TECHNICAL TRANSACTIONS 6/2017
CZASOPISMO TECHNICZNE 6/2017
PHYSICS
DOI: 10.4467/2353737XCT.17.103.6579

Stanisław Jadach
The Henryk Niewodniczański Institute of Nuclear Physics, Polish Academy of Sciences

Radosław A. Kycia (kycia.radoslaw@gmail.com)
Faculty of Physics, Mathematics and Computer Science,
Cracow University of Technology

# Software for calculations of the Higgs boson lineshape in future lepton colliders

# Program do obliczeń przekroju czynnego dla bozonu Higgsa w przyszłych zderzaczach leptonowych

**Abstract**

Simple software for Monte Carlo (MC) calculation of the Higgs boson lineshape and its beam broadening effect due to the beam energy dispersion is described. The software is based on the FOAM [1, 2] adaptive MC integrator from ROOT library [3]. This software enables the reproduction of the results presented in publication [4] and the tests with different parameters for the lineshapes and QED correction factors. A parallel version of the software based on MPI [5] is also described.

**Keywords:** QCD; the Higgs boson cross section; energy scan; Monte Carlo methods; FOAM; machine energy spread; initial state radiation

**Streszczenie**

Opisano prosty program do obliczeń Monte Carlo(MC) rozkładu przekroju czynnego dla bozonu Higgsa i poszerzenia tego rozkładu związanego z dyspersją energii wiązki. Program bazuje na adaptacyjnym programie do całkowania metodą MC - FOAM [1, 2] z biblioteki ROOT [3]. Oprogramowanie umożliwia reprodukcję rezultatów z publikacji [4] i testy tych wyników z różnymi parametrami wiązki i czynnikami korekcyjnymi QED. Zrównoleglona wersja oprogramowania bazująca na MPI [5] również została opisana.

**Słowa kluczowe:** QCD, przekrój czynny dla bozonu Higgsa, skanowanie energią, metody Monte Carlo, FOAM, rozrzut energii wiązki, promieniowanie w stanie końcowym

## 1. Introduction

One of the main aims of future lepton (electron and muon) colliders is detailed examination of the Higgs boson 's properties, and therefore, examination of effects that can influence this reaction is one of the most important aspect of their design. Initial state radiation (ISR) is one of the key factors – as shown in [4] – it reduces the cross section peak of the Higgs boson by a factor of about 35% for electron and 55% for muon colliders. These derivations use methods from analogous calculations of the same effect for the Z boson made for LEP [4].

Further detailed studies of ISR require specially designed software. The full featured Monte Carlo general generators that incorporate initial and final state radiation are currently developed – see detailed discussion in [12] and [13]. However, as they are multi-purposed software, the number of configuration options is large. In this paper a simplified and specialized software that allows one to simulate the Higgs boson lineshape with ISR effects aimed at use in designing future lepton colliders will be presented. It is useful in performing fast simulations of ISR cross section damping and predicting the cross section dependence on machine parameters.

The software presented here was used to produce results from paper [4] and therefore it is advisable to study this paper before commencing experiments with the programs. In [4] there is also all the theory required to understand the output. The software is available at [6].

This paper is organized as follows. In the next section requirements for running the software are provided followed by a short example. Next, the general idea of the software design is described. Finally a short introduction to the parallel version based on MPI [5] intended for specialists in this field is presented.

## 2. Requirements

The software requires standard and free software available for most Unix-like systems.
In order to compile and run the programs the following tools are mandatory:

- Unix-like system – POSIX compatible, e.g., Linux;
- Make compatible system, e.g., GNU make [7];
- g++ compiler from GNU Compiler Collection [8] or any other compatible compile
- ROOT library [3][1] .

There is also additional software that is required to perform more advanced operations:

- Doxygen [9] (optional) if one requires generation of the documentation from the code;
- Valgrind [10] (optional) if advanced debugging is required;
- MPI [5], e.g., OpenMPI (optional) if the user wants to use the parallel version of the program located in the MPI directory.

The repository [6] contains three directories:

[1] ROOT 6 or higher is compiled using C++ 2011 standard. Therefore, when GNU`g++` compiler is used then the flag `-std=c++11` should be added during compilation for compatibility, see Makefile.

- Simple – simplified version of software, suitable for learning its structure;
- Full – full version of the software;
- MPI – parallelized version.

In the next two sections the simplified version of the software will be presented.

## 3. One minute example

A simplified version of the program was constructed to familiarize new users with the philosophy of its design.

In order to run the program, a change directory to Simple is required and then execution of make run in the console. This command triggers the compilation and starts the program. When the program run ends, which should take no more than one minute on current desktop machines, the user will be able to see SimplePlot.eps file containing the example plot show in Fig. 1. These plots demonstrate only some capabilities of the library of functions belonging to the full program. In Fig. 1 the convolution of the Gaussian distribution that describes beam/machine energy spread:

$$G\left(E-E_0,\delta_E\right)=\frac{1}{\delta_E\sqrt{2\pi}}e^{-\frac{\left(E-E_0\right)^2}{2\delta_E^2}},$$

where:

$E_0$ – the central value of the beam energy,

$\delta_E$ – its spread, with a cross section is defined by;

$$\sigma_E^{\text{conv}}\left(E,\sigma_E\right)=\int dE'\sigma\left(E'\right)G\left(E'-E,\sigma_E\right),$$

where:

$\sigma$ – the cross section – the Born term without or with ISR corrections.

The details are described in [4].

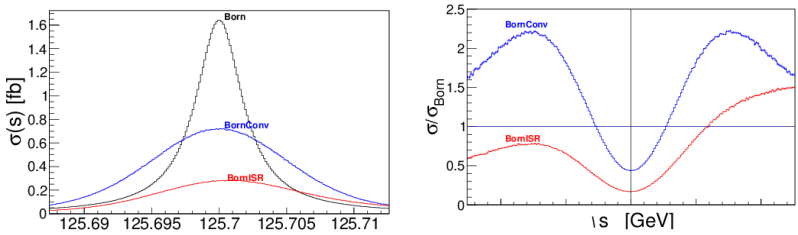


Fig. 1. Upper panel: The Born cross section for $e^+e^-$ à H process – black line, the Born cross section convoluted with the Gaussian distribution for the energy spread (in the centre of mass frame $\delta_E$ = 4.2 MeV – blue line, and the Born cross section with ISR corrections included. Lower panel: ratios of the last two plots by the Born term. OX axis is the same for both figures

## 4. Code analysis – simplified version

We start analysing the code from the main.cxx file from the Simple directory. At the end of the file there is the main() function which is the entry point of the program. The first relevant instruction is:

```
long NevTot = 10000000;
```

which sets up the statistic for the Monte Carlo integrator. The integration error on the ISR corrections or convolution with beam centre energy spread is proportional to $\frac{1}{\sqrt{\text{NevTot}}}$. On the other hand the time of integration is proportional to the statistics.

In the next lines, histograms are created. The line:

```
MakeBorn(string("BornH") );
```

creates the Born term histogram and saves it in BornH.root file. Then in the line

```
MakeConvBorn( string("histo-sig04-born"), 0.0042, NevTot );
```

the histogram for the Born term convoluted with the Gaussian distribution that describes beam energy deviation with the energy spread $\delta_E = 0.0042$ GeV is saved in the file histo-sig04-born.root. Finally, the similar plot including ISR corrections is created in

```
MakeISR( string("histo-sig04-isr2"), 2, 3, 0.0042, NevTot );
```

and it is saved in histo-sig04-isr2.root file. The plots are created in the function

```
plotSimpleFigs();
```

This function can be customized by the user to adjust the plots to her/his requirements. We will analyse the main parts of the function plotSimpleFigs() function. The first part

```
TFile DiskFileBorn(  "./histo-sig04-born.root");
TFile DiskFileISR2(  "./histo-sig04-isr2.root");
TFile DiskFileBornH(  "./BornH.root");

TH1D *h_Born       = (TH1D*)DiskFileBornH.Get("h_SigEne");
TH1D *h_SigEneBorn  = (TH1D*)DiskFileBorn.Get("h_SigEne");
TH1D *h_SigEneISR2  = (TH1D*)DiskFileISR2.Get("h_SigEne");
```

is responsible for retrieving the histograms from the root files. Next, the values of the Higss mass (m_MH) and width (m_GamH) is retrieved from the TDen0sity object:

```
TDensity* Density = new TDensity();
double MH   = Density->m_MH;
double GamH = Density->m_GamH;
```

The class `TDensity` is contained in an appropriate header and class implementation files. This class contains the integrand function and all the relevant physical constants.

In the next part ROOT specific operations are performed in order to format the plots.

The same idea on the large scale is used in the full software – there are more cases for the data generation and more plot functions that prepare customized plots. Fig. 2 describes the general concept of the control flow in the software.

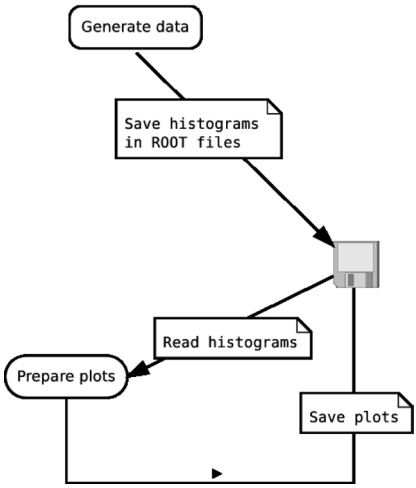


Fig. 2. Flowchart of the software. First, histograms are created and saved in root files on disk. Then they are retrieved, adjusted and saved on disk in PostScript format [14]

## 5. General overview of the program

In this section the description of general software contained in the Full directory is given. The software contains a few parts:

- `Makefile` compatible with GNU make;
- Class `TDensity` in files `TDensity.h` and `TDensity.cxx` that contains the convolution integrands for the Born term and ISR contributions. It inherits from `TFoamIntegrand class` as it is required by FOAM [1,2];
- Main program and function library in `main.cxx` that contains in addition to the main() the functions described in the previous section.

The following commands are defined in `Makefile`:

- `make run` – compile and run the program;
- `make clean` – clean executables;
- `make cleanests` – clean executables, root, pdf and eps files;

- `make Generate-doc` – generate documentation from the code and display the HTML version in the Firefox web browser;

Makefile is set up for parallel compilation on the maximal number of cores available on the computer. This default setup can be overwritten modifying MAKEFLAGS flag at the top of the file or providing `-j [number of processes]` flag during make call from the line command – for details see [7].

The FOAM [1,2] adaptive integrator requires the density distribution. The class `TDensity` implements the density distribution for the sole Born term, and the Born term convoluted with the Gaussian energy spread and with additional ISR corrections [4]. The class can calculate the cross section for electron and muon Higgs boson production $ll \to H$, where l is electron or muon. The type of reaction can be switched by (un)commenting the appropriate `define` statement in the `TDensity.h` file, e.g., default setup for $ee \to H$ reaction is as follows:

```
#define ELECTRON
//#define MUON
```

The class `TDensity` contains fields which are responsible for selecting appropriate distributions:

- `m_ISROn` – 0 – ISR correction is OFF, only the Born shape; 1 – ISR correction ON;
- `m_keyISR` – selects ISR type defined in details in [4]: 0 for the type (a); 1 for the type (b); 2 for the type (c);
- `m_kDim` – dimension of the distribution to integrate. The following combinations are possible:
  - If `m_ISROn=0` then `m_kDim = 2` – convolution of the Gauss distribution with Born term.
  - If `m_ISROn=1` then `m_kDim = 2` – beam/machine energy spread OFF; 3 – beam/machine energy spread ON;
- `m_sigE` – energy spread in GeV if applicable (`kDim = 3`).

For better understanding, a few examples of correct combinations are provided:

- `m_ISROn = 0, m_kDim = 2, m_sigE = 0.0042` – the Born term convoluted with the beam/machine energy distribution modelled by the Gaussian distribution for the energy spread(the Gaussian standard deviation);
- `m_ISROn = 1, m_kDim = 2, m_keyISR=2` – Born with ISR (c);
- `m_ISROn = 1, m_kDim = 3, m_keyISR=2, m_sigE = 0.0042` – Born with with ISR of type (c) (see [4]) convoluted with the Gaussian distribution for the energy spread.

A set of functions that set up all legitimate combinations is delivered. These were used in the simplified example and are in the details described below – their names start with `Make`, e.g., `MakeBorn(...)`.

The main.cxx file contains two types of functions. The first kind generates the cross section distribution and stores them in a root file. These functions set up appropriately the

TDensity object, and use FOAM to integrate the distributions, to create the histograms and save them into a root file. This root file contains two histograms

- h_Ene – contains the cross section as a function of energy;
- h_NORM – is the two bin histogram: the first one at 0.5 contains the integrated luminosity, and the second one at 1.5 that contains the number of events in h_Ene histogram.

The functions are as follows. The first function creates the Born term histogram and saves it in a root file:

```
void MakeBorn( string filename = "BornH" )
```

The second function calculates the Born term convolution with the Gauss distribution that describes beam energy spread, and saves it into root file:

```
void MakeConvBorn( string filename = "histo1",
Double_t sigE = 0.0041, long NevTot =  1000000 )
```

The last important 'production' function calculates the Born term with ISR contribution convoluted with the Gaussian distribution and saves it into root file

```
void MakeISR( string filename = "histo1", Int_t keyISR = 2,
Int_t kDim = 3, Double_t sigE = 0.0041, long NevTot =  1000000 )
```

The second kind of function retrieves the constructed histograms, does the formatting and stores the results into EPS files. These include

```
void plotISRabc( void )
```

which prepares the plots for the three types of the ISR contributions and saves them in the EPS file ISRabc.eps. The next one

```
void plotISR123( void )
```

prepares the plots with the ISR contribution and its convolution for $\delta_E$ = 4.2 MeV and 8MeV. Plots are saved in ISR123.eps. The next two functions

```
void plotBorndelta( void )
void plotISRdelta( void )
```

prepare the plots related to the Born term including the ISR corrections convolution with the Gauss distributions for different machine energy spread values $\delta_E$. The last two functions are responsible for creating and plotting the cross section dependence on the machine energy spread value:

```
int makeISRsigEDistribution( int nbins = 100, long NevTot
=  1000000 )
int plotISRsigEDistribution( int nbins = 100, long NevTot =
1000000 )
```

In particular, the resulting plots can be seen in paper [4], as well as, generated using the software.

## 6. Parallelization with MPI

In this section the description of the parallel version of the programs is provided. They are located in the MPI directory. The description is brief as it is aimed at specialists on parallelization and is only a technical improvement to the program.

The processes of the generation of histograms are separate, independent and require a large amount of time in order to obtain large statistics. Therefore, they are ideally suitable for parallelization. There are different approaches to the issue. Below, a conceptually simple one is applied. It relies on the MPI (Message Passing Interface) [5] idea, and specifically uses the OpenMP [15] technology for C/C++. The choice is dictated by the fact that standard distribution of FOAM [1, 2] integrator does not work when parallelization is applied on the level of threads, therefore, the most common technology for parallelization on the level of processes was selected, i.e., MPI.

MPI delivers the software infrastructure for the communication between the processes by applying the MPI library calls. It is a popular standard in High Performance Computing. The process management is implemented using the well-known producer-consumer design pattern [11]. The number of processes used in the computations is controlled by NOP variable in Makefile.

There are two programs in the delivered software package [6]. The first one contained in the subdirectory `Basic` generates the plots for all considered types of the cross sections and the second one stored in the subdirectory `sigEPlots` generates the plots for the dependence of the cross section on the centre of mass machine energy spread.

In the first program, the histograms are created according to the data in the array `data` defined at the beginning of `main()` function in `main.cxx` files. Every cell of this array is of the struct type `genData` which contains the generation parameters e.g. the machine energy spread or type of ISR correction. Then every process depending on its unique identification number, assigned by MPI interface, generates the histograms for its range of data in the array, see Fig. 3.

The range of indices of the array for given process is derived using its identification number and is performed by rather standard functions

```
int startIndex( int N, int workers, int rank);
int stopIndex( int N, int workers, int rank);
```

where $N$ is the dimension of the array, workers is the number of all processes and rank is the unique rank of the process. Therefore, every process processes initial data from the array data from the indices range returned by the above functions.

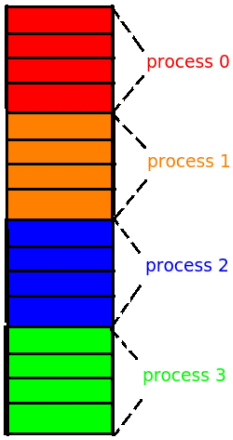


Fig. 3. Data distribution of the array among the processes. Each process, process some range of initial data array, which is determined by unique identification number of the process

A simple check for speeding up the first program is presented in Fig. 4. The calculations were performed using an Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz processor. Each process worked using a single thread. The maximum number of processes is 8; therefore they allowed 7 processes at maximum to run without disturbance from the operating system and other system tasks. Speedup grows almost linearly when the number of cores is increased, which shows that the plateau of Amdahl's law [11] is not reached in the test.

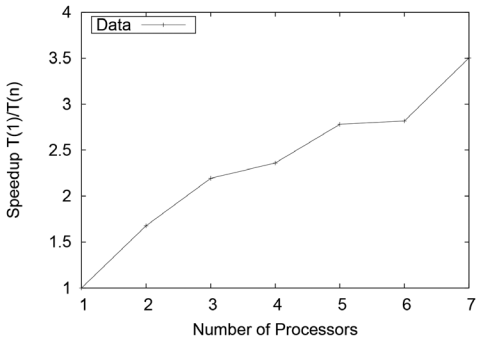


Fig. 4. Speedup for Basic program. The calculations were performed using an Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz processor. Each process worked on a single core and as there were only 8 cores it is therefore inefficient to run more than 7 processes at the same time. Here $T(i)$ is the time of computation for i processes

In the second program from sigEPlots directory every process generates the values of the cross section for a given range of values of the machine energy spread by calculating convolution integrals for different values of $\delta_E$ independently using parallelization. The value of the energy spread for the bin centres are stored in the shared array – every process calculates the cross section for a given range of the energy spread $\delta_E$ at given beam energies $E$.

These computed values are stored by processes in the arrays shared by them. The values are gathered by the main process at the end of the parallel section and then in a single process the plots are generated. In this program it is important that the size of the arrays shared between the processes have to be equal to the multiplicity of the number of processes used in the computation as there is a constraint that jobs are distributed equally among processes. The speedup of the calculations is significant comparing to the single-process program. However, as the number of computations for process have to be a multiplicity of the number of processes there is no precise measure of this speedup in this approach – the size of the job depends on the number of declared processes.

In the implementation, static scheduling of tasks was used, which is the simplest and the most ineffective type of scheduling. Therefore there is much room for improvement.

## 7. Conclusions

A description of the library and program for calculations of the machine energy spread influence on the cross section lineshape of the Higgs boson production in annihilation of lepton pairs was provided. The program structure can be easily adapted to other similar calculations. Also, a simple approach to program parallelization was described.

## References


[1] Jadach S., *Comput. Phys. Commun.*, 152, 2003, 55–100.
[2] Jadach S., *Comput. Phys. Commun.*, 130, 2000, 244–259.
[3] Brun R., Rademakers F., *Proceedings AIHENP'96 Workshop*, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389, 1997, 81–86.
[4] Jadach S., Kycia R.A., *Phys. Lett. B 755*, 2016, 58.
[5] MPI (Message Passing Interface), http://www.mpi-forum.org/.
[6] https://github.com/rkycia/HiggsLineshapeCalculator, http://fizyk.ifpk.pk.edu.pl/ rkycia/ software.html.
[7] GNU Make, https://www.gnu.org/software/make/.
[8] GNU Compiler Collection, https://gcc.gnu.org/.
[9] Doxygen, http://www.stack.nl/ dimitri/doxygen/.
[10] Valgrind, http://valgrind.org/.
[11] Ben-Ari M., *Principles of Concurrent and Distributed Programming*, Pearson; 2nd edition, 200.
[12] Jadach S., Ward B.F.L., Was Z.A., Yost S.A., *Phys. Rev.*, D 94, 074006, 2016.
[13] Jadach S., Ward B.F.L., Was Z., *Phys. Rev.*, D 63, 2001.
[14] Adobe Systems Inc. Addison-Wesley Educational Publishers Inc., Verlag, 1999.
[15] OpenMPI software, https://www.open-mpi.org/