Analysis of ν_μ Disappearance in SBN Detector

Castaly Fan

October 2019

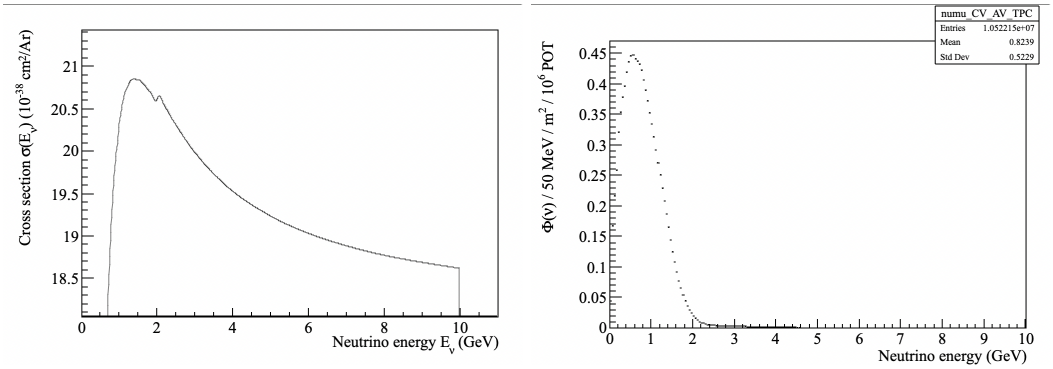
1 Background

For searching sterile neutrinos, one way is to look over the disappearence (reduction) of muon neutrinos (ν_μ) in a muon neutrino beam. The data of detector properties we have is shown as below:

Table 1: Data of detector properties

Detector	Baseline	Active LAr mass	POT (protons on target)
SBND	110m	112 tons	6.2×10^{20}
MicroBooNE	470m	89 tons	13.2×10^{20}
ICARUS	600m	476 tons	6.2×10^{20}

The cross section $\sigma(E_\nu)$ for CCQE ν_{mu} interactions on ^4Ar is shown as the left figure below. And the initial neutrino flux $\phi(E_\nu)$ without oscillations at MicroBooNE is shown as the right figure below.



(a) cross section (b) flux at MicroBooNE

Figure 1: The figure of cross section and flux.

Given the number of argon nuclei N_{Ar}, we can calculate the event rate

$$N_{ev}(E_\nu) = \sigma(E_\nu) \times N_{Ar} \times [\phi(E_\nu) \times \Delta E_\nu] \times POT \tag{1}$$

Armed with the equation 1 and the data of Table 1, we can start the task step by step.

2 Histograms of the Event Rate

Equation 1 shows that the event rate depends on the number of 4_{Ar}, which can be expressed as

$$N_{Ar} = \frac{M \times N_A}{W} \tag{2}$$

where M is the mass in gram, W is the atomic weight, and N_A is the number of nuclei per mole (namely, 6.02×10^{23}). Now, let us consider N_{Ar} of each detectors. Given the LAr mass of Table 1, we obtain:

- 1.6856×10^{30} for SBND
- 1.3395×10^{30} for MicroBooNE
- 7.1638×10^{30} for ICARUS

Then, we can create the event rate's histogram according to Equation 1.

Back to the Figure 1, the cross section for each detector is the same, whereas the curve of the flux would be scaled by the factor $\frac{1}{r^2}$ since the flux falls off with the distance r from the target. For instance, given that the flux at MicroBooNE depends on the distance 470 m, the flux at SBND would be scaled by $(470/110)^2$ m and the flux at ICARUS would be scaled by $(470/600)^2$ m. Given that Equation 1, the unit should be considered in order to create the event rate's histogram. So we can simply write down the event rate equation in terms of the units:

$$[N] = [\frac{cm^2}{Ar}] \times [Ar] \times [\frac{1}{MeV \cdot POT \cdot m^2}] \times [MeV] \times [POT] \tag{3}$$

Here, $[N]$ is denoted by the number of event. Then we have to notice the unit should be normalized, in particular, $[cm^2]$ is for cross section whereas $[m^2]$ is for flux.

To create the plot for event rate histograms, we can log into the ROOT framework and run the following command (some "//" means the note for the command line below):

```
  void draw_histogram(TH1D* h, TGraph* cross_section, double parameters)
{
        std::cout << parameters << std::endl;
        // for every bin in x-axis (GeV)
        for (int i = 1; i <= h->GetNbinsX(); ++i)
        {
                float binCenter = h->GetBinCenter(i);

                // value of cross section
                float y1 = cross_section->Eval(binCenter);

                // value of flux
                float y2 = h->GetBinContent(i);

                // eventrate formula
                float N = y2 * y1 * parameters;
```

```
                // assign the y value back to the plot
                h->SetBinContent(i,N);

                // debug output
                // std::cout << N << std::endl;
        }
        h->GetYaxis()->SetTitle("Number of Events / GeV");
        h->Draw("SAME");
        std::cout << h->Integral("width") << std::endl;
}

void eventrate()
{
        // find the directory where cross section file is
        TFile* cross_section_file = TFile::Open("cross_section.root");
        cross_section_file->ls();

        // read the cross section file
        TGraph* cross_section = (TGraph*) cross_section_file->Get("qel_cc_n;1");

        // draw the cross section figure
        // cross_section->Draw();

        // find the directory where flux file is
        TFile* flux = TFile::Open("flux.root");
        flux->ls();

        // read the flux file
        TH1D* h = (TH1D*) flux->Get("numu_CV_AV_TPC");

        // create three clones for three cases
        TH1D* h_sbnd = (TH1D*) h->Clone("h_sbnd");
        TH1D* h_microboone = (TH1D*) h->Clone("h_microboone");
        TH1D* h_icarus = (TH1D*) h->Clone("h_icarus");

        draw_histogram(h_sbnd, cross_section,
                    1E-38/1E4*1.6856E30/0.05*6.2E20/1E6*(470.0/110)*(470.0/110));
        draw_histogram(h_microboone, cross_section,
                    1E-38/1E4*1.3395E30/0.05*13.2E20/1E6);
        draw_histogram(h_icarus, cross_section,
                    1E-38/1E4*7.1638E30/0.05*6.2E20/1E6*(470.0/600)*(470.0/600));
}
```

As a result, the histogram of event rate for each detector can be created as Figure 2.

By calculating the total number of events according to the Figure2, we can obtain the value 3.13398×10^6, which is almost matched with the number of ν_μ events during $\nu_\mu n \to \mu^- p$ process based on the SBN proposal (namely, 3,122,600).

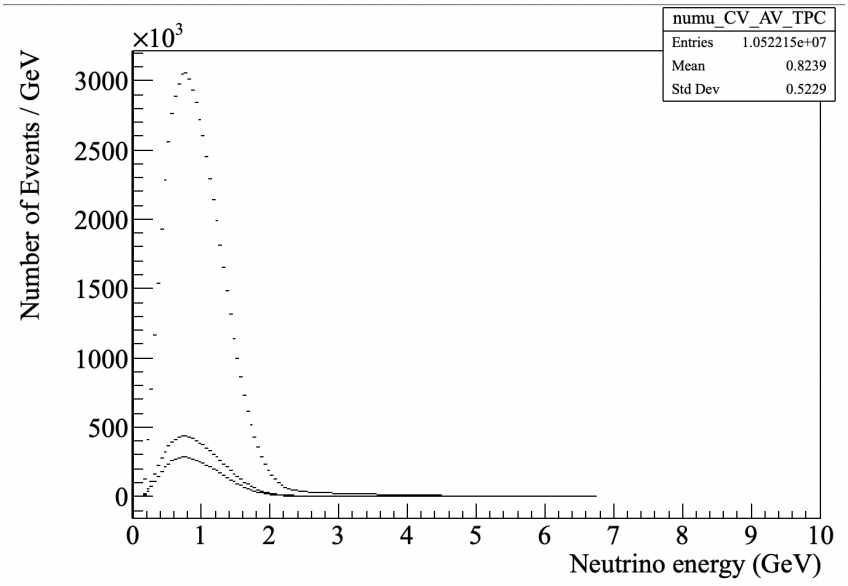


Figure 2: Event rate's histogram. The curves from up to down are respectively corresponding to SBND, ICARUS, and MicroBooNE.

3 Neutrino Oscillations

To explore the disappearance of ν_μ, we have to know the probability where ν_μ remain, denoted by $P_{\nu_\mu \to \nu_\mu}$. In general, if $P_{\nu_\mu \to \nu_x}$ means the probability that ν_μ turn into ν_e, ν_τ, or ν_s (sterile neutrinos), then $P_{\nu_\mu \to \nu_\mu}$ would be reasonably equal to $1 - P_{\nu_\mu \to \nu_x}$. Specifically, we have a formula describing ν_μ disappearance as below:

$$P^{3+1}_{\nu_\mu \to \nu_\mu} = 1 - \sin^2(2\theta_{\mu\mu}) \sin^2\left(\frac{\Delta m^2 L}{4E_\nu}\right) \tag{4}$$

The equation above is taken by natural units. Thus, we can write down the form in SI units:

$$P^{3+1}_{\nu_\mu \to \nu_\mu} = 1 - \sin^2(2\theta_{\mu\mu}) \sin^2\left(1.27\frac{\Delta m^2 L}{E_\nu}\right) \tag{5}$$

where θ is the mixing angle, L is the oscillation distance in kilometers, and E is the neutrino energy in GeV. Setting $\sin^2(2\theta_{\mu\mu}) = 0.1$ and $\Delta m^2 = 1eV$, the factor of probability can be simply written as

$$P^{3+1}_{\nu_\mu \to \nu_\mu} = 1 - 0.1 \sin^2\left(1.27\frac{L}{E_\nu}\right) \tag{6}$$

, which shows the probability of oscillations depends on the distance L and the neutrinos energy E_ν. To create the event rate histogram with neutrino oscillations, we can consider the factor of equation 6. The distance L is corresponding to each detector's baseline (see Table 1), as for the energy E_ν, it is given by the flux histogram as Figure 1.

The command here is slightly different from the first one, that is, we have introduced "baseline" as a new parameter:

```
  void draw_histogram(TH1D* h, TGraph* cross_section,
                      double parameters, double baseline)
{
        std::cout << parameters << std::endl;
        for (int i = 1; i <= h->GetNbinsX(); ++i)
        {
                double binCenter = h->GetBinCenter(i);
                double y1 = cross_section->Eval(binCenter);
                double y2 = h->GetBinContent(i);

                // eventrate formula
                double S = sin(1.27/1E3 * baseline/binCenter);
                //std::cout << i << ": " << y2 << std::endl;
                double N = (1.0 - (0.1) * S * S )* y1 * y2 * parameters;

                h->SetBinContent(i,N);
                // std::cout << N << std::endl;
        }
        h->GetYaxis()->SetTitle("Number of Events / GeV");
        h->Draw("SAME");
        std::cout << h->Integral("width") << std::endl;
}

void eventrate2()
{
        TFile* cross_section_file = TFile::Open("cross_section.root");
        cross_section_file->ls();
        TGraph* cross_section = (TGraph*) cross_section_file->Get("qel_cc_n;1");
        // cross_section->Draw();

        TFile* flux = TFile::Open("flux.root");
        flux->ls();
        TH1D* h = (TH1D*) flux->Get("numu_CV_AV_TPC");

        TH1D* h_sbnd = (TH1D*) h->Clone("h_sbnd");
        TH1D* h_microboone = (TH1D*) h->Clone("h_microboone");
        TH1D* h_icarus = (TH1D*) h->Clone("h_icarus");

        draw_histogram(h_sbnd, cross_section,
                1E-38/1E4*1.6856E30/0.05*6.2E20/1E6*(470.0/110.0)*(470.0/110.0),
                110.0);
        draw_histogram(h_microboone, cross_section,
                1E-38/1E4*1.3395E30/0.05*13.2E20/1E6,
                470.0);
        draw_histogram(h_icarus, cross_section,
                1E-38/1E4*7.1638E30/0.05*6.2E20/1E6*(470.0/600)*(470.0/600),
                600.0);
}
```

Now, we can obtain the event rate histogram with oscillations, which is shown as Figure 3.

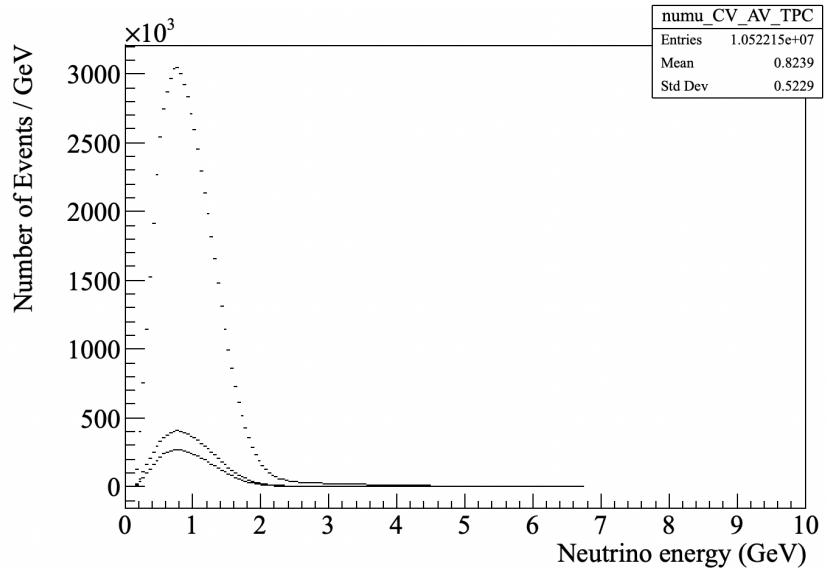


Figure 3: Event rate's histogram with oscillations. The curves of SBND, ICARUS, and MicroBooNE are same as Fig 2, whereas have subtle differences between both diagrams.

By looking at the Figure 3 roughly, it is hard to see obvious differences between that with the one without oscillations (Figure 2). However, there are actually few subtle differences between both histograms. Now what we have to do is to analyze the details of both curves via statistical ways.

4 Statistical Testing

Statistically, to do data fitting work, we need to use least squares in order to minimize the sum of squared errors and residuals. Consider a function $y_i, i = 1, ..., N$ as a function of another variable x_i without error. Each y_i has a different unknown mean λ_i and a known variance σ_i^2. Then, suppose the true value λ is the function of x and depends on unknown parameters $\theta = (\theta_1, ..., \theta_m)$, that is, $\lambda = \lambda(x; \theta)$. Now, consider the $\chi^2(\theta)$ quantity:

$$\chi^2(\theta) = \sum_{i=1}^{N} \frac{(y_i - \lambda(x_i; \theta))^2}{\sigma_i^2} \tag{7}$$

In other words, $y_1, ..., y_N$ are measured with errors $\sigma_1, ..., \sigma_N$ at each $x(x_1, ..., x_N)$ value without errors. The true value $\lambda_i = \lambda(x_i; \theta)$, where θ is adjusted to minimize the value of χ^2 given by the equation above.

Assume un-oscillated event rate histogram is y_i and oscillated one is another function λ. For the variance σ_i^2, consider it as λ^2. Thus, the chi-squared value of one detector can be simply obtained from:

$$\chi^2 = \sum_{i=1}^{N} \frac{(y_i - \lambda_i)^2}{\lambda_i} \tag{8}$$

As for the total chi-squared value for three detectors can be written as the sum

$$\chi^2_{total} = \chi^2_a + \chi^2_b + \chi^2_c \tag{9}$$

where the index a, b, and c can be represented as the detector SBND, MicroBooNE, and ICARUS respectively. To achieve that, we need to introduce the parameter s2 ($sin^2\theta$) and m (mass):

```
void draw_histogram(TH1D* h, TGraph* cross_section, double parameters,
                    double baseline, double s2, double m)
{
        std::cout << parameters << std::endl;
        for (int i = 1; i <= h->GetNbinsX(); ++i)
        {
                double binCenter = h->GetBinCenter(i);
                double y1 = cross_section->Eval(binCenter);
                double y2 = h->GetBinContent(i);
                double S = sin(1.27/1E3 * m * baseline/binCenter);
                //std::cout << i << ": " << y2 << std::endl;
                double y = y1 * y2 * parameters;
                double N = (1.0 - s2 * S * S )* y;
                h->SetBinContent(i,N);
                // std::cout << N << std::endl;
        }
        h->GetYaxis()->SetTitle("Number of Events / GeV");
        h->Draw("SAME");
        std::cout << h->Integral("width") << std::endl;
}

double chi2(TH1D* h, TH1D* g)
{
        double chi_sq = 0.0;
        //for (int i = 1; i <= h->GetNbinsX(); ++i)
        for (int i = 1; i <= 135; ++i)
        {
                double y = h->GetBinContent(i);
                double lambda = g->GetBinContent(i);
                if (lambda == 0) continue;
                double chi_sq_i = (y-lambda)*(y-lambda)/lambda;
                std::cout << "#" << i << "y:" << y << ", lambda:"
                          << lambda << std::endl;

                chi_sq = chi_sq + chi_sq_i;
        }
        return chi_sq;
}

void eventrate_chi()
{
        TFile* cross_section_file = TFile::Open("cross_section.root");
```

```
    cross_section_file->ls();

    TGraph* cross_section = (TGraph*) cross_section_file->Get("qel_cc_n;1");
    // cross_section->Draw();

    TFile* flux = TFile::Open("flux.root");
    flux->ls();
    TH1D* h = (TH1D*) flux->Get("numu_CV_AV_TPC");

    TH1D* h_sbnd = (TH1D*) h->Clone("h_sbnd");
    TH1D* h_microboone = (TH1D*) h->Clone("h_microboone");
    TH1D* h_icarus = (TH1D*) h->Clone("h_icarus");

    draw_histogram(h_sbnd, cross_section,
            1E-38/1E4*1.6856E30/0.05*6.2E20/1E6*(470.0/110.0)*(470.0/110.0),
            110.0, 0, 1);
    draw_histogram(h_microboone, cross_section,
            1E-38/1E4*1.3395E30/0.05*13.2E20/1E6,
            470.0, 0, 1);
    draw_histogram(h_icarus, cross_section,
            1E-38/1E4*7.1638E30/0.05*6.2E20/1E6*(470.0/600)*(470.0/600),
            600.0, 0, 1);

    TH1D* h_sbnd2 = (TH1D*) h->Clone("h_sbnd2");
    TH1D* h_microboone2 = (TH1D*) h->Clone("h_microboone2");
    TH1D* h_icarus2 = (TH1D*) h->Clone("h_icarus2");

    draw_histogram(h_sbnd2, cross_section,
            1E-38/1E4*1.6856E30/0.05*6.2E20/1E6*(470.0/110.0)*(470.0/110.0),
            110.0, 0.001, 1);
    draw_histogram(h_microboone2, cross_section,
            1E-38/1E4*1.3395E30/0.05*13.2E20/1E6,
            470.0, 0.001, 1);
    draw_histogram(h_icarus2, cross_section,
            1E-38/1E4*7.1638E30/0.05*6.2E20/1E6*(470.0/600)*(470.0/600),
            600.0, 0.001, 1);

    double chi2_sbnd = chi2(h_sbnd, h_sbnd2);
    double chi2_microboone = chi2(h_microboone, h_microboone2);
    double chi2_icarus = chi2(h_icarus, h_icarus2);

    std::cout << "chi2_sbnd: " << chi2_sbnd << std::endl;
    std::cout << "chi2_microboone: " << chi2_microboone << std::endl;
    std::cout << "chi2_icarus: " << chi2_icarus << std::endl;

}
```

Then we can print out the χ^2 value for each detector. we get:

- SBND: $\chi^2_a = 0.278707$
- MicorBooNE: $\chi^2_b = 1.54599$
- ICARUS: $\chi^2_c = 3.45765$

So that the total χ^2 value is $\chi^2_{total} = \chi^2_a + \chi^2_b + \chi^2_c = 5.282347$, which shows the difference between null hypothesis (un-oscillated) with oscillated histograms.

5 The Heat Map Plot

Given that the χ^2 value for each detector, we can make a heat map plot to show whether the experiment can be consistent with or rule out statistically. To realize it, we must make a 2-D plot, that is, TH2D in ROOT framework. Armed with the event rate histograms as well as the χ^2 values, we can write down the loop in our original function. The entire command lines are shown as below:

```
void draw_histogram(TH1D* h, TGraph* cross_section, double parameters,
                    double baseline, double s2, double m)
{
        //std::cout << parameters << std::endl;
        for (int i = 1; i <= h->GetNbinsX(); ++i)
        {
                double binCenter = h->GetBinCenter(i);
                double y1 = cross_section->Eval(binCenter);
                double y2 = h->GetBinContent(i);

                double S = sin(1.27/1E3 * m * baseline/binCenter);
                //std::cout << i << ": " << y2 << std::endl;
                double y = y1 * y2 * parameters;
                double N = (1.0 - s2 * S * S )* y;
                h->SetBinContent(i,N);
                // std::cout << N << std::endl;
        }
        h->GetYaxis()->SetTitle("Number of Events / GeV");
        h->Draw("SAME");
        //std::cout << h->Integral("width") << std::endl;
}

double chi2(TH1D* h, TH1D* g)
{
        double chi_sq = 0.0;
        //for (int i = 1; i <= h->GetNbinsX(); ++i)
        for (int i = 1; i <= 135; ++i)
        {
                double y = h->GetBinContent(i);
                double lambda = g->GetBinContent(i);
```

```
                if (lambda == 0) continue;
                double chi_sq_i = (y-lambda)*(y-lambda)/lambda;
                std::cout << "#" << i << " y: " << y << ", lambda: "
                          << lambda << std::endl;
                chi_sq = chi_sq + chi_sq_i;
        }
        return chi_sq;
}

void eventrate_heatmap()
{
        TFile* cross_section_file = TFile::Open("cross_section.root");
        cross_section_file->ls();
        TGraph* cross_section = (TGraph*) cross_section_file->Get("qel_cc_n;1");
        // cross_section->Draw();
        TFile* flux = TFile::Open("flux.root");
        flux->ls();
        TH1D* h = (TH1D*) flux->Get("numu_CV_AV_TPC");

        TH1D* h_sbnd = (TH1D*) h->Clone("h_sbnd");
        TH1D* h_microboone = (TH1D*) h->Clone("h_microboone");
        TH1D* h_icarus = (TH1D*) h->Clone("h_icarus");

        draw_histogram(h_sbnd, cross_section,
                1E-38/1E4*1.6856E30/0.05*6.2E20/1E6*(470.0/110.0)*(470.0/110.0),
                110.0, 0, 1);
        draw_histogram(h_microboone, cross_section,
                1E-38/1E4*1.3395E30/0.05*13.2E20/1E6,
                470.0, 0, 1);
        draw_histogram(h_icarus, cross_section,
                1E-38/1E4*7.1638E30/0.05*6.2E20/1E6*(470.0/600)*(470.0/600),
                600.0, 0, 1);


        auto heatmap = new TH2D("heatmap","heatmap; log(sin^2theta); log(Delta_m^2)",
                                100,-3,0,
                                100,-2,2);

        for (int i = 1; i < heatmap->GetNbinsX()+1; i++)
        {
            for (int j = 1; j < heatmap->GetNbinsY()+1; j++)
            {
                float x = pow(10, heatmap->GetXaxis()->GetBinCenter(i));
                float y = pow(10, heatmap->GetYaxis()->GetBinCenter(j));

            TH1D* h_sbnd2 = (TH1D*) h->Clone("h_sbnd2");
            TH1D* h_microboone2 = (TH1D*) h->Clone("h_microboone2");
            TH1D* h_icarus2 = (TH1D*) h->Clone("h_icarus2");
```

```
            draw_histogram(h_sbnd2, cross_section,
                    1E-38/1E4*1.6856E30/0.05*6.2E20/1E6*(470.0/110.0)*(470.0/110.0),
                    110.0, x, y);
            draw_histogram(h_microboone2, cross_section,
                    1E-38/1E4*1.3395E30/0.05*13.2E20/1E6,
                    470.0, x, y);
            draw_histogram(h_icarus2, cross_section,
                    1E-38/1E4*7.1638E30/0.05*6.2E20/1E6*(470.0/600)*(470.0/600),
                    600.0, x, y);

            double chi2_sbnd = chi2(h_sbnd, h_sbnd2);
            double chi2_microboone = chi2(h_microboone, h_microboone2);
            double chi2_icarus = chi2(h_icarus, h_icarus2);
            double chi2_total = chi2_sbnd + chi2_microboone + chi2_icarus;
            heatmap->SetBinContent(i,j,chi2_total);

            std::cout << "chi2_sbnd: " << chi2_sbnd << std::endl;
            std::cout << "chi2_microboone: " << chi2_microboone << std::endl;
            std::cout << "chi2_icarus: " << chi2_icarus << std::endl;

            std::cout << "x" << x << std::endl;
            std::cout << "y" << y << std::endl;

            delete h_sbnd2;
            delete h_microboone2;
            delete h_icarus2;
            }
        }
            heatmap->Draw("colz");
            gPad->SetLogz();
}
```

Ultimately, the plot on canvas is shown as Figure 4 in next page. The amazing result is important in testing the sensitivity for each detector. Note that the axis is set as logarithmic value. For the formal heat map plot, the curves can be shown more obviously, which is shown as Figure 5.

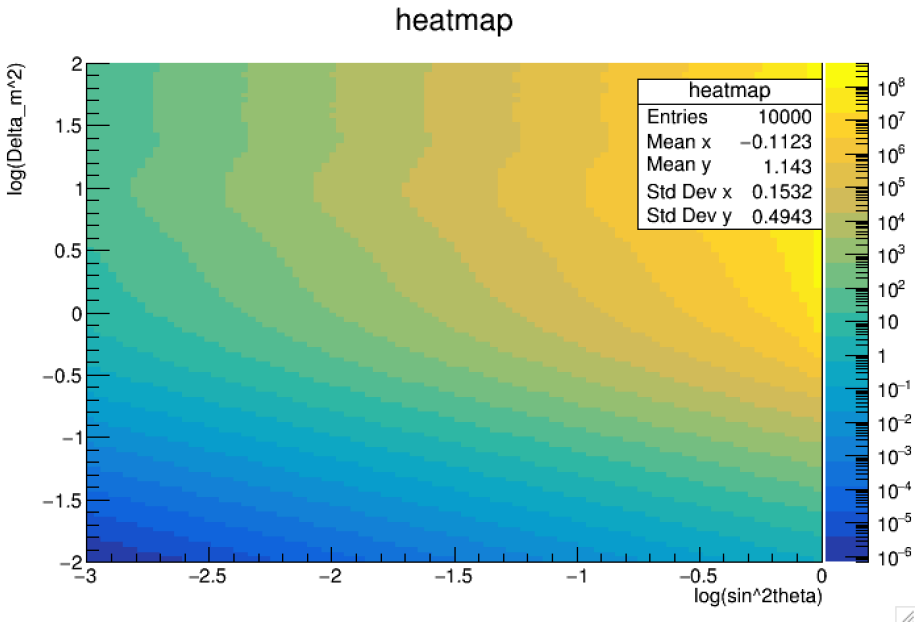


Figure 4: Heat map plot given from each detector's χ^2 value. The contour curves can be seen as the sensitivity prediction for the detectors.

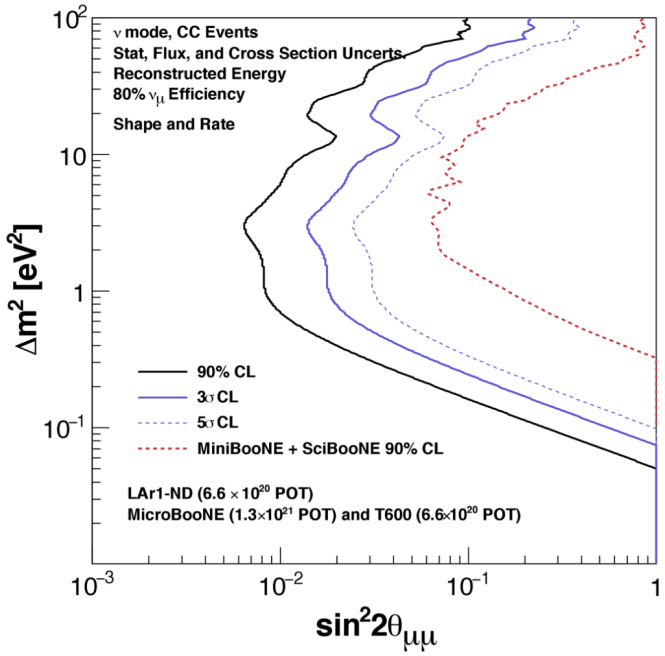


Figure 5: The formal heat map plot in SBN proposal (arxiv:1903.04608). Compared with both plots, the curves are almost similar so that the result would be correct.

6 Conclusion

To create the 2-D heat map plot, the first step is to calculate event rate according to cross section's and flux's data. Then, considering neutrino oscillations, we can see the slightly different between both event rate's histograms. To know the difference between null hypothesis (non-oscillated) with the oscillated one, we must figure out the χ^2 value. Finally, compiling the result into the original loop, we can attain the heat map plot with the curves. It is noteworthy that physicists can look over the sensitivity from such contour plot. In this way, it can let us know the experimental details of ν_μ disappearance in SBN detector.

7 Acknowledgement

I am grateful to Prof. Andrew Mastbaum for giving me a precious opportunity to be engaged in SBN program's research. It is a wonderful tour for me to learn plenty of technological skills as well as data analysis work, which is crucial to my physics career in the near future.

References

[1] Pedro A. N. Machado, Ornella Palamara, David W. Schmitz. *The Short-Baseline Neutrino Program at Fermilab.* (2019) arXiv:1903.04608.

[2] R. Acciarri, C. Adams, R. An, C. Andreopoulos, A.M. Ankowski, *et al. A Proposal for a Three Detector Short-Baseline Neutrino Oscillation Program in the Fermilab Booster Neutrino Beam.* (2015) arXiv:1503.01520.

[3] Glen Cowan. *Statistical Data Analysis.* (1998) New York: Oxford University Press.