = 4291.148 +- 2.979682 nb
## Sigma unpol 1st order = 36321.62 +- 2.981152 nb
## Sigma pol soft-photon 1st order = -0.001322356 +- 3.981649e-09 nb
## Sigma pol hard-photon 1st order = -0.0001086349 +- 6.23166e-08 nb
## Sigma pol Born = -0.001437277 +- 4.324964e-09 nb
## Sigma pol 1st order = -0.001430991 +- 6.629825e-08 nb
## Asymm 1st order = -3.939778e-08 +- -3.713238e-12
## Asymm Born = -4.161285e-08 +- -1.745507e-13
##########################################################################
```

9.1 Weight distributions

Event simulation is performed using the algorithm of Vegas which assigns a weight w_j to each event, calculated from partial cross sections defined on the internal grid of Vegas.

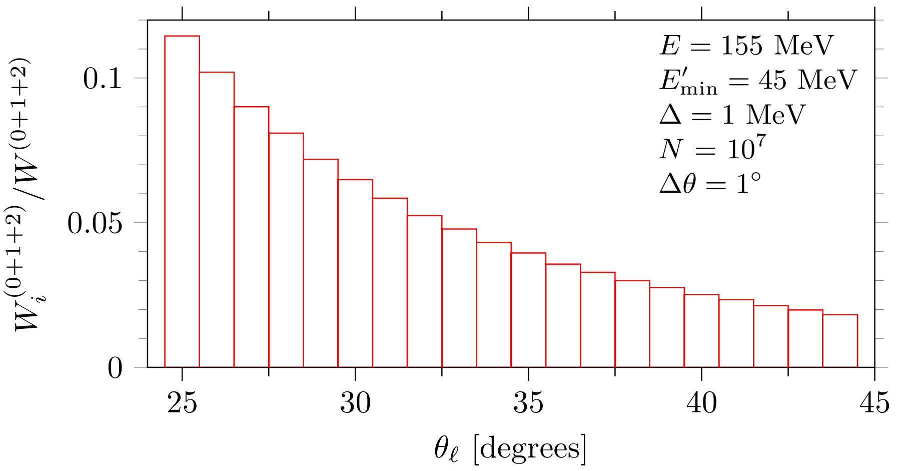

Figure 3: The distribution of the weights ratio W_i/W defined in Eq. (16) and Eq. (17) for the cross section including first and second order corrections $\sigma^{(0+1+2)}$.

These weights are defined in such way that the cross section is estimated from

$$\sigma = \frac{W}{N}, \tag{15}$$

where the total weight W is the sum of all individual event weights,

$$W = \sum_{j=1}^{N} w_j, \tag{16}$$

and N the total number of events. Note that for convenience we have included the normalization to the total cross section in this definition of the weights. We can also define the binned weight, which is the sum of all event weights in one particular bin. Choosing bins in the scattering angle θ_ℓ with the size of $\Delta\theta$, the binned weight is defined as

$$W_i = \sum_{j=1}^{N} w_j \Theta(\theta_\ell - \theta_i)\Theta(\theta_i + \Delta\theta - \theta_\ell), \tag{17}$$

where i is the bin in degrees and Θ the Heaviside function. Fig. 3 shows the distribution of the weights ratio $W_i^{(2)}/W^{(2)}$ for the cross section that includes first order and second order corrections $\sigma^{(0+1+2)}$ (see Sec. 2 for more details of how this cross section is defined). The weights ratio can be compared with the ratio of the cross section $\sigma_i^{(0+1+2)}$ in each bin relative to the total cross section $\sigma^{(0+1+2)}$,

$$\sigma_i^{(0+1+2)} = \int_{\theta_i}^{\theta_i+\Delta\theta} \mathrm{d}^4\sigma^{(0+1+2)} \quad \text{and} \quad \sigma^{(0+1+2)} = \int_{\theta_\ell^{\min}}^{\theta_\ell^{\max}} \mathrm{d}^4\sigma^{(0+1+2)} \tag{18}$$

which can be calculated without event simulation. Fig. 4 shows the result of this comparison. The cross sections $\sigma_i^{(0+1+2)}$ and $\sigma^{(0+1+2)}$ here are calculated with the Suave integration method (see [9]), while the weights are provided by the Vegas integration routine. We find good agreement between the two distributions, which allows us to conclude that weights of the event generator follow the correct distribution with a precision lower than 10^{-3}.
As an additional example we show results for the distribution of event weights, W_i/W, describing the differential cross section with two hard photons in the final state, $\mathrm{d}\sigma_{2h\gamma}/\mathrm{d}\theta_\gamma$ defined in Eq. (12), as a function of the polar angle θ_γ of one of the emitted photons in Fig. 5. In this figure we can clearly distinguish the expected enhancement in the region where $\theta_\gamma = \theta_\ell$. This enhancement is due to the well-known collinear peak from final-state radiation (see [6] for more details). In the present case, the peak is washed out since the electron scattering angle is integrated over the range $25° \leq \theta_\ell \leq 45°$.
Finally we present in Fig. 6 the result of a calculation for electron-^{12}C scattering. This figure shows the ^{12}C cross section measured as a function of the scattering angle θ_ℓ at energies of 420 MeV for incident electrons. The leading order result from this plot can be compared with one of the first measurements of nuclear form factors carried out at the Standford linear accelerator (see [26]). At leading order we can clearly distinguish two diffraction minima, while at first order we notice that these minima are washed out, due to photon radiation.

Acknowledgment

We thank Dominik Becker for his feedback and help to develop `POLARES`. We also want to thank Matthias Heller for providing us with useful analytical expressions for the more

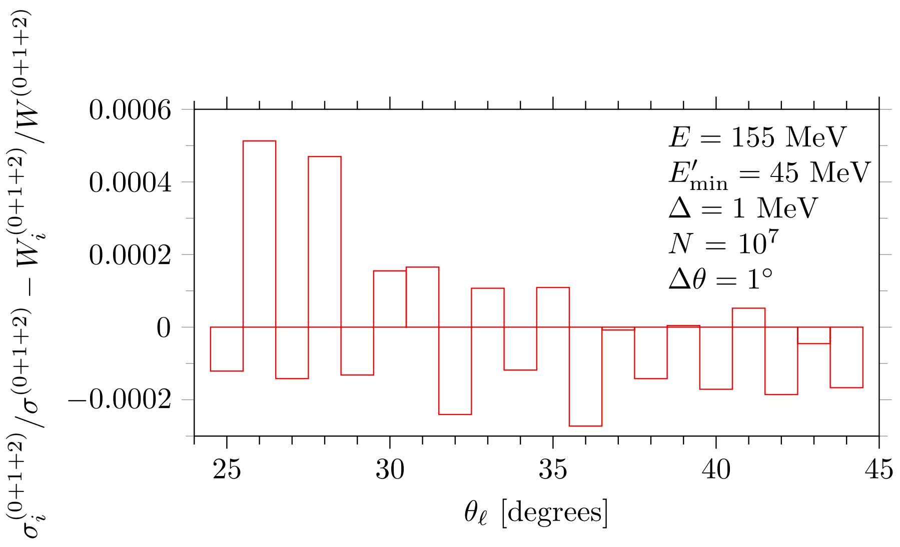

Figure 4: The ratio of event weights (see Fig. 3) compared with the ratio of the cross sections defined in Eq. (18), for each bin.

complex scalar integrals and Oleksandr Koshchii for useful discussions and for providing us with important information concerning carbon scattering. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) in the framework of the collaborative research center SFB1044 "The Low-Energy Frontier of the Standard Model: From Quarks and Gluons to Hadrons and Nuclei".

References

[1] D. Androić *et al.* [Qweak Collaboration], Nature **557** (2018) 207 [arXiv:1905.08283 [nucl-ex]].

[2] D. Becker *et al.*, Eur. Phys. J. A **54** (2018) 208 [arXiv:1802.04759 [nucl-ex]].

[3] L. W. Mo and Y. S. Tsai, Rev. Mod. Phys. **41** (1969) 205.

[4] Y. S. Tsai, Phys. Rev. **122** (1961) 1898.

[5] Y. S. Tsai, SLAC-PUB-0848.

[6] R.-D. Bucoveanu and H. Spiesberger, Eur. Phys. J. A **55** (2019) 57 [arXiv:1811.04970 [hep-ph]].

[7] A. V. Gramolin, V. S. Fadin, A. L. Feldman, R. E. Gerasimov, D. M. Nikolenko, I. A. Rachek and D. K. Toporkov, J. Phys. G **41** (2014) 115001 [arXiv:1401.2959 [nucl-ex]].

[8] M. Vanderhaeghen, J. M. Friedrich, D. Lhuillier, D. Marchand, L. Van Hoorebeke and J. Van de Wiele, Phys. Rev. C **62** (2000) 025501 [hep-ph/0001100].

[9] T. Hahn, Comput. Phys. Commun. **168** (2005) 78 [hep-ph/0404043].

[10] D. H. Beck and R. D. McKeown, Ann. Rev. Nucl. Part. Sci. **51** (2001) 189 [hep-ph/0102334].

[11] C. J. Horowitz, Phys. Rev. C **57** (1998) 3430 [nucl-th/9801011].

[12] J. Erler, A. Kurylov and M. J. Ramsey-Musolf, Phys. Rev. D **68** (2003) 016006 [hep-ph/0302149].

[13] A. Czarnecki and W. J. Marciano, Int. J. Mod. Phys. A **15** (2000) 2365 [hep-ph/0003049].

[14] L. C. Maximon and J. A. Tjon, Phys. Rev. C **62** (2000) 054320 [nucl-th/0002058].

[15] F. Jegerlehner, Nuovo Cim. C **034S1** (2011) 31 [arXiv:1107.4683] [hep-ph].

[16] R. K. Ellis and G. Zanderighi, JHEP **0802** (2008) 002 [arXiv:0712.1851 [hep-ph]].

[17] M. Heller, O. Tomalak, M. Vanderhaeghen, S. Wu, [arXiv:1906.02706 [hep-ph]].

[18] P. J. Mohr et al. , Rev. Mod. Phys. **88** (2016) 035009 [arXiv:1507.07956 [atom-ph]].

[19] J. C. Bernauer, PhD thesis, Mainz 2010, `http://wwwa1.kph.uni-mainz.de/A1/publications/doctor/`.

[20] J. C. Bernauer, P. Achenbach, C. A. Gayoso et al. , Phys. Rev. C **90** (2014) 015206 [arXiv:1307.6227 [nucl-ex]].

[21] J. Friedrich and T. Walcher, Eur. Phys. J. **A17** (2003) 607 [hep-ph/0303054].

[22] J. Piekarewicz, A. R. Linero, P. Giuliani and E. Chicken, Phys. Rev. C **94** (2016) 034316 [arXiv:1604.07799 [nucl-th]].

[23] F. Ignatov, `http://cmd.inp.nsk.su/~ignatov/vpl/`

[24] A. Keshavarzi, D. Nomura and T. Teubner, Phys. Rev. D **97** (2018), 114025 [arXiv:1802.02995 [hep-ph]].

[25] O. Tomalak and M. Vanderhaeghen, Phys. Rev. D **93** (2016) 013023 [arXiv:1508.03759 [hep-ph]].

[26] R. Hofstadter, Ann. Rev. Nucl. Part. Sci. **7** (1957) 231.

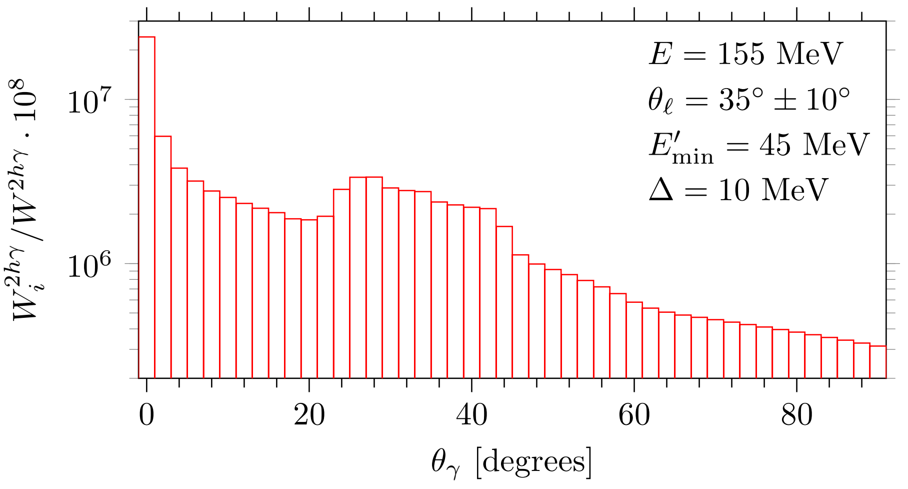

Figure 5: The distribution of the weights ratios W_i/W defined in Eq. (16) and Eq. (17) for the cross section with two hard photons in the final state.

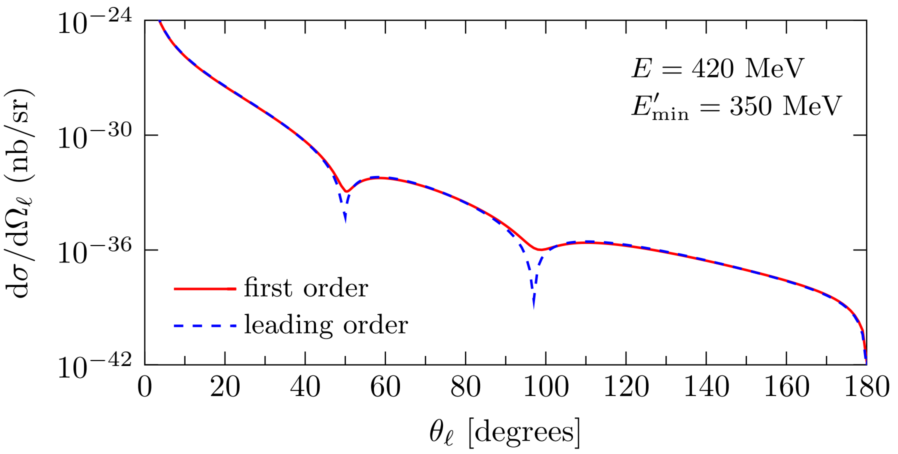

Figure 6: Cross section at Born level and including first-order radiative corrections for electron ^{12}C scattering. Photon radiation at first order reduces the depth of the diffraction minima, which are clearly visible at leading order.