

Signal Injection

Davio Cianci and Joseph Zennamo

What We Want

We would like to know how well we will be able to constrain the parameters of sterile neutrino oscillation with our detector setup.

Sensitivity Curves

The current state of the art in these studies involves comparing sample sterile neutrino oscillation signals over a phase space of Δm^2 and $\sin^2(2\theta)$ with a null hypothesis and measuring our ability to resolve signal.

The Process

- Initialize and fill vectors
- Build covariance matrix
- Compare sample predicted signals with null hypothesis
- Calculate X^2 values
- Draw beautiful plots

Shape Only Analysis

Nominal uncertainty matrix is created by adding these three components.

Since we lack information to properly account for the cross-section normalization, a shape-only analysis is a more realistic approach for the time being, so we subtract the normalization component from the overall matrix.

$$E_{i,j}^{shape} = E_{i,j} - \frac{N_j}{N_T} \sum_{k=1}^n E_{i,k} - \frac{N_i}{N_T} \sum_{k=1}^n E_{k,j} + \frac{N_i N_j}{N_T^2} \sum_{kl}^n E_{k,l}$$

$$E_{i,j}^{mixed} = \frac{N_j}{N_T} \sum_{k=1}^n E_{i,k} + \frac{N_i}{N_T} \sum_{k=1}^n E_{k,j} - 2 \frac{N_i N_j}{N_T^2} \sum_{kl}^n E_{k,l}$$

$$E_{i,j}^{norm} = \frac{N_i N_j}{N_T^2} \sum_{kl}^n E_{k,l}$$

Covariance Matrix

- Flux uncertainty matrix is built using:

$$M(i,j) = (\text{Nom} - \text{Sys}_n)_i (\text{Nom} - \text{Sys}_n)_j$$

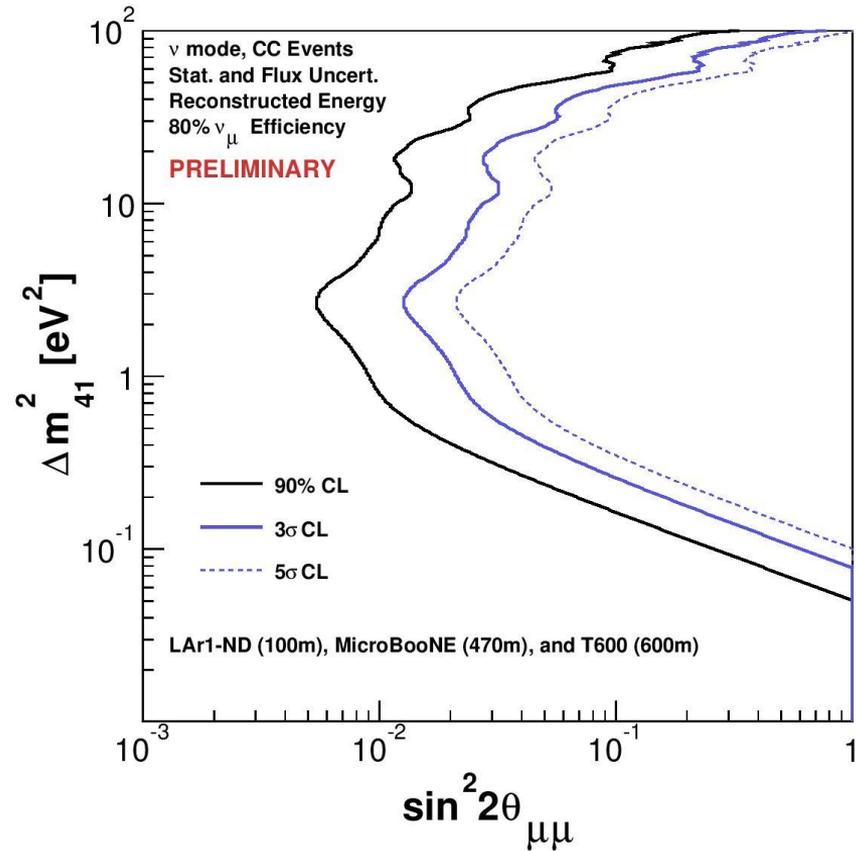
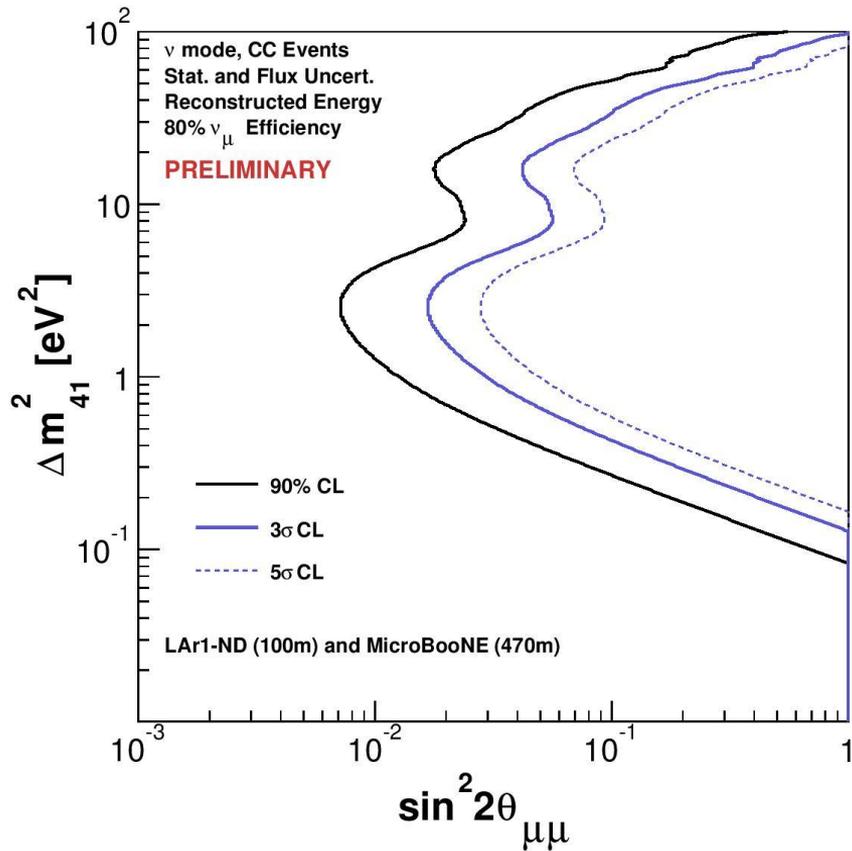
- i,j represent entries for each bin, separated by detector
- Entries are averaged over N systematic fluctuations
- All entries are fractionalized and then normalized to null hypothesis signal
- Statistical uncertainty added along the diagonal

χ^2 Calculation

- Fill prediction matrix with oscillations for each Δm^2 , $\sin^2(2\theta)$ subtracted from null hypothesis.
- Compare each prediction to null hypothesis
- Find χ^2 using:

$$\chi^2 = (E_{null} - E_{pred})_i \text{Cov}_{ij}^{-1} (E_{null} - E_{pred})_j$$

- Space is covered with a 1-directional Raster scan and contours are cut at 5σ ($\chi^2 < 25$), 3σ ($\chi^2 < 9$), and 90% ($\chi^2 < 1.64$) confidence levels.



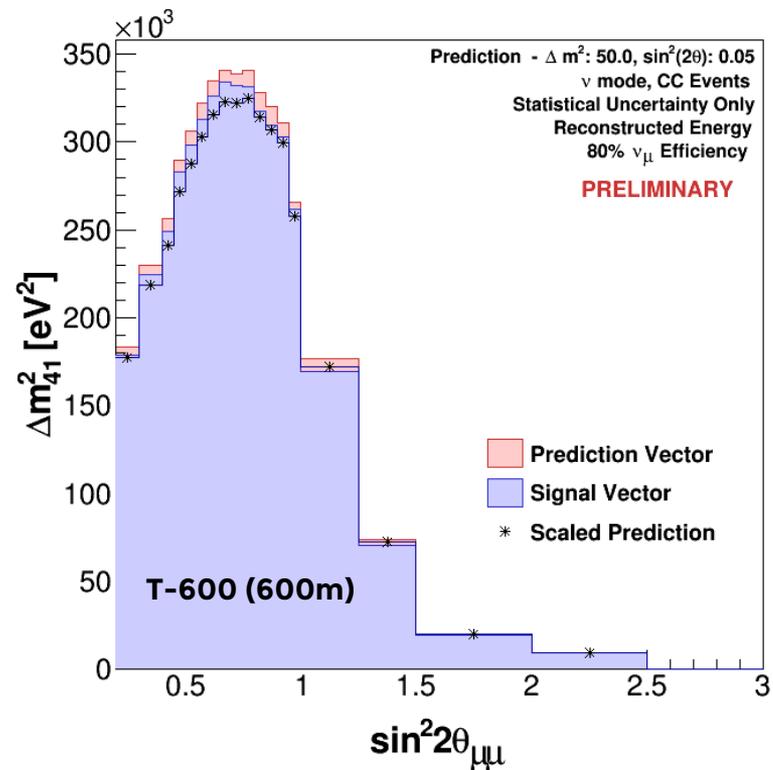
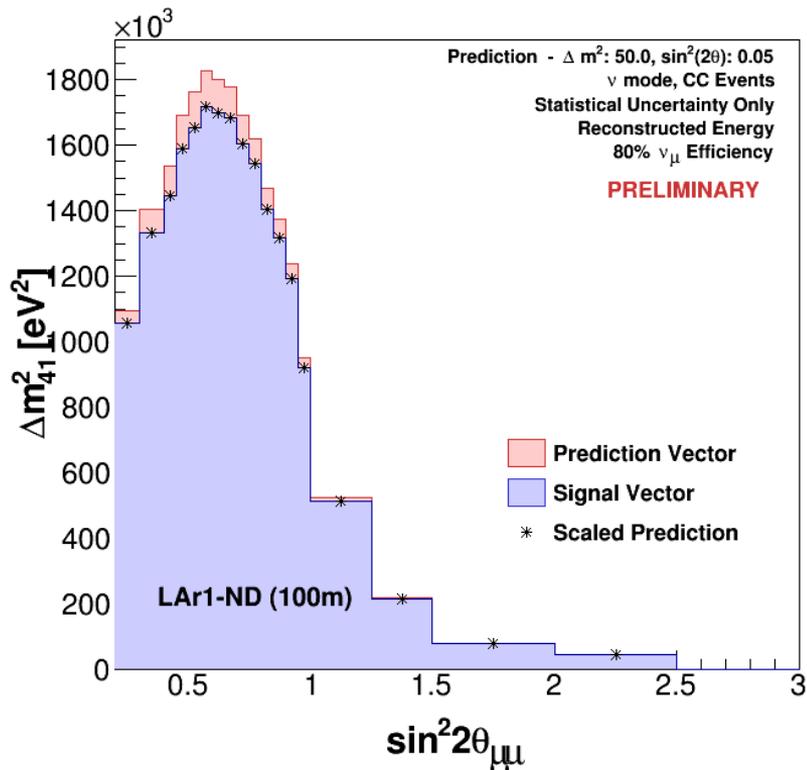
Sensitivity Curve

Signal Injection

Instead of comparing to a null hypothesis, we would like to see how strongly we will be able to constrain these parameters when faced with a “real” signal.

Changes to χ^2 Calculation

- We create a signal vector by subtracting the predicted oscillation for a set Δm^2 and $\sin^2(2\theta)$ from the null hypothesis.
- The prediction vector is created exactly as before, but this time it is scaled to the injected signal in the near detector. This gives a contribution to the χ^2 of 0 in the near detector, so all information comes from MicroBooNE and T600
- Calculate χ^2 as before using E_{sig} to replace E_{null}

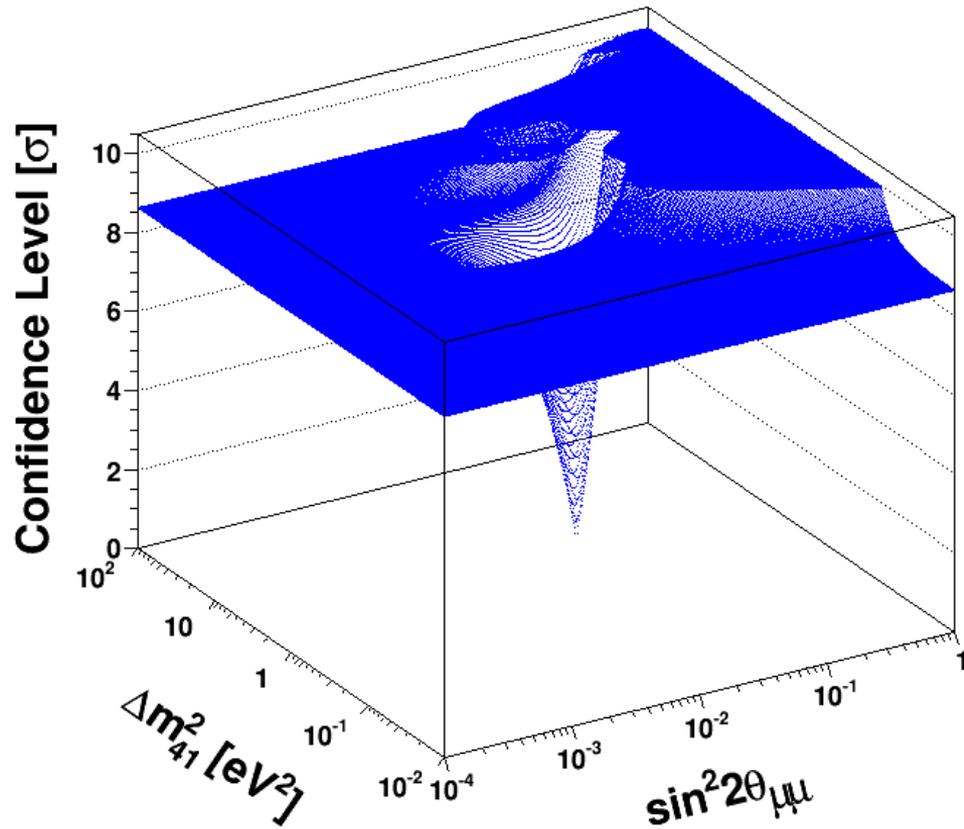


“Shape Only” Scaling of Prediction Vector

Best Fit

Next, we must find the best-fit point - where the code thinks the signal is most likely to lie.

This is calculated by finding the point on the plot of lowest X^2 .



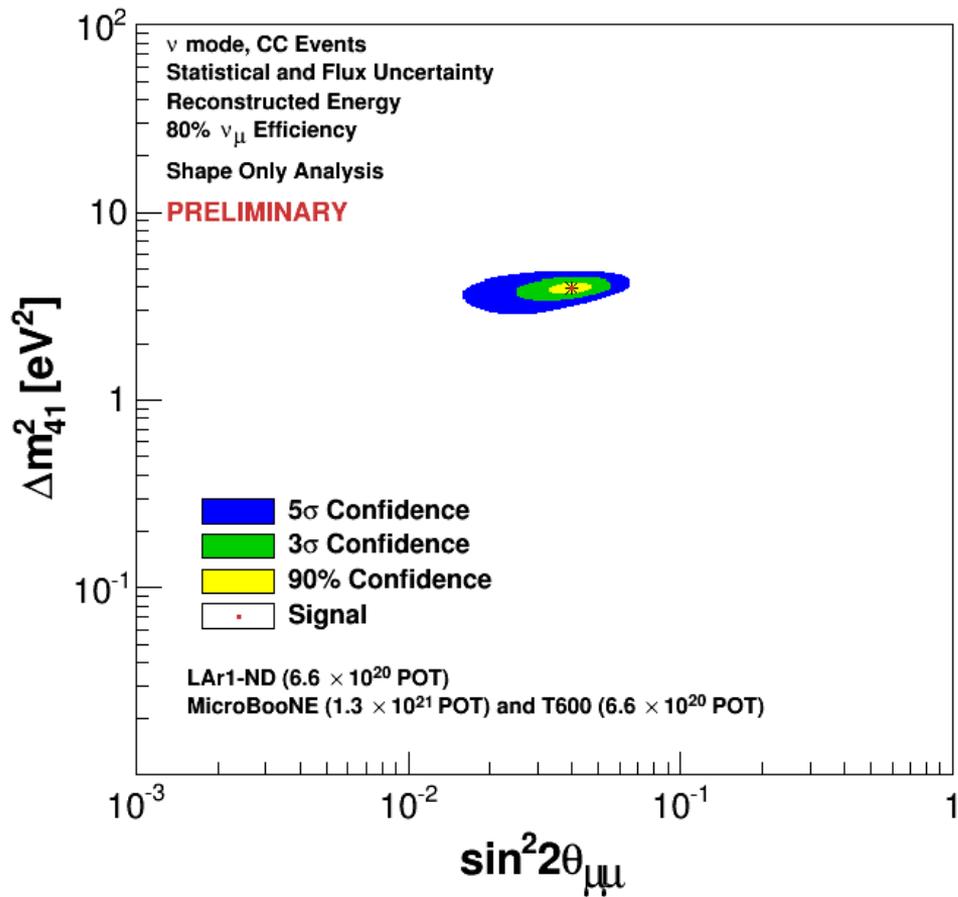
χ^2 Surface

Confidence Levels

Using the best fit, confidence contours are created, placing a point on the plot at each Δm^2 , $\sin^2(2\theta)$ where we have:

$$\Delta X^2 < X^2_{C.L} - X^2_{Best\ Fit}$$

For cleanliness, the code does not draw jellybeans that touch at least three sides of the plot.

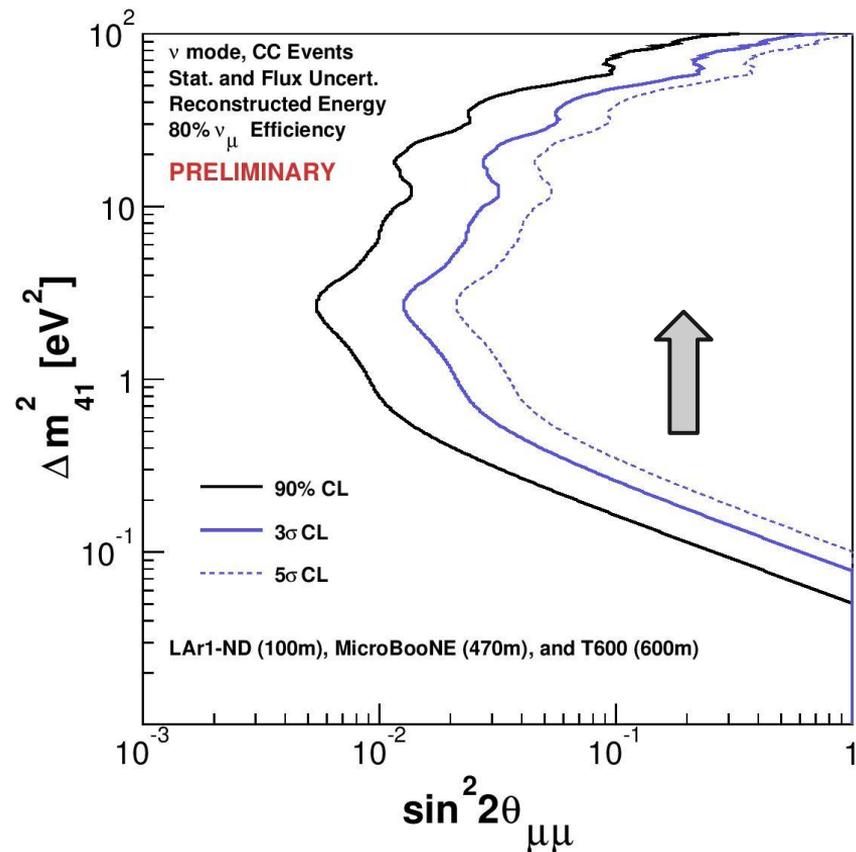
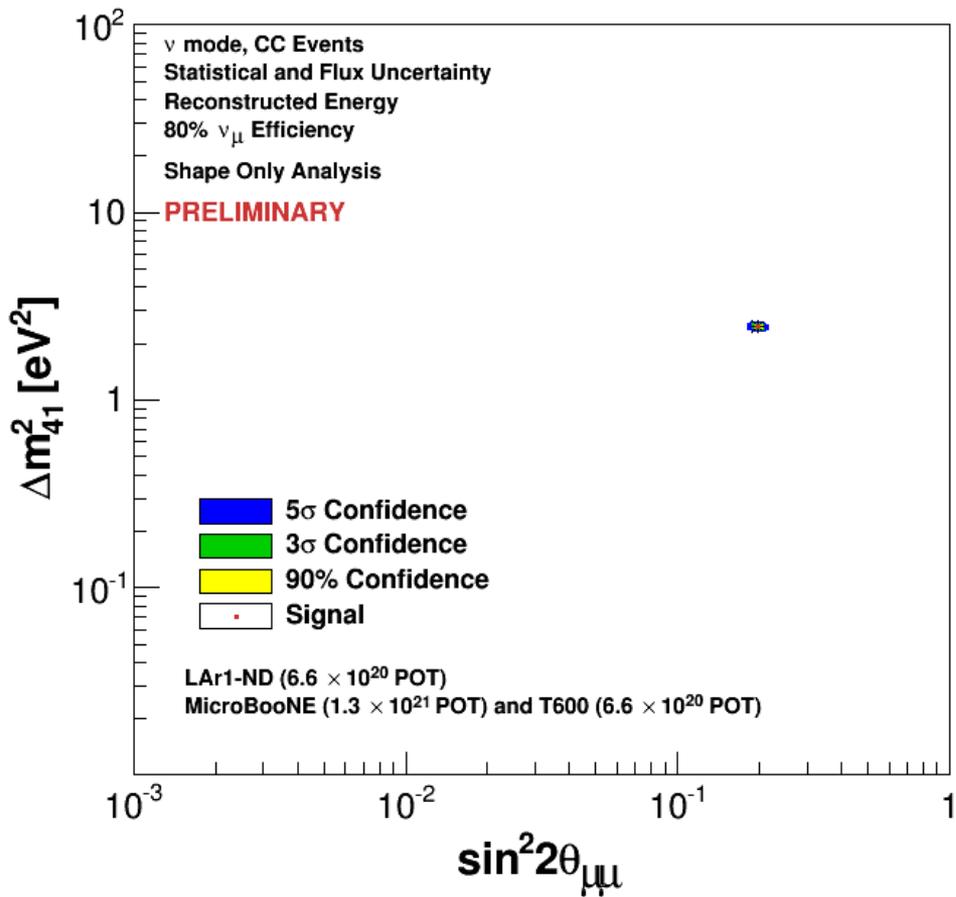


$$90\% \text{ C.L.} : X^2 \leq 1.64 + X^2_{Min}$$

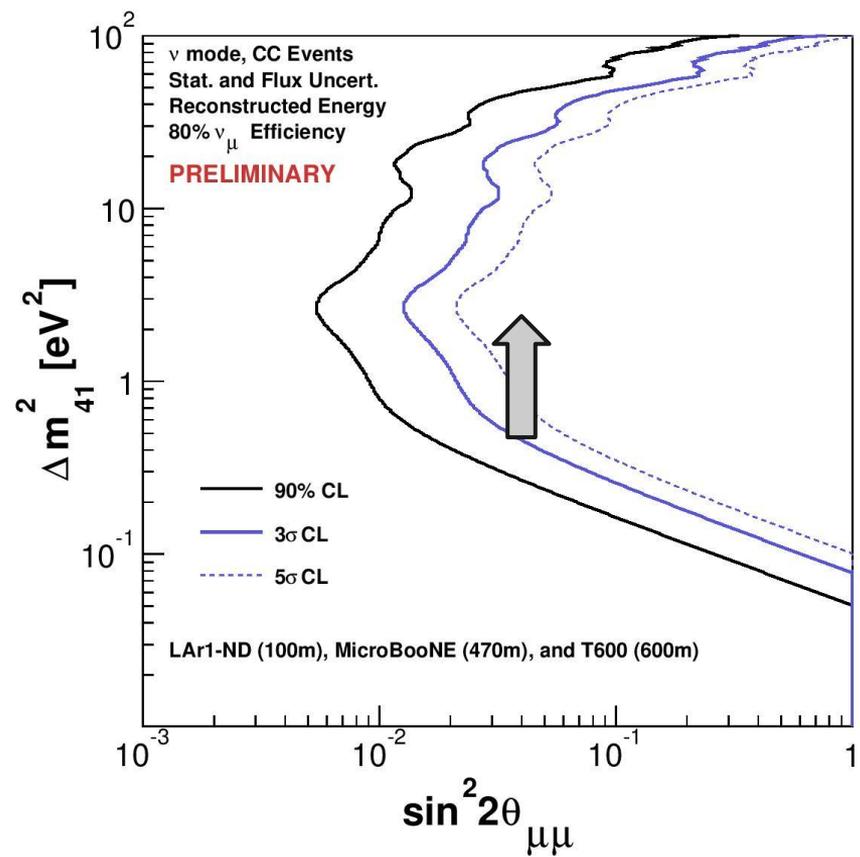
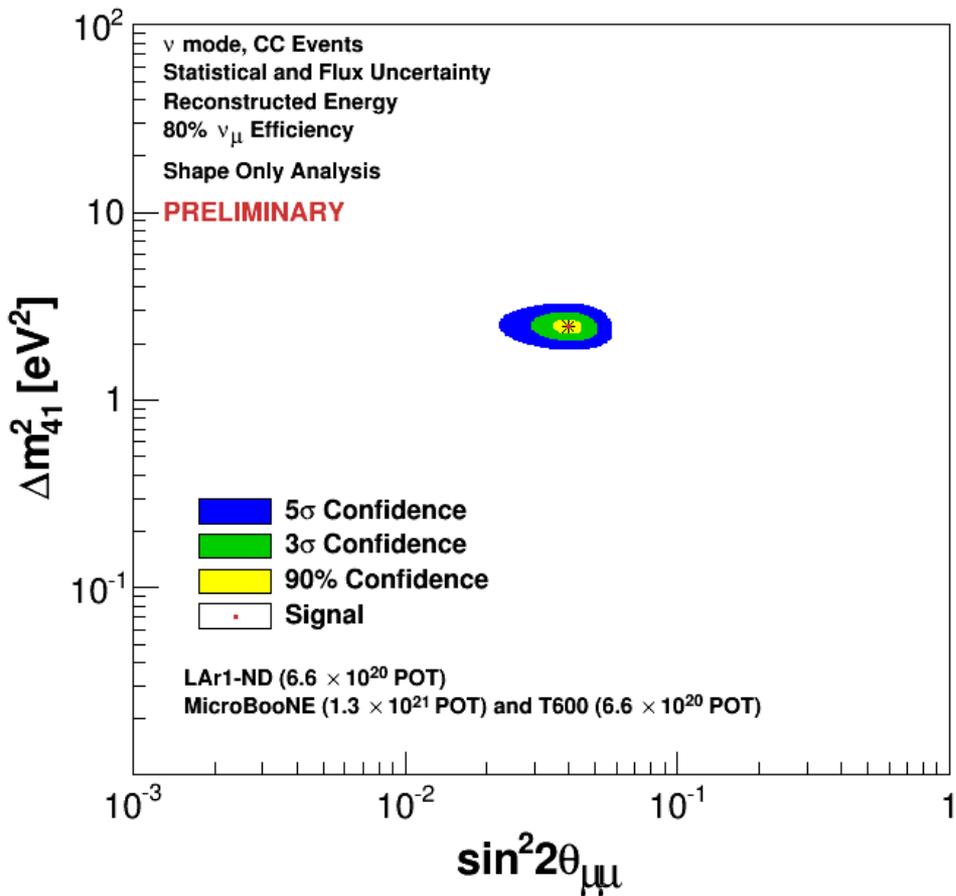
$$3\sigma \text{ C.L.} : X^2 \leq 9 + X^2_{Min}$$

$$5\sigma \text{ C.L.} : X^2 \leq 25 + X^2_{Min}$$

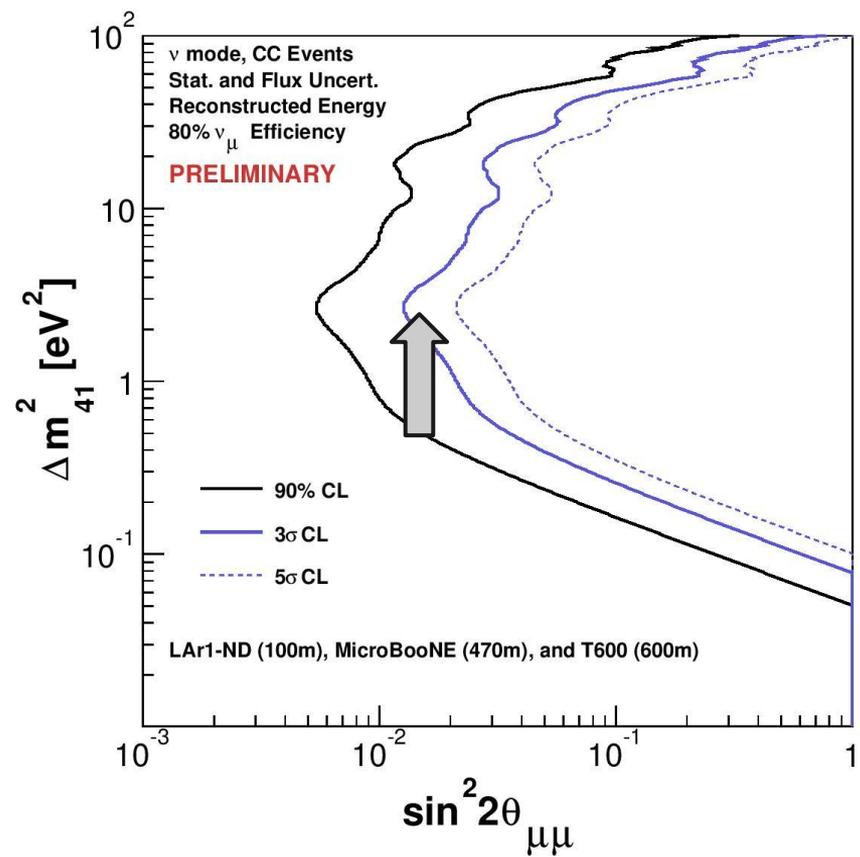
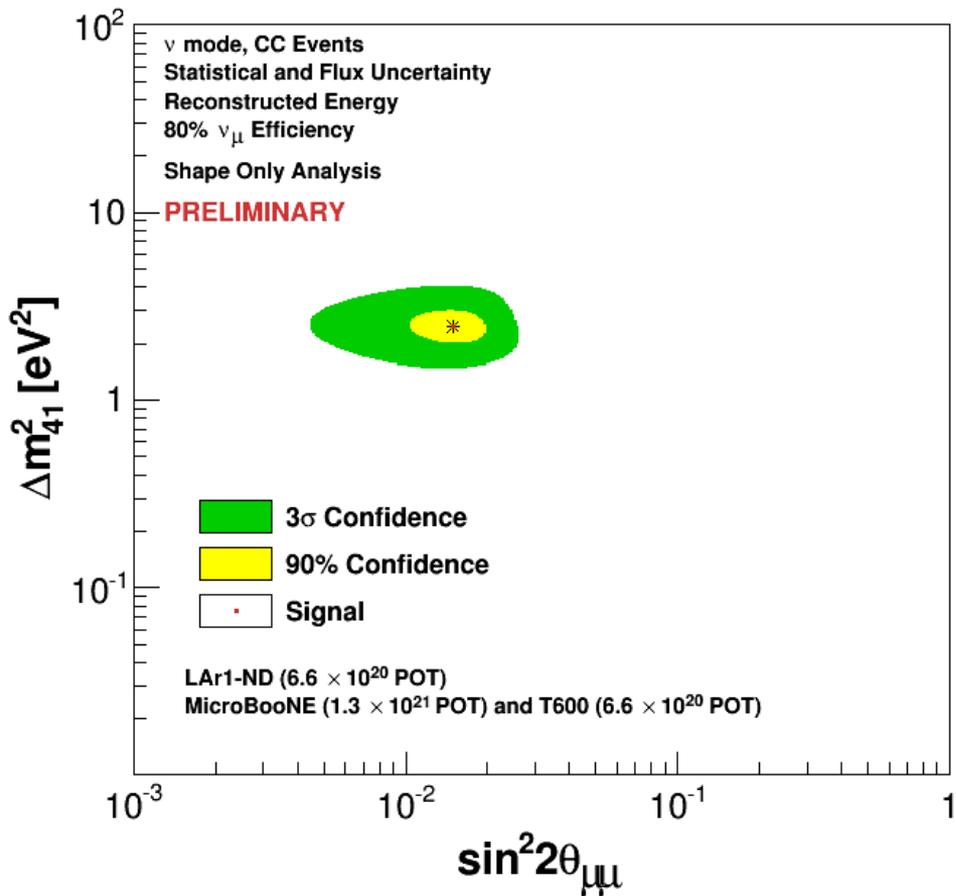
Jellybean Plot



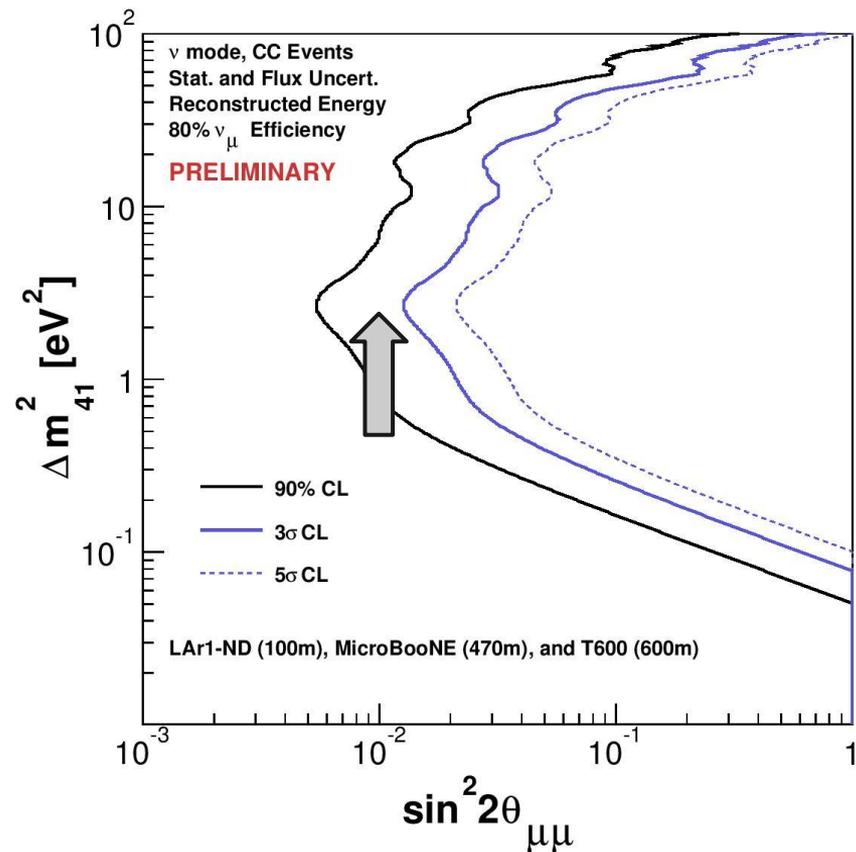
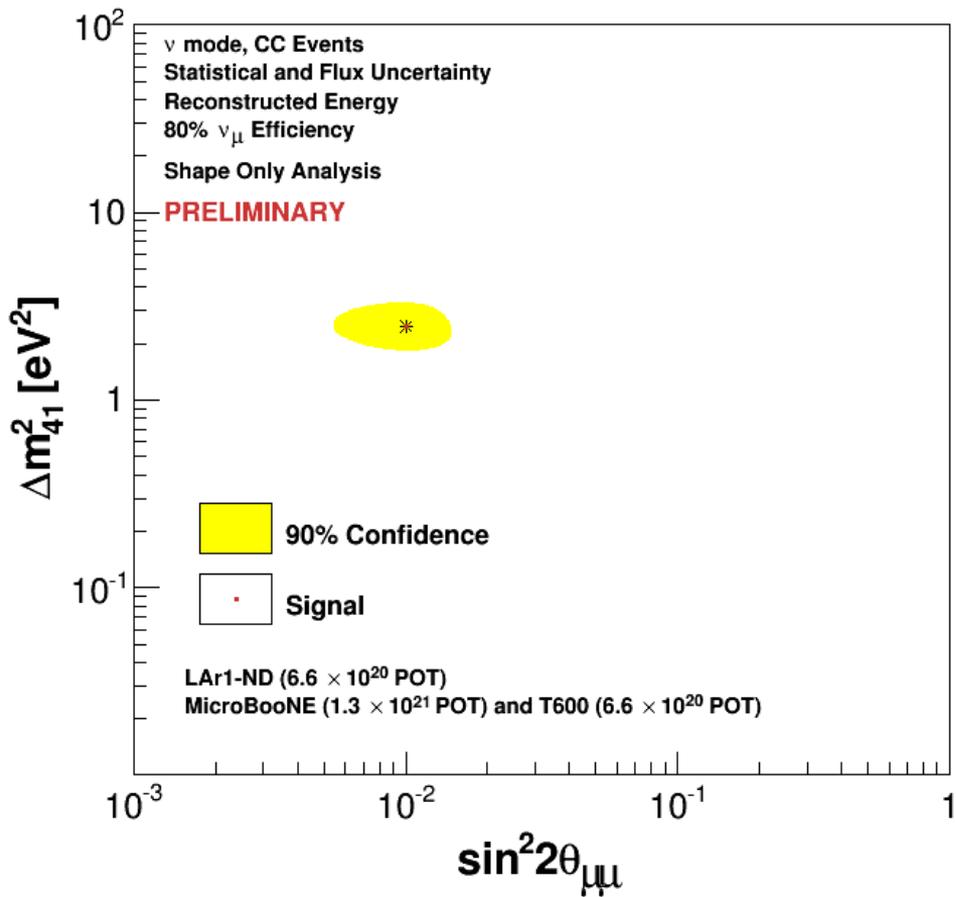
Jellybean vs. Sensitivity Curves 1



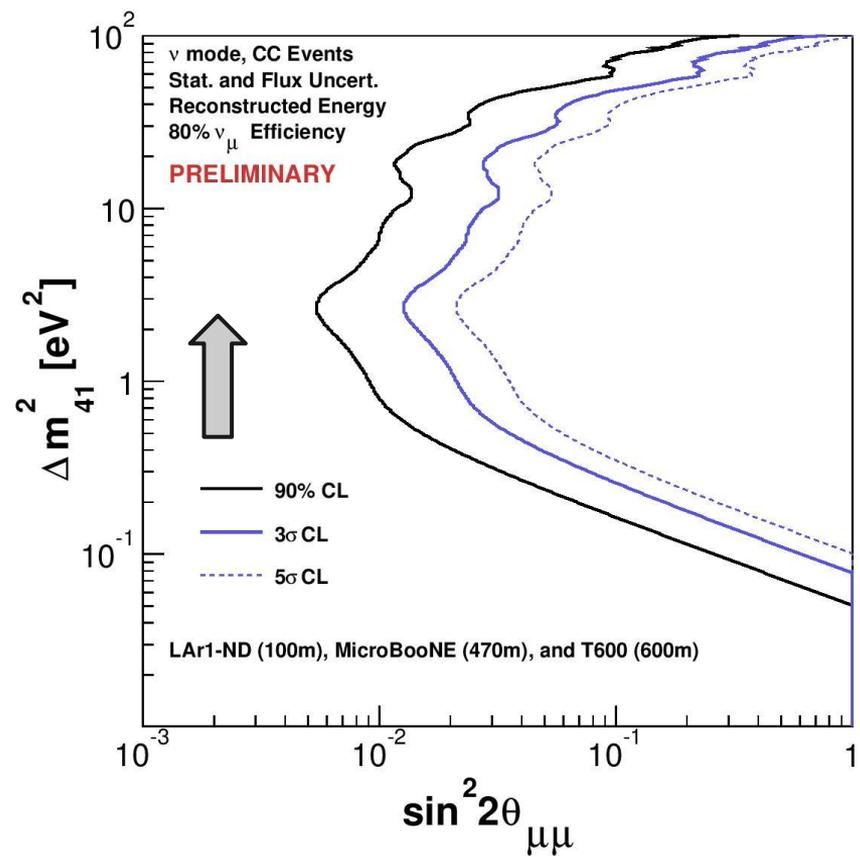
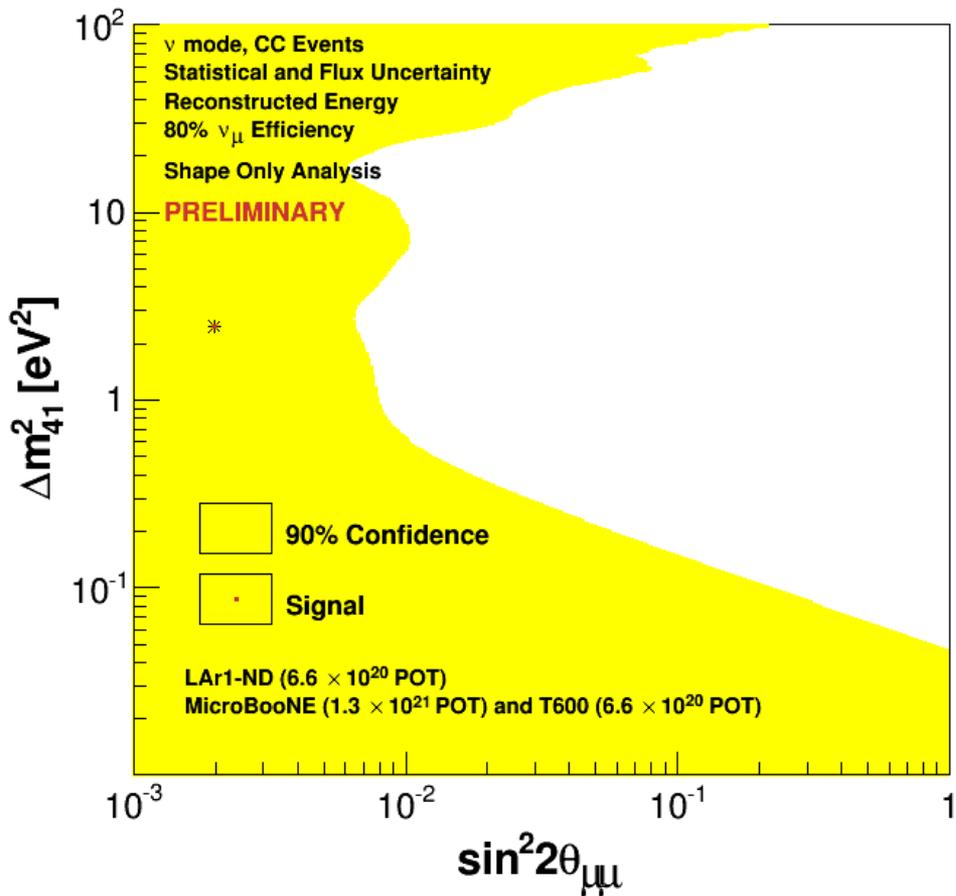
Jellybean vs. Sensitivity Curves 2



Jellybean vs. Sensitivity Curves 3



Jellybean vs. Sensitivity Curves 4



Jellybean vs. Sensitivity Curves 5

Too Good To Be True? Yes

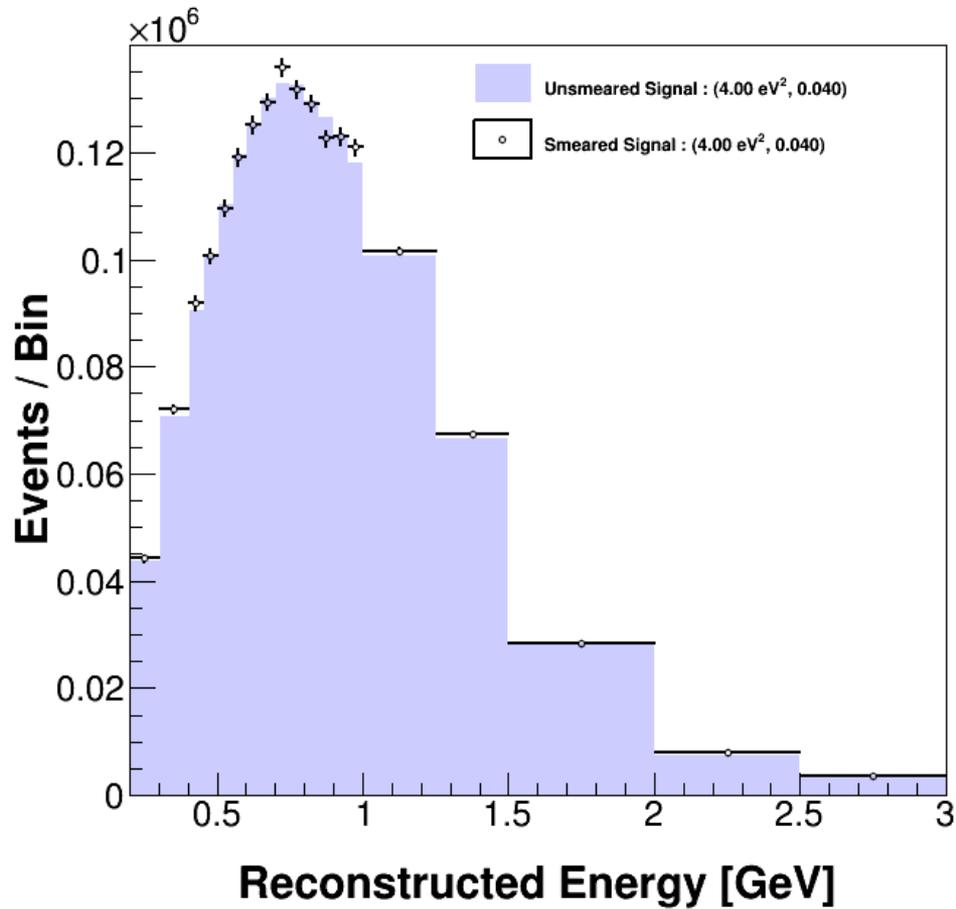
From those plots, we see that the code nails the best-fit exactly to the signal and that the minimum X^2 is always 0.

The signal and prediction vectors are all cut from the same cloth - there will always be a prediction to match the signal exactly, which is very idealistic

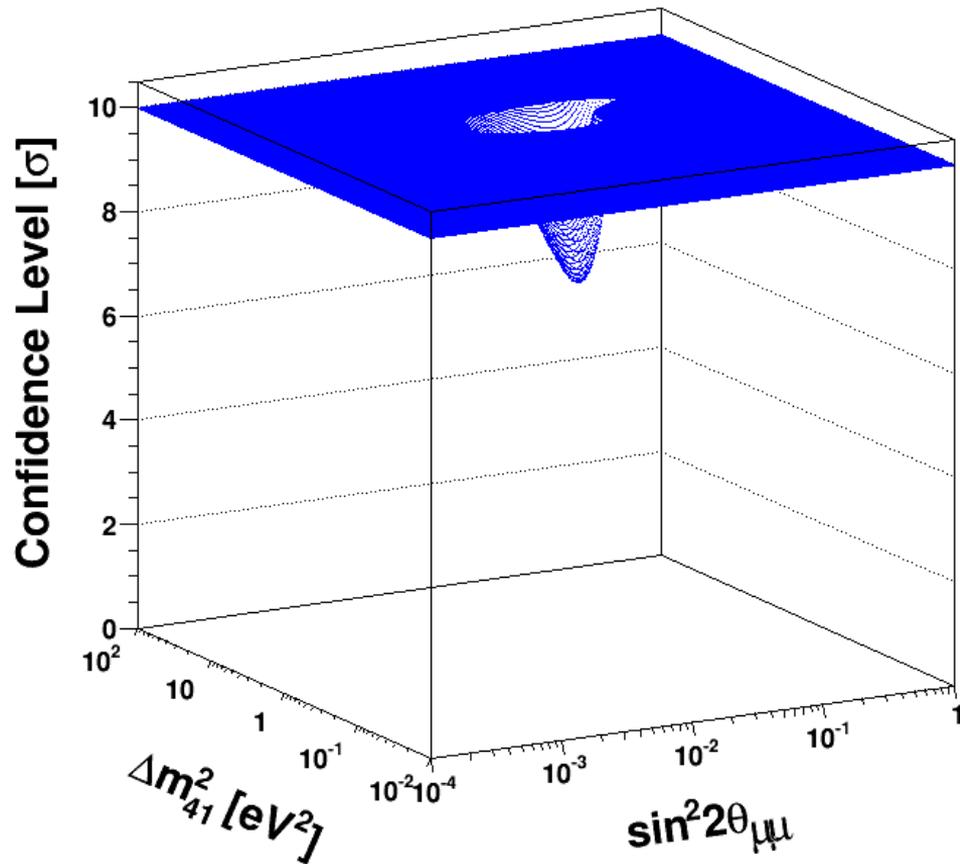
Fluctuation

To fix this, we need to fluctuate the signal to simulate something the detectors may actually find.

We build the signal vector just as before, but then fluctuate it using a Poisson distribution.



Fluctuated vs. Unfluctuated



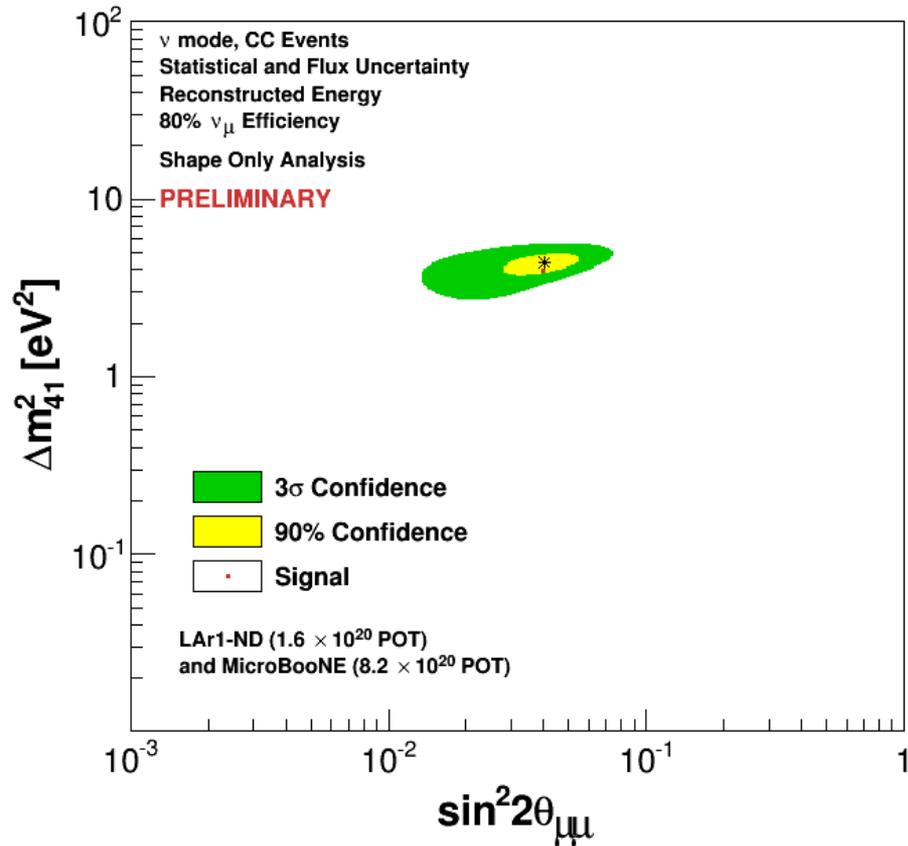
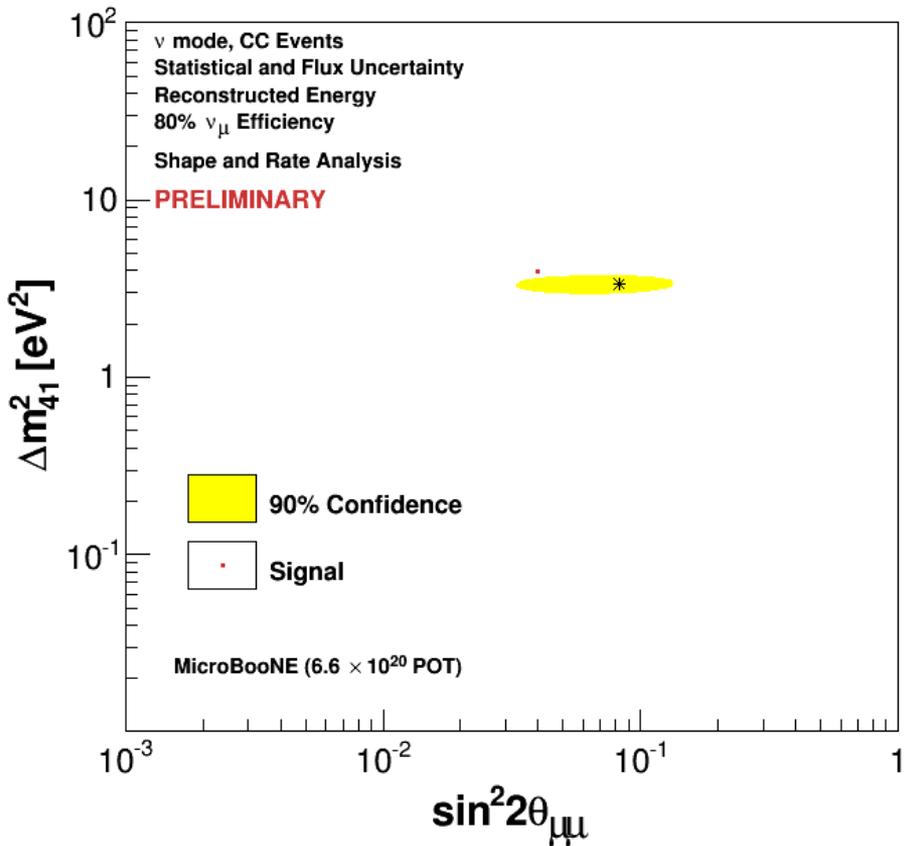
Same χ^2 Surface

Iteration

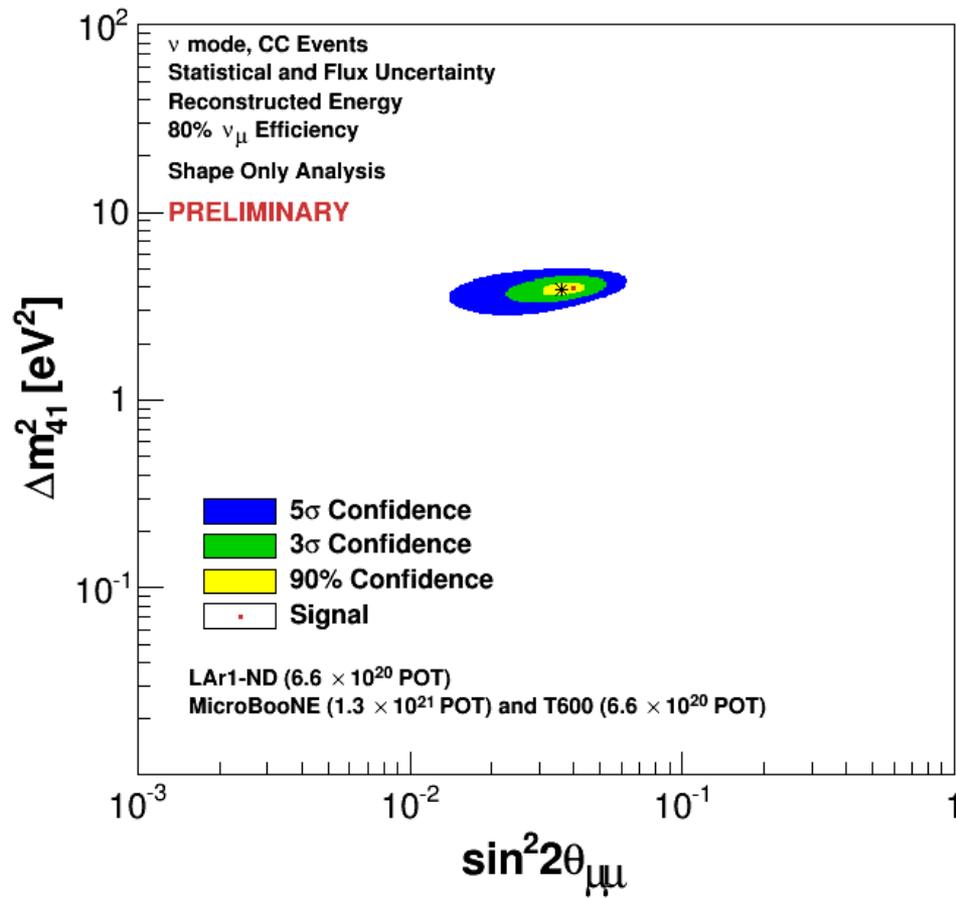
- With smearing, there is no longer a perfect match for the best-fit point.
- The covariance matrix is built without any assumptions about a signal (much like the real-life case).
- After some signal is detected, we need to update the matrix with the new information and run the χ^2 calculation again.

Covariance Matrix: part II

- Instead of scaling the flux uncertainty matrix to a null hypothesis, it is scaled to the best-fit signal.
- The χ^2 values are calculated again with the new covariance matrix
- This process is repeated until the change in $\Delta\chi^2$ is less than .002.



Jellybeans for Multiple Detector Setups

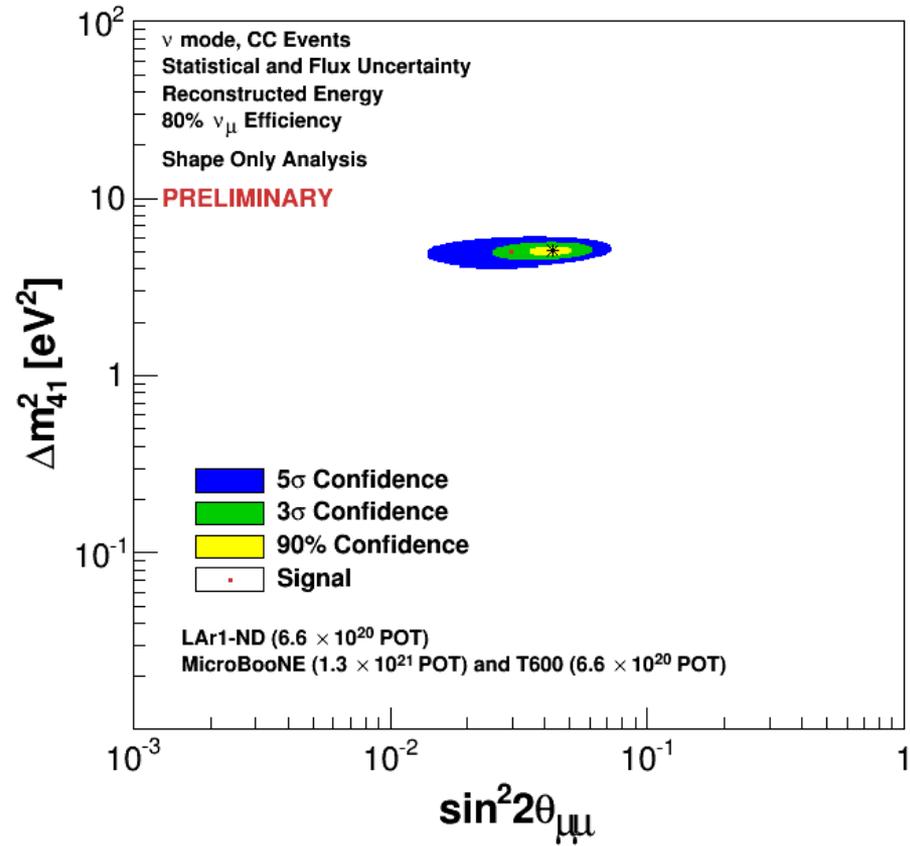
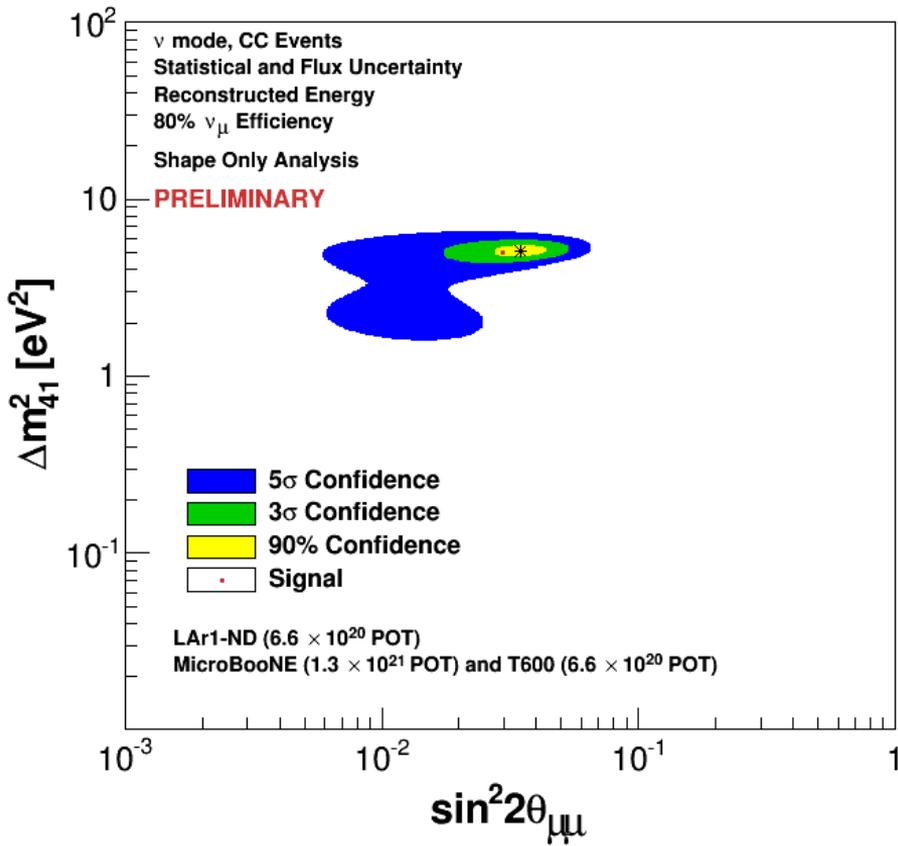


Jellybean for Setup with Each Detector

Moving Forward

The code still has a few kinks to work out, particularly with regards to stability of best-fit point.

Even with iteration, it often fails to find the signal exactly and since each run changes the fluctuation, it provides a very different best-fit.



Best-Fit Instability

Moving More Forward

A tech-note describing these studies is in progress and will be available soon on the SBN and MicroBooNE DocDBs.

Thanks All!