JOINT INSTITUTE FOR NUCLEAR RESEARCH
DZHELEPOV LABORATORY OF NUCLEAR PROBLEMS

FINAL REPORT ON THE
SUMMER STUDENT PROGRAM

GNA: data analysis software for JUNO and Daya Bay experiments

Supervisors:
Maxim Gonchar,
Konstantin Treskov

Student:
Anna Fatkina

Participation period:
June 12 — August 9

Dubna, 2017

Contents

1 Introduction

For the last few decades physicists have been trying to solve the mysteries of neutrino. The first neutrinos were detected in the middle of XX century and there are still a lot of questions that are not answered yet. Scientists suggest that understanding of neutrinos nature will help to explore the Universe secrets.

Neutrino is an elementary particle that interacts with other particles only through weak interactions, which makes it hard to be detected. It is the only truly neutral particle in the Standard Model of elementary particles. The fact that neutrino is a neutral particle opens a door for the intriguing question whether neutrinos and antineutrinos are the same particles or different — Majorana or Dirac particles?

According to the Standard Model of elementary particles there are three types of neutrino: electron, muon and tau neutrino. Those types are usually referred as flavors. Flavor neutrinos have no definite mass.

The phenomenon that neutrino being produced as a neutrino with definite flavor and after propagation may be detected as a neutrino with a *different* flavor is known as *neutrino oscillations*. For example, electron neutrino oscillations into muon and tau neutrino posed a long standing riddle why flux of neutrino from Sun was two times less then expected. Such a process may occur only if flavor neutrinos are composed out of neutrinos with definite mass. It is called neutrino mixing. The flavor and massive neutrinos are related to each other through a rotation in 3-dimensional space:

$$|\nu_\alpha\rangle = \sum_{i=1}^{3} V_{\alpha i} |\nu_i\rangle, \tag{1}$$

where $|\nu_\alpha\rangle$ is a flavor neutrino state, $|\nu_i\rangle$ is a massive neutrino state and $V_{\alpha i}$ is a complex unitary matrix known as a Pontecorvo-Maki-Nakagawa-Sakata (PMNS) matrix. The PMNS matrix is usually parameterized by three angles — θ_{12}, θ_{13}, θ_{23} and one phase — δ_{CP} [1].

The frequency of oscillations are defined by the mass splittings $\Delta m^2_{ij} = m^2_i - m^2_j$ of massive neutrinos. The general formula for oscillation probability reads:

$$\begin{aligned} P(\nu_\alpha \to \nu_\beta) = \delta_{\alpha\beta} &- 4\sum_{i>j} \mathrm{Re}(V^*_{\alpha i} V_{\beta i} V_{\alpha j} V^*_{\beta j}) \sin^2 \frac{\Delta m^2_{ij} L}{4E} \\ &+ 2\sum_{i>j} \mathrm{Im}(V^*_{\alpha i} V_{\beta i} V_{\alpha j} V^*_{\beta j}) \sin^2 \frac{\Delta m^2_{ij} L}{2E}. \end{aligned} \tag{2}$$

The existence of neutrino oscillations is the first indication that Standard Model is not complete and should be extended to explain the nature of neutrino masses and why they are so tiny in comparison with other particles.

Some of the open questions in the neutrino physics that physicists are striving to find an answer to:

- Neutrino mass hierarchy problem or which neutrino is heavier: ν_1 (inverted hierarchy) or ν_3 (normal hierarchy)?
- The absolute scale of neutrino masses is still unknown.
- Neutrino nature — Dirac or Majorana?
- What is the value of the phase δ_{CP}?

To answer these questions huge and complicated detectors are required. And the amount of data to be analyzed is also enormous.

In this work we consider two cutting-edge reactor antineutrino experiments — JUNO and Daya Bay.

1.1 JUNO

JUNO, Jiangmen Underground Neutrino Observatory, is an upcoming neutrino experiment located in the Southern China in Jiangmen province. It will observe the flux of reactor electron antineutrinos. The 20-kilotons liquid scintillator detector will be instrumented with 17000 of 20" PMTs and 25000 3" PMTs that would provide an unprecedented 3% energy resolution. The detector will be located underground on the depth of 700 meters. This is crucial for reducing the backgrounds related to cosmic muons passing through a detector [2]. Two nuclear power plants, Taishan and Yangjiang, located at average distance of 52.5 kilometers from JUNO location would serve as a powerful source of antineutrino producing $\approx 6 \times 10^{21}\,\bar{\nu}_e$ per second.

Such a setup would allow JUNO to cover a wide range of physical topics and searches [3]:

- Determination of the neutrino mass hierarchy at the confidence level of $3-4\,\sigma$. It is the main goal of the experiment;

- Precise measurement of the neutrino mixing parameters and mass splitting with a sub-percent accuracy;
- Measurement of neutrino flux from radioactive decays inside Earth (geoneutrinos) with a record statistics;
- Measurements of solar neutrino flux from ^{8}B and other isotopes;
- Measurements of atmospheric neutrinos;
- Potential measurement of neutrinos from supernova with high statistics;
- Exotic searches such as:
  - Sterile neutrinos;
  - Proton decay;
  - Non-standard interactions of neutrinos;
  - Neutrinos from dark matter annihilation.

The detector construction is underway and JUNO is expected to start data taking in 2020.

1.2 Daya Bay

The Daya Bay Reactor Antineutrino Experiment is the reactor experiment located near the Daya Bay Nuclear Power Plant in Dapeng City in China. The experimental setup consists of 8 identically designed detectors distributed over 3 sites and 6 reactor cores, 2.9 GW of thermal power each as shown at Figure 1.

Each detector contains 20 tons of gadolinium doped liquid scintillator in inner vessel of the detector and 20 tons of non-doped scintillator. Having both near and far detectors in terms of distance to the reactors allows to significantly decrease the correlated uncertainties of antineutrino flux from reactors and of the detection efficiency. High statistics and small backgrounds contamination lead to the currently most precise measurement of mixing angle θ_{13} and mass splitting Δm^2_{32}, Figure 2.

In addition to the oscillation parameters measurements Daya Bay also presents a number of important results:

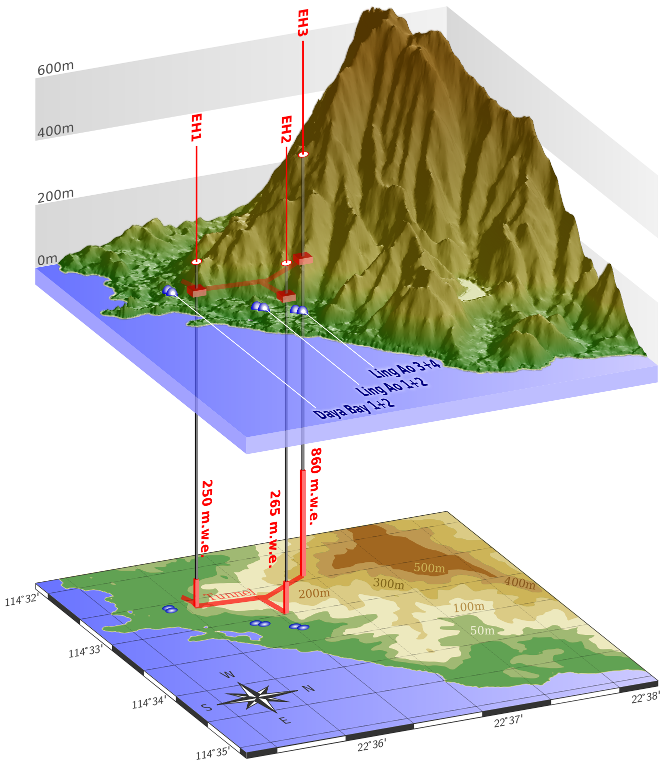


Figure 1: The scheme of Daya Bay detectors and reactors location.

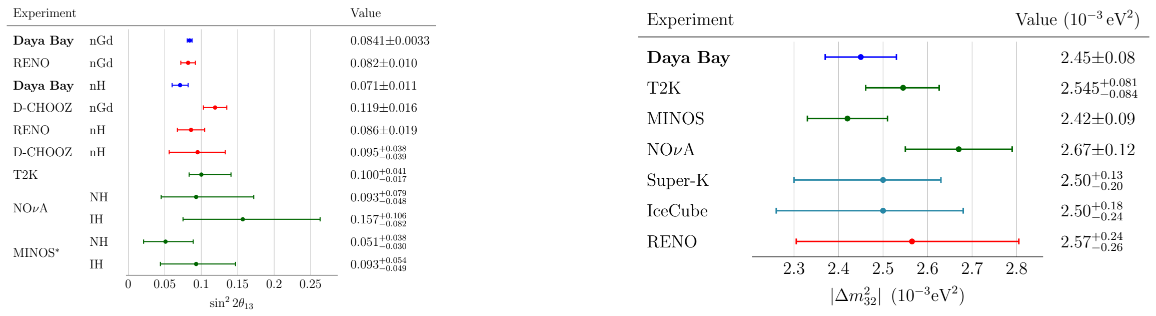


Figure 2: Left: comparison all recent measurements of $\sin^2 2\theta_{13}$. Right: comparison of all recent measurements of Δm^2_{32} [4].

- Study of the evolution of nuclear fuel in time [5];
- Combined search for sterile neutrinos with MINOS experiment [6];
- Study of quantum coherence of neutrino oscillations in wave packet approach [7] and etc.

2 Software

The analysis of experimental data introduces huge computational costs. For example, in the Daya Bay experiment more than 250 parameters are included into the minimization process, even though the majority of them are constrained. And it is just one intermediate step of the analyzing algorithm that is repeated a lot of times.

GNA (Global Neutrino Analysis) [8] is the software for analyzing data of neutrino experiments. Currently it is devoted to the analysis JUNO and Daya Bay. It also planned to include other experiments (accelerator, atmospheric) into analysis. It is designed with having joint analysis of various experiments in mind.

The flexible design is required to perform the data analysis, especially the joint one. GNA framework consists of separate modules that can be combined into the computational chain. The calculation result of each module is cached and recalculated only when it is really needed, i.e. when the parameters or inputs of these modules are modified. Such a design strongly suggests to investigate the possibility of using various kinds of parallel computing technologies.

Modules that represents different approaches to the similar problems are interchangeable and have the same interfaces. The approach is illustrated by the further example of Poisson and χ^2 modules. The mathematical description is presented in the section 3. An example of using it and comparing the results can be found in section 4.1.

There exist another tools for the neutrino analysis, but most of them are proprietary. An example of free software aimed to the similar problem is GLoBES, [9]. This package to date is obsolete, the latest release was presented at 2007.

The user interface of GNA is implemented in Python as it provides some useful features to parse command line commands, draw figures and link modules. Data analysis algorithms are implemented in C++. The interface is linked with C++ backend via PyROOT.

To run GNA it is necessary to call it via python. It works as a framework and can be executed with different parameters depending on the analysis requirements. Here is a short example of script that draws a spectrum with two peaks with different energies and widths and then saves it into file:

```
python gna -- gaussianpeak --name peak1 \
           -- gaussianpeak --name peak2 \
    -- ns --value peak1.E0 2 --value peak1.Width 0.2 \
    -- ns --value peak2.E0 4 --value peak2.Width 0.1 \
    -- spectrum --plot peak1/spectrum --plot peak2/spectrum \
        --savefig task1.png
```

On C++ side the Eigen library is used for working with matrices and vectors, [10]. Eigen is the state-of-the-art library providing useful features for math operations. It is implemented using the expression templates to build expression trees at compile time and to generate custom code to evaluate it. Expression templates allow to intelligently remove temporaries and enable lazy evaluation, when that is appropriate. The library performs its own loop unrolling and vectorization to achieve better performance. Bearing that in mind, Eigen provides easy to use interface and fine optimized computational kernels.

3 Mathematical background

In this section we describe mathematical methods used in considered parts of project. More details about applying it in the context of GNA framework can be found in section 4.

3.1 Gaussian probability density function

Normal (or Gaussian) distribution for the vector $x \in R^k$ can be defined by the probability density function. In case of multivariate normal distribution this function is usually presented the following way:

$$f(x) = \frac{1}{\sqrt{(2\pi)^k|V|}} e^{-\frac{1}{2}(x-\mu)^T V^{-1}(x-\mu)}, \tag{3}$$

where k — the length of vector x, μ — the mean or expectation of the distribution, V — non-singular covariance matrix representing the widths and correlations of vector x.

In case of V being an identity matrix and μ being zero vector it is called standard normal distribution.

3.2 χ^2 function

χ^2 distribution with k degrees of freedom is the distribution of sum of squares of k independent normal variables. The χ^2 function is commonly used in data analysis as experimental data is often close to be normally distributed.

Let x be an experimental data, a vector size of N, μ is the hypothesis that is tested. It depends on K unknown parameters θ and M parameters of model η_0. Lets consider the likelihood function:

$$L(x,\theta,\eta) = \frac{1}{\sqrt{(2\pi)^N |V_{stat}|}} e^{-\frac{1}{2}(x-\mu(\theta,\eta))} \frac{1}{\sqrt{(2\pi)^M |V_\eta|}} e^{-\frac{1}{2}(\eta-\eta_0)^T V_\eta^{-1}(\eta-\eta_0)}, \tag{4}$$

where V_{stat} — matrix of statistical errors of x and V_η is the matrix of η parameter errors.

This function shows a probability of spectrum detection with parameters θ and η in case of normal distribution hypothesis.

In case of linear approximation μ is the sum of normally distributed variables so this function seems the following way:

$$L(x,\theta) = \frac{1}{\sqrt{(2\pi)^N |V|}} e^{-\frac{1}{2}(x-\mu(\theta,\eta_0))^T V^{-1}(x-\mu(\theta,\eta_0))}, \tag{5}$$

$$\chi^2 = (x-\mu(\theta,\eta_0))^T V^{-1}(x-\mu(\theta,\eta_0)), \tag{6}$$

where V is a full covariance matrix includes statistics ans systematic errors.

The best asymptotically unbiased estimation (under some assumptions, see [11]) of the parameters θ is obtained by the maximization of the likelihood function:

$$\hat{\theta} = \arg\max_\theta L(x,\theta). \tag{7}$$

The minimization of χ^2 function, eq. (6), is equivalent to the maximization of likelihood function.

Construction of the confidence intervals for parameters estimated in Gaussian case is straightforward. It is defined by the boundaries of the parameter region inside which

$$\Delta\chi^2 = \chi^2(\theta) - \chi^2(\hat{\theta}) < \alpha_{CL} \tag{8}$$

is hold, where α_{CL} is corresponding quantile of χ^2-distribution. The common choice of $\Delta\chi^2 = 1$ corresponds to the 1 σ confidence interval.

Within the GNA framework χ^2 module is used to estimate the parameters and their uncertainties. Cholesky decomposition of the covariance matrix is used for computational efficiency:

$$V = LL^T, \tag{9}$$

where L is a lower triangular matrix. The χ^2 function can be expressed in the following form:

$$\chi^2 = y^T y, \tag{10}$$

$$y = L^{-1}(x - \mu). \tag{11}$$

More information can be found in [12].

3.3 Poisson probability density function

The probability mass function for Poisson distribution is

$$f(x, \mu) = \frac{(\mu^x e^{-\mu})}{x!}. \tag{12}$$

This means that $f(x, \mu)$ is a probability of observing x events, where μ is the average number of events.

For large values of x Poisson distribution converges to the normal distribution. Therefore, there is no difference what to use in case of high density of events, [13]. If the number of events is low, it is preferable to use Poisson statistic due to the so-called law of small numbers.

Let x and μ be N-dimensional vectors. The expression for Poisson likelihood reads:

$$L(x|\mu) = \prod_{i=1}^{N} \frac{(\mu_i^{x_i} e^{-\mu_i})}{x_i!}. \tag{13}$$

For the extremum finding purpose it can be replaced by a natural logarithm of L since a logarithm is a monotonic function:

$$-2\log L(x|\mu) = -2\sum_{i=1}^{N} (x_i \log(\mu_i) - \mu_i - \log(x_i!)). \tag{14}$$

The maximization of this function yields the same result for parameters estimation as in case of using (13).

3.4 Feldman-Cousins approach

The confidence levels that are derived in conventional methods of parameters estimation, like in (8), may not represent the true confidence interval in a number of situations:

- The systematic errors are not Gaussian. That is quite common case;
- The dependence of the model on the parameters of interest is not linear. Common case: a physical boundary in parameter space, for example $\sin^2 2\theta \leq 1$;
- Small number of events.

In order to construct correct confidence in such a cases two approaches can be used: Bayessian approach that brings subjective choice of priors, and frequentists approach known as Feldman-Cousins approach, [14]. Feldman-Cousins approach consists of the following steps:

- Extensive toy Monte-Carlo simulations of the experiment in every point of parameter space;
- Based on the generated sample the empirical distribution of $\Delta\chi^2$ between best fit and toy models is determined, so all models have to be fitted with the theoretical model;
- Based on that empirical distribution the confidence level is constructed.

It is *extremely computationally costly*, it can require a generation of more then 10000 toy experiments and fitting all of them at each point of the parameter space. One of the ways of accelerating that procedure is described in the section 4.2.

4 Summer Student Program experiences and results

During the first week of the Summer Student Program (SSP) the basics of the GNA project — structure and usage — were studied. The introductory task was to generate the energy distribution of events for the JUNO experiment in hypothesis of normal hierarchy with a fixed value of Δm^2_{ee} and then to see whether it can be fully described in the hypothesis of inverted hierarchy. The quantitative measure of how well the spectra can be distinguished is the difference of minimal values of χ^2 of the fits of both hypotheses to the generated spectrum. The fits have been performed over Δm^2_{ee} parameters.

The profiles of χ^2 versus Δm^2_{ee} can be found in the Appendix 1. On Figure 8a there is a profile of χ^2 versus Δm^2_{ee} in the hypothesis of normal hierarchy. On Figure 8b — the profile of χ^2 versus Δm^2_{ee} in the hypothesis of inverted hierarchy. One can see that minimums of χ^2 reside in a different regions of parameter space and there is a difference between them. Even the best fit of hypothesis of the inverted hierarchy does not fully reproduce the generated spectrum. It is illustrated on the Figure 3. It is impossible for them to be exactly the same, and it is the basis for future neutrino mass hierarchy determination in JUNO.

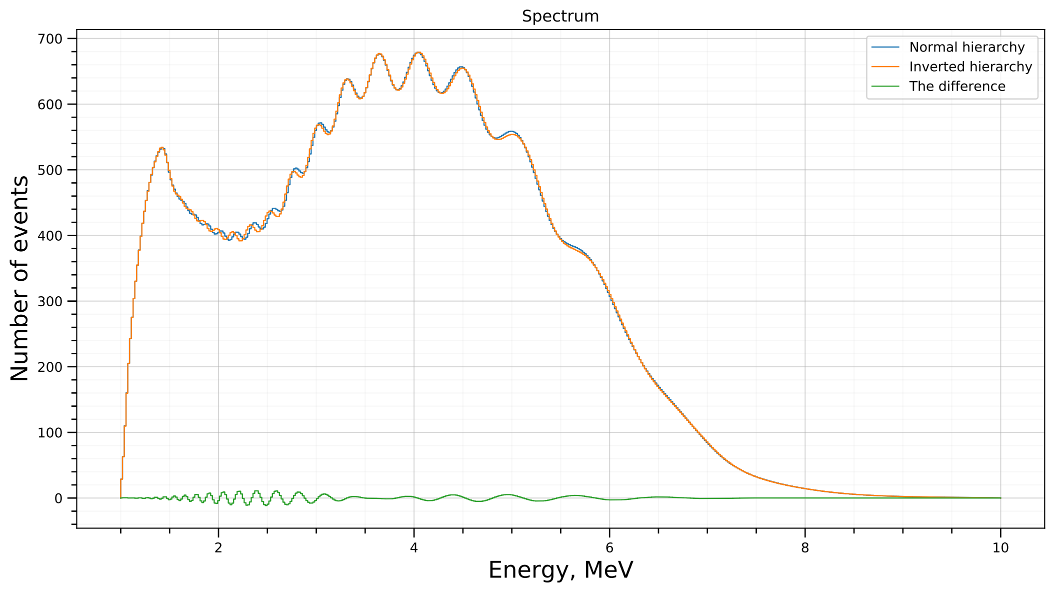


Figure 3: The comparison of the best matching spectra for different hierarchies.

Then there were a number of tasks to extend the framework. The following tasks are considered:

1. Poisson module: returns Poisson log-likelihood value;
2. Grid Filter: computes extended cross-section with a given value;
3. CUDA integration: CUDA and cuBLAS support added to the project;
4. Parallelism: opportunities of using parallel technologies were considered;
5. Unit tests and documentation were added for implemented for implemented modules.

In the GNA project there is a number of user guides that are generated by *Sphinx*. There is also the *doxygen* configuration file that generates documentation based on comments automatically. During the work with the implemented modules doxygen comments were written. After that the corresponding user guide pages were added.

4.1 Poisson

In the GNA there exists module for computing the value of the χ^2 function. The goal was to implement Poisson module to compute log-poisson value. The module should have the interfaces as χ^2 to enable easy substitution between them. The using of the Poisson statistic is more suitable for cases of low density of events, for example, for accelerator neutrino experiments such as NOνA.

The implementation of this module is based on the natural logarithm of Poisson likelihood. As the part of this sum is a factorial, gamma-function was used here to compute the approximate value $\Gamma(n) = (n-1)!$, where $n \in N$. An approximate expression may be used optionally: $\log(n!) \approx n\log(n) - n$.

There was added a set of test scripts to demonstrate the work of Poisson module and compare it to χ^2.

As a theoretical model the Gaussian-shaped peak with a flat background was used. The model is described by the following formula:

$$\frac{dN}{dE} = b + \mu \frac{1}{\sqrt{2\pi w}} \exp \frac{-(E-E_0)^2}{2w^2}, \tag{15}$$

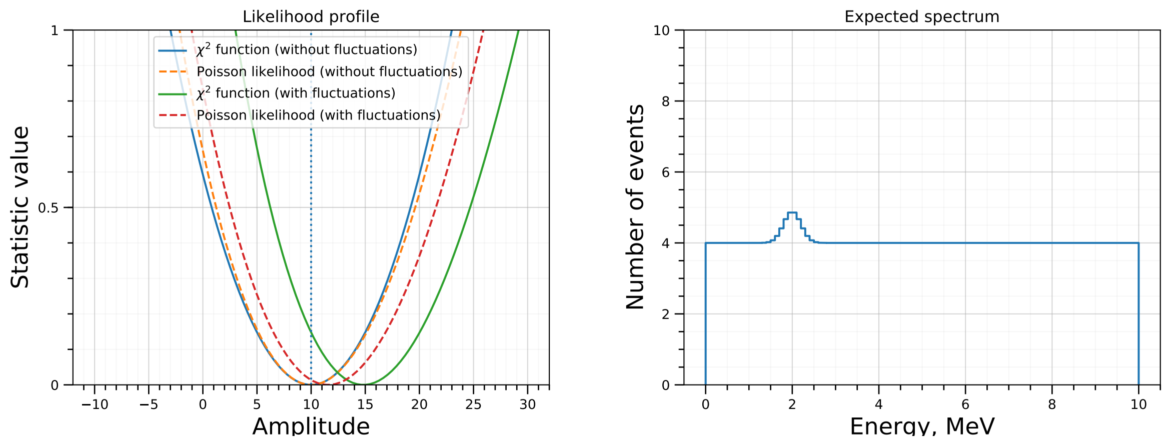


Figure 4: Few events per bin. On the left figure the true value of considered parameter marked by vertical line. On the right figure there is expected Gaussian peak spectrum.

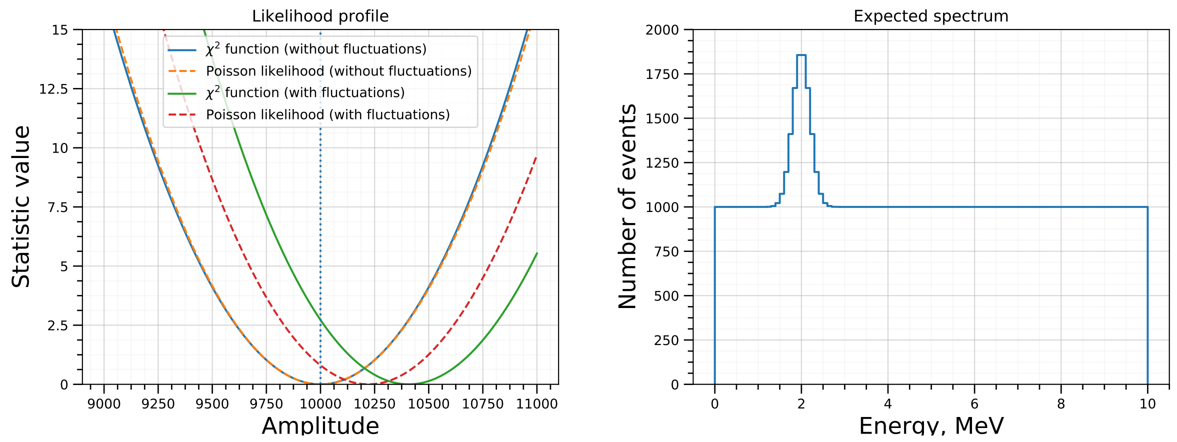


Figure 5: Many events per bin. On the left figure the true value of considered parameter marked by vertical lines. On the right figure there is expected Gaussian peak spectrum.

where: N — number of events; b — constant background, caused by equipment features, cosmic rays, etc.; μ — signal strength. The more μ is, the higher Gaussian peak raises. w — peak width; E_0 — position of the peak that is the position where curve maximum is obtained.

Background, signal strength, peak width and position of the peak can be set from the command line. Parameters that are not included in the formula such as the number of bins and peaks, maximum and minimum allowable values of E, integration order also can be set by user.

By formula (15) expected spectrum was generated. Another spectrum was created to be fit.

The two samples of data were generated:

- Asimov data sample that is basically theoretical predicted spectrum with a fixed parameters of interest;
- A sample obtained from Poisson distribution with Asimov data as mean values.

Using both of generated samples the value of the parameter μ has been estimated with both χ^2 and Poissonian statistics, (15). Statistic profiles are shown at right-hand side of Figures 4 and 5. On the left side figures four curves for different cases described above are presented.

On the Figure 4 there is a Poisson likelihood and χ^2 functions profile with a low density of events. On the Figure 5 there is a profile for the high density of events.

4.2 Grid Filter

`Grid Filter` is a helper module that will be integrated into Feldman-Cousins algorithm in the future. It is necessary to optimize computations as Feldman-Cousins is a computationally expensive algorithm. The idea is to discard points with a low probability from the analysis. That is the based on the observation that quite often Feldman-Cousins go quite close to conventional χ^2 one, Figure 6.

Here is a description of this filter. Consider $z = f(x, y)$ being a two-dimensional function. A 2-dimensional array consisting of z values is expected as input for the `Grid Filter` module. The range of x and y can be set by user. There are also can be set some input parameters: val, err. The API of this module provides the following possibilities:

1. Compute the cross-section. Output is a mask with 1 for the points at $z =$ val $\pm$ err and 0 for the other (discarded) points;

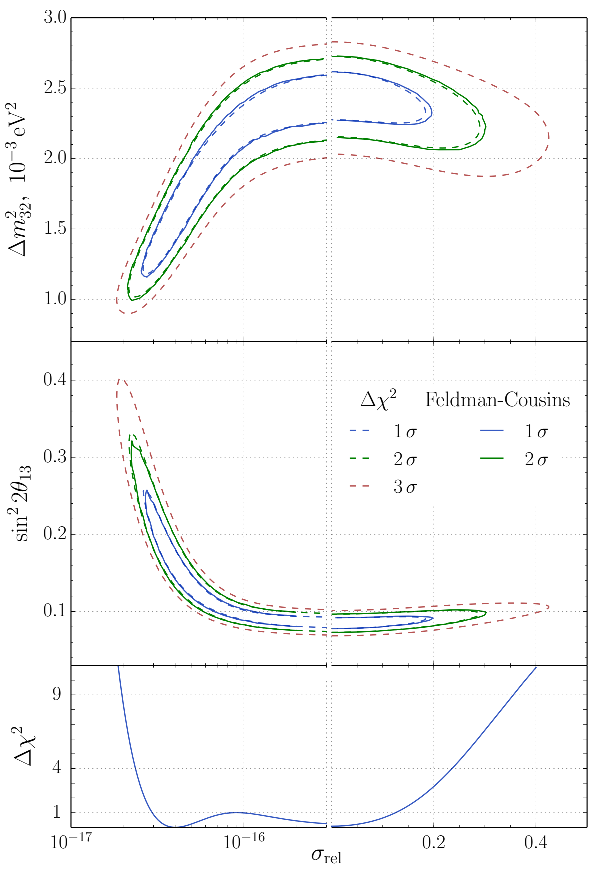


Figure 6: The Feldman-Cousins and conventional $\Delta\chi^2$ contours from [7]. It illustrates that quite commonly Feldman-Cousins contours are quite close to conventional $\Delta\chi^2$.

2. Compute extended cross-section:

   - With fixed deviation from the original cross-section set by user;
   - With the automatically computed deviation depending on the average value of gradient vectors length across the original cross-section;
   - With the automatically computed deviation depending on the gradient vector length at every point.

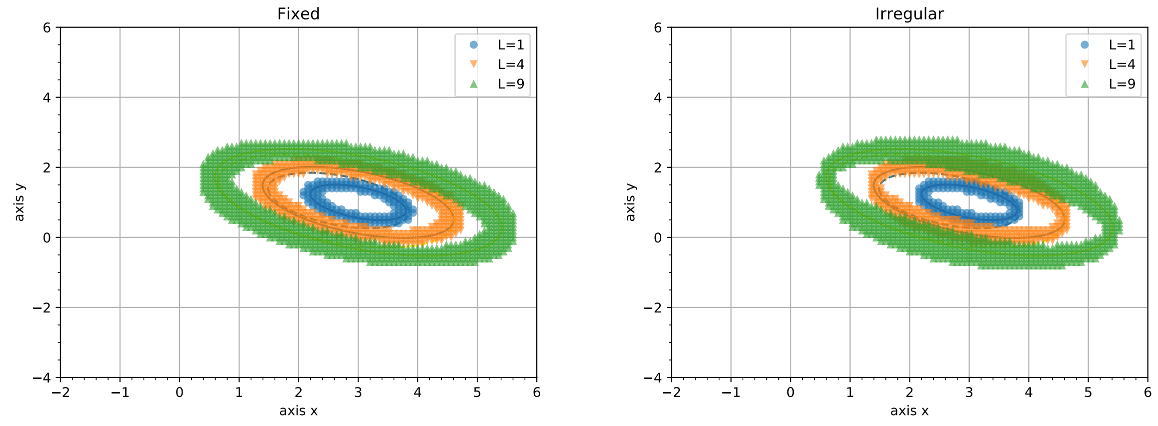


Figure 7: Left: Fixed contour width. Right: Irregular contour width. L is the value of function, around which the points are distributed.

The difference between cross-sections with the fixed and irregular width of extended contour is shown at Figure 7.

There also exist special parameters like *gradient influence* and *initial deviation* that are just multipliers for gradient value at the formulae of deviation computing. Parameter *tolerance* sets the allowable error at $z = \text{val} \pm \text{err}$ expression. It provides more flexible interface but does not make it more difficult to use as there are default values set. After initialization of the algorithm parameters described above can not be altered.

There are more examples in Appendix 2 and Appendix 3 of usage of this module. In the Appendix 2 it is shown how does the gradient multiplier influences the width of extended cross-section contour. Appendix 3 demonstrates the influence of tolerance parameter. Considered parameters are set from the command line and parsed by python test script.

4.3 CUDA integration

Graphical processing units (GPU) are widely used today not only for image processing but also in science applications. GPUs are known for the huge number of threads so it is appropriate for SIMD-oriented (Single Instruction Multiple Data) codes, such as independent operations over elements of arrays.

CUDA Toolkit (Compute Unified Device Architecture) is a powerful tool that helps to port code into NVIDIA GPGPUs (General-purpose GPU) and provides extensions for C and C++, [15]. CUDA architecture allows to accelerate software by effective using of GPU threads and memory access. Threads are logically unified into blocks that are, in turn, unified into grids. Sizes of grids and blocks can be set from the host code. There is a special memory hierarchy that provides some types of it with distinct permissions.

CUDA support was added to the project. There was implemented an example of using CUDA-based library cuBLAS (Basic Linear Algebra Subprograms). In the future it is planned to add GPU-based implementation of some algorithms and a possibility to choose whether GPU have to be used or not.

There is an API for basic math operations including matrices operations. By design the GNA is expected to deal with huge dense matrices. This suggests that using cuBLAS can lead to increase of the computational efficiency under certain conditions that are to be studied.

4.4 Parallelism

There is no custom multithreading in the current GNA implementation. Better performance is achieved by caching and highly optimized third-party libraries and compiler optimizations only. It was decided to find the code fragments that can run in parallel or be modified for it.

The first problem we faced was the absence of unit tests at the C++ side. Profiling through the python interface creates a lot of intermediate calls in call graph that hide actual computations. For that reason C++ unit tests implementation for all the computationally expensive modules is one of the tasks ahead.

The possibility of using multi-core CPUs (Central Processing Unit) and GPUs to accelerate the existing code is being considered now. We plan to use not only standard C++ multithreading for CPUs but also such external standards as OpenMP, [16]. It provides a convenient interface that will help to keep code clean. Furthermore, it is supported by the most compilers.

The main goal is to adapt GNA to heterogeneous computational systems and to provide to the user an interface to choose the desired architecture.

5 Conclusion

In this report modules that were implemented during the SSP (`Poisson` and `Grid Filter`) are described. A short introduction about the GNA framework, its mathematical and physical background is presented.

`Poisson` module was integrated into existing code base. Python interface for it was implemented. It is currently possible to use it along with χ^2 module that is a part of GNA implemented earlier.

`Grid Filter` module was added like standalone module. The reason for this is that `Grid Filter` is a part of a bigger algorithm not yet fully implemented.

For both modules python tests and detailed documentation were written.

We also analyzed existing code base from a perspective of better performance. We are planning to port a part of GNA code to GPGPU by using CUDA and CUDA-based libraries such as cuBLAS.

CuBLAS support was added to project and it is planned to add a possibility to port some SIMD-oriented code fragments to GPGPU upon user requests. We plan to modify project to adapt it for heterogeneous systems as it seems to be useful to deal with huge datasets.

It is also planned to implement Feldman-Cousins algorithm using existing `Grid Filter` to filter out the points with low probability to find the contour.

Acknowledgements

I would like to thank the organizers of Summer Student Program for providing education opportunities and valuable experience of being a part of JINR community. Thanks to University Center stuff for their assistance in solving all the issues. Special thanks to JINR for the financial support.

I am also very grateful to my supervisors Maxim Gonchar and Konstantin Treskov for their support during the program. They have always been patient and kind. Both of them definitely have qualities of an excellent leaders and are very kind colleagues.

Special thanks to my university and, in particular, to my academic adviser A. B. Degtyarev who made it possible for me to participate in SSP this summer.

References

[1] Carlo Giunti and Chung W. Kim. *Fundamentals of Neutrino Physics and Astrophysics*. 2007. ISBN: 9780198508717.

[2] Zelimir Djurcic et al. "JUNO Conceptual Design Report". In: (2015). arXiv: `1508.07166 [physics.ins-det]`.

[3] Fengpeng An et al. "Neutrino Physics with JUNO". In: *J. Phys.* G43.3 (2016), p. 030401. DOI: `10.1088/0954-3899/43/3/030401`. arXiv: `1507.05613 [physics.ins-det]`.

[4] Feng Peng An et al. "Measurement of electron antineutrino oscillation based on 1230 days of operation of the Daya Bay experiment". In: *Phys. Rev.* D95.7 (2017), p. 072006. DOI: `10.1103/PhysRevD.95.072006`. arXiv: `1610.04802 [hep-ex]`.

[5] F. P. An et al. "Evolution of the Reactor Antineutrino Flux and Spectrum at Daya Bay". In: *Phys. Rev. Lett.* 118.25 (2017), p. 251801. DOI: `10.1103/PhysRevLett.118.251801`. arXiv: `1704.01082 [hep-ex]`.

[6] P. Adamson et al. "Limits on Active to Sterile Neutrino Oscillations from Disappearance Searches in the MINOS, Daya Bay, and Bugey-3 Experiments". In: *Phys. Rev. Lett.* 117.15 (2016). [Addendum: Phys. Rev. Lett.117,no.20,209901(2016)], p. 151801. DOI: `10.1103/PhysRevLett.117.151801, 10.1103/PhysRevLett.117.209901`. arXiv: `1607.01177 [hep-ex]`.

[7] Feng Peng An et al. "Study of the wave packet treatment of neutrino oscillation at Daya Bay". In: (2016). arXiv: `1608.01661 [hep-ex]`.

[8] *GNA repository*. `https://git.jinr.ru/gna/gna`.

[9] Patrick Huber et al. "New features in the simulation of neutrino oscillation experiments with GLoBES 3.0: General Long Baseline Experiment Simulator". In: *Comput. Phys. Commun.* 177 (2007), pp. 432–438. DOI: `10.1016/j.cpc.2007.05.004`. arXiv: `hep-ph/0701187 [hep-ph]`.

[10] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. http://eigen.tuxfamily.org. 2010.

[11] Luca Lista. “Practical Statistics for Particle Physicists”. In: *2016 European School of High-Energy Physics (ESHEP 2016) Skeikampen, Norway, June 15-28, 2016*. 2016. arXiv: `1609.04150 [physics.data-an]`. URL: `https://inspirehep.net/record/1486520/files/arXiv:1609.04150.pdf`.

[12] Maxim Gonchar. “ Measurement of the neutrino mixing angle θ_{13} and of the neutrino mass splitting Δm^2_{32} in the Daya Bay experiment”. PhD thesis. JINR, 2017.

[13] Jacquelyn T McQueston Lawrence M Leemis. “Univariate Distribution Relationships”. In: *The American Statistician* 62.2 (2008), pp. 45–53. DOI: `10.1198/000313008x270448`.

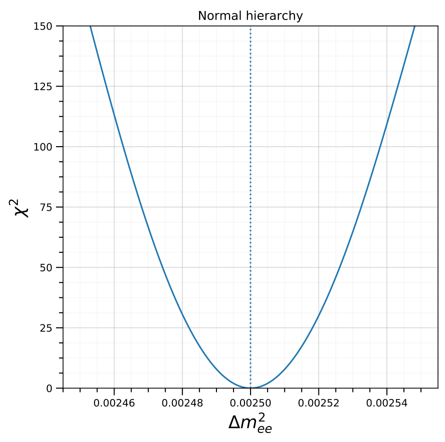
[14] Gary J. Feldman and Robert D. Cousins. “A Unified approach to the classical statistical analysis of small signals”. In: *Phys. Rev.* D57 (1998), pp. 3873–3889. DOI: `10.1103/PhysRevD.57.3873`. arXiv: `physics/9711021 [physics.data-an]`.

[15] CUDA Toolkit Documentation. *Programming Guide*. 2016.

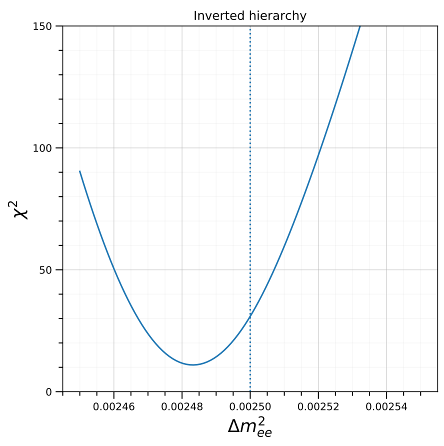
[16] Michael Klemm and Christian Terboven. “Full Throttle: OpenMP* 4.0”. In: *The Parallel Universe* 16 (2013), pp. 6–16.

Appendix 1

GNA test: fitting Δm^2_{ee} value in the hypothesis of normal and inverted hierarchy. The true hierarchy is normal.



(a) Normal



(b) Inverted

Figure 8: Profile of χ^2 in the hypothesis of normal and inverted hierarchy. The true value is marked by vertical line.

Appendix 2

Grid Filter tests for a different gradient influence values on three levels.

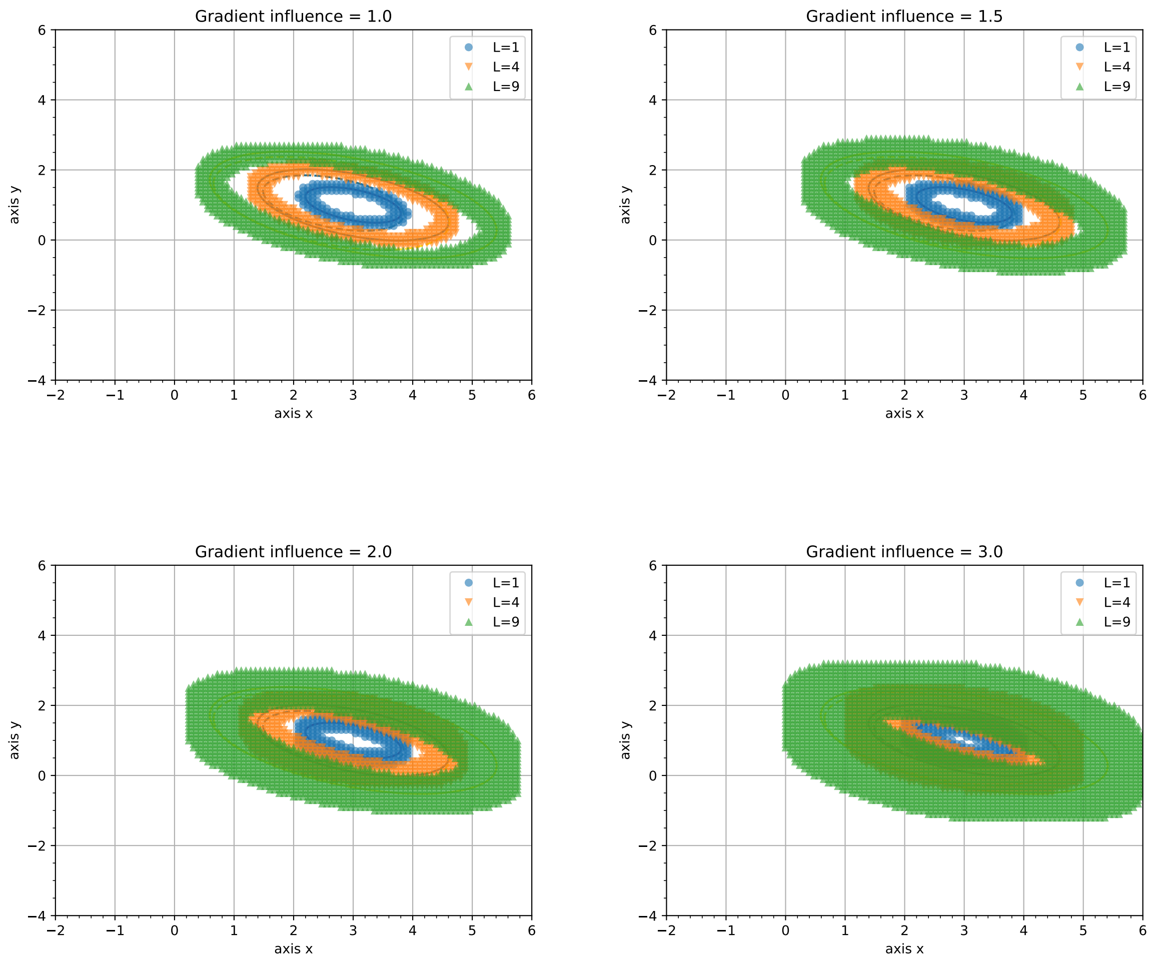


Figure 9: Test for different *gradient influence* parameter values. L is the value of tested function

Appendix 3

Grid Filter tests for a different tolerance values on three levels.

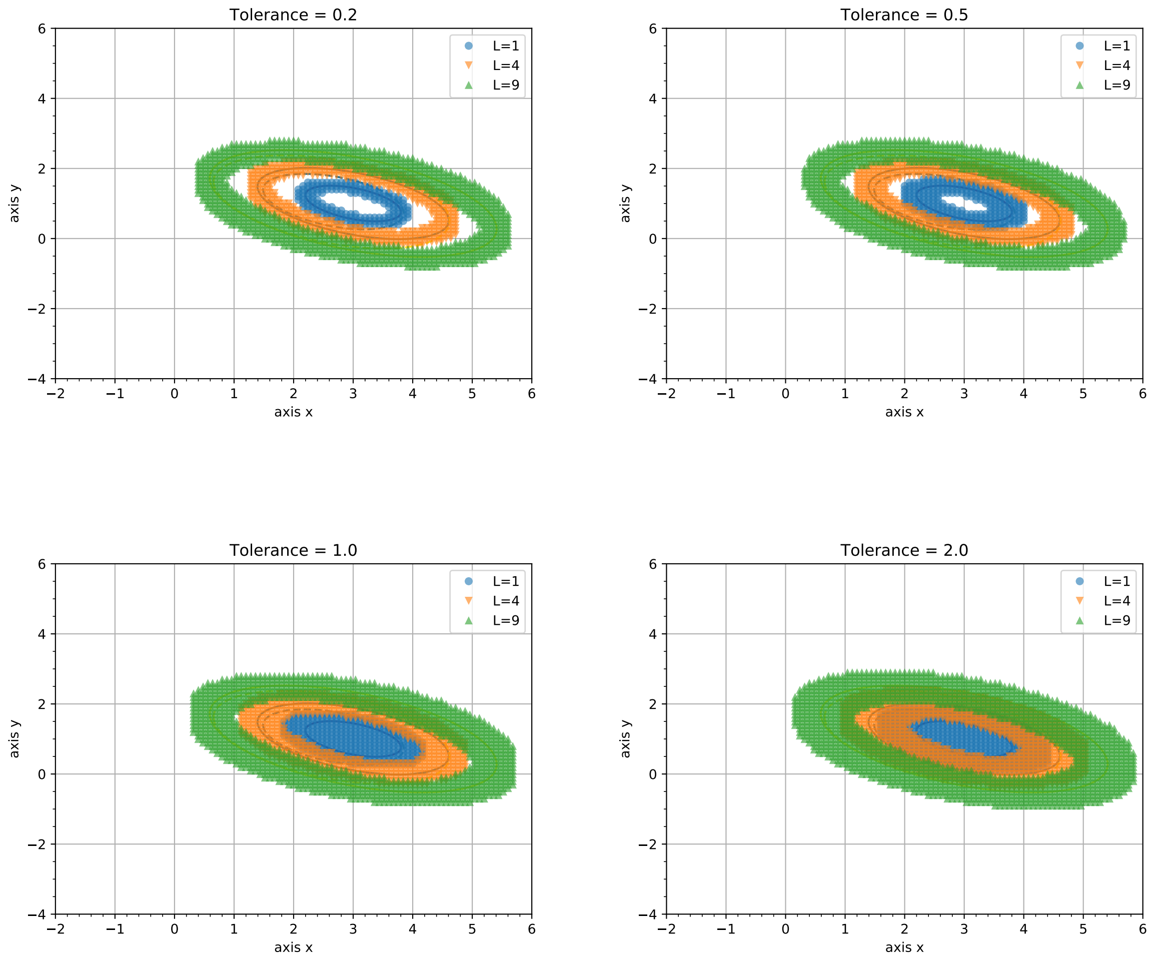


Figure 10: Test for different *tolerance* parameter values. L is the value of tested function