The c++ codes in each folder implements the different steps in the algorithm for the resummation procedure discussed in the paper for the divergent weak-field expansion for the Heisenberg-Euler Lagrangian (HEL). Each subdirectory contains a README.pdf file that gives a detailed explanation of the code. The directory tree is

```
FOR_PRSA/
 Electric
    1_5k_mom_1_5kdig
    1k_mom_1kdig
    2h_mom_3kdig
    2k_mom_2kdig
    5h_mom_550dig
    construct_P
    Delta
    moments
    Pade_1k
 Magnetic
     1_5k_mom_1kdig
     1h_mom_130dig
     1k_mom_1kdig
     20_mom_1hdig
     2h_mom_250_dig
     2k_mom_2kdig
     50_mom_1hdig
     5h_mom_5hdig
     5_mom_1hdig
     construct_p
     Delta
     moments
     Pade_50
```

The subdirectories Magnetic and Electric contain the source codes relevant in the resummation of the HEL in the cases of purely magnetic and electric background, respectively.

The subdirectory construct_p contains the code for constructing the matrix in equation (4.25) of the paper.

The subdirectory `Delta` contains the code that implements the resummation in reference [5].

The subdirectory `moments` contains the code that generates the perturbation coefficients in the the weak field expansion in equation (3.3)

The subdirectory `Pade_50` and `Pade_1k` implements Pade approximants of the corresponding divergent weak-field expansions for the HEL using say 50 and 1000 perturbation coefficients, respectively.

The rest of the directories are implementations of the resummation algorithm for various number of moments and working precisions. In the directory `Magnetic/1h_mom_130dig` for instance,

```
Magnetic/1h_mom_130dig/
   Constants
   first
   fourth
   function
   results
   second
   third
```

the subdirectory `Constants` contains codes for solving the system of linear equations in equation (4.24) using 100 perturbation coefficients a_n as inputs at 130-digit working precision.

The subdirectories `first`, `second`, `third`, `fourth` compute the first, second, third and fourth terms in equation 4.26. While codes in `function` computes the second term $\beta\Delta(\beta)$ in equation 4.20.

The code uses the `c++ Boost` Library. In particular, `Boost.Multiprecision` which requires the GNU GMP, GNU MPFR, GNU MPC libraries; `Boost.MPI` which requires an underlying MPI implementation for which we used Open-MPI v5.0.5. We also used the Eigen 3 library for the LU factorization routine that uses the `mpreal` datatypes from the `MPFR c++` library `http://www.holoborodko.com/pavel/mpfr/`.