# Looking for Matter Enhanced Neutrino Oscillations Via Day v. Night Asymmetries in the NCD Phase of the Sudbury Neutrino Observatory

by

Richard Anthony Ott, III

Submitted to the Department of Physics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Physics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2011

Author ..........................................................
Department of Physics
July 20, 2011

Certified by ....................................................
Joseph Formaggio
Associate Professor
Thesis Supervisor

Accepted by ....................................................
Krishna Rajagopal
Chairman, Department Education Committee

# Looking for Matter Enhanced Neutrino Oscillations Via Day v. Night Asymmetries in the NCD Phase of the Sudbury Neutrino Observatory

by

Richard Anthony Ott, III

Submitted to the Department of Physics
on July 20, 2011, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Physics

## Abstract

To measure the regeneration of electron neutrinos during passage through the Earth via the MSW effect, the difference in electron neutrino flux between day and night is measured at the Sudbury Neutrino Observatory (SNO). To be able to distinguish an actual discrepancy from an artifact, the detector properties and backgrounds are studied in regard to both a diurnal difference and a difference in the detector for upgoing v. downgoing neutrinos. This thesis focuses on the signal extraction part of this proccess, by which the neutrino fluxes are determined from the processed data by means of a Markov Chain Monte Carlo method. Recognizing that this effect is expected to have an energy dependence, the asymmetry is modeled as an affine function of energy. This measurement yields an asymmetry in the neutrino flux of $0.044^{+0.037}_{-0.031} + (-0.018 \pm 0.028)(E_\nu - 10)$, with $E_\nu$ measured in MeV.

Thesis Supervisor: Joseph Formaggio
Title: Associate Professor

# Acknowledgments

I have many people to thank for making it through the long path that lead me here. First, I thank my friends, family and coworkers for supporting me though the years, whether you realized it or not. I wish to thank the many faculty, staff and students I have had contact with during my time at MIT. I have learned much from all of you, and you've helped me understand where many of my strengths and weakness lie. Having an income from TAing so many times was helpful as well. I want to thank my advisor, Joe Formaggio. You took me in to your group and helped me all along the way, even though when we met I had no background in particle physics. Finally, I want to thank the many great scientists involved in SNO. Without your dedication, hard work and guidance, the analysis presented here wouldn't exist.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Neutrinos are the least understood Standard Model particle. They are unique among particles in that they only interact via the weak force[1] - they are electrically neutral (no electromagnetic force) leptons (no strong force). The weak force is, as the name implies, weak, so that neutrino interactions with matter are rare. This results in them being extremely difficult to detect - even with the large neutrino outputs from various nuclear processes[2], neutrino detectors must be extremely large and still have low count rates. Even so, neutrinos are now regularly detected and we are able to study their properties with ever increasing precision. The more we are able to learn about them, the closer we get to clues about what lies beyond the Standard Model.

Our current understanding of the Standard Model has matter composed of twelve spin $\frac{1}{2}$ particles: six quarks, three charged leptons (the familiar electron and its heavier cousins the muon and the tau) and three neutrinos. These particles interact via the four fundamental forces of electromagnetism, the strong nuclear force, the weak nuclear force and gravity. The forces, in turn, are transmitted via the force carrying bosons: the photon for electromagnetism, the gluon for the strong force and the W and Z bosons for the weak force. Gravity is supposedly transmitted by the graviton, but the weakness of gravity prevents us from measuring this, or from gravity having a measurable effect in particle physics. We will not mention it from now on.

[1]...and gravity, but it is too weak to measure in particle physics, so we ignore it

[2]$10^{11}$ neutrinos per cm$^2$ per second on the Earth, just from the sun

The classes of particles differ from each other in the forces that they interact with. The quarks "feel" all the forces: strong, weak and electromagnetic. The leptons do not feel the strong force, leaving the weak and electromagnetic. Neutrinos, with no charge, additionally don't feel the electromagnetic force.

One of the biggest, relatively recent discoveries is that neutrinos have mass. In the early theory of the weak interaction, and later the Standard Model, they were assumed to have precisely zero mass - so this discovery required one of the few changes to the Standard Model since its inception. While the absolute masses of the neutrinos is still unknown, we now know the *differences* between the masses of the three types of neutrinos rather well. For the three neutrinos, we have two mass differences, $\Delta m_{21}$ and $\Delta m_{32}$. These mass differences allow the three different types of neutrinos to mix, changing from one type to a superposition of the types as they pass through space. The amount of mixing is characterized by both the mass differences and the mixing angles $\theta_{12}$, $\theta_{23}$ and $\theta_{13}$.

In this thesis, we seek to refine the measurement of the day/night effect. This is a measurement of how much these mixing properties of neutrinos generated in the sun are affected during their passage through the Earth. Since neutrinos detected during the day do not pass through the Earth and those detected at night do, this appears as a difference in properties between day and night, hence the name. Current models predict an approximately 2% to 5% effect, much smaller than any measurement to date has been able to measure. Here we use the data gathered through the Sudbury Neutrino Observatory (SNO) experiment, with particular emphasis on data from the final data collecting period, to improve this measurement.

The primary measurement from SNO was the neutrino flux from the sun, separately measuring the rate of electron neutrinos ($\nu_e$) and the total rate of all three flavors of neutrinos ($\nu_x$). While this allowed SNO to conclusively show that the neutrinos generated in the sun and arriving in the detector were not all $\nu_e$, it is insufficient to make a strong statement about the neutrino mixing parameters driving this process. Figure 1-1 shows an example plot of the mixing parameters, plotting $\tan^2(\theta_{21})$ versus $\Delta m^2_{21}$ (these are the parameters that dominate the mixing for neutrinos coming from

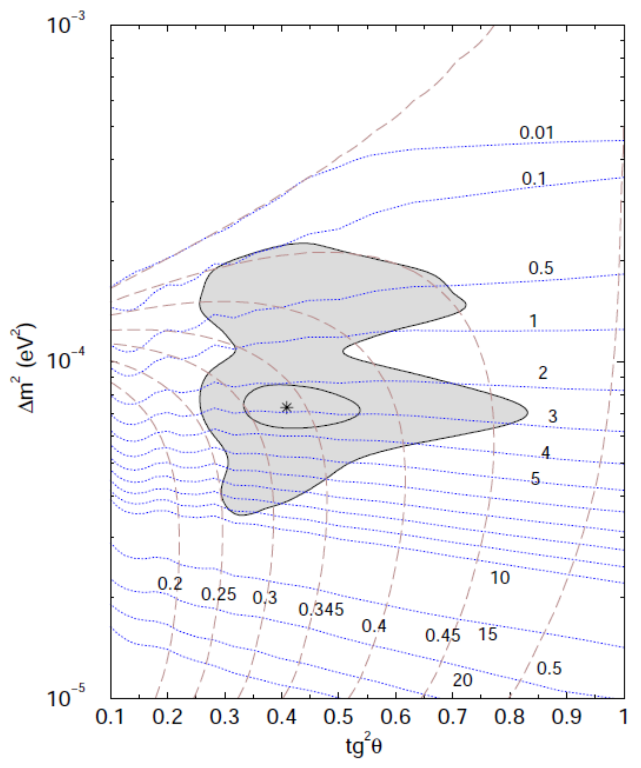


Figure 1-1: An example plot showing the effect of the day/night measurement on knowledge of the mixing parameters $\Delta m^2_{21}$ and $\theta_{12}$. The dashed lines are constant amounts of measured $\nu_e$ v. $\nu_x$ ($\nu_x$ is the total flux, counting all flavors of neutrino, i.e. this is the proportion of detected solar neutrinos that are $\nu_e$'s when detected), SNO's primary measurement. The dotted blue lines are lines of constant day/night effect, in percent. The grey region is an example confidence region, i.e. any value inside the grey mass gives predictions in agreement with the measured results, within a specified confidence interval. Note the importance of the day/night effect in constraining the measurement. From [1].

the sun). The dashed lines are lines of constant proportion of $\nu_e$ to $\nu_x$, i.e. a value of 0.3 indicates that 30% of the measured neutrinos are $\nu_e$. The dotted blue lines are are lines of constant day/night measurement, measured in percent. The graph makes it clear that the day/night measurement provides a complement to the main measurement, restricting the valid parameter range.

Beyond the mixing parameters, there is still quite a bit we don't know about neutrinos. Without the knowledge of the absolute mass of one of the neutrinos, there is an open question as to whether the neutrino mass splittings are large or small compared to the mass of the lightest neutrino, which may even be zero. Additionally, there is an ambiguity in the mass splittings. We know $\Delta m_{32} \gg \Delta m_{12}$, but do not know if this means that we have two neutrinos close in mass and one much heavier, or one much lighter. This is the "hierarchy" question - the former is the "normal hierarchy", the latter the "inverted hierarcy". Even some of the very fundamental properties of neutrinos remain open questions. Are neutrinos and antineutrinos the same particle? If so, they would be the only known Majorana particles (in fact, they're the only known particles that even *could* be Majorana). Are there other types of neutrinos we haven't detected? They would have to either be very massive or not interact via the weak force (i.e. be "sterile" - only interacting via gravity, they would be impossible to detect directly). Is the parameter linking $\nu_1$ and $\nu_3$, known as $\theta_{13}$, zero? Relatedly, is there CP violation in the neutrinos? All we currently know is that $\theta_{13}$ is small, but recent results are starting to indicate it is not exactly zero.

There are many experiments beginning to attempt to answer these open questions, making the upcoming period an exciting one in neutrino physics. While we cannot address these questions directly with the measurement presented in this thesis, improving our knowledge of the mixing parameters is crucial to their success. An example of particular relevance is NO$\nu$A, which will pass a neutrino beam through roughly 500 km of earth to measure the mass hierarchy. This exploits the fact that the matter-enhanced oscillations that drive the day/night effect, due to a process called the MSW effect, are actually sensitive to the hierarchy, unlike simple oscillations. While SNO is unable to measure this itself, the improved knowledge of the MSW effect and the mixing parameters gathered in this analysis and others like it are needed to be able to design and understand this experiment.

It is our hope that our contribution to the measurement of the mixing parameters will play a role in furthering our understanding of the neutrino.

# Chapter 2

# Neutrino History and Theory

## 2.1 Invention and Discovery

The neutrino was originally proposed Wolfgang Pauli in 1930 in his famous "Radioactive Ladies and Gentlemen" letter, reprinted in English in [2], as a solution to a problem with $\beta$ decay. At the time, it had been observed by Chadwick [3] that the electrons emitted in $\beta$ decay often had a continuous spectrum of energies, rather than the monoenergetic lines typical of $\gamma$ decay. There was initial controversy over whether this was due to a spread in energies of the electron emitted from the nucleus, or due to the emission of a monoenergetic electron that went through a secondary process broadening its energy spectrum. This was settled through a calorimetric measurement of the total energy emitted in the $\beta$ decay of Radium-E, now know as $^{210}$Bi, in favor of the former theory [4, 5]. Knowing that the measured electrons were those emitted by the nucleus, it became apparent that conservation of energy appeared to be violated: the nuclear initial and final states had discrete energy, while the emitted electron had a range of energies. Niels Bohr favored a resolution where conservation of energy was only true in a statistical manner in nuclear systems, whereas Pauli (correctly) suggested instead that an additional, unobserved particle was being emitted as well. In his letter, he also proposed that this particle was part of the nucleus to resolve the discrepancy between the measured spins of nuclei and the model that the nucleus consisted solely of spin-$\frac{1}{2}$ protons. This latter idea was quickly abandoned in favor of what are now known as neutrons. Pauli vastly underestimated the difficulty of observing this particle, which we now call a neutrino, in his letter, but by 1931 had realised that the particle must be extremely penetrating [2]. After Pauli's suggestion, Fermi created a theory explaining $\beta$ decay incorporating the neutrino; this theory was the starting point for the modern theory of the weak force.

The first measurement of the neutrino came in 1956 by Cowan and Reines [6]. Many measurements of neutrino properties followed, establishing the properites we know today. Certain properties must be true for the neutrino to solve the $\beta$ decay problem: they must be spin $\frac{1}{2}$, electrically neutral leptons. Subsequently, it was determined that neutrinos and anti-neutrinos cannot initiate the same reactions [7] and that there were more than one type of neutrino: the ones involved in $\beta$ decay, associated with electrons, became $\nu_e$. These were shown to be different from those associated with muons, $\nu_\mu$ [8] and both were different from those associated with $\tau$ particles, $\nu_\tau$ [9]. The famous Z-pole experiment at LEP measured that there are only three neutrino flavors that couple to the Z boson (i.e. interact via the weak force) with mass $m_\nu < \frac{1}{2} m_Z$ [10]. The masses of the three neutrinos are all too small to measure, with current limits at $m_\nu < 2$ eV [11] and with a Standard Model prediction of precisely zero mass.

As the properties of the neutrino were being discovered, the weak force was being explored and the current model of that force developed. Neutrinos were instrumental in that development, as they are direct probes of the weak force. Two famous examples are the discovery of parity violation [12] and the discovery of the neutral current component of the weak force [13].

A brief summary of the leptonic part of the weak force is sufficient for the needs of this discussion. The weak force is moderated by the exchange of three particles, the charged W$^+$ and W$^-$ and the neutral Z. Figure 2-1 shows the Feynman diagrams for weak interactions with leptons. The most salient points for this discussion are that the W interaction, called the charged current, couples $\nu_x$ with $x$ for $x = e, \mu, \tau$ and the Z interaction, called the neutral current, couples $x$ to $x$ and $\nu_x$ to $\nu_x$.

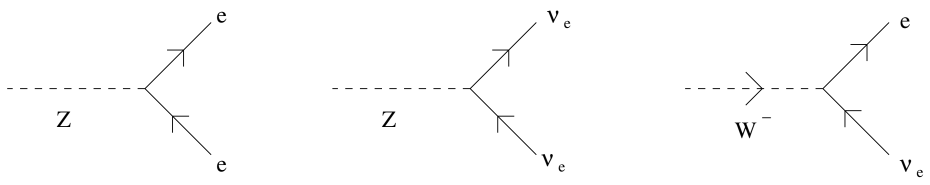


Figure 2-1: Feynman diagrams for weak force interactions of leptons. Note that diagrams with all arrows reversed are also valid, as are ones with global replacement of $e \to \mu$ or $e \to \tau$, i.e. electrons were chosen as an example, but $\nu_x$ and $x$ always go together for any $x$. As per normal for Feynman diagrams, particles propagating backward in time are antiparticles.

## 2.2 Solar Neutrinos

One relatively high-flux source of neutrinos is the sun, and neutrinos from it have been studied in many experiments. We will return to these experiments shortly, but first we should look at the sun itself, and how it generates neutrinos.

The sun is a reasonably typical star. It consists mostly of hydrogen and helium, with a small proportion of heavier elements, of sufficiently high temperature to be plasma. The sun is sufficiently large, with a radius of $7 \cdot 10^8$ m and a mass of $2 \cdot 10^{30}$ kg, that its core density (150 g/cm$^3$) and temperature ($1.6 \cdot 10^7$ K) allow protons to overcome their Coloumb repulsion and fuse into helium nuclei ($\alpha$ particles) [14]. This liberates energy, creating outward pressure from heat and photon pressure, which balances the inward pressure from gravity and keeps the sun in a state of hydrostatic equilibrium. The evolution of stars from pre-stellar clouds of primordial elements to nuclear-fusion-maintained equilibrium to exhausting their nuclear fuel and perishing is a very heavily studied area in astrophysics, known as stellar or solar modeling. A detailed and interesting treatment from the viewpoint of the nuclear processes in stars is given in [14], and the following discussion draws heavily from that work. The solar models used by John Bahcall et al in [15, 16] are generally the ones used in the field.

The idea that stars are powered by nuclear fusion was first proposed by Arthur Eddington in 1920. This was later expanded upon, most notably by Hans Bethe in [17], which layed the groundwork for the modern understanding of the fusion processes in stars and introduced the idea of the CNO cycle. A summary of this history is presented in [14].

In the sun, two processes dominate energy production: the proton-proton (p-p) chain (98.5%) and the CNO-I cycle (1.5%) [15]. Both of these processes are hydrogen-burning processes that fuse four hydrogen nuclei (protons) in to a helium nucleus. This reduces the nuclear charge from $4e$ to $2e$, so for conservation of charge to hold (and for the sun to remain electrically neutral), two electrons are consumed as well, either directly through electron capture or through annihilation with an emitted positron. To then conserve lepton number, two electron neutrinos must be emitted. This gives a net reaction of

$$4p + 2e^- \rightarrow \alpha + 2\nu_e + 26.7 \text{ MeV} \tag{2.1}$$

where the liberated energy is in some combination of thermal energy, neutrino energy and photons.

The simpler of the two processes is the p-p chain, which only requires the presense of hydrogen and helium. Looking at it in detail, we see the p-p chain actually consists of many branches of possible reactions, show in Figure 2-2. The branching ratios shown on the figure are for the sun, given current solar models.

The other dominant process, the CNO-I cycle, is one possible route of catalyzed fusion in stars. In these cycles, a closed loop of fusion processes take an existing element and transmute it repeatedly, ultimately ending up with an element that has an $\alpha$-producing reaction with a proton, reproducing the original element. There are four cycles that are commonly known to occur in stars, but the heavy dependence on temperature and initial isotope abundance results in only the CNO-I cycle and the CNO-II cycle occuring in the sun. In all four cycles, the two neutrinos come from $\beta^+$ decay of an isotope. Table 2.1 shows the CNO-I and CNO-II cycles.

The difference between all of the processes, from the viewpoint of solar neutrino physics, comes in the origin of the two neutrinos. Most are from processes resulting in three (or more) bodies, resulting in $\beta$-decay like energy spectra. These are from $p + p \rightarrow\ ^2\text{H} + e^+ + \nu_e$ (called pp), from $^3\text{He} + p \rightarrow \alpha + e^+ + \nu_e$ (called hep)

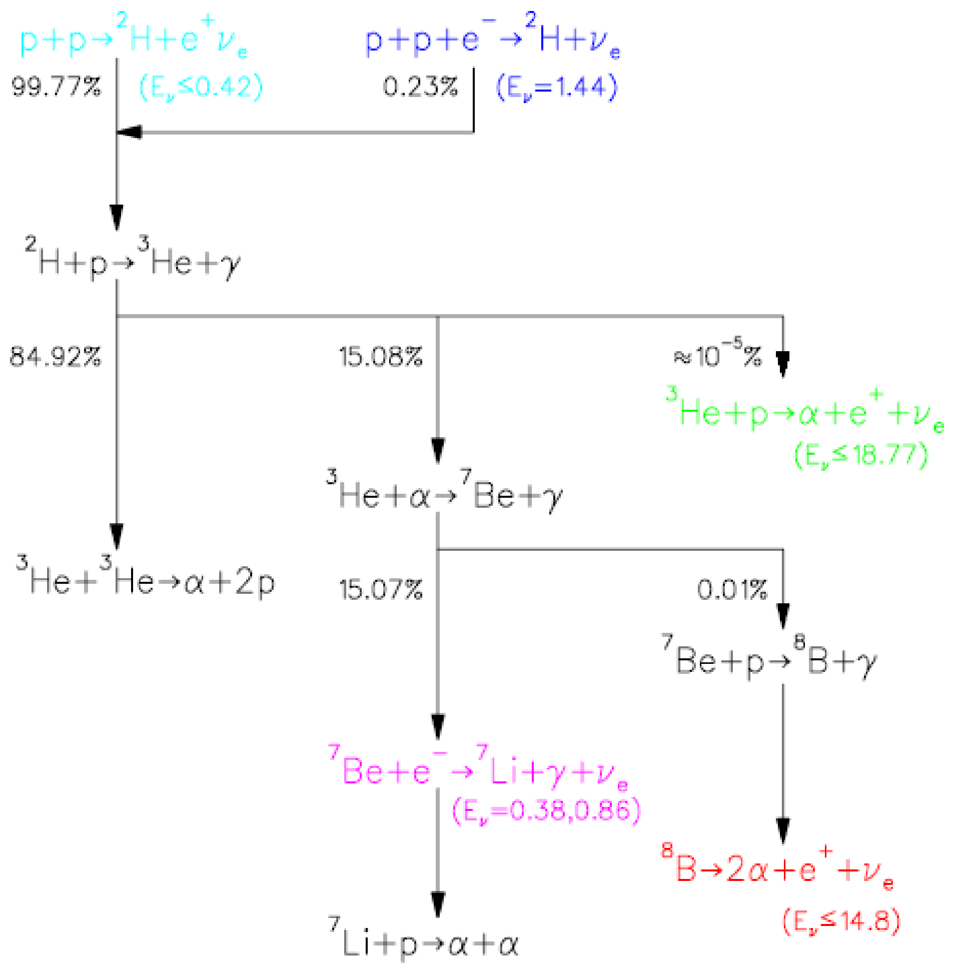


Figure 2-2: Chart of p-p chain branches, from the SNO image library

from $^8\text{B} \rightarrow 2\alpha + e^+ + \nu_e$ (called $^8$B) and from $\beta^+$ decays in the CNO cycles (labelled by isotope). Two come from electron-capture processes, resulting in line spectra (i.e. monoenergetic neutrinos). Those are from $p + p + e^- \rightarrow^2 \text{H} + \nu_e$ (called pep) and from $^7\text{Be} + e^- \rightarrow\ ^7\text{Li} + \gamma + \nu_e$ (called $^7$Be). Three additional, less well known line sources arise from electron capture in the CNO cycle, one for each CNO isotope at 1.022 MeV (two electron masses) above the maximum $\beta^+$ energy [18]. The neutrino output of the Sun is the sum of all of these spectra, shown component-wise in Figure 2-3 (which does not include the CNO line spectra).

The first successful attempt to measure neutrinos from the sun was the radiochemical experiment by Davis in the Homestake mine. This experiment looked for neutrino capture on chlorine in $C_2Cl_4$, producing an isotope of Ar, which was sep-

| CNO-I | CNO-II |
|---|---|
| $^{12}C + p \rightarrow ^{13} N + \gamma$ | $^{14}N + p \rightarrow ^{15} O + \gamma$ |
| $^{13}N \rightarrow ^{13} C + \beta^+ + \nu_e$ | $^{15}O \rightarrow ^{15} N + \beta^+ + \nu_e$ |
| $^{13}C + p \rightarrow ^{14} N + \gamma$ | $^{15}N + p \rightarrow ^{16} O + \gamma$ |
| $^{14}N + p \rightarrow ^{15} O + \gamma$ | $^{16}O + p \rightarrow ^{17} F + \gamma$ |
| $^{15}O \rightarrow ^{15} N + \beta^+ + \nu_e$ | $^{17}F \rightarrow ^{17} O + \beta^+ + \nu_e$ |
| $^{15}N + p \rightarrow ^{12} C + \alpha$ | $^{17}O + p \rightarrow ^{14} N + \alpha$ |

Table 2.1: The CNO-I and CNO-II cycles of catalyzed fusion. Taken from [14], p. 397

arated chemically and counted. The measured flux of neutrinos from the sun was approximately $\frac{1}{3}$ that precited in the solar models [19]. Many other experiments utilizing either radiochemical methods [20, 21, 22, 23] or measuring Cherenkov light from neutrino interactions in water [24] were performed, and all detected fewer $\nu_e$ than models predicted. This was called the "solar neutrino problem".

Different experiments see different portions of the neutrino spectrum. The radiochemical experiments generally have energy thresholds in the hundreds of keV, giving contributions from most of the processes in Figure 2-3. Water Cherenkov detectors like SNO and SuperK have much higher energy thresholds, in the few MeV range. This gives sensitivity only to the $^8$B and hep neutrinos, with $^8$B dominant (roughly 1000 times the flux of hep). This means that the experiments have different expectations, with predictions for major experiment groups from Bahcall show in Figure 2-4.

## 2.3 Neutrino Oscillations

Why do the theoretical predictions and experimental results in Figure 2-4 disagree so strongly (with one exception)? This is the "solar neutrino problem" mentioned earlier. The answer is that the model predicts the neutrino flux from the sun *at the sun.* It doesn't take in to account what might happen to those neutrinos on their way to a detector on the Earth.

At first glance, it doesn't seem like anything should happen - neutrinos interact *extremely* weakly with matter. The predicted interaction length with normal matter

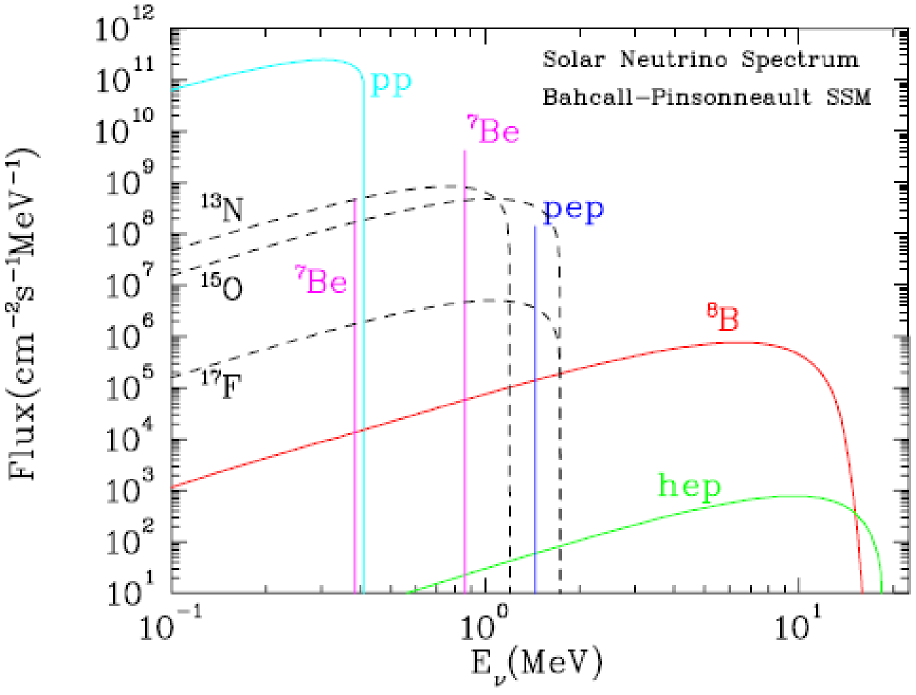


Figure 2-3: Spectrum of solar neutrinos, from the SNO image library, from [16]. This does not include the line spectra from the CNO cycle

is measured in light-years, and they only need pass through, at most, $10^9$ m of solar material, $10^{11}$ m of space (about 8 light-minutes) and $10^7$ m of rock in the Earth. It turns out that each of these has an effect, which we will examine in the next few sections, starting with the vacuum portion.

The possible effects of propagating though vaccuum were suggested in [26] and expanded upon in [27] and [28]. We will follow a more modern and less technical derivation.

In passing through vacuum, any particle propagates at a given velocity dependent on its kinetic energy and mass. Though not actually the case, we can to good approximation treat a propagating particle as if it were in a quantum mechanical eigenstate of momentum ($\vec{p}$). In a vacuum, this is also an eigenstate of energy (E) with $E = \sqrt{m^2c^4 + p^2c^2}$, where $p^2 = |\vec{p}|^2$

The Schrödinger equation, $i\hbar\frac{\partial}{\partial t}\psi = H\psi$ tells us how quantum mechanical states evolve. $H$ is the Hamiltonian, which describes how a particle interacts with a system.

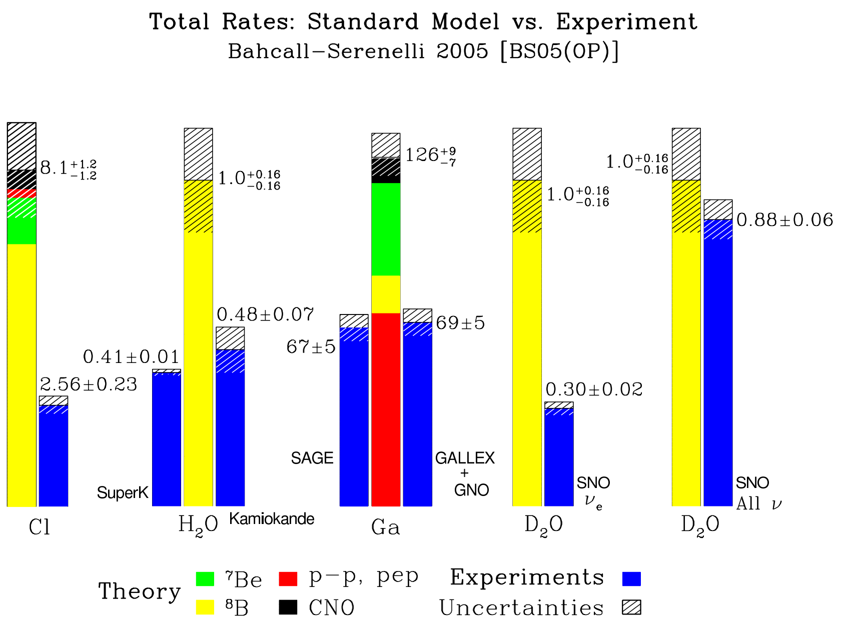


Figure 2-4: Theoretical expectations and results of neutrino counts for different experiments, from [25]

We are free to pick whatever basis we like to describe the system, and often choose one of energy eigenstates, i.e. those states where $H\psi = E\psi$. If we have three neutrinos, with masses $m_1$, $m_2$ and $m_3$, then we have a general superposition state (in an eigenstate of $\vec{p}$) evolving as

$$\psi_\nu = e^{i\frac{1}{\hbar}t\sqrt{m_1^2c^4+p^2c^2}}\,|\nu_1\rangle + e^{i\frac{1}{\hbar}t\sqrt{m_2^2c^4+p^2c^2}}\,|\nu_2\rangle + e^{i\frac{1}{\hbar}t\sqrt{m_3^2c^4+p^2c^2}}\,|\nu_3\rangle \tag{2.2}$$

Now we run in to a difficulty: we need to know what the masses of the neutrinos are. These have been measured to be extremely small; the current limit is $m_\nu < 2$ eV [11]. Additionally, we need to ask which states we are talking about when we talk about the mass of a neutrino. These will not necessarily be the same as the flavor states - neutrinos are unusual particles that only interact weakly. We know from quark measurements that weak decays can change quark flavor, i.e. the weak interaction doesn't couple to the strong force eigenstates (the flavor states for quarks: up, down, charm, strange, top, bottom) but to linear combinations of them (see any text on elementary particle physics for more details, for example [29] and [30]). If the same is occuring for neutrinos, those neutrinos that we call $\nu_e$, $\nu_\mu$ and $\nu_\tau$ may be linear combinations of the states that freely propagate since we only observe them as a consequence of weak interactions.

To see the effects of neutrinos having a small but non-zero mass, we start by looking at the simplified case of a world with only two neutrinos. We do this for two reasons: it is easier to follow and understand, and turns out to be a good approximation to what actually occurs in many cases. Since neutrinos have energies much larger than their masses, they are ultra-relativistic and propagate at a velocity so close to $c$ we can't measure the difference. Thus we won't have the mass states separating spatially and we can ignore that concern. In addition, this means that we are justified in a Taylor expansion of the phase factor above for small masses

$$E = \sqrt{m_i^2 c^4 + p^2 c^2} \approx pc + \frac{m_i^2 c^4}{2pc} \approx E + \frac{m_i^2 c^4}{2E} \tag{2.3}$$

where we have also used the approximation $E \approx pc$ for an ultrarelativistic particle.

In general, if we have $n$ particles mixing, we will have

$$|\nu_\alpha\rangle = \sum_i U_{\alpha i}^* \, |\nu_i\rangle \tag{2.4}$$

where $\alpha$ indicates a flavor state, $i$ indicates a mass state and the complex conjugate is a convention. This $U$ must be unitary to conserve probability, so that particles are neither created nor destroyed by the mixing. In the case of two neutrinos, $U$ is $2 \times 2$ and all complex phases are unobservable, allowing us to write this as

$$\begin{pmatrix} |\nu_e\rangle \\ |\nu_\mu\rangle \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} |\nu_1\rangle \\ |\nu_2\rangle \end{pmatrix} \tag{2.5}$$

$\theta$ is called the mixing angle, and is a constant of nature measuring how the weak interaction mixes states. If a neutrino starts as a $\nu_e$, it will propagate as

$$|\psi\rangle = e^{i\frac{E}{\hbar}}\left(\cos(\theta)e^{i\frac{1}{\hbar}\frac{m_1^2c^4}{2E}t}\,|\nu_1\rangle + \sin(\theta)e^{i\frac{1}{\hbar}\frac{m_2^2c^4}{2E}t}\,|\nu_2\rangle\right) \tag{2.6}$$

Any neutrino interaction (and thus detection) will be via the weak force and hence in the flavor basis, so we transform back to that basis (by inverting equation 2.5 and substituting for $|\nu_i\rangle$ above). In addition, we note that if a particle is travelling at $c$ (a very fixed velocity), we can choose to parameterize in distance instead of time, i.e. we can substitute $t = \dfrac{L}{c}$. Then a particle starting in $\nu_e$ at $t = L = 0$ will, at some distance $L$ away, be in the state

$$\begin{aligned}|\psi\rangle = e^{i\frac{E}{\hbar}}\left(\cos^2(\theta)e^{i\frac{im_1^2c^3L}{2\hbar E}} + \sin^2(\theta)e^{i\frac{im_2^2c^3L}{2\hbar E}}\right)|\nu_e\rangle + \\ \cos(\theta)\sin(\theta)\left(e^{i\frac{im_2^2c^3L}{2\hbar E}} - e^{i\frac{im_1^2c^3L}{2\hbar E}}\right)|\nu_\mu\rangle\end{aligned} \tag{2.7}$$

Then the probability of detecting a $\nu_e$ given that it started as a $\nu_e$, called the survival probability $P_{ee}$ is

$$P_{ee} = |\langle\nu_e\,|\psi\rangle|^2 = \cos^4(\theta) + \sin^4(\theta) + 2\cos^2(\theta)\sin^2(\theta)\cos\left(\frac{c^3}{\hbar}\Delta m^2\frac{L}{2E}\right) \tag{2.8}$$

where $\Delta m^2 = m_2^2 - m_1^2$. We simplify this using double angle and half angle formulas to get

$$P_{ee} = 1 - \sin^2(2\theta)\sin^2\left(\frac{c^3}{\hbar}\Delta m^2\frac{L}{4E}\right) = 1 - \sin^2(2\theta)\sin^2\left(1.27\Delta m^2\frac{L}{E}\right) \tag{2.9}$$

where in the final formula we have evaluated the constants and forced units on the variables: $E$ in GeV, $\Delta m^2$ in eV$^2$, and L in km.

This tells us something very interesting happens - if neutrinos have different masses, then they will oscillate. A neutrino produced in one flavor state will have a probability of being measured as a different flavor state that varies sinusoidally with the distance traveled. This amplitude depends on two properties inherent to the neutrinos, their mixing angle $\theta$ and their mass-squared difference $\Delta m^2$. It is worth noting that since these show up as $\sin^2(x)$ terms, these neutrino oscillations are insensitive to the sign of $\Delta m^2$ and $\theta$. In addition, since $\sin(\pi - x) = \sin(x)$ an additional ambiguity exists for $\theta$ versus $\frac{\pi}{2} - \theta$.

This is a possible solution to the "solar neutrino problem" - neutrinos are created in the sun as $\nu_e$ at the rates that solar model predict, but are changing to other flavors on their way to the Earth. More precisely, they are emitted in flavor eigenstates, but these are not propagation eigenstates. As the neutrinos propagate, the relative phases between the mass eigenstates vary, so when they are projected back in to the flavor basis by a measurement, interference between the mass states has occurred and they are now in a mixture of flavor states.

Since we know there are three (light, active) flavors of neutrinos in nature, we should look at what occurs in the full three neutrino case. We start again by looking at the mixing matrix. It is now a $3 \times 3$ unitary matrix, which is significantly more complicated. We will be following [31] for this case. We write this mixing matrix as the product of three effective $2\nu$ mixing matricies

$$U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{23} & s_{23} \\ 0 & -s_{23} & c_{23} \end{pmatrix} \begin{pmatrix} c_{13} & 0 & s_{13}e^{-i\delta} \\ 0 & 1 & 0 \\ -s_{13}e^{i\delta} & 0 & c_{13} \end{pmatrix} \begin{pmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.10}$$

Where $c_{ij} = \cos(\theta_{ij})$, $s_{ij} = \sin(\theta_{ij})$, $\theta_{ij}$ is the mixing angle between mass states $i$ and $j$, and $\delta$ is a CP-violating phase. All complex phases other than $\delta$ are unobservable and removed.

This $3\nu$ case has much richer structure, but we know from observations that $\Delta m^2_{12} \ll \Delta m^2_{23} \approx \Delta m^2_{13}$. This simplifies the resulting expressions for oscillations, and lets different types of experiments probe the individual parts of the matrix more readily. Solar neutrino experiments are in looking at a regime where $\Delta m^2_{32} L/E \gg 1$ (and $\Delta m^2_{31} L/E \gg 1$). Since there is an initial spread in both neutrino energy and position (since the Sun is large), the faster oscillations associated with $\Delta m^2_{31}$ and $\Delta m^2_{32}$ "average out" over the many oscillations between $\nu$ creation and detection.

This simplifies the survival probability to

$$P_{ee} = \cos^4(\theta_{13}) \left(1 - \sin^2(2\theta_{12}) \sin^2\left(\frac{c^3}{\hbar}\Delta m^2 \frac{L}{4E}\right)\right) + \sin^4(\theta_{13}) \tag{2.11}$$

This is just the $2\nu$ case with a multiplicative and an additive constant. An additional piece of information simplifies this further: measurements have shown that $\sin^2(2\theta_{13}) < 0.19$ (at 90% confidence) [11], so we make a very small error by ignoring these additional terms.

This all assumes that we are detecting a monoenergetic beam of neutrinos from a well-localized source. Since neutrinos are generated over a range of positions in the Sun and a range of energies, these need to be averaged over [32]. The net effect is that the vaccuum oscillations above, which depend on the coherence of the neutrinos, will average out. This leaves an expectation of an incoherent sum, giving

$$P_{ee} = k_1|U_{e1}|^2 + k_2|U_{e2}|^2 + k_3|U_{e3}|^2 \tag{2.12}$$

where $k_i$ is the fraction of neutrinos in the $i^{th}$ mass state reaching the detector, and $U_{ei}$ is the element in the mixing matrix linking $\nu_e$ and $\nu_i$. Note that this is simply a sum of the probabilities of each $\nu_i$ being measured as a $\nu_e$.

## 2.4 MSW Effect in the Sun

The previous section gave us one possible solution to the solar neutrino problem: vacuum oscillations. This, unfortunately, is not sufficient. We now look at the effect on neutrinos of passing through solar material on their way to our detector, the second on our list of things that happen to the neutrino between creation and detection.

It was first proposed by Wolfenstein [33] and later expanded upon by Mikheyev and Smirnov [34] that passing through matter can affect the oscillations of neutrinos. Conceptually, as neutrinos pass through matter, they undergo coherent forward scattering, i.e. scattering where their direction of propagation and state do not change appreciably. This is analogous to the index of refraction for light in a transparent medium. Neutral current, Z mediated processes affect all of the states equally, so they have no affect on oscillations. Charged current, W moderated interactions, on the other hand, do not. Since they couple neutrinos to their charged lepton partners, the fact that matter contains many electrons and no muons or taus means that $\nu_e$'s will experience a charged current effect while $\nu_\mu$'s and $\nu_\tau$'s will not. This creates an effective potential that only $\nu_e$'s will feel, which we will call $V_e$.

Since neutrino interactions are so weak, we can safely assume that the interaction strength will vary proportionally to the number of scatterers (electrons) available, so our effective potential will have the form $V_e \propto N_e$. The expression usually used is $V_e = \sqrt{2} G_F N_e$, where $G_F$ is the Fermi constant, a measure of the strength of the weak interaction at low energies. Note that this ignores absorption effects, which are negligible.

We will again start by adopting a $2\nu$ model for clarity. We write down our vacuum propagation matrix explicitly this time, and we switch to units where $\hbar = c = 1$ for simpicity

$$H_{mass} = \begin{pmatrix} E + \frac{m_1^2}{2E} & 0 \\ 0 & E + \frac{m_2^2}{2E} \end{pmatrix} \tag{2.13}$$

in the mass state basis. Recalling from before that the E term contributes an irrelevant overall phase, we drop it. Going a step further, this tells us that any Hamiltonian of the form $H - k\mathcal{I}$, where $\mathcal{I}$ is the identity matrix, will change the solution only by an overall phase. We then can rewrite our Hamiltonian as

$$H'_{mass} = H - (E + \frac{m_1^2 + m_2^2}{4E})\mathcal{I} = \begin{pmatrix} -\frac{\Delta m^2}{4E} & 0 \\ 0 & \frac{\Delta m^2}{4E} \end{pmatrix} \tag{2.14}$$

To convert to the flavor basis, basic quantum mechanics tells us

$$H_{\alpha\beta} = \langle \nu_\alpha | \, H \, | \nu_\beta \rangle = \sum_{i,j} \langle \nu_\alpha | \nu_i \rangle \langle \nu_i | \, H \, | \nu_j \rangle \langle \nu_j | \nu_\beta \rangle = (UHU^\dagger)_{\alpha\beta} \tag{2.15}$$

where U is our mixing matrix from before.

$$H_{flavor} = UH_{mass}U^{\dagger} = \begin{pmatrix} -\frac{\Delta m^2}{4E}\cos(2\theta) & \frac{\Delta m^2}{4E}\sin(2\theta) \\ \frac{\Delta m^2}{4E}\sin(2\theta) & \frac{\Delta m^2}{4E}\cos(2\theta) \end{pmatrix} \tag{2.16}$$

Since our effective potential only affects $\nu_e$'s, it adds to the Hamiltonian only in the first entry, giving (after trig substitution)

$$H = \begin{pmatrix} -\frac{\Delta m^2}{4E}\cos(2\theta) + V_e & \frac{\Delta m^2}{4E}\sin(2\theta) \\ \frac{\Delta m^2}{4E}\sin(2\theta) & \frac{\Delta m^2}{4E}\cos(2\theta) \end{pmatrix} \tag{2.17}$$

This Hamiltonian governs the propagation of neutrinos in matter. To find out the eigenstates of propagation, i.e. the new "effective mass" states, we look at the eigensystem of this matrix. The eigenvectors tell us the flavor composition of the new propagating states. This will give us the unitary transformation between flavor and propagating state, which we can write as

$$\begin{pmatrix} |\nu_{1m}\rangle \\ |\nu_{2m}\rangle \end{pmatrix} = \begin{pmatrix} \cos(\theta_m) & -\sin(\theta_m) \\ \sin(\theta_m) & \cos(\theta_m) \end{pmatrix} \begin{pmatrix} |\nu_e\rangle \\ |\nu_\mu\rangle \end{pmatrix} \tag{2.18}$$

where $\theta_m$ is the mixing angle for the states in matter. The relationship between $\theta_m$ and other parameters was derived by Wolfenstein in [33], but we use an expression that makes relations we are interested in more obvious (following most modern treatments, in this case [35] and [36]). This gives us

$$\tan(2\theta_m) = \frac{\frac{\Delta m^2}{2E}\sin(2\theta)}{\frac{\Delta m^2}{2E}\cos(2\theta) - \sqrt{2}G_F N_e} \tag{2.19}$$

Two interesting and relevant special cases exist. The most obvious is the resonant condition $\dfrac{\Delta m^2}{2E}\cos(2\theta) = \sqrt{2}G_F N_e$, which results in $\theta_m = \frac{\pi}{4}$. This is maximal mixing, where $\nu_e$ and $\nu_\mu$ are equal mixtures of $\nu_1$ and $\nu_2$, and was first discussed in [34]. This can occur even for very small vacuum mixing, and was proposed as a possible explanation for the solar neutrino problem: resonant oscillations almost certainly occur somewhere in the sun, given the large range of densities from the solar core to the solar surface, and would thoroughly mix $\nu_e$'s with $\nu_\mu$'s, even nearly completely converting $\nu_e$'s to $\nu_\mu$'s under certain conditions. However, experimental results indicate a large mixing angle $\sin^2(2\theta_{12}) = 0.86^{+0.03}_{-0.04}$ [11], $\theta_{12} \approx 34°$, making the latter not possible.

The second interesting special case is for very large $N_e$. In that case, the matter mixing angle limits to $\theta_m = \frac{\pi}{2}$. That is to say, $\nu_e \approx \nu_2$. If the solar core is dense enough for this to happen, then the emitted $\nu_e$'s will be $\nu_2$'s. If the change in the density of the sun is slow, we can make an adiabatic approximation which says that a particle in an eigenstate of propagation will remain in that eigenstate, even though the eigenstates themselves are changing. Then all $\nu_e$'s in the sun will leave the solar surface as $\nu_2$'s and will not experience vacuum oscillations.

From [35] we find that the numerical value of $V_e$ is

$$V_e = \sqrt{2} G_F N_e \approx (0.76 \cdot 10^{-7}) \frac{\rho_e}{N_A/\text{cm}^3} \frac{\text{eV}^2}{MeV} \tag{2.20}$$

where $\rho_e$ is the electron density and $N_A$ is Avagadro's number. Assuming a solar core density of $\rho_e \approx 100\ N_A/\text{cm}^3$ from solar models [15], and $\Delta m^2_{21} = 8 \cdot 10^{-5}$ eV$^2$ and $\sin^2(2\theta_{12}) = 0.86$ from [11], the $\theta_m$ as a function of neutrino energy is shown in Figure 2-5. Resonance occurs at $\approx 2$ MeV. SNO is sensitive to higher energy neutrinos, mostly in the range $6 - 20$ MeV and with a most likely energy of 10 MeV. This gives $\theta_m$ in the range 65° to 80°, corresponding to $|\langle \nu_2 | \nu_e \rangle|^2 \approx 0.82$ to 0.98. At higher energies, then, the $\nu_e \approx \nu_2$ is a good approximation, and becomes less so at lower energies. This is particularly important in comparing the results of different *types* of experiments, many of which are sensitive to lower energies than SNO and cannot make the $\nu_e \approx \nu_2$ approximation at all. This also suggests that SNO should see an energy dependence in oscillations from this effect, independent of those from vacuum oscillations.

It is worthwhile to take a moment to look at the adiabaticity issue. In general, if we have a set of eigenstates that are changing with time (or distance, for a propagating

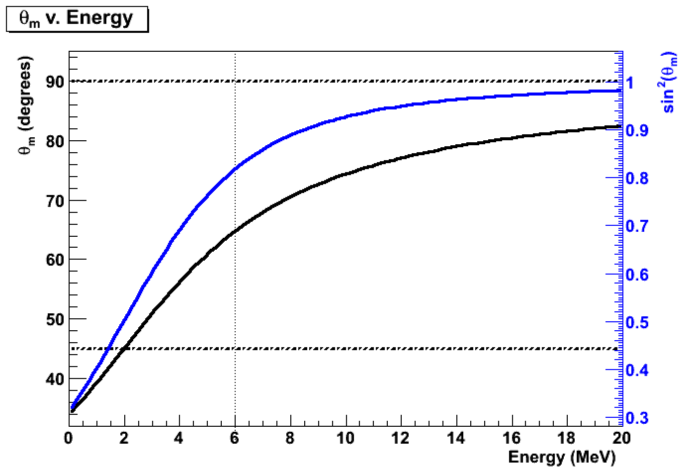


Figure 2-5: $\theta_m$ v. $E_\nu$ at solar core densities. The dotted vertical line is SNO's lower energy threshold (a few lower energy neutrinos) and the two horizontal dashed lines are $\theta_m = 45°$ and $\theta_m = 90°$, maximal mixing and $\nu_e \approx \nu_2$ respectively.

object), there is some probability $P_{jump}$ of crossing or jumping from one eigenstate to the other. The size of $P_{jump}$ depends on how quickly the eigenstates are changing, relative to the system's evolution if the eigenstates weren't changing. In the case of neutrinos in the sun, this is comparing the rate of change of $\theta_m$ versus the oscillation length. This is worked out in [35], which characterizes the needed condition as

$$\tilde{\gamma} = \frac{\pi \Delta m^2 / E_\nu}{|d(lnV_e)/dx|_{res}} \gg 1 \tag{2.21}$$

where the $res$ subscript indicates the constant should be evaluated at the MSW resonance density, where jumping is most likely. This constant is of order $10^6$ for the sun, so it is safe to assume that the mass composition of neutrinos exiting the sun will be the same as those generated in the sun. The flavor compositions, of course, will be different.

The three neutrino case is, again, much more complicated. However, in [37] we find that the measured values of $\Delta m^2_{21} \ll \Delta m^2_{23}$ and $\sin^2(\theta_{13})$ simplify things substantially. We find $|\nu_{3m}\rangle = |\nu_3\rangle$, so the $\nu_3$ effectively decouples from the matter mixing. We get an effective two flavor case, with $V_{e,eff} = \cos^2(\theta_{13})V_e$, with the two flavors now being $\nu_e$ and $\nu_x$, where $x$ is some combination of $\mu$ and $\tau$. An additional concern arises: if $\theta_m$ is changing across the Sun, then the mass composition of the created $\nu_e$'s will not be uniform and will need to be averaged over. This averaging results in

$$P_{ee} = \sin^4(\theta_{13}) + \frac{1}{2}\cos^4(\theta_{13})(1 + D_{3\nu}\cos(2\theta_{12})) \tag{2.22}$$

where

$$D_{3\nu} = \int_0^{R_{sun}} f(r)\cos(2\theta_{12m}(r))(1 - 2P_{jump})dr \tag{2.23}$$

where $f(r)$ is the probability distribution for a $\nu_e$ being generated at a radius $r$ in the sun, $R_{sun}$ is the radius of the Sun, $\theta_{12m}$ is the two neutrino matter mixing angle for the effective potiential $\cos^2(\theta_{13})V_e$ and $P_{jump}$ is negligibly small in the Sun. $D_{3\nu}$ must be calculated numerically, but can be approximated reasonably well by assuming an exponentially decaying matter density in the Sun.

## 2.5 MSW Effect in the Earth

Finally, we arrive at the third thing that can happen on neutrinos that reach our detector from the Sun: they can pass through the Earth. The MSW effect described in the previous section affects neutrinos in the Earth as well. Since the Earth is much less dense, so we would expect the effect to be much smaller. However, the density of the Earth is *not* changing adiabatically. There are very sharp transitions in density from air to ground, then from mantle to core, then back again. The Earth's density profile is shown in Figure 2-6. This will give a non-zero $P_{jump}$, which will change the relative proportions of the mass eigenstates and hence the relative proportions of $\nu_e$'s versus other flavors. In addition, the regenerated $\nu$'s will have a coherent portion, but much of it will average out. As explained in [37], this non-adiabaticity is the dominant effect. This is quite reasonable: if the change were adiabatic, we would have the same situation we have in the sun, where the neutrinos maintain their mass eigenstate composition but the corresponding flavor composition changes. When they are detected, they will have the flavor composition corresponding to the density of the material surrounding the detector. Since the Earth has low density at the crust, this would be the same as the vaccuum composition.

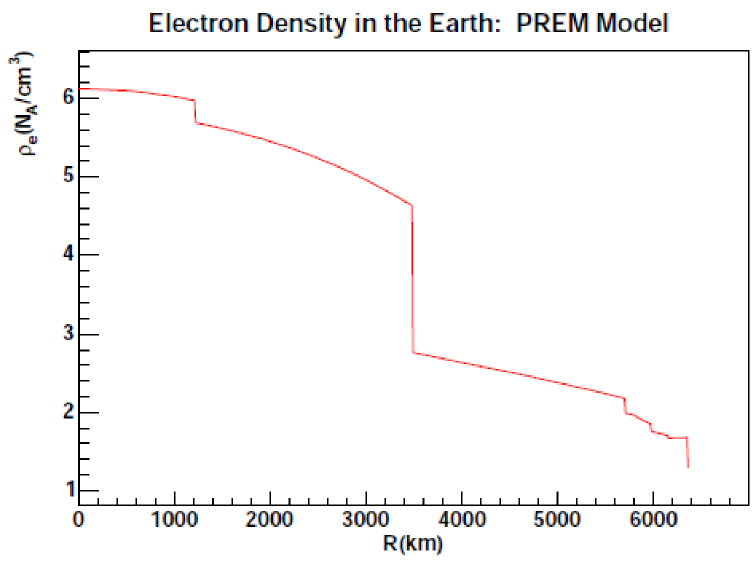


Figure 2-6: The density profile of the Earth, from the PREM model [38]. Figure from [39].

This creates a difference between $P_{ee}$ for neutrinos that must pass through the Earth and those that do not. This is known as the "Day/Night effect", since neutrinos only have to pass through the Earth to reach a detector when the bulk of the Earth is between the Sun (the source) and the detector, i.e. at night. The resulting difference is

$$P_{ee,Night} - P_{ee,Day} = -2\cos^6(\theta_{13}) D_{3\nu} \frac{EV_e}{\Delta m^2_{21}} \sin^2(2\theta_{12}) \sin^2\left(\frac{\Delta m^2_{21}}{4E} L\right) \tag{2.24}$$

where $D_{3\nu}$ is the same as before, for the Sun, and $V_e$ is for inside the Earth for the path being evalutated.

For any experiment, counting statistics require integrating across a range of solar positions. Usually, experiments divide their data in to a "Day" bin and a "Night" bin, which integrate over all neutrinos that do not have to pass through the Earth and all those that do, respectively. For the latter, this means integrating over a range of different paths through the Earth, ranging from relatively glancing passes (if the sun has just set) to going all the way through the Earth including the core. Both the computation of values needed for Eq. 2.24 and integrating over these paths is usually done numerically, though [40] makes a set of analytic approximations that reproduce the numerical results well. In addition, each detector has a unique energy and angular response, which also tend to average over some of the parameters and must again be dealt with numerically. This is done in [37], which gives a prediction of 4.5% for SNO.

At SNO, this is dealt with by the PhysInt group. They simulate the generation of neutrinos in the sun and their passage through the Earth, taking the MSW effect in to account. This allows them to find the predicted amount of both primary signal (amount of $\nu_e$ relative to $\nu_x$ detected) and the day/night effect for a given set of neutrino mixing parameters. This allows them to find those values of the mixing parameters that correspond to the measured results of SNO.

# Chapter 3

# Sudbury Neutrino Observatory

The data used in the analysis presented in this thesis is from the Sudbury Neutrino Observatory (SNO), a large water-Cherenkov neutrino detector. SNO is unique in its use of heavy water, $D_2O$, as its neutrino target, which gives it the ability to separately measure the flux of $\nu_e$ and the flux of $\nu_x$ (all active flavors of neutrinos) from the Sun. This benefit to using heavy water was first recognized by Chen [41], who was instrumental in the creation of SNO. This advantage over other detectors allowed it to definitively resolve the solar neutrino problem in 2001 [42, 43], showing that the solar models predicted the correct number of neutrinos, but that their flavor had changed by the time they reached the detector.

## 3.1 Detector

SNO looks for neutrino interactions in water. It contains both a region of heavy water, the primary target, and two regions of light water ($H_2O$). In all three of these regions, particles that are charged and moving faster than the local speed of light in water are detected, via light detectors sensing their Cherenkov radiation. This radiation is conceptually similar to a sonic boom; it is an effect caused by a charged particle affecting the medium it is travelling through faster than it can respond. This leads to electromagnetic constructive intereference and light emission in a cone in the direction of propagation of the particle. The emitted light is strongest in the UV

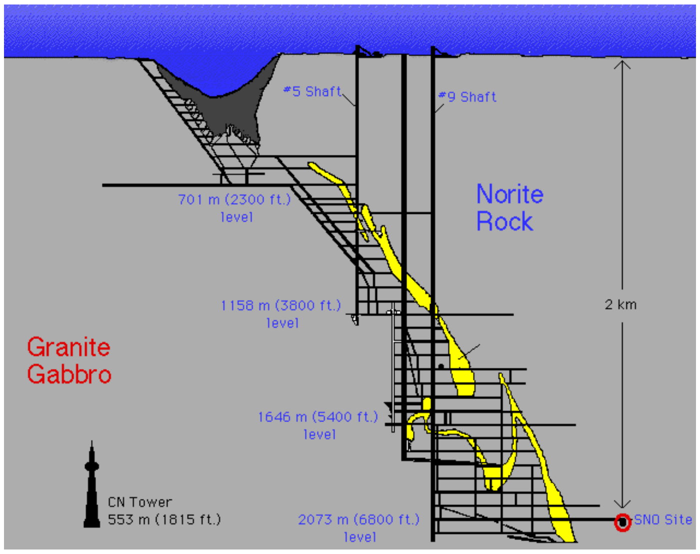


Figure 3-1: Diagram of SNO detector's location in the Creighton mine. Taken from the SNO image library.

portion of the spectrum, but extends to lower frequencies. The angle of the cone is a property of the medium, for $D_2O$ it is approximately 42°. Electrons in water decelerate rapidly, giving a narrow circle of photons when detected.

The detector is located in the Creighton Mine, owned by Vale Inco Ltd., near Sudbury, Ontario. See Figure 3-1 for a diagram of the mine area. This is an active nickel mine, with the detector at a depth of 2092 meters ($5890 \pm 94$ meters water equivalent), making it one of the deepest in the world. This depth is primarily needed to shield the detector from incoming cosmic ray muons. The overburden reduces the rate to approximately three per hour, compared to the surface rate for a detector this size of order $10^6$ muons/s. This shielding is aided by the flat overburden of rock, making all paths to the detector have at least 2092 meters of rock as shielding. In the mine there is a great deal of rock dust and other particulates, so the SNO collaboration maintains a clean room environment near the detector [44].

Figure 3-2 is a diagram of the SNO detector showing its major parts (except the NCDs). The basic design is a spherical volume of 99.92 % isotopically pure $D_2O$ surrounded by photomultiplier tubes (PMTs) that look for light signals from the $D_2O$ region. The detector geometry is approximately spherical, arranged in a set of spherical shell layers. The 1000 tonnes of heavy water is the innermost, housed in a 12 meter diameter vessel of UV-transparent acrylic, appropriately named the acrylic vessel (AV). The AV is nearly spherical, except for a hole in the top that allows access to the $D_2O$, with a cylindrical section leading from this hole to the top of the detector (collectively called the "neck"). This is surround by a layer of light water, approximately 1700 tonnes, that acts to shield the AV and heavy water from external radiation, as well as physically support the AV. This is housed in the photomultiplier support structure (PSUP), an 18 meter diameter stainless steel geodesic sphere structure which containts the PMTs. There are 9456 PMTs looking "inward", toward the heavy water, that act as the principle signal detectors. Each PMT has a light collector, which increases the total light collection coverage to 59%. In addition, there are 91 PMTs facing "outward", away from the heavy water, which look in the much larger external $H_2O$ region (approximately 5700 tonnes, physically separated from and not allowed to contact the inner light water by the PSUP) to look for cosmic ray muons emitting Cherenkov radiation to aid in vetoing these muons. In addition, this large volume of water acts to shield the detector from radiation from the rock walls surrounding the detector cavity, particularly neutrons and energetic photons from radioactive decays. Many details about the design and construction of SNO can be found in [44], from which we draw much of the following discussion.

### Water

To preserve the proper functioning of the detector, both the light and heavy water must be kept extremely clean. This is to both keep out radioactive contaminants that give rise to backgrounds, particularly radon and its daughters, and to prevent the buildup of any particulates or biological agents that can cloud the water. This is accomplished with an elaborate filtration process for the each of the water streams, which are never allowed to come in contact.

The light water is continuously drawn from the inner light water region and piped to a utility room near the detector. There it goes through a several stage purification process and is returned to the detector. The process begins with irradiation with 185 nm UV light to kill any bacteria and break up organic compounds, then a set of ion

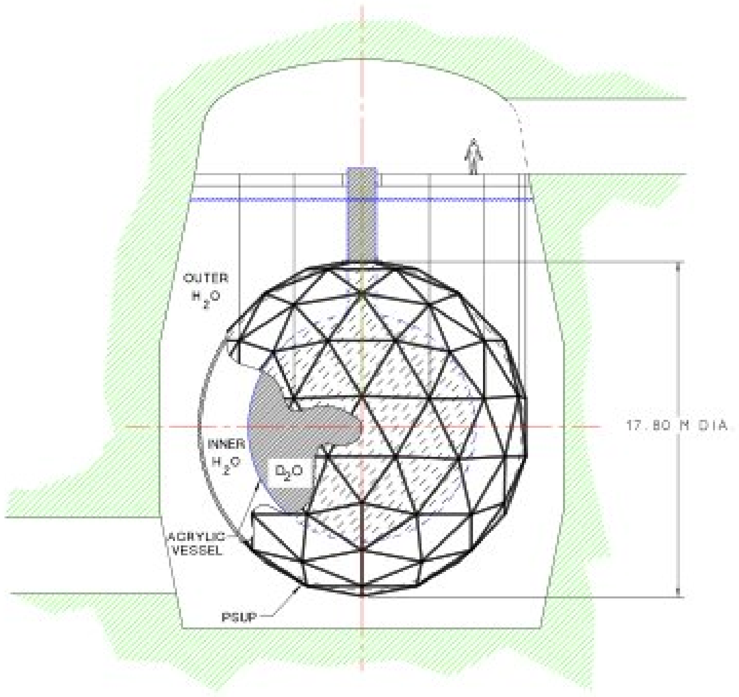


Figure 3-2: Diagram of the SNO detector. Note that the NCDs are not pictured. Taken from the SNO image library, appeared in [44]

exchange columns to removed dissolved ions. This is followed by a degassing unit designed to remove $O_2$ and radon, but that removes all dissolved gasses. The water is then regassed with ultrapure $N_2$, to avoid electrical problems from gas diffusing out of the PMTs. The water is then filtered through a set of 0.1 $\mu$m filters to remove particulates, followed by exposure to 254 nm UV light to kill any bacteria introduced by the cleaning process. The water is then run through a battery of purity and clarity tests before finally being chilled to 10°C and returned to the detector.

The light water in the outer region gets less purification treatment, as it is inherently dirtier by virtue of being in contact with the walls of the cavity and the various cables and supports associated with the external part of the PSUP. It is, however, periodically purified using a similar system. In addition, it is kept from coming in contact with the cavity walls by a plastic liner.

The $D_2O$ has a much more intensive purification process. This process includes several stages of reverse osmosis filtration, several ultrafiltration units, and a degassing unit. Here there is no need to regas the water. The water is then sent through a stringent set of assays, to test the density, conductivity, pH, clarity and levels of various contaminants. The design goal was to have sufficiently low levels of uranium and thorium decay chain elements to assure that less than 10% of neutral current events (described below) are from backgrounds. This results in limits of $3 \times 10^{-15}$ g/g of thorium and $4.5 \times 10^{-14}$ g/g uranium. In the NCD phase (see Section 3.2), these levels were measured via the assays plus in-situ measurements to be $(0.58 \pm 0.35) \times 10^{-15}$ g/g thorium and $(5.10 \pm 1.80) \times 10^{-15}$ g/g urainum [45], well below the target levels.

### Acrylic Vessel

The AV houses the heavy water. It is made of a UV transparent acrylic, except for the 1.46 m diameter “chimney” or “neck” region, which is UV opaque. It is spherical in shape and 12 m across, with the only access to the water volume through the neck. It consists of 122 acrylic panels joined together with a special adhesive; this design allowed it to be assembled in the cavity. Most of the panels are 5.6 cm thick, except along the equator of the sphere, where a set of 11.4 cm thick “belly” plates were used to attach ropes to support the weight of the sphere. The acrylic used was chosen for its optical properties, long term stability and low radioactive contaminant levels.

### PMTs and PSUP

The PMTs are the primary detector elements for SNO (in the NCD phase are complimented by the NCDs). The glass used was custom designed and handblown for SNO (and LSND) to be extremely radiopure. The PMTs themselves were Hamamatsu R1408’s with a waterproof enclosure. These were chosen for their excellent electronic characteristics, with a timing resolution of 1.7 ns and a mean noise rate of 500 Hz. As mentioned, each PMT has a light concentrator, increasing the photocathode coverage to about 59%. To protect the PMTs from the Earth’s magnetic field, 14 field-compensation coils are embedded in the cavity walls, cancelling out the horizontal components.

These PMTs (and associated hardware such as readout cables) are housed in the PMT support structure (PSUP). It is a 889-cm radius geodesic sphere of stainless steel, with a hole in the top to allow the chimney to pass through. It is designed to be impermeable to water, to separate the inner $H_2O$ volume from the outer $H_2O$ volume. The PSUP contains a housing for each PMT to support it structurally, stabilize the direction it is pointing and separate its photosensitive face, which must face the $D_2O$, from its electronics end, which is attached to a cable in the outer $H_2O$. This separation is both in terms of keeping the water separate and keeping light from one region getting in to the other.

#### Electronics and DAQ

The electronic pulses from each PMTs are sent through a separate cable to a set of electronics outside of the detector making up the Data Acquisition System (DAQ). The DAQ serves primarily to decide whether an event is sufficiently physics-like to record, and to store that event. The primary triggers to record an event occurs when 16 PMTs fire within 100 ns (the transit time for light reflecting off of the AV) and when the total charge collection in all PMTs is greater than 150 photoelectrons. A secondary trigger occurs at 5 Hz to allow for accurate measurement of backgrounds, both physical and instrumental. All triggers together led to a trigger rate between 15 and 20 Hz. Which trigger caused the event is recorded.

Accurate timing is critically important at SNO. The signal from the GPS system is used for primary, absolute timing. This serves as a 10 MHz clock, which is fed from the surface receiver to the detector. A secondary 50 MHz clock, located underground near the detector, is used for relative timing between events.

### 3.1.1 Neutrino Interactions

As SNO is only sensitive to energetic charged particles and neutrons, to detect a neutrino one or the other must be generated. Since only the $D_2O$ region is used for neutrino detection, we look at neutrino interactions with heavy water. Given the restriction of the generation of particles with high velocity, only three interaction routes with heavy water are available.

### Elastic scattering

The first of these, elastic scattering (ES), is also available in light water, and is the only detection method available to most water Cherenkov detectors. It is the elastic scattering of a neutrino off of an electron

$$e + \nu_x \to e + \nu_x \tag{3.1}$$

where $x = e$, $\mu$ or $\tau$ is the flavor of the neutrino. This imparts energy to the electron, pushing it above the Cherenkov threshold. While all three neutrino flavors participate in this reaction, it is dominated by $\nu_e$. All three neutrinos have a Feynman diagram involving the Z for this, but only $\nu_e$ has an additional diagram involving the W (the diagrams are shown in Figure 3-3). This results in approximately 6 times a much cross-section for $\nu_e$ at a neutrino energy of a few MeV, the region of interest for water Cherenkov solar detectors. SNO separates its simulations in to an electron ES component (ES) and a muon and tau component ($\text{ES}_{\mu\tau}$) to simplify dealing with this distinction.

The scattered electron is detected. For an incident neutrino within SNO's energy range, the scattering cross section is very strongly peaked in the direction of the incident neutrino. This behavior is derived in [46], with results for a scattering imparting an electron with 9 MeV shown in Figure 3-4. This varies a bit with energy, but even at the lowest energy in SNO's analysis window (6 MeV), 90% of electrons are scattered within 13°. This strong direction correlation is very useful in identifying ES events, see section 4.3.

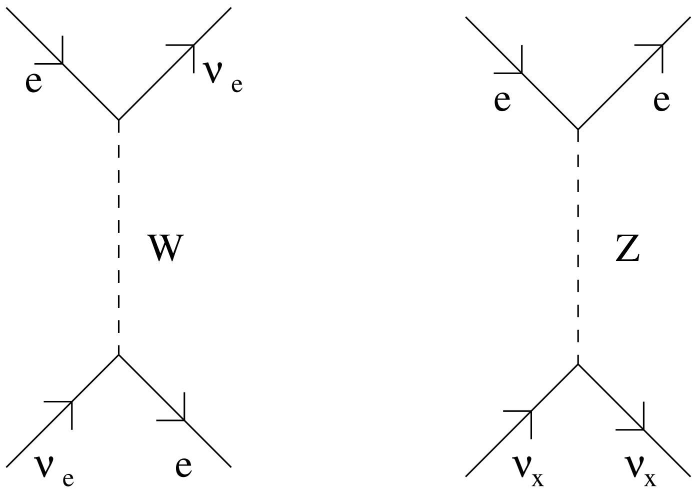


Figure 3-3: The Feynman diagrams for the elastic scattering interaction. $x = e, \mu, \tau$ is a neutrino of any flavor. Time is increasing to the right.

### Charged Current

The second process available is the charged current (CC) reaction. This involves a $\nu_e$ interacting with a deuteron to break it apart via

$$d + \nu_e \rightarrow p + p + e^- \tag{3.2}$$

This only has a W-exchange diagram and is only available for $\nu_e$, as the corresponding processes for $\nu_\mu$ or $\nu_\tau$ would involve a $\mu$ or a $\tau$ in the final state and the neutrinos from the Sun are not sufficiently energetic for this. This is extremely important, as it gives a pure measurement of the $\nu_e$ flux. In [42], the CC and ES measurements were compared as evidence for non-$\nu_e$ neutrinos in the solar neutrino flux.

The electron is detected via its Cherenkov radiation, but the protons are too massive to gain enough energy to radiate. Working out the cross section here we find that the direction of electron emission is weakly anti-correlated with the neutrino direction, but the energy is strongly correlated [47]. Breaking up the deuteron takes energy, with the final state having 1.442 MeV less kinetic energy than the initial state (giving a lower limit on the neutrino energy that can initiate this reaction).

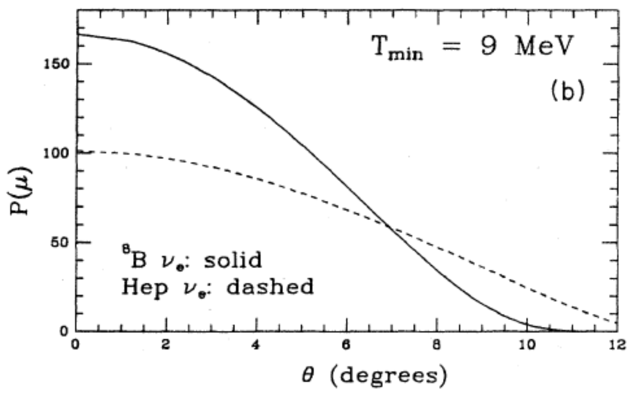


Figure 3-4: Scattering probability versus angle between incident neutrino and scattered electron, for the ES process. Vertical axis is in arbitrary units and $P(\mu)$ is the differential probability per unit of $\cos(\theta)$. The solid line is relevant for SNO, and shows that the electron is very likely to be scattered within a few degrees of the neutrino direction. $T_{min}$ is the energy of the scattered electron. From [46].

### Neutral Current

The third process available is the neutral current (NC) reaction. This also involves a $\nu$ breaking up the deuteron, but with a different final state

$$d + \nu_x \rightarrow p + n + \nu_x \tag{3.3}$$

This only has a Z-exchange diagram and is insensitive to the neutrino flavor, that is to say it occurs equally for all flavors. The final state here is more massive than in CC, having 2.224 MeV less kinetic energy than the initial state, again setting a minimum on the energy of the incident neutrino.

None of these particles give off Cherenkov radiation. Instead, the neutron thermalizes in the detector and is captured, and its capture gives off a detectable signal. Unfortunately, this thermalization process makes it impossible to tell the origin of the neutron, so any neturon that makes it in to the detector will appear to be part of the signal. In addition, any photon with sufficient energy can also cause this process, and several photons in the uranium and thorium decay chains have enough energy. See section 4.6 for more details.

## 3.2 Three phases

The signal given off by the neutron has been detected in three distinct ways at SNO. These methods involve reconfiguring the detector, a major undertaking, and divide the experiment in to "phases". In each of these phases the systematics and backgrounds of the detector were remeasured, to break correlations between phases and detect any changes due to the reconfiguration.

### 3.2.1 $D_2O$

In the first phase, the "$D_2O$ phase", lasted from November 1999 to May 2001. In this phase, the $D_2O$ itself was used to detect the neutron, which captured on a deuterium to form tritium. The capture reaction has a cross section of 0.52 millibarns and results in a single 6.25 MeV photon emitted from the excited final-state nucleus. This photon Compton scatters in the water, creating electrons that Cherenkov radiate and are detected.[1]

### 3.2.2 Salt

The second phase ran from July 2001 to September 2003. It was called the "salt phase" as two tons of NaCl were added to the heavy water. Chlorine has a much larger capture cross-section for neutrons of 0.44 barns, resulting in a much higher percentage of captured neutrons than in the previous phase. The result is a chlorine nucleus in an excited state that then gives off a cascade of photons with total energy 8.6 MeV, detected as in the previous phase.

### 3.2.3 NCD

The third phase ran from November 27, 2004 to December 28, 2006 with a livetime of 385.17 days. In this phase, the "NCD phase", the salt was removed from the heavy water and a set of neutron detectors were added. These "neutral current

[1]The subsequent tritium beta decay has a maximum energy of 18.6 keV, too low for the electron to Cherenkov radiate.

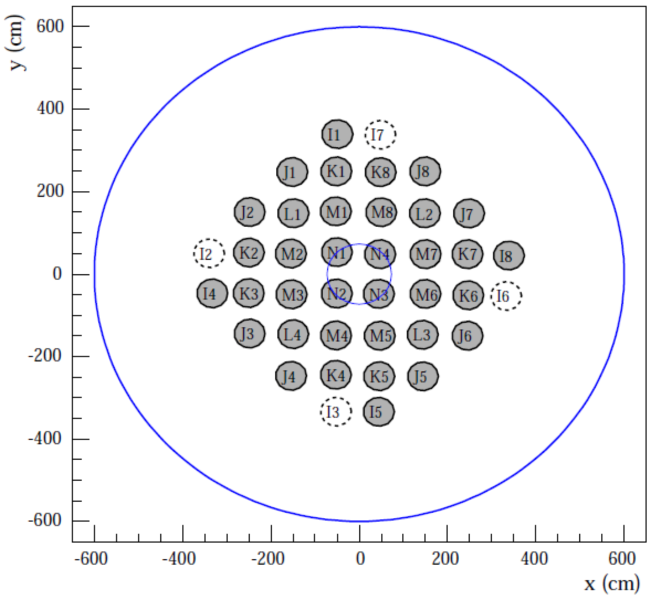


Figure 3-5: Diagram of the NCD array. The circular boundary is the equator of the AV. The grey circles mark $^3$He strings, the white circles mark $^4$He strings. Note that the markers indicating NCDs are not to scale. From [45].

detectors" (NCDs) were $^3$He proportional counters (nickel tubes with an interal gas mixture of 85% $^3$He and 15% $CF_4$) inserted directly in to the $D_2O$. These acted as an independent measure of the NC event rate, as both their detection mechanism and their electronics were distinct from the PMTs. The previous phases relied on light signals, so this serves as a truly indepenent measure and helps reduce correlations between phases. Note, however, that the capture of neutrons on deuterium (as in the first phase) occurs as well and forms part of the signal. The NCD and PMT measurements of NC act as cross-checks on each other.

As mentioned, the NCDs are directly in the heavy water. On Figure 3-2, they would appear as a set of vertical tubes between 9 and 11 meters long arranged in an array in the inner sphere of the detector. The array, as projected in to the equatorial plane of the AV, is shown in Figure 3-5. There are 36 $^3$He NCD strings[2], each an independent proportional counter operating via neutron capture

$$^3\text{He} + n \rightarrow p + {}^3\text{H} \tag{3.4}$$

[2]There are also four NCD strings with $^4$He, insensitive to neutrons, used for calibrations

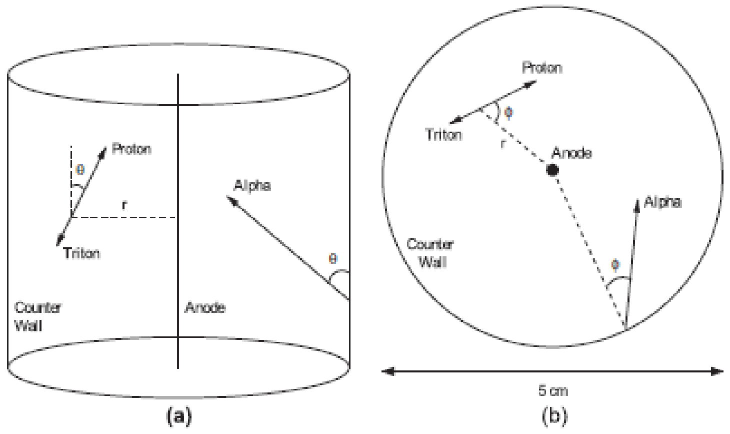


Figure 3-6: Cross section of an NCD, showing both a neutron interaction and an alpha particle. (a) shows the NCD from the side, (b) from above. From [48].

with the proton and the triton ionizing the gas, which is then amplified by the electric field in the tube. This signal is then read off the central wire, as per normal for a proportional counter. A cross section diagram is shown in Figure 3-6. This signal is then recorded by two separate sets of electronics, one of which measures the integrated signal and thus the energy of the event (the shaper-ADC), while the other records the entire signal pulse (the Multiplexer or MUX-scope). The energy spectrum seen in the ADC due to neutron capture is shown in Figure 3-8, note that it is peaked at 764 keV. A typical neutron MUX pulse is shown in Figure 3-7.

Radioactive decays in the tube also ionize the gas, giving rise to a complicated background. These are dominated by alpha decays, which are peaked at much higher energy (several MeV), but occur within the nickel body of the NCD. This leads to much lower measured ionization energies, as they deposit much of their energy in the nickel walls before they get to the gas. The net result is a relatively flat spectrum in energy and a different pulse shape. To improve signal to noise, SNO created an elaborate analysis to separate out the neutrons from backgrounds using this pulse shape difference, called the Pulse Shape Analysis (PSA). It is described in detail in [49], with the net result of extracting the number of neutrons detected by the NCDs. Details about how this is integrated in to this analysis are in Section 8.3.

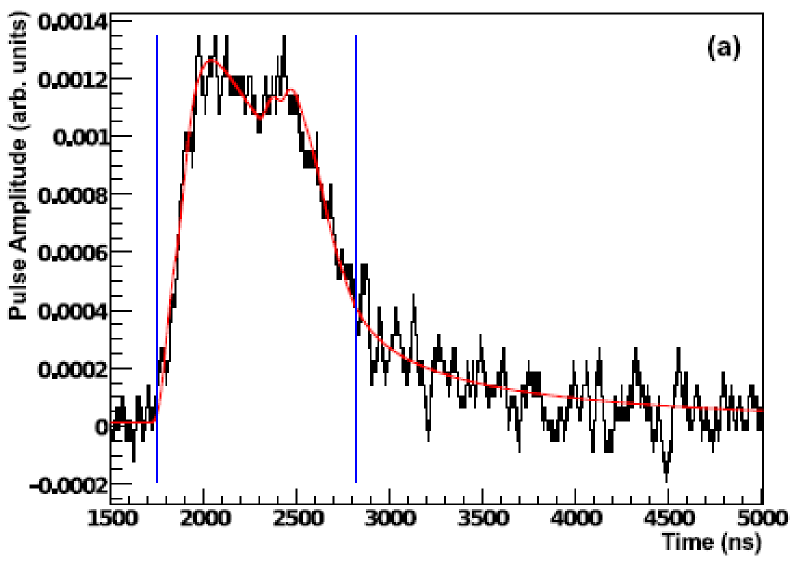


Figure 3-7: Sample pulse from an NCD. This pulse is a fairly typical pulse for a neutron capture. From [48].

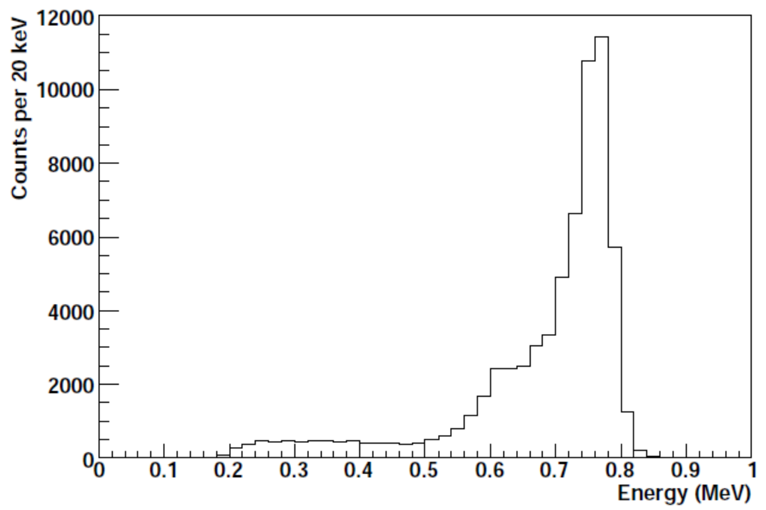


Figure 3-8: ADC energy spectrum for the NCDs, from neutron capture. The main peak at 764 keV is from both the proton and the triton depositing all of their energy as ionization. The long tail is from space charge effects and the proton or triton being absorbed by the NCD wall before it can deposit all of its energy. From [48], which also has more details.

# Chapter 4

# SNO Analysis

## 4.1 Overview

The goal of the SNO analysis is to convert the raw data (for the NCD phase described here, this includes the light in the detector and the signal in the NCDs) into a measurement of both the total flux of neutrinos from the sun and the proportion of that flux that are $\nu_e$. This is a many stage process, even discounting the work done in designing and running the detector. The major parts of SNO's analysis chain are summarized in Figure 4-1. Each involves a team scientists and many have multiple thesis projects as part of their development. As such, only a brief summary of most of them are possible in this thesis, except the "Sigex", "PDF Construction" and part of the "Systematics" sections, which are the focus of the analysis presented here.

#### Run Selection

The highest level in the chart is the "Run Selection". SNO's data taking is separated temporally in to runs, data taking periods generally between half an hour and a day long. This is done both for practical reasons such as operator shift changes, calibration measurements in the detector and activity in the nickel mine, and for analysis purposes of breaking the data in to smaller, more managable chunks over which the detector parameters are stable. The decision is then made about which runs to keep in the final data set for analysis. Those runs during or close in time to a calibra-

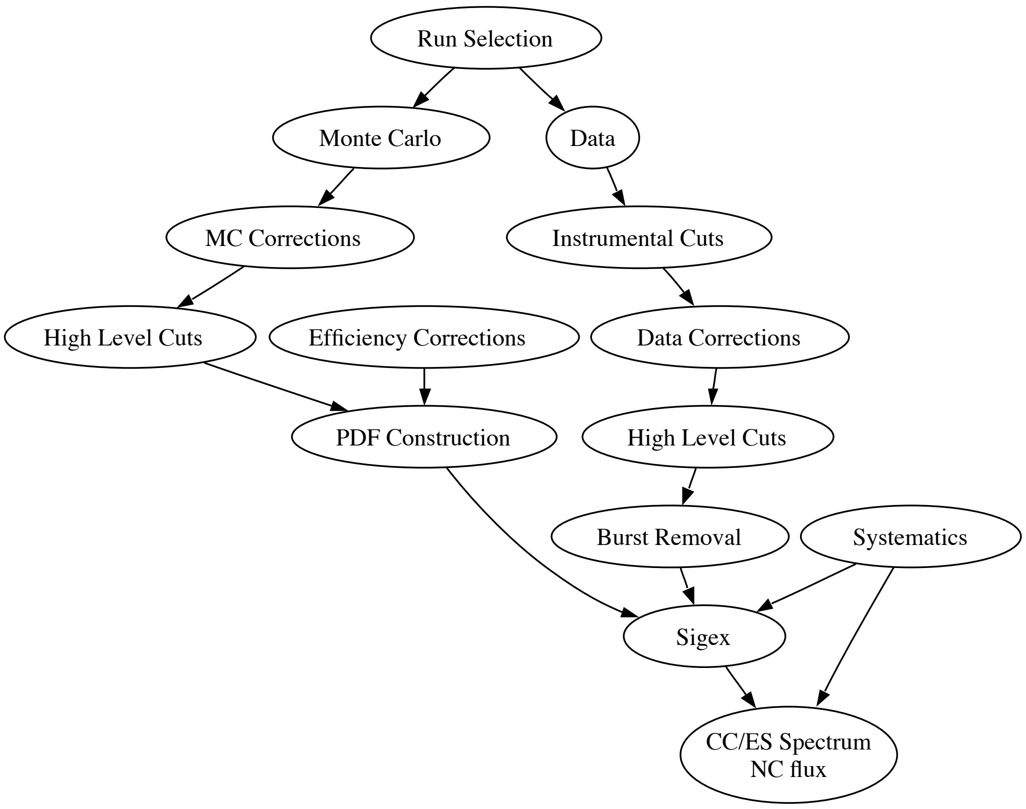


Figure 4-1: A graph showing the major steps in the overall SNO analysis chain. This image was taken from [49].

tion, which often involved adding radioactive materials to the $D_2O$, were removed, as were those that showed any form of hardware problems, those corresponding to detector maintenance and those with mine-activity related disruptions (such as power failures). After selection, 1834 runs with a total livetime of 385.17 days, divided in to 176.59 days and 208.58 nights, were in the final data set for the NCD phase. A detailed description of the run selection process can be found in [50].

### Data

For each run, the data is extracted from the raw PMT and NCD signals. This is the "Data" section in the chart. The NCD signals are processed through the PSA analysis (see Section 8.3) and the PMT signals are analyzed by a process called reconstruction. This looks at the spatial and timing distributions of the light from the PMTs to determine the physical event that took place. It is described in detail in [50, 51, 45]. We will summarize here.

The reconstruction generates three pieces of information about an event: the position, direction of travel and kinetic energy of the Cherenkov radiating electron. The position and direction reconstruction process starts by computing the PMT hit characteristics of Cherenkov light from an electron moving directly toward the hit region from a grid of points in the detector volume. Since electrons stop quickly in the detector (picoseconds), the electron's path is approximated as a point. The results most resembling the actual hit pattern (in both time and space) are then used as starting points for a fit, maximizing the same likelihood used to determine these best starting points. The best result out of these fits is then the reconstructed $x$, $y$, $z$, $u_x$, $u_y$ and $u_z$ (where the velocity $\vec{u}$ is normalized to 1, to only preserve direction information).

The kinetic energy reconstruction behaves similarly. It assumes the electron was at its reconstructed $\vec{x}$ with direction $\vec{u}$. The number of PMT hits (corresponding to the number of detected photons) is computed via a MC taking account of the optical properties of the detector for a given electron kinetic energy $E$. This $E$ is varied and fit for the best agreement with the measured number of PMT hits.

This gives a total set of variables for the data event of $x$, $y$, $z$, $u_x$, $u_y$, $u_z$ and $E$.

### Monte Carlo

Each run has a corresponding Monte Carlo run, generated using the simulation package SNOMAN. SNOMAN derives from EGS4 for electron and photon propagation, MCNP for neutron propagation and GEANT for muon physics, and contains code custom written by the SNO collaboration. In addition, the package was tested and tuned by comparing it to callibration measurements throughout the SNO project [44]. The MC is given the run times and the detector parameters and simulates the detector under those conditions, resulting in a prediction of what results we expect to see. This is accomplished by simulating the light signal expected from an interaction and running the reconstruction algorithm on this simulated light signal. This gives a great deal of additional information about each event - the parameters used to generate it. We know the underlying primary process (in fact, we separate CC, ES, etc into separate MC event sets for each run), the energy of the incident neutrino ($E_\nu$), and the true value for the event location, energy deposited and direction (denoted by $x_{true}$, $E_{true}$, etc.). Each of these is treated as an additional variable present for each MC event, giving the set of variables of $x$, $y$, $z$, $E$, $u_x$, $u_y$, $u_z$, $x_{true}$, $y_{true}$, $z_{true}$, $E_{true}$, $u_{x,true}$, $u_{y,true}$, $u_{z,true}$, $E_\nu$, plus several variables not used in the analysis.

While most aspects of the detector are modeled as well as possible, certain assumptions about the neutrino flux are made. The MC assumes the neutrino flux from the Sun in [16] and the $^8$B neutrino spectrum in [52] with no oscillations or other flavor changing effects. That is to say that the neutrinos that arrive at the detector are assumed to be 100% $\nu_e$ with an undistorted $^8$B beta-decay energy spectrum. The exception to this is the separate simulation for ES$_{\mu\tau}$, which assumes the undistorted $^8$B energy spectrum with 100% $\nu_\mu$ or $\nu_\tau$ (SNO is unable to distinguish between these two weak eigenstates).

#### MC Corrections

These are adjustments made to the MC reconstructed energy $E$ to correct for a known error in the Monte Carlo. They are described in [53] and are quite small.

#### Instrumental Cuts

For each event, an automated system evaluates its temporal position in a run, the state of the detector, its relation to other events and its spatial and temporal distributions of PMT hits. These are used to determine if it matches the profile for certain types of events and the results are stored to the “DAMN” bits. This cut rejects any event with certain bits set to true. The full set is provided in [49]. These cuts reject events too close in time to a muon passing through the system (to avoid followers) or the start or end of a run, events in the neck region, events with significant light in the outer water region (indicating a muon in that region), events with a PMT generating light or otherwise malfunctioning, events with electronic crosstalk between PMTs, and events too close in time to each other (indicating reflections or instrumental effect after-pulses).

### High Level Cuts

The same set of high level cuts are applied to both MC and data. These are cuts to further reject instrumental events and backgrounds. For the NCD phase data, this consists of two cuts. The first is on the "In Time Ratio" or ITR, which measures the fraction of PMT events that occur within a certain time of each other, related to the time for light to cross the detector, for a given event. We keep events with ITR $> 0.55$. The second is on $\beta_{14}$, a measure of the spatial isotropy of the PMT hits for an event. This involves a Legendre polynomial fit to the distribution of hit PMTs on the sphere of PMTs, with $\beta_{14}$ related to the coefficients of the first and fourth terms in the polynomial. This is described in more detail in [51, 53]. We keep events with $-0.12 < \beta_{14} < 0.95$.

### Efficiency Corrections

A few very small corrections are needed to make sure that the MC accurately predicts the correct number of events for each flux. For the NCD phase, there are eight of these, which give corrections to the expected number of events and correspond to better measurements of quantities made after the MC had been designed. These corrections are already included in the predictions presented in this analysis. The values for each type of correction are given in Table 4.1; the expected number of events in the given flux is multiplied by the numbers in this table.

### Data Corrections

Here the measured kinetic energy $E$ is corrected for known temporal and spatial variations in the detector. One correction is for the total number of active PMTs slowly changing as the experiment progresses and PMTs fail. This results in progressively more light being lost, leading to underestimates of the energy. In addition, a small vertical ($z$-direction) variation in the detector's PMT efficiency is corrected for. These are described in detail in [53].

| Correction | CC | ES | NC PMT | NC NCD |
|---|---|---|---|---|
| Mean Orbital Radius | 1.00000 | 1.00000 | 1.00101 | 1.00101 |
| Deuteron density | 1.01170 | 1.00000 | 1.00000 | 1.00000 |
| Number of deuterons | 1.00000 | 1.00000 | 0.99870 | 0.99870 |
| Number of electrons | 1.00000 | 1.01310 | 1.00000 | 1.00000 |
| Cut acceptance | 0.99170 | 0.99170 | 0.99170 | 0.86200 |
| Livetime acceptance | 0.98035 | 0.98035 | 0.98035 | 0.98035 |
| Geometry Errors | 1.01230 | 1.00730 | 1.00000 | 1.00000 |
| Radiative correction | 1.00000 | 1.00000 | 0.97656 | 0.97656 |
| Total correction | 0.9956869 | 0.9921400 | 0.9491510 | 0.8250158 |

Table 4.1: Values of efficiency corrections. The expected number of events in each category is multiplied by the total correction, which is the product of the other corrections.

**PDF Construction**

This is the process for turning the MC events in to an expected probability distribution to compare against the data. It is the topic of Section 8.1.

**Burst Removal**

Any event passing all other cuts but within 100 ms of another event is removed. This removes a variety of delayed instrumental responses in the PMTs.

**Systematics**

The systematics are the systematic uncertainties in our measurement. These are uncertainties that affect all the data points in the same manner, such as an offset in the center of the detector, which would shift all events one direction. They are measured by SNO's calibrations and discussed in Section 4.7.

**Sigex**

Sigex is short for "Signal Extraction". This is the process of using SNO's processed data and MC to determine the number of $\nu_e$'s and $\nu_x$'s detected, by determining the amounts of the three principle fluxes CC, ES and NC. This is the subject of this thesis; Chapter 8 deals most directly with it.

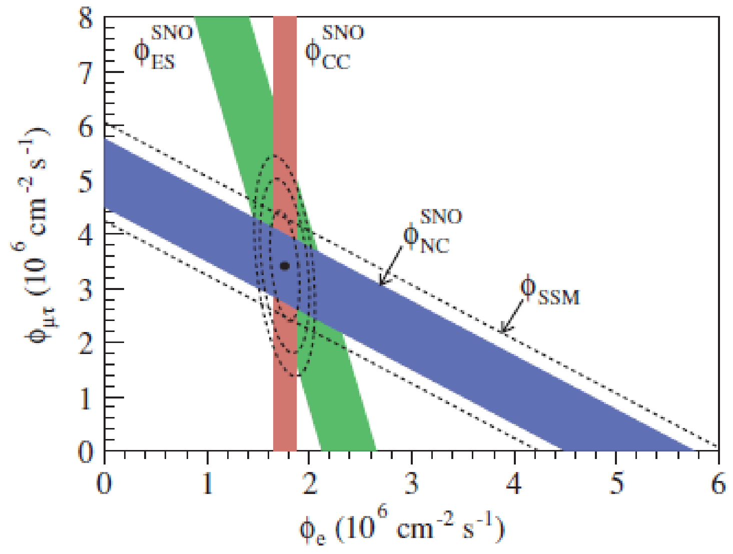


Figure 4-2: Results from the $D_2O$ phase. The three fluxes (CC, ES, NC) are shown as color bands, the axes are the fluxes of $\nu_e$ and $\nu_{\mu,\tau}$. Taken together, this shows that the solar neutrino problem is solved by neutrinos changing flavor. Taken from [50].

**CC/ES Spectrum, NC Flux**

This is the output of the analysis: the energy spectrum and flux amount of each of the principle fluxes, ES, CC and NC.

## 4.2 Previous analyses

This analysis is by no means the first one to look at SNO's data. A number of analyses are done with the data, but we will only highlight the "main" analyses here: those looking to extract the principle signal fluxes (CC, ES and NC) and their Day/Night differences (see Chapter 5).

The most critically important was the first analysis, the $D_2O$ phase results [54, 55, 50]. This definitively settled the solar neutrino problem discussed in Chapter 2 by showing that the predicted flux of neutrinos from the sun was arriving at the detector, but only a fraction of them were arriving as $\nu_e$. Its final result is summarized in Figure 4-2, showing the measured fluxes and the corresponding neutrino-flavor fluxes.

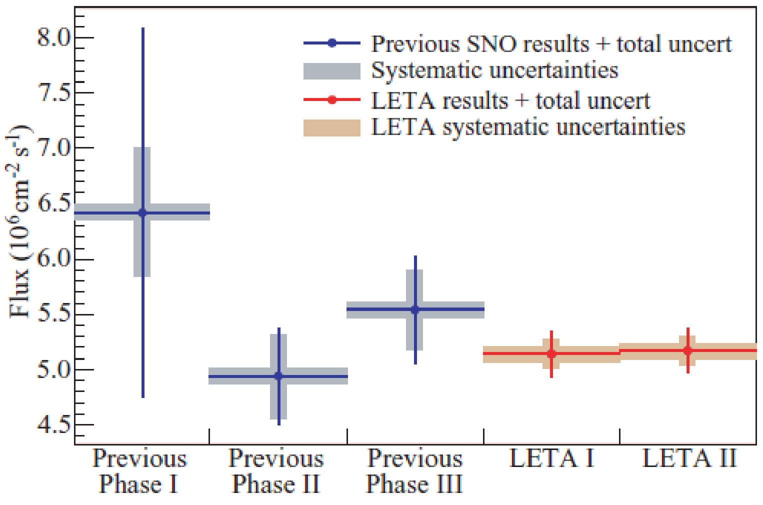


Figure 4-3: Measured total neutrino flux from the sun for the various phases of SNO. "LETA I" and "LETA II" are two different signal extractions within LETA. Note the large improvement in both systematic and statistical uncertainty. Taken from [51].

The second phase of SNO, the Salt phase, served to reinforce this result by detecting the neutrons from the NC reaction via a different mechanism [56].

The first two phases of SNO were combined with a much improved analysis as the "Low Energy Threshold Analysis" or LETA [51]. By very careful background and calibration studies, they were able to lower SNO's threshold in $E$ from 6 MeV to 3.5 MeV, resulting in greatly improved statistics, and significantly lower SNO's systematic uncertainties. The improvement in measuring the total neutrino flux from the sun is shown in Figure 4-3.

The third phase of SNO, the NCD phase, served as yet another independent check [57, 45]. For this phase, the neutrons from the NC were measured by a completely independent system of detectors. In addition, this analysis was the first SNO analysis to use the Markov Chain Monte Carlo (MCMC) method described in Chapter 7 and used in the analysis presented in this thesis.

In addition, each phase of SNO did a separate analysis to look for the "Day/Night effect". This is the altering of the proportion of $\nu_e$ in the measured neutrino flux due to the MSW effect on neutrinos passing through the Earth, described in Chapter 5. Those neutrinos detected during the day have only passed through the atmosphere and the small layer of rock above the detector; those detected at night have passed through the bulk of the Earth, so this effect manifests as a difference in the solar neutrino flux measured during the day versus that measured at night. This was measured to be $0.070 \pm 0.049 \pm 0.013$ for the $D_2O$ phase in [58] and $0.037 \pm 0.040$ for the Salt phase in [59]. It was included in LETA's primary analysis in [51], with a resulting value of $0.027 \pm 0.043$.

The analysis presented in this thesis is the Day/Night analysis for the NCD phase. We take this measurement one step further, however, and incorporate the LETA results to create a Day/Night measurement including all three phases of SNO.

## 4.3 Separation of Fluxes

In the NCD phase, we have PMT (light) signals from all three primary neutrino interaction processes (ES, CC, NC), and a signal from the NCDs that only measures NC. The ES signal is actually split in to two parts, the elastic scattering of electron neutrinos (just labelled ES) and the elastic scattering of $\mu$ and $\tau$ neutrinos, labelled $\text{ES}_{\mu\tau}$. This is due both to the two signals having slightly different distributions due to the extra Feynman diagram (though this difference turned out to be too small to observe) and for ease of computation later on, as described in Section 4.5. However, for the purposes of this section, the two ES's are indistinguishable and "ES" refers to both.

For now, we focus on the PMT aspect as the NCDs are ultimately handled by a separate analysis called PSA, as discussed in Section 8.3. We cannot separate the signals on an event-by-event basis, due to the random nature of the processes - how would we know that an individual electron event is from ES or CC, given that the neutrinos enter the detector with a range of energies, and the processes themselves are random? Instead, we separate them statistically.

From the reconstruction process, we have several pieces of information about each event: its position in the detector ($x$, $y$ and $z$), its energy ($E$), its direction of travel in the detector ($u_x$, $u_y$, $u_z$), its light-cone characteristics and the state of the detector at the time. In this analysis, the latter two are used only to reject problematic events – those resembling backgrounds or instrumental effects as described in the "High Level Cuts" and "Instrumental Cuts" sections above. We can now look at how each of the signals is expected to be distributed in each measured variable through the Monte Carlo. However, we have a large number of variables, which will be problematic as we are effectively looking to histogram our events. Thus, we need to look at the physics of the system to find which variables or combinations of variables are actually useful in distinguishing the fluxes from one another.

We start with the positions $x$, $y$ and $z$. Since the detector is nearly spherical, they carry mostly redundant information – rotation symmetry tells us that only the distance from the center of the detector is important. This being the case, we can reduce our number of variables by defining

$$\rho^3 = \left( \frac{\sqrt{x^2 + y^2 + z^2}}{600 \text{ cm}} \right)^3 \tag{4.1}$$

The normalizing distance 600 cm is chosen as it is the radius of the AV, i.e. the edge of the $D_2O$. The cube is taken so that bins of equal $\rho^3$ will contain equal volumes.

For the direction of travel, again with a spherical detector we have no reason to prefer one of our (arbitrary) axes. But here the symmetry of the detector is broken by the neutrino flux itself - it is coming from the Sun, meaning the neutrinos are all entering the detector from the same direction. Of course, this direction varies over the course of the day and the year. So we define the variable

$$\cos(\theta_{sun}) = -\frac{\vec{R}_{sun} \cdot \vec{v}}{|\vec{v}|} \tag{4.2}$$

where $\vec{R}_{sun}$ is a normalized vector in the direction of the sun. This measures what actually matters - the direction of travel relative to the direction of the incident neutrino, with the property $\cos(\theta_{sun}) = 1$ is a particle moving directly away from the sun. We keep the polar angle and not the azimuthal angle as the physical interactions are invariant under a rotation about the axis defined by the direction toward the sun.

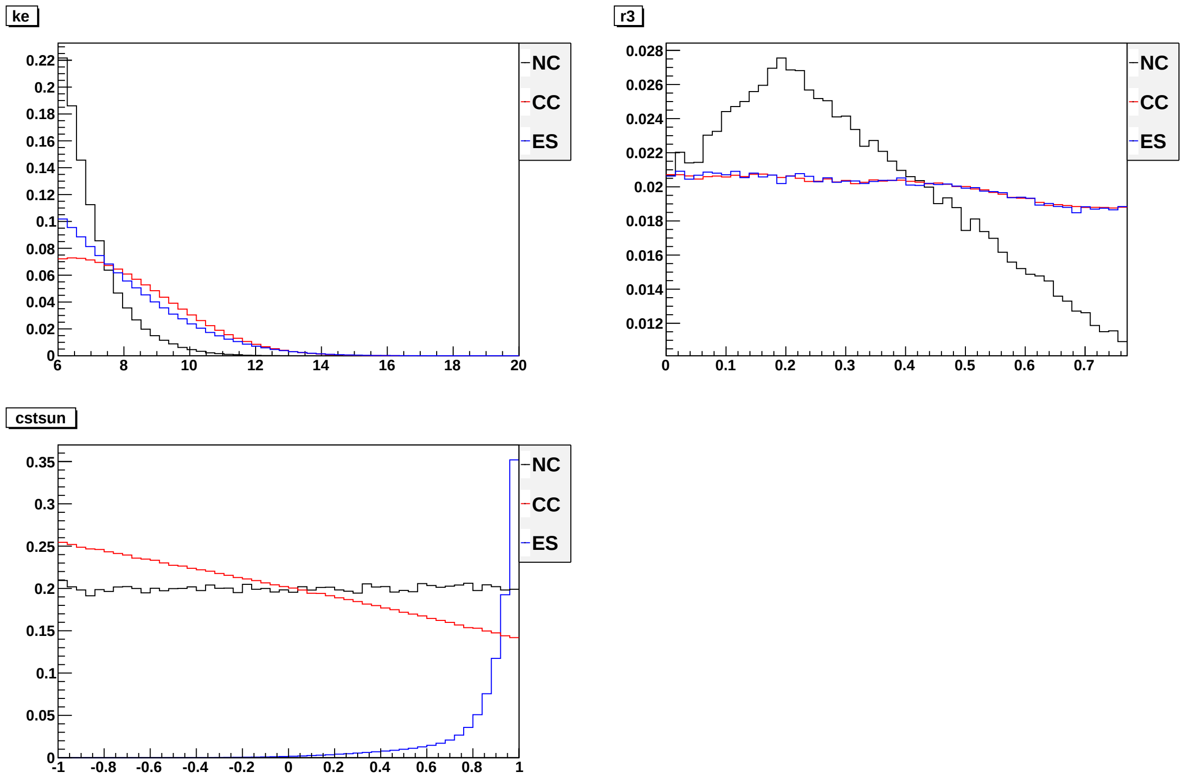


Figure 4-4: Monte Carlo for the three signal fluxes, ES, CC and NC, are shown in each of the variables used for analysis, E (labeled ke), $\rho^3$ (labeled r3) and $\cos(\theta_{sun})$ (labeled cstsun). The vertical scale is arbitrary.

We have now reduced down to three measured variables, a manageable number, which are $E$, $\rho^3$ and $\cos(\theta_{sun})$. We now look at the characteristics of each signal in each of these variables to see what characteristics stand out. Figure 4-4 shows this breakup from MC, which includes all known detector effects (in particular resolution in the three variables). We see clearly that the ES signal is sharply peaked in $\cos(\theta_{sun})$, while the CC shows a negative slope and the NC is flat. We also see that the NC has a distinctive shape in both $E$, as it is the scattered energy of a monoenergetic photon, and in $\rho^3$. The reduction in flux at low $\rho^3$ is a result of the high density of NCDs there, which capture the majority of the neutrons, while the reduction at high $\rho^3$ comes from the long path that neutrons wander in the detector before being absorbed interacting with the neutron-absorbing AV.

## 4.4 Signal Extraction

With the characteristics from the previous section, we should be able to separate the fluxes. If only the signal fluxes were present, we would just vary the relative proportion of each until we matched the histogrammed data. This is the basic idea that we will follow, but there is much more going on in the physical system - there are backgrounds and systematics that must be accounted for and the NCDs must be taken in to account. In addition, we stand to gain in accuracy by moving to an unbinned method, since we have a relatively small number of events.

The signal extraction method used for the analysis presented in this thesis is described in detail in Chapter 8, but a few general points are needed to understand the rest of this chapter. First, we seek to compare the data to the MC and adjust the MC until it matches the data. We do this by taking the MC events, altering them according to our systematics, doing the same for the backgrounds, and histogramming the three variables previously described. We can vary the relative amounts of each flux or background by varying how many events we take from each, though we do this not by taking fewer events but by making each MC event count as more or less than one event as it is added to the histogram. If we call our set of relative proportions of fluxes and backgrounds, systematic values and total number of events $\vec{\alpha}$, we are seeking to vary $\vec{\alpha}$ to minimize the difference between the MC histogram and the data. This will be made more precise in Chapter 8. The important thing for this chapter is that we are rebuilding this MC histogram at each step, varying the number of events in each flux and background and the values of the systematics within their measured constraints.

We use the data from the NCD phase, both the PMT data and the NCD data. However, the analysis is for all three phases of SNO simultaneously. To do this, we need to include the results of the LETA analysis. While deciding how to implement all three phases simultaneously, the SNO collaboration realised that the number of correlated backgrounds and systematics between the NCD phase and LETA were quite small, thanks to the presence of the NCDs and the recalibration done during the NCD phase. Thus we can run the LETA and the NCD analysis separately, and simply add the LETA results as a constraint term on the NCD results. How this was accomplished will be explained in Section 8.2.

Similarly, we only discuss the PMT side of the analysis here. The NCD side is almost entirely handled by the separate PSA analysis, described in Section 8.3.

That all said, we will describe how the flux separation is dealt with, and backgrounds and systematics that go in to the analysis below.

## 4.5 $\mathbf{P}_{ee}$

So far, we have been saying that we vary the relative amounts of the four primary fluxes (ES, $\text{ES}_{\mu\tau}$, CC, NC). This has been the case for every phase of SNO, but the manner in which this is done has changed. In SNO's first analysis, described in the $D_2O$ phase paper [50], this was done in the most straightforward way possible. ES and $\text{ES}_{\mu\tau}$ were combined into a single flux and the overall number of events in each flux were allowed to vary. This does not assume anything about the ratio of $\nu_e$ to $\nu_{\mu,\tau}$, but does assumes that the neutrino energy spectrum will be the same as the $^8$B beta decay spectrum. This is equivalent to saying that the survival probability from Chapter 2, $P_{ee}$, which is the probability that a $\nu_e$ generated in the sun is still a $\nu_e$ when it is detected, is a constant.

In Chapter 2 we showed that there are mechanisms for possible energy dependence in $P_{ee}$. One possible method for taking this in to account is by making no assumptions about the energy spectrum of the incident neutrinos. This gives an analysis unconstrained in energy, meaning that the energy spectrum is either ignored or allowed to vary. In [50], the former was presented as a secondary analysis, which did not use the reconstructed energy information at all other than to remove events outside the analysis window. In the NCD phase analysis in [57], the MC events were separated in to energy bins (as we do in this analysis, see Section 8.1), and the height of each bin was allowed to vary independently. This both makes use of the energy data and gives more information about the energy spectrum, but is much more computationally intense.

A third, less obvious option is to directly deal with $P_{ee}$. This is the method SNO has chosen both this 3-Phase analysis and the LETA analysis [51]. Here we balance the two approaches: we known what the energy spectrum of $^8$B beta decay and how the detector responds to $\nu_e$'s and $\nu_{\mu,\tau}$'s and use that information, but we allow for possible distortions in the energy spectrum. To allow for an energy dependent $P_{ee}$, we parameterize it as a polynomial in $E_\nu$, as

$$P_{ee} = P_0 + P_1(E_\nu - 10) + P_2(E_\nu - 10)^2 \tag{4.3}$$

where $E_\nu$ is measured in MeV and the polynomial is centered around 10 MeV, the most probable neutrino energy for our detector, to reduce correlations between the $P_i$'s. We use a second order polynomial because we expect $P_{ee}$ to be slowly varying over our region of interest and because we have insufficient data to extract information about a cubic component. This latter point was discovered by testing on simulated data.

When we build our MC histogram, each event is added with a weight. In a traditional histogram, when we add an event, the total for the bin that event falls into is incremented by 1. Instead, we increment by a different value, which we can vary, called the weight ($W$). This gives us a simple method for altering the number of events for each flux, by simply adjusting the relative $W$'s. Since our MC assumes 100% $\nu_e$ for ES, NC and CC (and 100% $\nu_{\mu,\tau}$ for ES$_{\mu\tau}$), we adjust the number of MC events for $P_{ee} \neq 1$ by weighting events by $P_{ee}$. As a simple example, if $P_{ee} = \frac{1}{3}$ (i.e. it is a constant), we would have $\frac{1}{3}$ as many events in ES and CC and would multiply their respective $W$'s by $\frac{1}{3}$, while NC would be unaffected. Since we have a $P_{ee}$ dependent on energy, we have a weight that varies with $E_\nu$ and thus varies event to event. If we call our overall scale $B8$ and ignore systematics, our weights are then

$$\begin{aligned}
W_{CC} &= f_{CC} B8 \cdot P_{ee} \\
W_{NC} &= f_{NC} B8 \\
W_{ES} &= f_{ES} B8 \cdot P_{ee} \\
W_{ES_{\mu\tau}} &= f_{ES_{\mu\tau}} B8 \cdot (1 - P_{ee}) \qquad (4.4)
\end{aligned}$$

where $f_x$ is the scaling factor to go from $B8$ to number of expected events for the respective flux. This is a constant for each flux that is determined by the units we choose for $B8$ and the number of MC events relative to the number of expected data events.

Now we are no longer fitting the relative number of events in each flux, but rather the $P_{ee}$ function directly. We can still extract the fluxes if we wish, we need merely count the number of events in each flux at the best-fit values of $P_{ee}$. One concern is that this has removed the freedom of ES and CC to vary relative to each other. This is partially true, though we have separated ES into ES and ES$_{\mu\tau}$ to compensate for this. Beyond this, we assume that our model gives the correct ratio of CC to ES events. This is the assumption that we understand how neutrinos interact with our detector, a necessary one.

## 4.6 Backgrounds

The SNO detector is extremely sensitive to radiation, by design. This means that backgrounds must be carefully measured and controlled. SNO spent a great deal of time and effort on this process, in experimental design, analysis optimization and in calibration.

The backgrounds fell in to several major classes: direct backgrounds from electrons and photons, instrumental backgrounds, cosmic-ray muons and neutrons from various sources.

The direct backgrounds were associated with radioactive decays in the uranium and thorium chains. Any $\beta$ decay with sufficient energy can cause a signal, as can a photon via Compton scattering. To combat this, extreme care was taken to reduce the levels of these elements in all aspects of the detector (see [44] for the system design, especially regarding purfication of the $H_2O$ and $D_2O$, special order PMTs and design of the AV, and [45] for the NCD-phase impurity measurements). Most of these decays are relatively low energy; a cut removing events with energy $E < 6$ MeV is used to remove them. In addition, the AV contained more impurities than the $D_2O$. The AV is at a radius of 600 cm, so events with radius greater than 550 cm were cut. This served to reduce these backgrounds to negligible levels [45]. Finally, since the $^8$B neutrino flux is expected to have a maximum neutrino energy of approximately 15 MeV, any event with energy greater than 20 MeV was cut.

As mentioned before, a major motivation in building the detector deep underground was the reduction in the cosmic ray muon rate. In addition, the experiment looked for muons in either the light water or the heavy water and vetoed any event within 20 seconds of one. This was to avoid both prompt effects from the heavy ionization, which generates a great deal of light in the detector, and the delayed effects of "muon followers". Muon followers are neutrons produced by spallation from the passing muon, which then thermalize in the detector and are captured. The length of the cut window is to allow enough time for this process to occur, with an estimated "leak through" of less than one event. There is some possibility that a muon could interact in the rock surrounding the detector, producing neutrons that may wander in to the detector but no corresponding light signal to veto, but this rate was calculated to be so low as to be ignorable at 0.18 neutrons per year [45].

Though we are focusing on the neutrinos from the $^8$B beta decay in the solar spectrum, neutrinos from the hep reaction are also present in the energy range we measure. They have a different energy spectrum and a much lower flux, as shown in Figure 2-3, so we treat them as a background. They interact identically to the main fluxes in the detector via the three interactions (ES, CC, NC), giving rise to three separate backgrounds called hepes, hepcc, and hepnc. The expected number of events is very small, as shown in Table 4.2. The numbers in the table lack uncertainties as they were not floated, and lack PMT-to-NCD conversions because they were not included in the NCD analysis as only a fraction of an event was expected.

As mentioned before, any neutrons in the detector were indistinguishable from NC signal. This makes them irreducible backgrounds, so they can't be removed from the data set. It was imperative, then, to understand how many such neutrons we expect in the detector, both in the NCD and the PMT signals. These are, of course, correlated - a neutron background that affects one affects the other, though not necessarily equally. Since the NCDs are near the center of the $D_2O$ volume, neutrons coming from outside the detector have less of an effect on them. However, they are more sensitive, so more events are expected in the NCDs in general. From Monte Carlo simulations, the ratio of NCD events to PMT events was computed for each type of background (described below), with conversion factors appearing in Table 4.2. Each background has its own Monte Carlo, giving expected distributions, which was treated identically to the principle fluxes CC, ES, $ES_{\mu\tau}$ and NC.

One prominent source of neutrons was photodisintigration of deuterium. As mentioned before, any photon with more than 2.2 MeV of energy can break up the deuteron, creating a neutron. The dominant source of these photons in SNO is the decay of $^{214}$Bi in the $^{238}$U chain and $^{208}$Tl in the thorium chain. While great care was taken to purify the $D_2O$, some of these elements remained. This is the background "d2opd" ($D_2O$ photodisintigration). The material of the NCDs and the cables connecting them to the external electronics both contained these elements as well (the backgrounds "ncdpd" and "cab", which were combined in to one background), as did the AV (part of the "ex" background). In addition, the NCDs contained two "hot spots", places where some contamination was introduced during the installation process. These two hotspot backgrounds are "k2pd" and "k5pd", named for the NCDs that they appear on.

An additional source of neutrons was from the $(\alpha, n)$ reaction. Alpha particles from polonium decays in the U and Th decay chains can induce this background by interacting with other nuclei in the detector, in particular deuterium and oxygen in the heavy water and carbon and oxygen in the AV. These were found to be negligible

| Background | NCD-to-PMT Conversion factor | NCD Events |
|---|---|---|
| ex | 0.5037 | $40.9 \pm 20.6$ |
| d2opd | 0.2677 | $31.0 \pm 4.7$ |
| ncdpd | 0.1667 | $27.6 \pm 11.0$ |
| k2pd | 0.2854 | $32.8 \pm 5.3$ |
| k5pd | 0.2650 | $45.5 \pm 8.0$ |
| cab | 0.1407 | $8.0 \pm 5.2$ |
| atmos | 1.8134 | $13.6 \pm 2.7$ |
| **Hep flux** | **NCD-to-PMT Conversion** | **PMT Events** |
| hepcc | N/A | 31.291 |
| hepes | N/A | 1.938 |
| hepnc | N/A | 0.986 |

Table 4.2: Number of events for each neutron background, with number of expected events in the NCDs given. Number of events in the PMTs is computed by multiplying the number in the NCDs by the conversion factor, so that the number of events in the PMTs and NCDs are 100% correlated.

for the heavy water and NCDs [45]. The rate from the AV was determined by directly counting $\alpha$ rates on the surface of the AV both before and after the NCD phase, and these were included with the AV's photodisintigration rates to create the "externals" or "ex" background.

Neutrinos from sources other than the Sun can also cause the NC reaction. The only significant contribution is from atmospheric neutrinos, from cosmic ray events in the upper atmosphere. This is the "atmospheric" or "atmos" background.

From a combination of measurements and Monte Carlo studies, summarized in [60], an expected number of events for each background was computed. These are shown in Table 4.2. The conversion factor between the PMTs and the NCDs is assumed to be fixed.

## 4.7 Systematics

SNO relies heavily on its Monte Carlo. While this simulation of the detector is very detailed and accurate, it is not perfect. If any parameter of the detector model is off, this can introduce a systematic error - as an example, if the PMT response is mismodeled slightly, this could result in a systematic mis-measurement of particle

| Source | Use |
|---|---|
| Electronic pulser | Sends pulses to PMT electronics. Calibrates timing and charge characteristics |
| Laserball and LED sources | Light sources controllable in frequency, intensity, direction and timing. Calibrates PMT response to light and optical properties of detector. |
| Contained radioactive sources: $^{16}$N, $^{252}$Cf, $^{8}$Li, Th, U | Source inside containers placed in $D_2O$. Calibrates detector reponse to photons, neutrons and energetic electrons. |
| Diffuse radioactive sources: Rn, $^{24}$Na | Radioactive sources mixed in to $D_2O$ and removed by water purification system. Calibrates as previous, but over longer time period and over entire detector volume. |
| $^{3}$H(p,$\gamma$)$^{3}$He accelerator source | Accelerator, creates high energy photons. Calibrates detector response to high energy photons. |
| Tagged radioactive sources: $^{252}$Cf, $^{24}$Na, $^{228}$Th | Contained radioactive source with coincidince counter to detect decays, allowing background-free measurements. Calibrates detector photon and neutron response. |
| Radioassays | Measures containimination levels in detector components (water, AV, cables, etc) |

Table 4.3: Table of calibration sources used in SNO. More detail is available in [44]

energy. Thus these systematics are taken as potential adjustments to the MC.

A great deal of time was spent doing a variety of calibrations at SNO, setting limits on as many of these systematics as possible. Many different calibrations using a variety of sources were performed, as described in [45, 44]. A brief summary is given in Table 4.3. The limits set by these calibrations set constraints on the possible values these systematics could take - we interpreted a measurement of $0 \pm \sigma$ for a systematic to mean that its possible values were a Gaussian distribution with mean 0 and width $\sigma$.

Most of these systematics are floated in the analysis within their constraints, in order to get their uncertainty contributions correct. The full list of floated systematics in the analysis, except day-night aspects that are discussed in Chapter 5, is in Table 4.4, and a description of each appears below. These descriptions and values were agreed upon by the 3-Phase working group in SNO and this list appears in [49].

Since the MC generation proccess is exceedingly computationally expensive, instead of re-running the MC for each potential adjustment the effects are applied to the resulting MC events. This actually aligns well with how the systematics are measured - in general, the event variables (energy, position, etc) are measured and compared to the MC or to known quantities, rather than directly measuring the input MC parameters (water properties, physics of the PMTs, etc). Thus we characterize our systematics as functions that we apply to the MC events to make their distribution agree with the data events. We write them in terms of how they adjust the parameters, which we then sum over for the final result. As an example, suppose energy had two systematics, which give $\Delta E_1$ and $\Delta E_2$. Then we would take our "new", adjusted MC energy to be $E_{new} = E + \Delta E_1 + \Delta E_2$.

### Energy

First we look at those systematics affecting the reconstructed energy $E$. These can have a large impact on the results – all three primary fluxes have most of their events at the lowest energies in the analysis window, so changing the lower boundary can alter the number of included events significantly. The adjustments we allow are the energy scale, the resolution and a non-linearity term. The scale and resolution were measured in [61] and the non-linearity was measured in [62].

### Energy Scale

The energy scale is the simplest, as it adjusts the energy by

$$\Delta E = (a_{0,c}^E + a_0^E)E \tag{4.5}$$

where the two terms are the part that is correlated between the three phases of SNO, $a_{0,c}^E$, and the part specific to the NCD phase, $a_0^E$; the $E$ is the reconstructed energy of the MC event. As expected, this is effectively multiplying the energy by a constant.

### Energy Resolution

The energy resolution is a bit more complicated. This is the resolution of the detector, and it relies on a variable present in the MC that isn't available in the data: the true energy of the interaction ($E_{true}$). This is the energy that the charged particle actually has, as opposed to that measured via reconstruction. The width of the distribution of the difference between this value and the reconstructed energy *is* the detector's energy resolution. To broaden or narrow this resolution, we adjust the energy by

$$\Delta E = b_0^E (E - E_{true}) \tag{4.6}$$

this has the property that when $b_0^E = -1$, $E_{new} = E$, that is our energy measurement is perfect. In the systematics table, two values appear for this number, $b_0^E(e^-)$ and $b_0^E(n)$. These are the values of resolution applied to electron events (ES, CC) and neutron events (NC, many backgrounds). These come from physically distinct processes, as the neutrons are actually measured via an emitted photon that then Compton scatters electrons in the $D_2O$. In [61], the conversion from the measured photon scattering (NC-like behavior) to monoenergetic electrons (ES and CC-like behavior) finds that the difference between calibration data and MC for the energy resolution is narrower for neutrons by a factor of 1.36. The resolution is a property of the detector, so changes affect neutrons and electrons simultaneously. Thus the two $b_0^E$'s are 100% correlated and are varied as one parameter in the analysis.

### Energy Nonlinearity

The posibility of a non-linearity in the detector's energy response is handled by the energy nonlinearity systematic, $c_0^E$. This is implemented as

$$\Delta E = E \left( c_0^E \frac{(E - 5.05)}{(19.0 - 5.05)} \right) \tag{4.7}$$

where the constants come from from the most probable (5.05 MeV) and highest (19.0 MeV) energies in the calibration and are effectively a normalization.

### Position

Next we look at the position systematics. Here we treat $x$, $y$ and $z$ separately and recombine them into $\rho^3$ later. This is done because there are a few known vertical variations in the detector, corresponding to the $z$ direction.

### Position Offset

The simplest is the offset. In each direction, we have a potential offset

$$\Delta x = a_0^x \tag{4.8}$$

$$\Delta y = a_0^y \tag{4.9}$$

$$\Delta z = a_0^z \tag{4.10}$$

The measurement of these values was described in [63].

### Position Scale

Next is the position scale. This uses different values for $z$ v. $x$ and $y$, and is applied as

$$\Delta x = a_1^{x,y,z} x \tag{4.11}$$

$$\Delta y = a_1^{x,y,z} y \tag{4.12}$$

$$\Delta z = (a_1^{x,y,z} + a_1^z) z \tag{4.13}$$

so that there is a scale correlated to all three directions, effectively a radial scale, and a z-specific component. These were described in [64].

### Position Resolution

The position resolutions are more complicated. They are conceptually the same as the energy resolution, but they are known to vary vertically in the detector. This leads to a z-dependent resolution, as if the $b_0$ were a function of z. This is applied as

$$\Delta x = (b_0^{xy} + b_1^{xy} z + b_2^{xy} z^2)(x - x_{true}) \tag{4.14}$$

$$\Delta y = (b_0^{xy} + b_1^{xy} z + b_2^{xy} z^2)(y - y_{true}) \tag{4.15}$$

$$\Delta z = (b_0^{z} + b_1^{z} z)(z - z_{true}) \tag{4.16}$$

An extra complication is that these parameters are correlated. Their measured values do not have standard uncertainties, instead the constraints appear as a covariance matrix $\mathbf{\Sigma}$. Just as a standard uncertainty describes a Gaussian probability distribution for a parameter, this describes a mulit-dimensional Gaussian for a set of parameters $\vec{\beta}$, with central values $\bar{\vec{\beta}}$

$$p(\vec{\beta}) = \frac{1}{(2\pi)^{m/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\vec{\beta} - \bar{\vec{\beta}})^T \mathbf{\Sigma}^{-1} (\vec{\beta} - \bar{\vec{\beta}})\right) \tag{4.17}$$

One matrix describes $b_0^{xy}$, $b_1^{xy}$ and $b_2^{xy}$, it is

$$\mathbf{\Sigma}_{xy} = \begin{pmatrix} 0.000818124 & -2.24984 \times 10^{-7} & -4.19131 \times 10^{-9} \\ -2.24984 \times 10^{-7} & 3.66098 \times 10^{-9} & 3.71423 \times 10^{-12} \\ -4.19131 \times 10^{-9} & 3.71423 \times 10^{-12} & 3.92118 \times 10^{-14} \end{pmatrix} \tag{4.18}$$

Another describes $b_0^z$ and $b_1^z$, it is

$$\mathbf{\Sigma}_{z} = \begin{pmatrix} 0.00078696 & 3.47188 \times 10^{-7} \\ 3.47188 \times 10^{-7} & 6.80761 \times 10^{-9} \end{pmatrix} \tag{4.19}$$

These parameters are described in [63].

### $\cos(\theta_{sun})$ Resolution

The third variable, $\cos(\theta_{sun})$, only has a resolution parameter. In addition, this resolution parameter is only applied to the ES signal, as it has no effect on NC and very little on CC. It parameterization, however, can cause problems in those fluxes if the resolution parameter is negative. The parameterization is

$$\cos(\theta_{sun})_{new} = 1 + (1 + a_0^\theta)(\cos(\theta_{sun}) - 1) \tag{4.20}$$

We have departed from our $\Delta x$ description here for clarity as to what this is doing. For positive values of $a_0^\theta$, it moves events further away from 1, about which the ES signal is sharply peaked. This may cause points to "fall off the edge", i.e. gain an unphysical value of $\cos(\theta_{sun}) < -1$. To compensate for this, any event that does this is instead given a random value from the uniform distribution $[-1, 1]$. If $a_0^\theta$ is negative, values are moved closer to 1. This parameterization is designed to avoid the $\cos(\theta_{sun}) > 1$ issue, but can result in a region near $-1$ having no events; this is not a problem for ES (there are very few or zero events there anyway), but would be a large problem for CC or NC, hence why it is not applied to them. This parameter was measured in [65].

### Weighting Systematics

The remaining four systematics behave differently from the previous ones. They do not change the value of one of the event variables; instead, they change the weight given to an event. This is best thought of in the context of a histogram: when an event is added to a histogram, the number of events in that bin usually increases by one. Instead, we allow each event added to increase the number of events by a real number amount $W$, the weight of that event. An example of this is a decrease in efficiency of the detector, so that if only half of the events in the MC are actually in the data, we would apply $W = 0.5$. These are applied as a product rather than a sum, i.e. if an event had two weight systematics $W_1$ and $W_2$, it would be added to the histogram with a weight $W = 1 \cdot W_1 \cdot W_2$. The weights discussed in Section 4.5 are applied as systematics of this form.

### Energy Dependent Fiducial Volume

The first weight-affecting systematic is the energy dependent fiducial volume. This treats the possibility that the detector's position reconstruction depends on the energy of the event. It only treats the net effect of possibly pushing events out of the fiducial volume of the detector, changing the overall number of events. This is applied as

$$W = 1 + c_0^{R^3}(E - 5.05) \tag{4.21}$$

the 5.05 is the same as in the energy nonlinearity. This was described in [64].

### Neutron Detection Efficiencies

The next two weight affecting systematics are the neutron detection efficiencies. These measure what proportion of generated neutrons are detected by the PMTs ($\epsilon_{PMT}$) and by the NCDs ($\epsilon_{NCD}$). These are applied as

$$W = \epsilon_x \tag{4.22}$$

to the PMT NC events or the NCD NC events, as is appropriate. Note that in the actual analysis described in Chapter 8, these are normalized to have central value 1 and the number of expected events in the relevant NC flux is adjusted appropriately. This was done for computational reasons.

### Winter Spectrum Uncertainty

The final weight affecting systematic is the uncertainty in the $^8$B energy spectrum. This spectrum comes from [52], a paper by Winter, et al. that measured the spectrum of neutrinos from a terrestrial source of $^8$B, hence the title Winter Spectrum Uncertainty for this parameter. This is applied as

$$W = 1 + \frac{1}{3}\sigma_{^8B}(0.018 - 0.001999E_\nu - 8.8769 \cdot 10^{-5}E_\nu^2) \tag{4.23}$$

| Name | Symbol | Value |
|---|---|---|
| Correlated Energy Scale | $a_{0,c}^{E}$ | $0.0000 \pm 0.0041$ |
| Energy Scale | $a_0^E$ | $0 \pm 0.0081$ |
| Energy Resolution (neutron) | $b_0^E(n)$ | $0.0000 \pm 0.0104$ |
| Energy Resolution ($e^-$) | $b_0^E(e^-)$ | $1.36 \cdot b_0(n)$ |
| Energy Nonlinearity | $c_0^E$ | $0.0000 \pm 0.0069$ |
| Position Offset X | $a_0^x$ | $0.0 \pm 4.0$ |
| Position Offset Y | $a_0^x$ | $0.0 \pm 4.0$ |
| Position Offset Z | $a_0^x$ | $5.0 \pm 4.0$ |
| Position Scale XYZ | $a_1^{x,y,z}$ | $0.0000^{+0.0029}_{-0.0077}$ |
| Position Scale Z addition | $a_1^z$ | $0.0000^{+0.0015}_{-0.0012}$ |
| Position Resolution XY Constant | $b_0^{xy}$ | $0.06546^{*}$ |
| Position Resolution XY Linear | $b_1^{xy}$ | $-5.501 \cdot 10^{-5}$ * |
| Position Resolution XY Quad | $b_2^{xy}$ | $3.9 \cdot 10^{-7}$ * |
| Position Resolution Z Constant | $b_0^z$ | $0.07096^{**}$ |
| Position Resolution Z Linear | $b_1^z$ | $1.155 \cdot 10^{-4}$ ** |
| $\cos(\theta_{sun})$ Resolution | $a_0^\theta$ | $0.00 \pm 0.12$ |
| Energy Dependent Fiducial Volume | $c_0^{R^3}$ | $0.0000^{+0.0088}_{-0.0067}$ |
| PMT NC Efficiency | $\epsilon_{PMT}$ | $0.046725 \pm 0.0000603$ |
| NCD NC Efficiency | $\epsilon_{NCD}$ | $0.211 \pm 0.005$ |
| Winter Spectrum Uncertainty | $\sigma_{^8B}$ | $0 \pm 1$ |

* The position resolution XY systematics have a constraint matrix, see text

** The position resoultion Z systematics have a constraint matrix, see text

Table 4.4: List of systematics floated in the analysis. Does not include day/night aspects. See text for explanation of how these are applied.

where $E_\nu$ is the energy of the incident *neutrino*, rather than the measured energy.

# Chapter 5

# Day/Night

## 5.1 Introduction

The analysis described in Chapter 4 is the main analysis as described in [57] (which was for the NCD phase only). That analysis assumes that $P_{ee}$ is a constant as a function of time. Our analysis is an extension of this, called the Day/Night analysis, that allows for $P_{ee}$ to be different during the night versus during the day.

As discussed in Section 2.5, the oscillations of neutrinos passing through the Earth are expected to be modified by the MSW effect. This has the effect of altering the expected rate of $\nu_e$'s detected during the night, when $\nu$'s must pass through the Earth to reach the detector, relative to the rate detected during the day. It should have no effect on the total rate of $\nu$'s of all flavors reaching the detector. This is known as the "Day/Night effect", and measuring this is our goal.

We will be discussing the difference between day and night for many quantities. As such, we define a few general terms we will see repeatedly. In general, the values measured in the main analysis of the data do not distinguish between day and night; instead they measure an average value of sorts. If the value of a parameter is independent of the counting time, for example most of the systematics, the day/night averaged value is a simple average

$$\bar{\alpha} = \frac{\alpha_N + \alpha_D}{2} \tag{5.1}$$

where $\alpha$ is our parameter, and the subscripts $N$ and $D$ indicate night and day respectively. Following common practice, we measure the day v. night difference with the asymmetry

$$A_\alpha = 2\frac{\alpha_N - \alpha_D}{\alpha_N + \alpha_D} \tag{5.2}$$

which is the difference divided by the average. Since we would like to compare against the main analysis, we characterize our parameters by $\bar{\alpha}$ and $A_\alpha$ rather than $\alpha_D$ and $\alpha_N$. We solve for the latter in terms of the former to get

$$\begin{aligned} \alpha_D &= (1 - \frac{A}{2})\bar{\alpha} \\ \alpha_N &= (1 + \frac{A}{2})\bar{\alpha} \end{aligned} \tag{5.3}$$

This is obviously a problem for a quantity with $\bar{\alpha} = 0$. In those cases (which include many of the systematics), we avoid the problem by redefining our parameterization so that we use the quantity $1 + \alpha$, which alters asymmetry to be

$$A_\alpha = 2\frac{\alpha_N - \alpha_D}{2 + \alpha_N + \alpha_D} \tag{5.4}$$

this changes our day and night values to

$$\begin{aligned} \alpha_D &= \bar{\alpha} - \frac{\bar{\alpha}A}{2} - \frac{A}{2} \\ \alpha_N &= \bar{\alpha} + \frac{\bar{\alpha}A}{2} + \frac{A}{2} \end{aligned} \tag{5.5}$$

In those cases where values depend on the counting time, such as the number of background events, we must modify our approach. As expained in [60], in those cases we are really concerned about a day/night difference in the underlying rates

$$A = 2\frac{R_N - R_D}{R_N + R_D} \tag{5.6}$$

but we actually measure the number of events, so our constraint is on

$$N = R_N T_N + R_D T_D \tag{5.7}$$

where $T_D$ and $T_N$ are the amount of counting time (livetime) for the day and night, respectively. We solve this to arrive at the expected rates for day and night

$$\begin{aligned} R_D &= \frac{N}{T_D + T_N \frac{1+A/2}{1-A/2}} \\ R_N &= \frac{N}{T_N + T_D \frac{1-A/2}{1+A/2}} \end{aligned} \tag{5.8}$$

which we can convert to counts by simply multiplying by the appropriate livetime.

We then treat the day and night as nearly independent analyses, except that all of the parameters are linked. We separate both the Monte Carlo (MC) and the data in to its day and night components, and have a $\vec{\alpha}_D$ and a $\vec{\alpha}_N$. We then fit the day data to the day MC and the night data to the night MC, but we do not vary $\vec{\alpha}_D$ and a $\vec{\alpha}_N$ directly. Instead, we vary the parameters $\bar{\alpha}$ and $A$ and compute the corresponding $\alpha_D$ and $\alpha_N$ at each step in the fit. We then evaluate the log likelihood for the day and night separately and sum them to arrive at ELL($\vec{\alpha}$), where our $\vec{\alpha}$ is expanded to now include $\bar{\alpha}$ and $A$ for each parameter. In this way, we are using all of our data in our fit, rather than doing two smaller fits for day and night separately. This has the large benefit of automatically taking into account correlations between day and night values for the parameters.

## 5.2 Survival probability

The survival probability $P_{ee}$ described in Section 4.5 was the $P_{ee}$ neglecting MSW effects in the Earth. The day/night signal is a measure of how this changes when we do include these effects. We choose to parameterize the measurement as $P_{ee,day}$ and $A_{P_{ee}}$, with $A$ defined above. We expect both of these quantities to be energy dependent, so we treat them as polynomials in neutrino energy $E_\nu$. We initially chose a third order polynomial for $P_{ee,day}$, but discovered that we could gain no information about the $E_\nu^3$ term, so we chose a simpler quadratic. For similar reasons (lack of statistics but an expected energy dependence), we parameterize the $A$ term as a linear equation. This gives us five parameters

$$\begin{aligned} P_{ee,day} &= P_0 + P_1(E_\nu - 10) + P_2(E_\nu - 10)^2 \\ A_{P_{ee}} &= A_0 + A_1(E_\nu - 10) \end{aligned} \tag{5.9}$$

where $E_\nu$ is in MeV, and the polynomials are centered at 10, the peak of our expected neutrino signal, to reduce correlations between terms. Since we are choosing $P_{ee,day}$ instead of $\bar{P}$ as our parameter of choice, we re-solve the system of equations to find

$$\begin{aligned} P_{ee,day} &= P_0 + P_1(E_\nu - 10) + P_2(E_\nu - 10)^2 \\ P_{ee,night} &= P_{ee,day}\frac{2+A}{2-A} \end{aligned} \tag{5.10}$$

This gives us the separate day and night $P_{ee}$'s, which are applied to the separate day and night MCs as described in Section 4.5. Note that we do not have an asymmetry on the $^8$B flux, so that we are assuming that the NC rate is constant, as it is unaffected by the $P_{ee}$.

## 5.3 Systematics and Backgrounds

We are looking for a very small change in the neutrino flux for day versus night. If the detector itself has day/night variations, these may mask or mimic this change. For example, if one of the backgrounds has a day/night difference, we may see this as an excess of counts during the day or night. Similarly for systematics, as they can change the sensitivity of the detector or the analysis region, which is particularly important for low energies. Also, long term variations in the detector will appear to be day/night differences as the seasons and the length of the day change.

The main analysis for the NCD phase in [57] is mostly insensitive to these effects.

However, since they can creep in to the main analysis, the Monte Carlo already models most of the known differences. Anything that was missed in the MC, however, needs to be accounted for. This was the goal of the Day/Night group within SNO, which studied as many of the systematics and backgrounds as possible for unmodeled day/night differences. These differences fell in to two categories: diurnal and directional.

Diurnal differences correspond to parameters that are truly different during the day versus at night. For example, the energy scale of the detector may have a true difference between day and night, if PMT voltages were less stable during the day due to more activity in the mine. These studies relied on looking at calibrations or in-situ measurements that had both day and night information available.

Directional differences are spatial (top v. bottom) differences in the detector that make parameters for upgoing versus downgoing events appear different. Note that these are upgoing or downgoing charged particles, not neutrinos. These produce Cherenkov radiation that illuminate the top or bottom of the detector, respectively. The question is how these up-down differences will then affect the apparent neutrino signal. Looking at Figure 5-1, we can see the answer. For NC, the answer is "not at all", as there is no correlation between the neutrino direction and the direction of motion of the detected charged particles. For CC, the answer is "not much", as the correlation is low. For ES, however, the answer is "a lot", since the scattered electron distribution is very sharply peaked in the direction of travel of the incident neutrino. Since neutrinos during the night will mostly be upgoing and those during the day will mostly be downgoing, this will mimic a day/night difference. As such, we only apply these asymmetries to ES and $\text{ES}_{\mu\tau}$ MC events. Measuring the directional asymmetries of many of the systematics was the main contribution to the day/night group by the author of this thesis. The results appear below, but the details of how these are measured are in Chapter 6.

The results of the asymmetry measurements of the Day/Night group are collected in [60]. We expect our detector to be the same day v. night, so those measurements which see a non-zero difference $\mu \pm \sigma$ are treated instead as if they had measured a

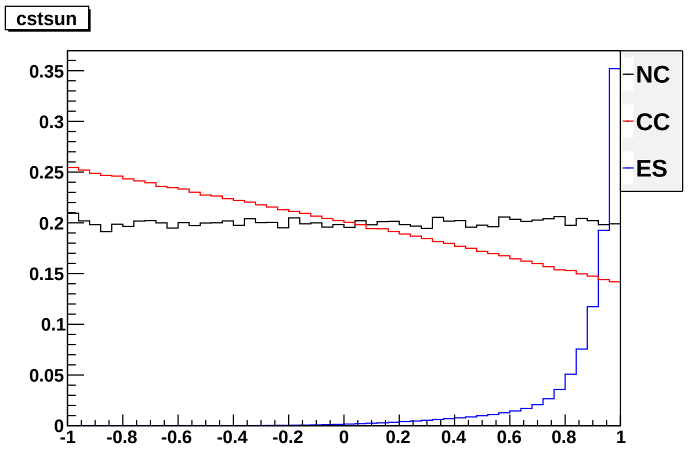


Figure 5-1: Flux event distribution for $\cos(\theta_{sun})$. The vertical axis is in arbitrary units.

zero difference with a larger error $0 \pm (\mu + \sigma)$ (there are two exceptions to this, both are measured background rates with real, expected asymmetries). This allows the fitting procedure to wander in to the larger area if the data takes it there, without biasing our results toward a non-zero asymmetry. Many of the asymmetry measurements found an asymmetry too small to be a concern, these had $A$ fixed to 0, i.e. they had $\alpha_D = \alpha_N$, to reduce the number of parameters in our fit. A parameter with $A = 0$ is described as not having an asymmetry of that type (directional or diurnal). The values for the measured asymmetries appear in Table 5.1, those with no entry have $A = 0$.

For a systematic with only a diurnal asymmetry, denoted $A_{diur}$, we simply use $A = A_{diur}$ and the formulas above for all fluxes and backgrounds. For one with only a directional asymmetry, denoted $A_{dir}$, we use $A = A_{dir}$ for the ES and ES$_{\mu\tau}$ fluxes and $A = 0$ for all other fluxes and backgrounds. For a systematic with both $A_{dir} \neq 0$ and $A_{diur} \neq 0$, we use $A = A_{dir} + A_{diur}$ for ES and ES$_{\mu\tau}$ and $A = A_{diur}$ for all other fluxes and backgrounds.

As in Chapter 4, here we only describe the asymmetries for the PMTs. As mentioned in Section 5.2, the NC flux is assumed to be constant, so the NCD rates are assumed to be the same day v. night and have $A = 0$ for all parameters. The NCDs are handled by the PSA analysis, described in Section 8.3.

| Parameter | Type | Center | Value | Measurement |
|---|---|---|---|---|
| Energy Scale $(a_0^E + a_{0,c}^E)^1$ | diur | 1 | $0 \pm 0.0038$ | $^{16}$N [66] |
| Energy Scale $(a_0^E + a_{0,c}^E)^1$ | dir | 1 | $0 \pm 0.0099$ | $^{16}$N, Chapter 6 |
| Energy Resolution $(b_0^E)^2$ | dir | 0 | $0 \pm 0.012$ | $^{16}$N, Chapter 6 |
| Position Scale $(a_1^{x,y,z})^3$ | diur | 1 | $0 \pm 0.0015$ | $^{16}$N [67] |
| Position Scale $(a_1^{x,y,z})^3$ | dir | 1 | $0 \pm 0.0018$ | $^{16}$N, Chapter 6 |
| $\cos(\theta_{sun})$ Resolution | dir | 1 | $0 \pm 0.069$ | $^{16}$N, Chapter 6 |
| ex background | diur | B | $-0.0195 \pm 0.0112$ | Radioactivity on NCDs [68] |
| d2opd background | diur | B | $-0.034 \pm 0.112$ | Radioactivity on NCDs [68] |

1 The Energy Scale is the sum of two parameters, see Section 4.7. The asymmetry is applied to their sum.

2 The Energy Resolution has different values when applied to neutrons v. electrons, this asymmetry is only applied to the electron value.

3 The Position Scale is $a_1^{x,y,z}$ for $x$ and $y$, and $a_1^{x,y,z} + a_1^z$ for $z$. The asymmetry is applied to these values.

Table 5.1: All day/night asymmetries for background and systematics. Parameters not listed have $A = 0$. "Type" indicates whether the asymmetry is diurnal (diur) or directional (dir), and "center" indicates whether the asymmetry is defined as equation 5.2 (0), equation 5.4 (1), or equation 5.8 (B).

# Chapter 6

# Asymmetry Analysis

## 6.1 Introduction

The Day-Night measurement in the SNO detector is, fundamentally, simply subtracting the electron neutrino flux during the day[1] from the flux during the night. Of course, that simple measurement could suffer from many problems that we hope to avoid, including artificial biases of many natures. To counteract this, we make a number of measurements of the properties of the detector that could artificially cause a day v. night difference. Any difference we see is a systematic error we need to either correct for or keep track of.

This report describes one of these measurements: differences between the response of the top and the bottom of the detector. In SNO, we are not capable of directly detecting neutrinos; instead, we detect electrons or neutrons that are generated when neutrinos interact with the water in our detector. There are three main types of interactions, charged current (CC), neutral current (NC) and elastic scattering (ES). The ES signal is neutrino-electron elastic scattering, which is dominated by electron neutrinos; the CC signal is the process $\nu_e + d \rightarrow p + p + e^-$, where $d$ is a deuteron, which only occurs for electron neutrinos; and the NC signal is $\nu_x + d \rightarrow \nu_x + p + n$, which occurs equally for all three neutrino flavors. For elastic scattering, the direction of motion of the electrons thus generated is highly correlated with the direction of

[1]When the sun is above the $\theta = \pi/2$ horizon in detector coordinates

motion of the neutrino itself. Since the sun is our source of neutrinos, the “night” signal will mostly be electrons moving toward the top of the detector and the “day” signal will mostly be toward the bottom of the detector. Thus, if the top of the detector and the bottom of the detector differ in some way, this will seem like a difference between day and night, but may be too subtle to be apparent in analyses that are not specifically looking for it.

The detector itself has several known top-bottom differences. The most prominent is the “neck”, a hole in both the acrylic vessel (AV) and the PMT support structure (PSUP). It is the cylindrical object at the top of the detector in Figure 3-2. The neck provides necessary physical access to the $D_2O$ volume, but has a corresponding hole in PMT coverage. In addition, the neck is made of a different material than the rest of the AV and is opaque to ultraviolet light. These are both very significant, as the electrons that we detect are actually detected via their Cerenkov radiation. This hole results in light being lost, and the algorithm that determines the electron’s track from the light turns this in to lowered energy for the electron. This is, for the most part, a well understood and well modeled problem, but is still the dominant top-bottom difference. In addition, the detector is a large tank of very pure water, leading to vertical thermal and density gradients. Finally, the PMTs in the detector are not all from the same batch (a batch is a set of PMTs produced at the same time), in particular the top and bottom of the detector are from different batches. These differences are supposed to be corrected by calibrations and adjusting the operating voltages and parameters of the PMTs, but subtle differences may remain.

## 6.2 Basic Methodology

The basic method followed is to look at events that illuminate the top of the detector, characterize their behavior and compare it with the behavior of events that illuminate the bottom of the detector. The top versus bottom asymmetries of these characterizations are what we seek, traditionally calculated in terms of some parameter S that characterizes the distribution[2]

$$A_S = 2\frac{S_{night} - S_{day}}{S_{night} + S_{day}} \tag{6.1}$$

Here we are not examining day versus night differences per se, but rather top versus bottom differences that can mimic them. However, we ultimately want a systematic uncertainty we can apply to our actual Day/Night measurements, so we will need to convert our measured asymmetry in to an $A_S$ of this form. To do this, we will weight our results by the expected spatial distribution of the neutrino flux, as will be discussed later in this section.

The method we describe here is derived from and similar to those used for this analysis for the salt [69] and $D_2O$ [58] phases of SNO. However, we make several additions and alterations to the method.

### 6.2.1 $^{16}$N source

To actually measure the physical asymmetries in the detector, we can use any calibration source that illuminates both the top and the bottom of the detector. The $^{16}$N calibration runs were used in the analysis described here, as they have both this property and many additional properties that make them desirable for this analysis: they sample the detector spatially, occur at intervals throughout the NCD phase and thus sample the detector temporally, have an energy spectrum that is in the range we expect neutrino induced events to fall in, are well modeled with a corresponding Monte Carlo, and have a large number of events to provide good statistics.

$^{16}$N is a radioactive isotope of nitrogen with a half-life of 7.1 s, which decays to $^{16}$O via $\beta$ decay. The $^{16}$O is created in an excited state and releases photons en route to the stable $^{16}$O ground state. With such a short half life, the $^{16}$N was produced on site by the reaction $^{16}O(n,p)^{16}N$. The generated $^{16}$N was then flowed into a stainless steel decay chamber in the $D_2O$, which absorbed the $\beta$ particle and

[2]An example of S examined in this report is the center of a Gaussian fit to the distribution of electron energies

allowed the photon to enter the detector. Three branches dominate the $\beta$ decay: one produces no photon (28% of decays), another produces a photon with 6.1 MeV (66% of decays) and the third produces a photon with 7.1 MeV (5%). Other photon energies between 2 MeV and 8.9 MeV are possible, but occur in less than 1% of decays. The emitted photons undergo Compton scattering with the electrons in the heavy water, producing electrons that give off Cerenkov radiation that SNO's PMTs detect. It is these electrons that we are talking about when we say "electrons" in measurements in this report.

For each data run, this source is placed in the detector at a location that varies from run to run but is constant for each run, and $^{16}$N is pumped in for the duration of the run. The detector otherwise behaves normally, so the light generated by the electrons is captured by the PMTs. Given the distribution and timing of the PMT response, a reconstruction algorithm determines where the electron originated from (the event's position), its energy and the direction of its velocity. It is this information that is stored in the $^{16}$N data set and is examined in this report.

In addition, a very well developed Monte Carlo simulation of the detector exists for each data run. It simulates the expected results for the run, with the $^{16}$N source at the run's position. Well known top-bottom asymmetries present, such as the neck, are included in the model and corrected for when the neutrino signal is extracted. We are not concerned with these asymmetries, and need to subtract out their effects - we are looking to determine a correction to apply to the MC. To do this, for each parameter we measure, we also measure the same parameter in the Monte Carlo simulation. It is only if these two differ that we are concerned, so we either subtract (or divide) the two, and then look for deviations from either zero (or one). This leaves us with only those asymmetries that are corrections to our model, which are systematics of the form discussed in Section 4.7.

### 6.2.2 Detector binning

Not all up going events (or down going events) are equal. Those that originate from the bottom of the detector (or top of the detector) will have a larger Cerenkov ring

| Bin | Radius (cm) | $\theta_{spatial}$ |
|---|---|---|
| 0 | $r \leq 150$ | $0 \leq \theta \leq \pi$ |
| 1 | $150 < r \leq 375$ | $0 \leq \theta < \frac{\pi}{8}$ |
| 2 | $150 < r \leq 375$ | $\frac{\pi}{8} \leq \theta < \frac{3\pi}{8}$ |
| 3 | $150 < r \leq 375$ | $\frac{3\pi}{8} \leq \theta < \frac{5\pi}{8}$ |
| 4 | $150 < r \leq 375$ | $\frac{5\pi}{8} \leq \theta < \frac{7\pi}{8}$ |
| 5 | $150 < r \leq 375$ | $\frac{7\pi}{8} \leq \theta < \pi$ |
| 6 | $375 < r \leq 550$ | $0 \leq \theta < \frac{\pi}{8}$ |
| 7 | $375 < r \leq 550$ | $\frac{\pi}{8} \leq \theta < \frac{3\pi}{8}$ |
| 8 | $375 < r \leq 550$ | $\frac{3\pi}{8} \leq \theta < \frac{5\pi}{8}$ |
| 9 | $375 < r \leq 550$ | $\frac{5\pi}{8} \leq \theta < \frac{7\pi}{8}$ |
| 10 | $375 < r \leq 550$ | $\frac{7\pi}{8} \leq \theta < \pi$ |

Table 6.1: Spatial bins used in the analysis. $r$ and $\theta_{spatial}$ defined relative to the detector center and zenith.

and thus affect more PMTs and a larger area of the detector. If some form of non-uniformity is present in the detector, these events could be affected by it differently. Since we have ample statistics, we divide the detector in to spatial bins to separate these events.

Following the lead of Jeff Secrest's Energy Systematics study [61], we use "LETA style" bins, giving us 11 spatial bins. These bins are defined in terms of divisions in $r$ (radius relative to the center of the detector) and $\theta_{spatial}$ (polar angle relative to the zenith direction in the detector, i.e. straight up). See Table 6.1 for the bins used. These bins are annular sections spherical shells in the detector. For simplicity, an event is placed in the bin corresponding to the location of the $^{16}$N source during that data run. This division, however, is insufficient for determining whether an event is up going or down going.

To actually separate up going events from down going events, we further divide these spatial bins in to direction bins, defined in terms of the direction of the electron's reconstructed velocity, $\frac{\vec{v}}{|\vec{v}|}$. Since we are only concerned about the direction of motion, we only need two variables to describe it. The polar coordinates $\theta_{direction}$ and $\phi_{direction}$ are the traditional choice, measured relative to the same fixed detector coordinates as the spatial bins. Here we use 5 equal bins each in $\cos\theta_{direction}$ and a modified $\phi$, which we call $\phi_{center}$.

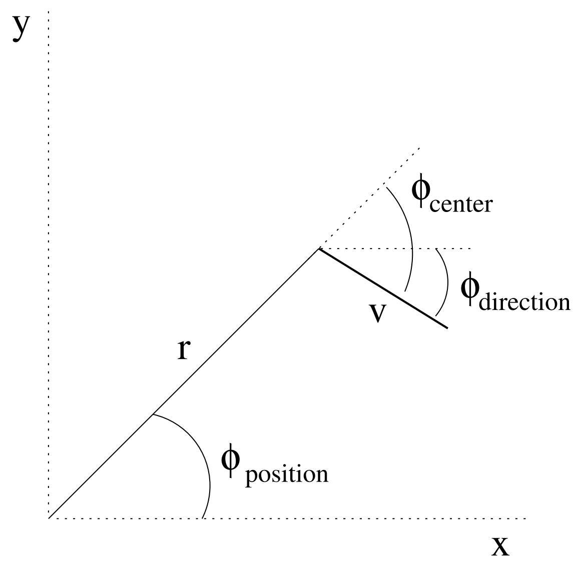


Figure 6-1: The x and y axes are fixed in the detector. Note that both $\phi_{direction}$ and $\phi_{center}$ are negative.

The choice to use $\phi_{center}$ comes from the realizations that the dominant physical top-bottom asymmetry in the detector comes from the neck and that the detector is approximately azimuthally symmetric. Hence taking all of the events in a ring-shaped spatial bin and looking at those moving "left" or "right", i.e. binned in $\phi_{direction}$ in a set of detector-fixed coordinates, will average out any neck effects. Also, all of the $\phi_{direction}$ bins would look approximately the same, making them not a useful division. Instead, we use a set of bins that measure whether an event is moving toward z-axis of the detector. The simplest way to do this is

$$\phi_{center} = \phi_{direction} - \phi_{position} \tag{6.2}$$

Where $\phi_{direction}$ is the detector-fixed direction $\phi$ and $\phi_{position}$ is the azimuthal angle $\phi$ of the event's position, relative to the same detector-fixed coordinates from the spatial bins. $\phi_{center}$ is then the angle between the electron's direction of travel and the direction from the center of the detector to it. This is illustrated in Figure 6-1.

As noted earlier, each data run had the $^{16}$N source at a fixed position in space, and the runs occurred throughout the NCD phase. To be able to look for trends in time, we actually do our full binning and fitting procedure on each run individually.

This puts runs as wholes in spatial bins. We then average each of the 25 direction bins over the runs in a given spatial bin to give the resulting directional bin values for each spatial bin.

### 6.2.3 Weighting

Now that a binning scheme is defined, we can finally return to the issue of transforming from the top-bottom asymmetry to the day-night asymmetry. To accomplish this, we use an additional set of data: the expected neutrino signal. Given theoretical knowledge, the previous two phases of SNO, and a lot of hard work on the part of the simulation group [44], a very good Monte Carlo simulation of the detector and its expected behavior in response to the neutrino flux from the Sun exists[3]. We use this MC to convert our measurements in terms of our spatial and direction bins in to a measurement of the expected effect on the signal fluxes.

First, we separate this MC in to the three signals (ES, CC, NC) and in to day and night. For each signal, we then bin the MC events in the same 275 bins (11 spatial x 25 direction) as the $^{16}$N data, for day and night separately.

Next, to convert a measurement in to day and night, we sum the measured values across the bins, weighted by the expected neutrino signal in that bin. Since we are generally looking at the difference between data and Monte Carlo, the weighted sum is

$$\begin{aligned} \Delta S_i &= S_{i,data} - S_{i,mc} \\ A_{day} &= \frac{\sum_{i=1}^{275} \Delta S_i \ \nu_{i,day}}{\sum_{i=1}^{275} \nu_{i,day}} \end{aligned} \tag{6.3}$$

where $S$ is the measured parameter and $\nu_i$ is the expected neutrino signal in bin $i$. $A_{night}$ is calculated in an identical manner. We use two different methods of calculating $A_{day}$ in this document. For the energy values, a fit is done for each run,

[3]We obtained this signal MC from [70]

to give $S_{i,data}$ and $S_{i,mc}$ for each run. We then do a simple weighted average over runs for data and MC separately, then subtract to arrive at the $\Delta S_i$ in the previous equation. For both the angular resolution and position reconstruction, the individual run fits are still done, but then $\Delta S_i$ is calculated for each run, and a weighted average is done over these $\Delta S_i$'s. This latter way is more appropriate and a better measure of comparing like to like. The energy was done differently because it didn't appreciably affect the results and was simpler to implement.

This serves two purposes: it averages our measurements across the detector, and weights bins according to how they contribute to the day-night asymmetry. So if a bin is expected to have the same signal in day and night, when we take $A_{night} - A_{day}$ that bin will not contribute. This becomes more apparent when we rewrite the difference as:

$$N_{day} = \sum_{j=1}^{275} \nu_{j,day} \ , \qquad N_{night} = \sum_{j=1}^{275} \nu_{j,night}$$

$$A_{night} - A_{day} = \sum_{i=1}^{275} \Delta S_i \left( \frac{\nu_{i,night}}{N_{night}} - \frac{\nu_{i,day}}{N_{day}} \right) \tag{6.4}$$

To compute the error on this $A_{day}$ and $A_{night}$, we simply do propagation of errors. We define two parameters, $N$ and $B$, to make the equation more readable.

$$N = \sum_{j=1}^{275} \nu_{j,day} \ , \qquad B = \sum_{j=1}^{275} \Delta S_j \, \nu_{j,day}$$

$$\sigma^2_{A_{day}} = \frac{1}{N^2} \sum_{i=1}^{275} (\nu_{i,day} \, \sigma_{\Delta S_i})^2 + \sum_{i=1}^{275} \left[ \left( \frac{\Delta S_i}{N} - \frac{B}{N^2} \right) \sigma_{\nu_{i,day}} \right]^2 \tag{6.5}$$

We note that since $\nu_{i,day}$ is simply counting, its uncertainty is its square root. The uncertainty for $A_{night}$ is calculated in an identical manner.

For each of the different quantities studied in the detector, we make small adjustments to this method as needed. For each quantity we will explain what $S$ we are measuring, how we define $\Delta S$ (which can be either $\Delta S = S_{data} - S_{mc}$ as above or $\Delta S = \frac{S_{data}}{S_{mc}} - 1$) and how $\sigma_{\Delta S}$ is calculated.

## 6.3 Energy

The first set of parameters we study are those related to the ability of the detector to measure energy. All measurements at SNO are sensitive to shifts in the energy scale, and this is measured as a systematic for other SNO results [61]. However, we are concerned with any shift in energy scale that is asymmetric across the detector as opposed to an overall shift.

To measure the energy scale, we look at the measured energies for the electrons in our data. In each of our 275 bins, we build a histogram of measured energies of electrons, which we expect to be the same for each bin, and fit it. Since the energy spectrum is mostly Gaussian, but with non-Gaussian behavior in the tails (see Figure 6-2), we use an iterated Gaussian fit. For each histogram, the mean $\mu$ and RMS are determined, and a Gaussian is fit over $\mu \pm$ RMS. The $\mu$ and $\sigma$ of the fit Gaussian are extracted, and the fit is repeated over $\mu \pm 1.6\sigma$. This is repeated twice, and the resulting $\mu$ and $\sigma$ are the $S$'s for our first two asymmetry measurements: the energy mean and the energy width. For a sample fit, see Figure 6-2.

For the energy mean, $S = \mu$ and we use $\Delta S = 1 - S_{data}/S_{mc}$, so that we are measuring and averaging percentage changes in the energy. This corresponds to changes in the energy scale, a quantity that is well understood. For $\sigma_{\Delta S}$, we use the uncertainty on the fit value of $\mu$ as given by Root's Minuit fitter.

For a few runs, the fit was very poor. To avoid these runs causing problems, they were cut by a generic cut on any run with less than 200 events. This cut approximately 5000 events out of approximately 7,000,000 events for the total set by cutting 124 out of 1300 runs.

Figure 6-3 shows the fit $\mu$ value versus run number for a randomly chosen bin. Since increasing run number corresponds to increasing time, any trend here would be a change over time in the detector. None seems to be present.

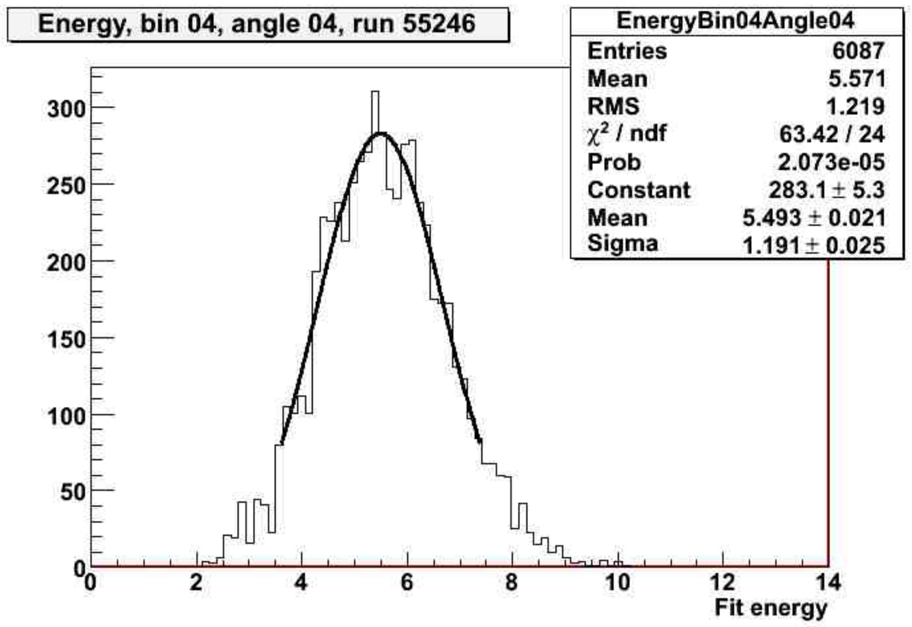


Figure 6-2: Histogram of event energy for randomly selected run, fit with iterated Gaussian

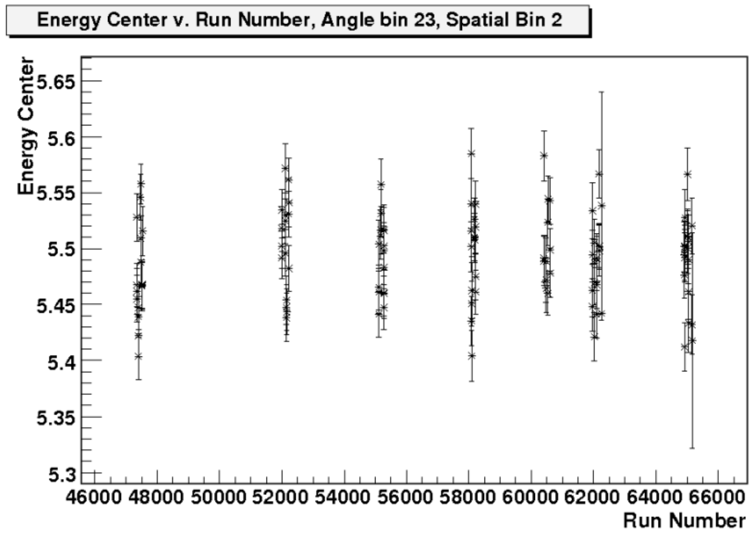


Figure 6-3: Fit center v. run number for randomly selected bin, data only

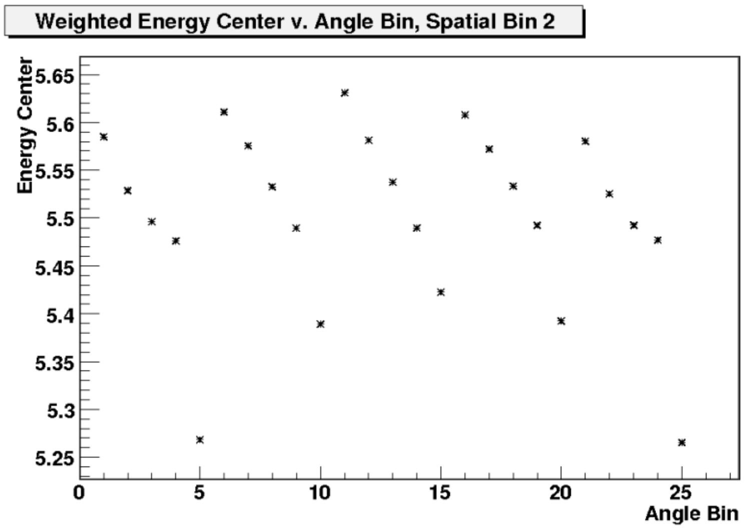


Figure 6-4: Fit $\mu$, averaged over runs v. angle bin for spatial bin 2, data only. See text for an explanation of the pattern in this and subsequent graphs.

Figure 6-4 shows the 25 angular bins in spatial bin 2, after averaging over the data runs. The same bin in Monte Carlo is shown in Figure 6-5. An interesting trend is the clear progression in groups of 5 bins, these correspond to going from the bottom of the detector to the top. Apparently light and energy are lost in the neck region, which is not surprising. This pattern is matched between data and Monte Carlo, and does not affect our final result.

To proceed to the final result, we apply our weighting procedure. This results in six measurements, one each for ES, CC and NC, for day and for night. The difference between day and night for each of these is our total asymmetry, shown in Table 6.2.

| Signal | Weighted Average |
|---|---|
| ES, Day | -0.0082 $\pm$ 0.0096 |
| ES, Night | 0.0017 $\pm$ 0.0095 |
| CC, Day | -0.0024 $\pm$ 0.0080 |
| CC, Night | -0.0036 $\pm$ 0.0080 |
| NC, Day | -0.0036 $\pm$ 0.0080 |
| NC, Night | -0.0036 $\pm$ 0.0080 |

Table 6.2: Energy center shift as fraction, weighted average over detector

For the energy width, we repeat this procedure for $\sigma$ from the same Gaussian fit. Figure 6-6 shows the fit $\sigma$ value versus run number for the same bin as before.

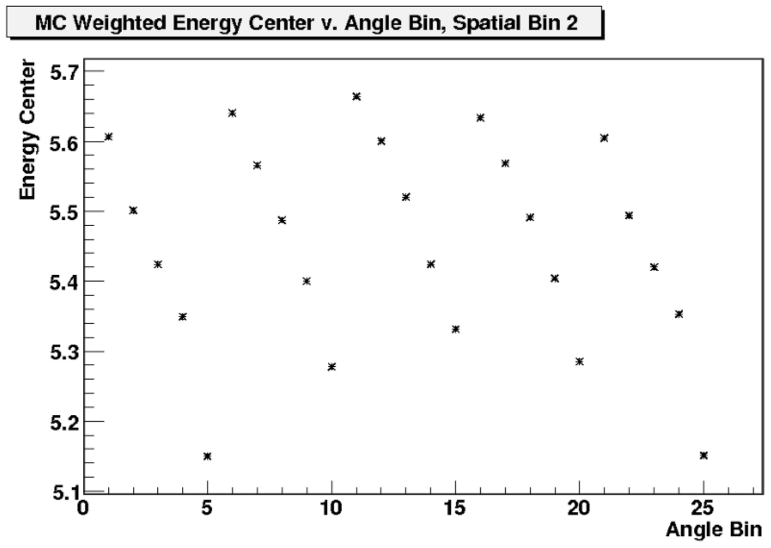


Figure 6-5: Fit $\mu$, averaged over runs v. angle bin for spatial bin 2, MC only

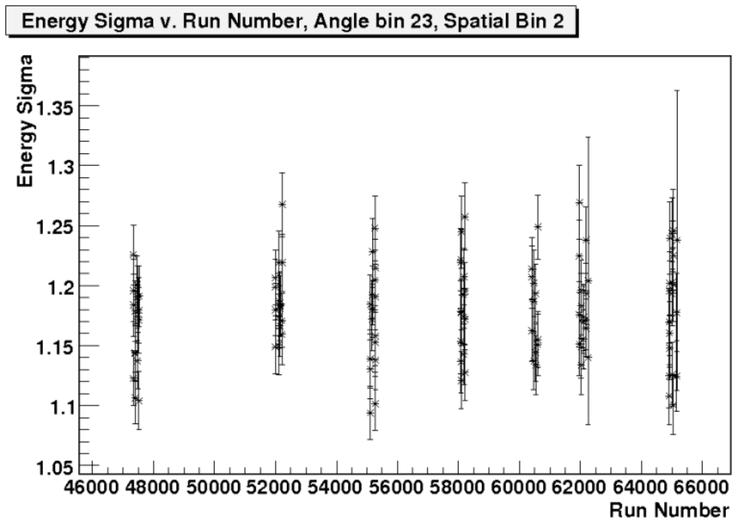


Figure 6-6: Fit $\sigma$ v. run number for randomly selected bin, data only

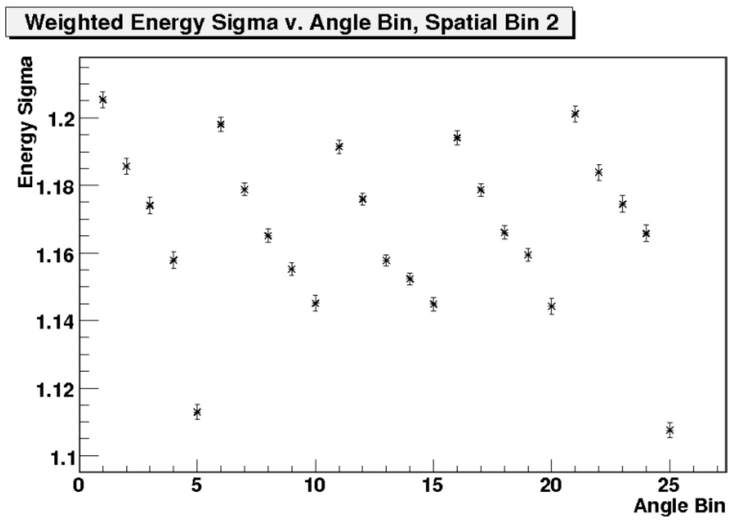


Figure 6-7: Fit $\sigma$, averaged over runs v. angle bin for spatial bin 2, data only

Figure 6-7 shows the 25 angular bins in spatial bin 2, as in the $\mu$ case. The same loss of energy to the neck is seen, and is again matched in Monte Carlo. Averaging across all bins leads to the final results, summarized in Table 6.3.

| Signal | Weighted Average |
|---|---|
| ES, Day | $0.001 \pm 0.022$ |
| ES, Night | $0.013 \pm 0.022$ |
| CC, Day | $0.009 \pm 0.018$ |
| CC, Night | $0.007 \pm 0.018$ |
| NC, Day | $0.008 \pm 0.018$ |
| NC, Night | $0.008 \pm 0.018$ |

Table 6.3: Energy sigma shift as fraction, weighted average over detector

## 6.4 Angular Resolution

Another aspect of the detector that is of great concern is the angular resolution. The angular resolution is how well the detector is able to resolve the direction of motion of the object we are trying to measure - neutrinos in the case of SNO's normal data taking, the photons from $^{16}$N in the case of the calibration runs. We measure this for the calibration data by measuring the angle between the electron's velocity and the straight line between the source and the event location. This line is the path

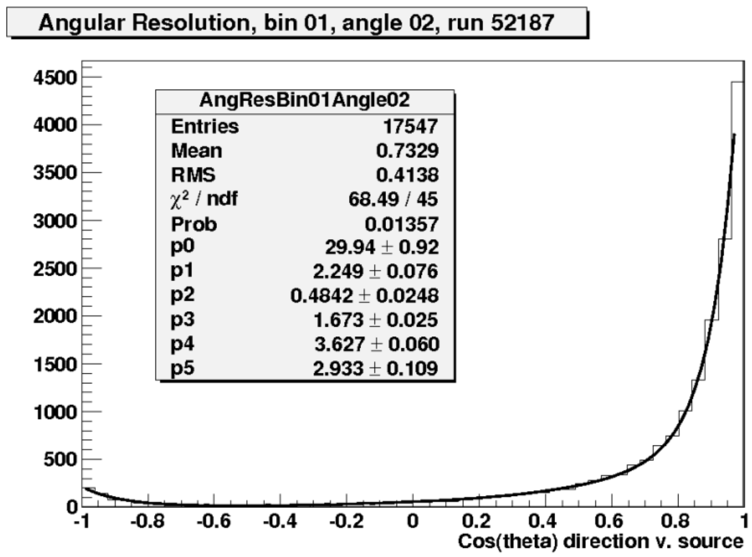


Figure 6-8: Sample fit for $\cos\theta$. Order of parameters: A, $\alpha_1$, $\alpha_2$, $\beta_1$, $\beta_2$, $\beta_3$
Note that parameters here are square roots of respective quantities

the photon travelled from the source (we neglect multiple scattering), and hence its direction of motion. In equation form, this is:

$$\cos\theta = \frac{(\vec{x}_{event} - \vec{x}_{source}) \cdot \vec{v}}{|\vec{x}_{event} - \vec{x}_{source}||\vec{v}|} \tag{6.6}$$

We expect this distribution to be strongly peaked near $\cos\theta = 1$, i.e. $\theta = 0$. Looking at a sample histogram (Figure 6-8), we see that this is the case, and that this is clearly a non-Gaussian distribution. Typically, this distribution is fit to the sum of two exponentials, but our statistics are sufficient that we can see a small rise near $\cos\theta = -1$ as well, so our fit to the histogram is to three exponentials (where x is $\cos\theta$):

$$N(x) = A(e^{\beta_1(x-1)} + \alpha_1\, e^{\beta_2(x-1)} + \alpha_2\, e^{\beta_3(x+1)}) \tag{6.7}$$

Figure 6-8 also shows a sample fit. We note that there is an ambiguity in the fitting process: the first two terms are identical, and can be switched by letting $\alpha_1 \to \alpha_1^{-1}$ and $\beta_1 \leftrightarrow \beta_2$. To account for this, if $\alpha_1 < 1$, we make this switch in our graphs and analysis. Due to our choice of initial conditions for our fitter, this means we are choosing $\beta_2 > \beta_1$, so $\beta_2$ corresponds to the steep slope and $\beta_1$ to the shallow slope.

For this measurement, our binning becomes a problem. In particular, using $\phi_{center}$ is a very poor choice. For a source position on the z-axis, effectively all events are "outgoing" events. The "in going" bins then have only the small tail near $\cos\theta = -1$, but exaggerated due to the removal of most of the other events. In essence, our $\phi_{center}$ bins are correlated with our measured value. Since the $\phi_{direction}$ bins, i.e. those fixed with respect to the detector, are basically identical due to axial symmetry, we only use the 5 bins in $\cos\theta_{direction}$. Our averaging scheme as defined earlier still applies - we simply sum over the 5 bins instead of 25 angular bins (so 55 bins total, rather than 275).

Following our prescription for our analysis, we need to define $S$, $\Delta S$ and $\sigma_S$. We actually have three separate $S$'s, the $\alpha_1$, $\beta_1$, and $\beta_2$ from our fit, so we can see how the distribution changes across the detector. We aren't actually interested in characterizing the tail at $\cos\theta = -1$, so we do not look at $\alpha_2$ or $\beta_3$; they are only used to improve the quality of the fits. As in the energy section, we define $\Delta S = \frac{S_{data}}{S_{mc}} - 1$, here calculated for each run, and use the error on the fit produced by Root's Minuit fitter for $\sigma_S$. We have an additonal cut on events for this section, rejecting those events less than 60 cm away from the source, to allow for the various angles to be well measured. Due to some fitting problems, we reject those fits with $\chi^2/dof > 4$, where dof is number of degrees of freedom. In addition, a few bad fits slipped past this cut, so fits with values of $\Delta S$ more than 6 $\sigma_{\Delta S}$ from their bin's average are removed (as a simple outlier rejection). These two cuts together cause most bins to lose 0 to 2 runs, but one exceptional bin loses 10 runs out of 52, and the next worst bin loses 8 runs out of 113.

A sample of $\beta_2$ v. run number is shown in Figure 6-9. No pattern or drift over time is apparent. We now average across runs to get $\frac{\beta_{2,data}}{\beta_{2,mc}} - 1$ for each of our 55 bins (11 spatial and 5 angular, for this section). All 55 bins are shown in Figure 6-10, with each group of five bins being one spatial bin, so bins 0 through 4 are in spatial bin 0, bins 5 through 10 are in spatial bin 1, etc.

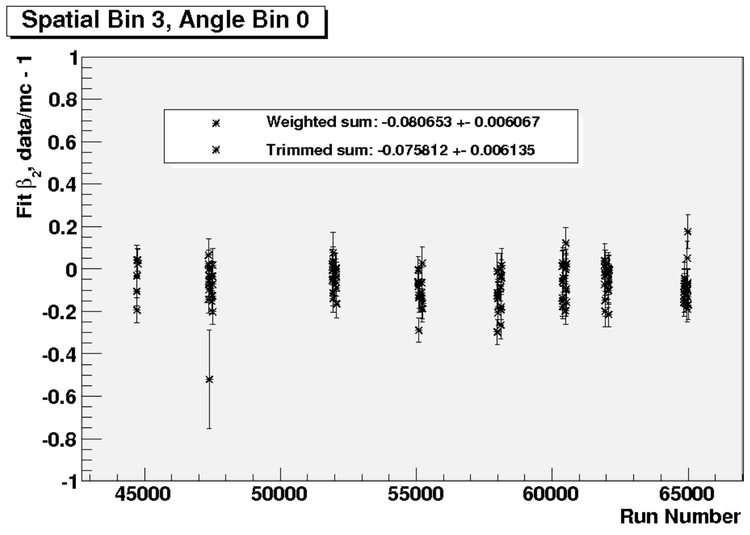


Figure 6-9: Sample $\frac{\beta_{2,data}}{\beta_{2,mc}} - 1$ v. run number

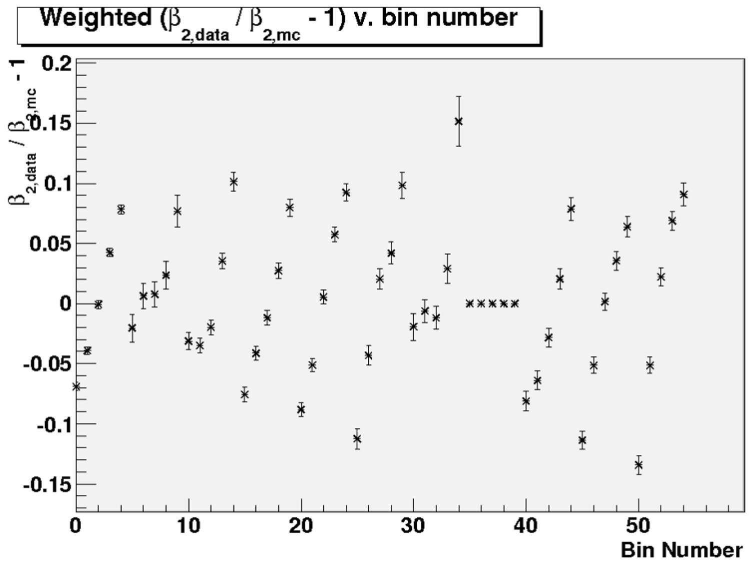


Figure 6-10: Averaged $\frac{\beta_{2,data}}{\beta_{2,mc}} - 1$ for all bins. Each group of five is one spatial bin, with the flat region being the empty bin 7, which is set to zero for graphing.

This process is repeated for $\beta_1$ and $\alpha_1$. The results are weighted using the neutrino flux weightings, using the 55 bins in this section rather than the 275 bins from before. The resulting day and night values are summarized in Table 6.4.

| | $\alpha_1$ | $\beta_1$ | $\beta_2$ |
|---|---|---|---|
| ES, Day | -0.042 ± 0.059 | -0.017 ± 0.048 | -0.043 ± 0.009 |
| ES, Night | 0.007 ± 0.058 | 0.044 ± 0.047 | 0.026 ± 0.009 |
| CC, Day | -0.017 ± 0.049 | 0.019 ± 0.040 | -0.003 ± 0.008 |
| CC, Night | -0.018 ± 0.049 | 0.011 ± 0.040 | -0.011 ± 0.008 |
| NC, Day | -0.016 ± 0.047 | 0.014 ± 0.039 | -0.009 ± 0.007 |
| NC, Night | -0.017 ± 0.048 | 0.014 ± 0.039 | -0.009 ± 0.007 |

Table 6.4: Angular resolution fit parameters as fractional change data v. MC, weighted average over detector

## 6.5 Position Reconstruction

In this section, we will be looking for any offsets in space in the reconstruction algorithm and detector response. To do this, we look at $\vec{x} - \vec{x}_{source}$, i.e. the distance of an event from the $^{16}$N source, as three separate components (X, Y, and Z). Here we use 25 angular bins, as in the Energy section. We apply the same algorithm as before, except that we are now looking at $\Delta S = S_{data} - S_{mc}$ instead of $\Delta S = \frac{S_{data}}{S_{mc}} - 1$ as in previous sections. The reason for this change is that we expect the difference to be approximately zero, which makes any quantity relying on division not useful (and potentially infinite).

The quantity $X - X_{source}$ (or Y or Z) is expected to be roughly gaussian, but with heavy tails. This is usually fit to an empirical function that is the sum of a gaussian and a decaying exponential, i.e.

$$A \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right) + B \exp\left(\frac{|x-\mu|}{\tau}\right) \tag{6.8}$$

This equation fits the data reasonably well. However, it was not used here due to problems with the fitting algorithm. Instead, since we are merely characterizing the center of the peak, we opted for the iterated gaussian fit used in the Energy section.

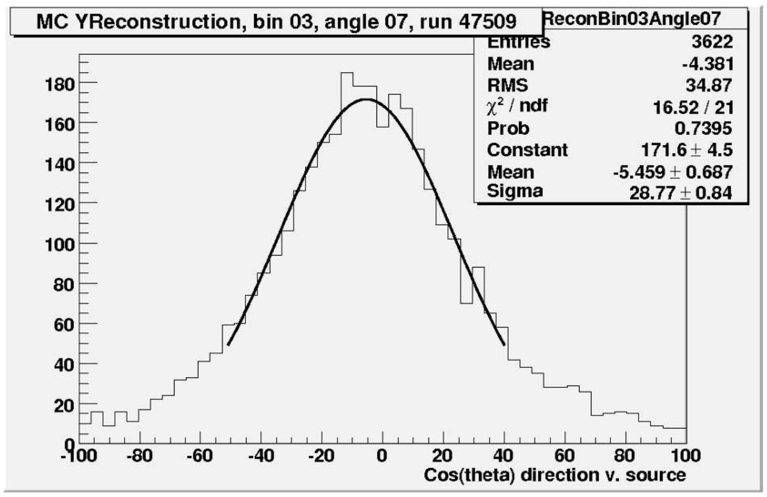


Figure 6-11: Sample fit for $(X - X_{source})$

This does a good job of finding the center of the distribution, since it is restricted to $\mu \pm 1.6\sigma$ and thus not strongly influenced by the non-gaussian tails of the distribution. A sample distribution and fit is shown in Figure 6-11.

As in the angular resolution, each run is fit individually, then a weighted average of $\Delta S = \mu_{data} - \mu_{mc}$ is done to arrive at the $\Delta\mu$ for each of the 275 bins. However, here the uncertainty on $\mu_{data}$ is not simply the fitter error from Minuit, as there is also an uncertainty in determining the position of the source. To account for this, a 3 cm uncertainty is attributed to $X_{source}$ for the data, adding in quadrature to the fitter error. This is representative of a typical uncertainty in determining the manipulator position. Its exact value has very little effect on the results of this analysis, as long as it is of order a few centimeters. The $\mu$ for each run for a randomly selected bin is shown in Figure 6-12. No particular pattern is apparent, as expected. An example of the averaged difference in the angular bins of a randomly selected spatial bin is shown in (Figure 6-13). The groups of five bins are due to the bin ordering, with every fifth bin the start of a new $\phi_{center}$, as this is a 1-D listing of a 2-D bin ordering. Finally, averaging over all bins gives the final results in Table 6.5. We expect to see an offset in Z, but one that is the same for day and night, as is present.

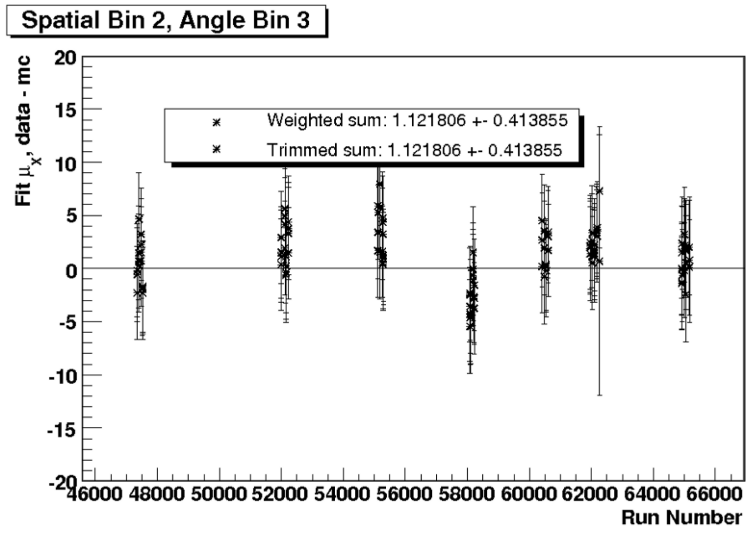


Figure 6-12: Fit $(X - X_{source})$, data - MC, v. Run Number

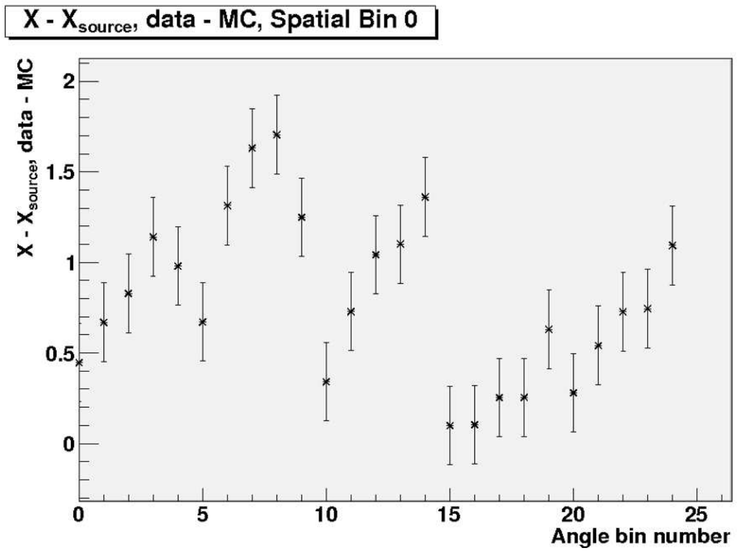


Figure 6-13: Averaged $(X - X_{source})$, data - MC, v. angle bin, for spatial bin 0

|  | X | Y | Z |
|---|---|---|---|
| ES, Day | 0.92 ± 0.30 | 0.89 ± 0.75 | 5.01 ± 0.84 |
| ES, Night | 1.24 ± 0.30 | 0.75 ± 0.73 | 6.08 ± 0.82 |
| CC, Day | 1.10 ± 0.24 | 0.80 ± 0.62 | 5.61 ± 0.69 |
| CC, Night | 1.06 ± 0.24 | 0.82 ± 0.62 | 5.50 ± 0.69 |
| NC, Day | 1.09 ± 0.22 | 0.83 ± 0.58 | 5.54 ± 0.64 |
| NC, Night | 1.09 ± 0.23 | 0.84 ± 0.58 | 5.53 ± 0.65 |

Table 6.5: Shift of position, in cm

## 6.6 Conclusion

Overall, the differences between day and night are small, almost all within uncertainties with zero. As expected, the difference for ES is larger than for CC or NC, in all measurements. Table 6.6 shows the final asymmetries, as a night minus day difference, for ES for each of the measurements made. The relevant point is that the energy asymmetries are approximately 1%, the angular resolution parameter asymmetries are all within a few percent (but not as well measured), and the reconstructed spatial asymmetries are all less than 1 cm.

The error bars quoted for the night-day asymmetries are simply the day and night value uncertainties added in quadrature. This is most likely an overestimation, as this assumes that the day and night have no systematics in common. This is not possible given how these values were calculated. However, in the NCD Day/Night analysis, only the measured central values will be used as the size of the systematic error. The uncertainties quoted here will not be used in the analysis, except possibly as a limit on how far the floated asymmetry systematic is allowed to wander.

These values are converted to day/night directional asymmetries in [60]. We assume that all of the day/night differences are zero, with uncertainty given by the central value. The suggested directional asymmetries for the systematics and their corresponding parameter from this study are given in Table 6.7.

| | ES | CC | NC |
|---|---|---|---|
| Energy $\mu$ | 0.010 $\pm$ 0.014 | -0.001 $\pm$ 0.011 | 0.000 $\pm$ 0.011 |
| Energy $\sigma$ | 0.012 $\pm$ 0.031 | -0.002 $\pm$ 0.025 | 0.000 $\pm$ 0.025 |
| Angular Resolution $\alpha_1$ | 0.049 $\pm$ 0.083 | -0.001 $\pm$ 0.069 | -0.001 $\pm$ 0.067 |
| Angular Resolution $\beta_1$ | 0.061 $\pm$ 0.083 | -0.008 $\pm$ 0.057 | 0.000 $\pm$ 0.055 |
| Angular Resolution $\beta_2$ | 0.069 $\pm$ 0.013 | -0.008 $\pm$ 0.013 | 0.000 $\pm$ 0.010 |
| Reconstructed X (cm) | (0.32 $\pm$ 0.42) | (-0.04 $\pm$ 0.34) | (0.00 $\pm$ 0.32) |
| Reconstructed Y (cm) | (-0.14 $\pm$ 1.05) | (0.02 $\pm$ 0.88) | (0.01 $\pm$ 0.82) |
| Reconstructed Z (cm) | (1.07 $\pm$ 1.17) | (-0.11 $\pm$ 0.96) | (-0.01 $\pm$ 0.91) |

Table 6.6: Night - Day asymmetries for all measurements made. All but Reconstructed positions are a fractional shift, (data/mc - 1)

| Systematic | Parameter | Suggested Value |
|---|---|---|
| Energy Scale | Energy $\mu$ | $0 \pm 0.0099$ |
| Energy Resolution | Energy $\sigma$ | $0 \pm 0.012$ |
| $\cos(\theta_{sun})$ resolution | Angular Resolution $\beta_2$ | $0 \pm 0.069$ |
| XShift | Reconstructed X | 0 (negligible) |
| YShift | Reconstructed Y | 0 (negligible) |
| ZShift | Reconstructed Z | 0 (negligible) |

Table 6.7: Translation of values from the $^{16}$N directional asymmetry study to suggested directional asymmetries for ES events

# Chapter 7

# Markov Chain Monte Carlo

The main signal extraction presented in Chapter 8 minimizes the difference between the Monte Carlo and the data, varying proportions of the various signals and backgrounds and distortions in the MC (the systematics). This involves a large number of parameters and each step is very computationally expensive, making a typical minimizer such as Minuit [71] impractical. Instead, we turn to the Markov Chain Monte Carlo (MCMC) method. Here we describe the basic ideas of the method, the particular MCMC algorithm we have chosen, and why this method is preferable for this problem.

## 7.1 Definitions

The Markov Chain Monte Carlo (MCMC) method derives from the principles of Bayesian statistics. We will not go in to the fundamentals of this, which can be found in a number of references [72, 73]. But we will need a few definitions of terms that are commonly used in this field, as the following discussion will rely on this notation. In addition, we define several terms as shorthand or for convenience in the following discussion. To illustrate the meaning of these quantities, we look at a hypothetical problem: we have a model of the weather in Boston, and wish to test its validity.

$p(x)$ is the probability distribution function (pdf) for x. This requires context -

if x is Temperature, then $p(x)$ for Boston will be different from $p(x)$ for Orlando, which will be very different than $p(x)$ for a quark-gluon plasma.

$p(x|y)$ is a conditional probability of x, given y (i.e. assuming y is true). Continuing our example, $p(x|\text{summer})$ will have much more weight near 80°F than $p(x|\text{winter})$.

$\{\vec{x}\}$ is a set of data points. Individual points will be denoted $\vec{x}_i$. Our example could be a set of measurements of temperature and wind speed at various times.

$\{\vec{y}\}$ is a set of simulated events from a Monte Carlo based on our model. Individual points will be denoted $\vec{y}_i$.

$\vec{\alpha}$ is a set of model parameters. Again, individual points will be denoted $\alpha_i$. Our example could be the season and the time of day, so that we are asking our model to only look at summer afternoons.

## 7.2 Markov Chain Monte Carlo

The method of the Markov Chain Monte Carlo is a combination of two parts: a "Markov Chain", a random walk with the next step only dependent on the current step; and a "Monte Carlo", a simulation method using repeated random sampling. It requires a known relation between the data distribution and a set of model parameters, giving $p(\vec{x}|\vec{\alpha})$ and a measured set of data points $\{\vec{x}\}$; it returns information about the values of $\vec{\alpha}$, i.e. about the probability distribution $p(\vec{\alpha}|\{\vec{x}\})$ (called the posterior distribution). The method randomly walks through the space of $\vec{\alpha}$ in a particular way (which defines the algorithm but is always a Markov Chain) and returns a random sample (the Monte Carlo aspect) from $p(\vec{\alpha}|\{\vec{x}\})$, usually called a "chain".

This random sample is akin to the output of a "normal" fitter, most of which assume that this distribution is Gaussian to report some value $\alpha_i \pm \sigma_{\alpha_i}$ for each model parameter. To convert from $p(\vec{\alpha}|\{\vec{x}\})$ to $\alpha_i \pm \sigma_{\alpha_i}$, we need merely assume that $p(\vec{\alpha}|\{\vec{x}\})$ is Gaussian near its maximum and, for each parameter individually, either fit with a Gaussian or numerically compute the mean and standard deviation of the random sample. But with the MCMC methods, we have the flexibility of looking directly at the probability distribution if we wish.

The method relies on having a way to define the probability that a data set was drawn from a distribution, since we will be varying our distribution as we vary $\vec{\alpha}$. This is most naturally done with the Likelihood function, described in the next section. In the context of the overall analysis, the question arises of how to describe $p(\vec{x}|\vec{\alpha})$ at all, given the complexity of the experiment forces us to rely on Monte Carlo simulations. We will delay answering this question until Chapter 8; for the rest of the chapter we will assume we have this available to us.

There are many MCMC algorithms (see [72, 74] for an overview), differing mainly in how the random walk through parameter space is implemented. The simplest, and first to be discussed, is the Metropolis algorithm [75]. This is the one we will use. Many factors went in to this decision. The most proiminent are that the more advanced algorithms often require the analytic derivative of $p(\vec{x}|\vec{\alpha})$, which we do not have available, or they require more evaluations of $p(\vec{x}|\vec{\alpha})$, which we will find is very computationally expensive to evaluate. In addition, the Metropolis algorithm was used is the previous SNO analysis presented in [57], so the SNO collaboration had experience with the algorithm and had already approved its use.

## 7.3 Extended Log Likelihood

In general terms, we have an experiment that produces a set of data points $\{\vec{x}_i\}$, with each point consisting of a number of values. Assuming that these data points represent statistically independent events (interactions in our detector), we can say that these are random draws from some probability distrubtion $p(\vec{x}|\vec{\alpha})$, where $\vec{\alpha}$ is a set of parameters describing this distrubtion's shape. Since it is a probability distribution, we have $\int_{\text{all } \vec{x}} p(\vec{x}|\vec{\alpha}) = 1$ for any $\vec{\alpha}$. We don't know this distrubtion a priori (else we wouldn't need to do the experiment), but we assume that we have a good enough understanding of the system that our ignorance can be characterized by the set of parameters $\vec{\alpha}$. Our goal is now to find the $\vec{\alpha}$ that best matches our data.

As per usual, the Likelihood is defined as the product of the probabilites of drawing each of the points.

$$\mathcal{L}(\vec{\alpha}) = \prod_{i}^{n} p(\vec{x}_i|\vec{\alpha}) \tag{7.1}$$

This gives a measure of the "probability" that the data set was drawn from $p(\vec{x}|\vec{\alpha})$, in the sense that the better that $p(\vec{x}|\vec{\alpha})$ agrees with the data, the larger $\mathcal{L}$.

Unfortunately, this is too simple. First, this assumes that we are drawing from a normalized probability density function (PDF), when in reality the number of events in our data set is very important. Second, many of the elements of $\vec{\alpha}$ are constrained by external measurements. The former is dealt with by using the Extended Likelihood, which penalizes differences between the number of events expected and the number of data events, and the latter by adding terms to the likelihood.

The Extended Likelihood is a standard prescription. We now have a "PDF" $N \cdot p(\vec{x}|\vec{\alpha})$, which will have $N$ events in it - integrating across the distribution gives $N \cdot 1$. To incorporate this, we multipy the Likelihood in equation 7.1 by a Poisson distribution for the number of data events, with mean equal to the number of events $N$ in the "PDF" (which is now no longer normalized to one). We call the number of data events $n$ and we will drop the quotes around PDF from now on, with the knowledge that we are now talking about $Np(\vec{x}|\vec{\alpha})$ instead of $p(\vec{x}|\vec{\alpha})$. This gives

$$\mathrm{EL}(\vec{\alpha}) = \frac{N^n e^{-N}}{n!}\mathcal{L}(\vec{\alpha}) = \frac{N^n e^{-N}}{n!}\prod_{i}^{n} p(\vec{x}_i|\vec{\alpha}) \tag{7.2}$$

We will later take the logarithm of this for computational reasons. Quickly evaluating that logarithm gives

$$\log(\mathrm{EL}) = n\log(N) - N - \log(n!) + \sum_{i}^{n}\log(p(\vec{x}_i|\vec{\alpha})) = \sum_{i}^{n}\log(Np(\vec{x}_i|\vec{\alpha})) - N - \log(n!) \tag{7.3}$$

Note that $n$ is a constant (since the set of data points is fixed), while $N$ is not (since this is a parameter of our PDF, which we will later take to be part of $\vec{\alpha}$).

The constraint terms are external measurements for our parameters, giving independent information about the values the parameter can take. We simply multiply our $\mathcal{L}$ by the (normalized) pdf that corresponds to the measurement. This is stating that we have a prior distribution for that parameter, so that our external information tells us what we think the probability distribution for that parameter should be. In our analysis, these take one of three forms: a Gaussian, a "two sided Gaussian", and a covariance matrix.

The simplest and most common case is the Gaussian. If a parameter $\alpha_j$ has a value $\bar{\alpha}_j \pm \sigma$, then we multiply $\mathcal{L}$ by a gaussian distribution

$$p(\alpha_j) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\alpha_j - \bar{\alpha}_j)^2}{2\sigma^2}\right) \tag{7.4}$$

The "two-sided Gaussian" is the case of asymmetric uncertainties, i.e. a measured value of $\bar{\alpha}^{+\sigma_+}_{-\sigma_-}$. This corresponds to a Gaussian with one value of $\sigma$ above the mean and a different one below, so we multiply $\mathcal{L}$ by

$$p(\alpha_j) = \frac{\sqrt{2}}{\sqrt{\pi}(\sigma_+ + \sigma_-)} \begin{cases} \exp\left(-\frac{(\alpha_j - \bar{\alpha}_j)^2}{2\sigma_-^2}\right) & \text{if } \alpha_j \leq \bar{\alpha}_j \\ \exp\left(-\frac{(\alpha_j - \bar{\alpha}_j)^2}{2\sigma_+^2}\right) & \text{if } \alpha_j > \bar{\alpha}_j \end{cases} \tag{7.5}$$

A covariance matrix is somewhat more complicated, as it links the values of several parameters. It corresponds to a multi-dimensional Gaussian. Given an $m \times m$ covariance matrix $\mathbf{\Sigma}$ linking $m$ parameters (label them $\vec{\beta}$)

$$p(\vec{\beta}) = \frac{1}{(2\pi)^{m/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\vec{\beta} - \bar{\vec{\beta}})^T \mathbf{\Sigma}^{-1} (\vec{\beta} - \bar{\vec{\beta}})\right) \tag{7.6}$$

For ease of computation, and computational accuracy on a finite-precision machine, we take the logarithm of the product of the extended likelihood and the constraints. Since $\log(x)$ is a monotonically increasing function of $x$, maximizing one is equivalent to maximizing the other. We drop all terms that are constants and hence do not contribute to the maximization process to end up with the Extended Log

Likelihood:

$$\mathrm{ELL}(\vec{\alpha}) = \log(\mathrm{EL}) + \sum_{constraints} \log(p(\vec{\alpha})) = \sum_{i=0}^{n} \log(Np(\vec{x}_i|\vec{y})) - N + \sum_{constraints} \log(p(\vec{\alpha})) \tag{7.7}$$

When the constraint terms are expanded, in general the normalization coefficients, being constant, are dropped.

In sum, we now have a way of measuring the probability that our data set $\{\vec{x}\}$ was drawn from the distribution characterized by $\vec{\alpha}$, taking in to account the known constraints on $\vec{\alpha}$. How this ELL changes as we change $\vec{\alpha}$ will tell us how we want to direct our random walk through parameter space, as codified in the Metropolis algorithm.

## 7.4 Metropolis Algorithm

The Metropolis algorithm consists of a loop which walks across the parameter space of $\vec{\alpha}$. At the end of the $i^{th}$ step, it will have the parameter values $\vec{\alpha}_i$. The $(i+1)^{th}$ step starts by varying each of the parameters by adding a random draw from a Gaussian with $\mu = 0$ and $\sigma = \sigma_{\alpha_j}$, i.e. each parameter is varied around its current position with a width specific to that parameter (determined before the process starts, see Section 7.5). This creates a "proposed" step $\vec{\alpha}_{i+1}^{prop}$. The Extended Likelihood (including all constraints, so exp(EL($\vec{\alpha}$)) is then compared between the previous and the current step by taking the ratio $p(\text{step}) = \dfrac{EL(\vec{\alpha}_{i+1}^{prop})}{EL(\vec{\alpha}_i)}$. A random number is drawn from the uniform distribution between 0 and 1, call it $\text{rand}(0,1)$. If $\text{rand}(0,1) < p(\text{step})$ then we keep the proposed value and set $\vec{\alpha}_{i+1} = \vec{\alpha}_{i+1}^{prop}$, if not we don't take the step and keep our old one, so $\vec{\alpha}_{i+1} = \vec{\alpha}_i$. This latter point is important - the repetition in our data set of "better points" is a large part of what gives rise to the useful behavior of this algorithm. In more algorithmic form this is:

$$\begin{aligned}
&\vec{\alpha}_{i+1}^{prop} = \vec{\alpha}_i + \vec{g}(\vec{\sigma}_{\vec{\alpha}}) \\
&p(\text{step}) = \frac{EL(\vec{\alpha}_{i+1}^{prop})}{EL(\vec{\alpha}_i)} \\
&\textbf{if } \text{rand}(0,1) < p(\text{step}) \\
&\quad \textbf{then } \vec{\alpha}_{i+1} = \vec{\alpha}_{i+1}^{prop} \\
&\quad \textbf{else } \vec{\alpha}_{i+1} = \vec{\alpha}_i
\end{aligned}$$

The output of the algorithm is the set $\{\vec{\alpha}_i\}$. The first few steps depend on the initial starting position $\vec{\alpha}_0$, so we discard those as the "burn-in" period (discussed in Section 7.5). The rest of the points are a random draw from the posterior distribution of the parameters $p(\vec{\alpha}|\{\vec{x}_i\})$ - the probability distribution for our parameters given the data points. This is exactly what any fitting algorithm (or experiment, for that matter) sets out to find. Since we have a collection of random draws from this distribution, we bin them to create an approximate PDF, or we assume that they will be Gaussian (or two-sided Gaussian) near the maximum and do an unbinned fit, to find out what values of parameters best correspond to our data. An example fit, from the final analysis, is shown in Figure 7-1. Since this is directly the probability distribution, its mean is the parameter value and its standard deviation ($\sigma$) is the parameter uncertainty. With many parameters, if we do this for a single parameter (so we only fit or histogram one element of the vector), we get the distribution integrating across all other parameters. If we wish to find the covaraince between two parameters, we merely histogram them against each other, fit them as a 2D fit, or directly compute the covariance of the two sets of values.

## 7.5 Step Size Finding

In the MCMC process, at each step each parameter is varied by adding a random draw from a Gaussian, with a different width for each parameter. These widths don't affect the theoretical behavior of the chain, in the sense that any choice of widths will

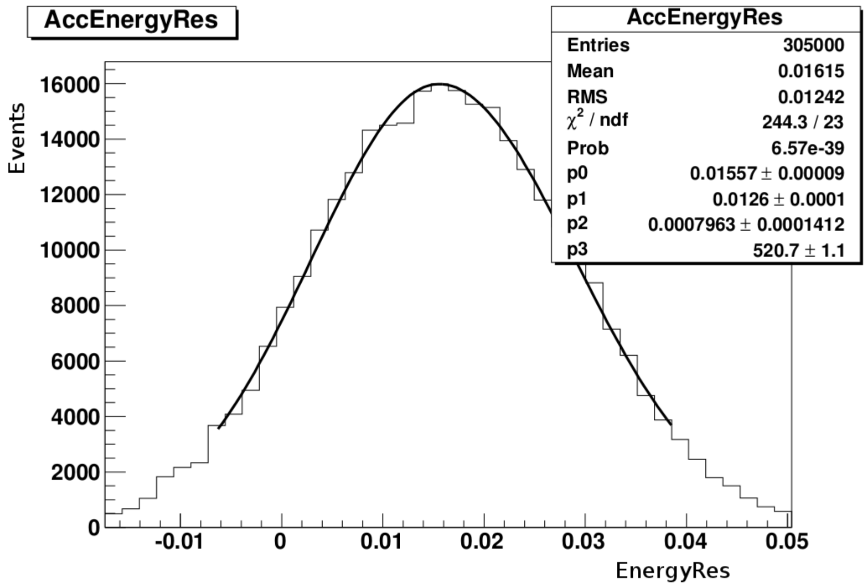


Figure 7-1: An example posterior distribution, with an asymmetric Gaussian fit. This is the energy resolution systematic from the final fit.

eventually produce a random sampling of $p(\vec{\alpha}|\{\vec{x}\})$. They do, however, have a very large impact on the practical behavior of the MCMC.

This can be illustrated by thinking about a case with one parameter, for example one with the Likelihood versus parameter value shown in Figure 7-2. Suppose we start our chain at the point marked with an arrow. The dimension marked $\sigma$ is a reasonably good width for our Gaussian. Since the Likelihood is steep there, we are much more likely to take a step "uphill" and will reach the maximum in a few steps. More importantly, the region containing most of the probability is only approximately ten $\sigma$'s wide. Once a parameter value near the maximum is reached, the algorithm should have a good probability of stepping back "downhill" in one direction or the other, and thus exploring the space around the maximum, as a step of size $\sigma$ will not change the likelihood value too dramatically. If the width were a tenth of $\sigma$, however, we would have two problems: first, it would take many steps to get to the maximum (this corresponds to a longer burn-in period) and, once at the maximum, it would take a very large number of steps to explore the majority of the probability. If the

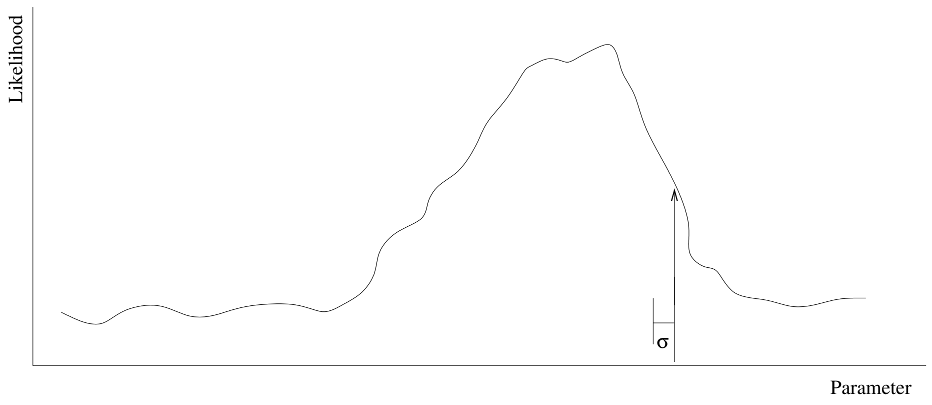


Figure 7-2: An example of a possible Likelihood versus parameter value graph, in the case of a system with a single parameter. See text for details.

width were ten times $\sigma$, the opposite problem would occur: it would be very likely that the step would take us away from the maximum completely, and with a very low likelihood away from the maximum, that step would be rejected by the algorithm. We seek to balance between these two cases.

Looking at Figure 7-2, we can immediately see that the starting location of our chain plays a role. Thankfully, it can be proven that this role is transient, that is that regardless of the starting point the chain will settle in to a proper sampling $p(\vec{\alpha}|\{\vec{x}\})$ [74]. We need to discard this initial transient, called the burn-in period, as it is not part of the sample we want. The characteristic behavior of this burn-in period is a rapidly changing log-likelihood, as the chain wanders through low-probability regions where it is very probable for it to take steps "uphill". Once the chain reaches a high probability region, the stable sampling behavior will set in; this is called a stationary chain or stationarity. This gives rise to graphs of log-likelihood versus step such as Figure 7-3. The end of the burn-in period is fairly obvious on the graph, occuring around step 200, when the changes become the size of the long-term fluctuations. We customarily add 50% or 100% to the size of the burn-in period to be safe, as occasionally a parameter will not have settled in to stationarity at that point, if it has a very small impact on the likelihood. Ideally, we should check each parameter individually, but this is not practical when we are dealing with hundreds of runs each containing dozens of parameters. Instead, we choose a few sample runs and check to

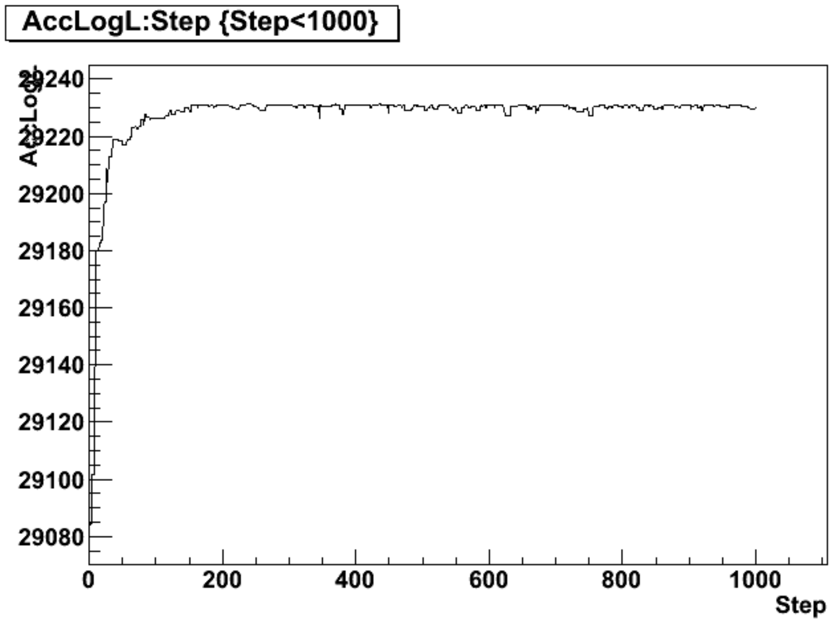


Figure 7-3: Sample log likelihood v. step plot showing burn-in

make sure the safety margin is sufficient. We can, of course, force the burn-in period to be shorter by moving the initial point closer to the maximum likelihood point (as with any fitter).

Before discussing how to optimize these step sizes, we need a goodness measurement for our step sizes. We seek to avoid two problems: too large a step, which results in few steps taken, and too small a step, which results in the space not being sampled properly. The former is simple to measure, we just measure how often the algorithm takes a step. Denote that "% steps". The rule of thumb is that it should be roughly 23.4% for a system with multiple parameters [72, 76], though the precision of this number belies the range of acceptable values - any % step between 20% and 30% is quite good, and something between 15% and 35% is acceptable. The latter problem is less obvious, but is well measured by the autocorrelation. This measures how many steps the algorithm takes before values are no longer correlated with earlier values, in colloquial terms how long before the chain "forgets" where it was before. The smaller this is, the better we are sampling the space, so it measures how independent our samples of $p(\vec{\alpha}|\{\vec{x}\})$ are. The autocorrelation is not an indepedent measure from % steps, as when very few steps are taken, the chain "remembers" its location for a long time. In fact, [72, 74] point out that the autocorrelation is the correct measure of what we desire in a chain, while the % step is just a useful rule of thumb. We note, however, that letting % step get too small, even if it gives a better autocorrelation, can unacceptably increase the burn-in period. The autocorrelation is defined as

$$\rho(h) = \frac{\sum_t [(x_t - \bar{x})(x_{t+h} - \bar{x})]}{\sqrt{\sum_t (x_t - \bar{x})^2 \cdot \sum_t (x_{t+h} - \bar{x})^2}} \tag{7.8}$$

where $x_i$ is an element in a sequence and $h$ is called the "delay". $\rho(h)$ is equal to the traditional definition of correlation between two sequences for the sequence $x_i$ and $x_{i+h}$. Note that this is only a useful measure once the burn-in period has been removed, as we wish to measure the stationary sampling behavior of the chain. The autocorrelation should decay exponentially, with a fluctuating background level due to the finite number of points available. A sample autocorrelation is shown in Figure 7-4, which includes an exponential fit of the form

$$\rho(h) = A \exp(-\frac{h}{\tau})$$

The exponential's decay-time parameter $\tau$ gives us a single, real number measure of the autocorrelation. The smaller this $\tau$, the faster the chain forgets its previous position, giving "more independent" random draws from $p(\vec{\alpha}|\{\vec{x}\})$.

We now seek to minimize $\tau$ for all parameters in our chain simultaneously while keeping the % step in an acceptable range. A way of automating this process is still an active area of research, called Adaptive MCMC (see, for example, [76]). Having found no accepted method for finding the $\sigma_{\alpha_j}$'s, through trial and error we settled on a methodology that, while a bit slow and labor intensive, produces good results. This method attempts to make best use of our parallel computing resources by running many chains with different step sizes in parallel at each step. We will refer to each instance of MCMC running in parallel as a "run". Each run has a fixed step size for each parameter.

We start by running a large sweep over possible step sizes. Since we are assuming

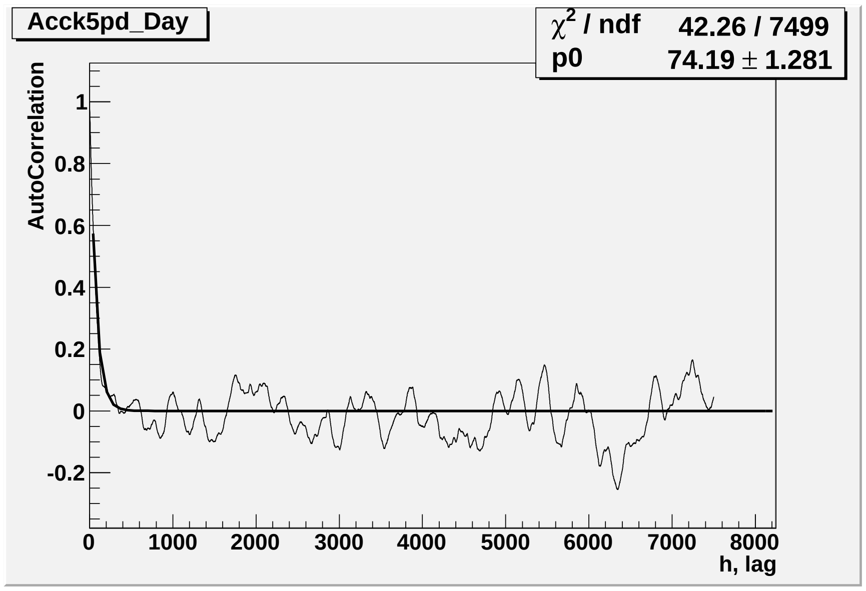


Figure 7-4: Sample autocorrelation with exponential fit

to start with very little knowledge of the correct step size, we will look over several orders of magnitude and sample logarithmically, so that we get approximately the same number of samples in each decade. First we select a range to sample over for each parameter. In the case of a parameter with an external constraint, this is straightfoward - the step size will be less than the width of the constraint, but could be much smaller if the data more sharply constrains it. We will denote the constraint width as $\delta$. If we know nothing else about the system, we take a range $[10^{-4}\delta, \delta]$. We restrict this range if we have more information, such as a similar system of parameters to compare against. In the case of a parameter with no constraint, we need to guess. If we have an idea of the ranges of values that we expect the parameter to take, we use these in place of $\delta$, possibly extending the range to $[10^{-6}\delta, 10\,\delta]$. Otherwise, we simply run as broad of a sweep as we can, starting with a range of $[10^{-8}, 10^{2}]$ when we run 100 runs in parallel (to keep about 10 runs per decade). We may find we are still outside the range we want, in which case we repeat with a different range. We then randomly sample within this range for each parameter, to generate a number of runs with different step sizes. We randomly choose instead of picking a progression as the large number of parameters (over 50 in the main analysis) would require a unacceptably large number of runs. We then run these in parallel, compute the autocorrelation for each parameter and % step for each run, and generate a plot of $\tau$ versus step size and % step versus step size for each parameter. The burn-in period can be a problem here, as it will vary for various step sizes. We attempt to control for this by starting the chains as close to the maximum as we can. Since we do this step size finding process on simulated data to avoid bias and blindness problems, we know the actual maximum and can start arbitrarily close to it. In addition, those runs with too small a step size to find the maximum will return very bad autocorrelations, which we need to keep in mind when looking at results.

A sample $\tau$ plot is shown in Figure 7-5. The key characteristics to note are that $\tau$ tends to decrease as the step size increases. Figure 7-6 shows a sample % step versus step size plot. Here we see that as step size increases, % step decreases. This is the fundamental trade-off in the step size finding process: increasing the step size of a parameter decreases that parameter's autocorrelation, but decreases the % step, which increases the autocorrelation of all parameters. We have found that the "corner" in the $\tau$ graph, where $\tau$ stops decreasing as quickly (about 0.4 in Figure 7-5, though we note that the $\tau$ v. step size behavior is actually power-law like) is approximately the best value. We then find this corner for each parameter and record it; we will denote this value $\varsigma$. If the corner is not present, we have not looked over a sufficiently large range of step sizes so we repeat this step with altered ranges.

With the $\varsigma$'s in hand, we know approximately the correct values. Unfortunately, the $\tau$'s are correlated, both through the % step change and through the MCMC process itself. Thus we must search again, though on a more restricted range. We again create runs with logarithmically randomly selected step sizes for each parameter, this time from the range $\frac{1}{3}\varsigma, 3\varsigma$ (this is, of course, arbitrary). We again run these in parallel and compute the autocorrelations. Instead of graphing the results, we search though them for those with desirable characteristics. We search for two types of

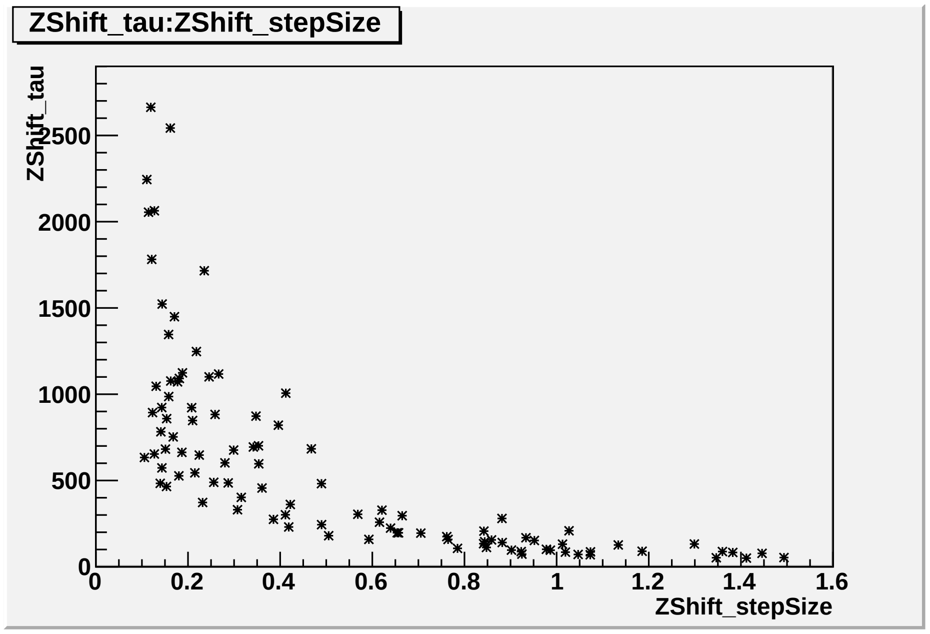


Figure 7-5: Sample autocorrelation $\tau$ v. step size

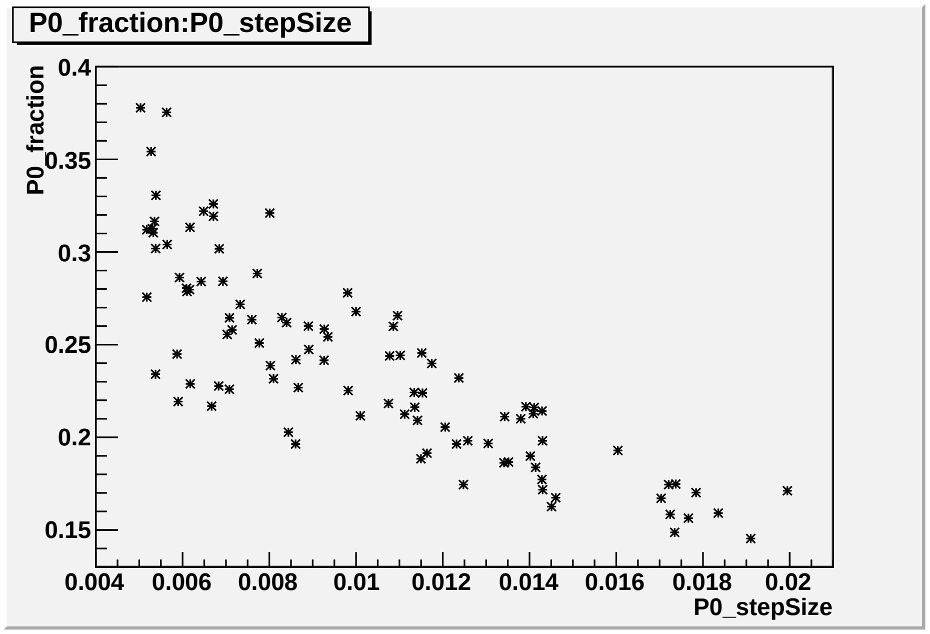


Figure 7-6: Sample % step v. step size

runs: those with the smallest $\tau$'s (in particular ones with all $\tau$'s less than a particular value, which we lower until only a few runs qualify) and those with the most uniform $\tau$ values. Ideally, we'll find a run with all $\tau$'s similar and small, in the case of the final experimental run this was a $\tau$ of approximately 200. If we do, and this has a reasonable % step, we have found our step sizes and we stop. If not, we look at the best runs and see what needs to change.

If the runs are close to what we need, we adjust by hand and test. In the case of one or two parameters with too large $\tau$'s, we increase the step sizes of those problem parameters and decrease the step sizes of a few parameters with much lower $\tau$'s to compensate (unless the % step is too large, in which case we only increase). If all of the $\tau$'s are similar but too large, we look at the % step. If it is too large, we increase all step sizes, if too small we reduce all step sizes. We must be careful here - if all of the $\tau$'s are much too large, this can be the result of the autocorrelation not fitting well to an exponential, for example Figure 7-7. This usually arises from a run that hasn't actually passed through the burn-in period, though it can also result from a run with a $\tau$ so large that we simply did not run long enough to properly measure it. We reject those runs, as they tell us nothing. Either way, this gives a new set of values, $\tilde{\varsigma}$. We now again randomly sample, this time of the linear range $[\frac{1}{2}\tilde{\varsigma}, \frac{3}{2}\tilde{\varsigma}]$, but include an additional run with all parameter step sizes at $\tilde{\varsigma}$. We then repeat the search described in this paragraph, adjusting as necessary and iterating. This usually only takes one or two iterations, but occasionally can become problematic.

In an attempt to automate and streamline this process, we implemented the Adaptive MCMC described in [76]. This is a variant on the Metropolis algorithm which adjusts the step sizes at each step, in an ever decreasing way. This adjusts the MCMC

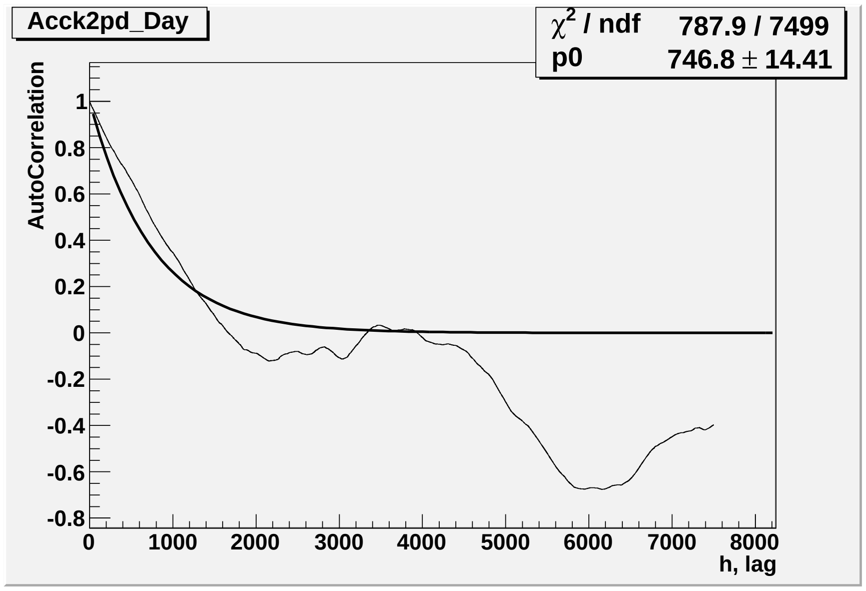


Figure 7-7: Sample autocorrelation plot where exponential fit failed

algorithm to

$$\vec{\alpha}_{i+1}^{prop} = \vec{\alpha}_i + \vec{g}(\vec{\sigma}_{\vec{\alpha},i})$$

$$p(\text{step}) = \frac{EL(\alpha, \vec{i+1}^{prop})}{EL(\vec{\alpha}_i)}$$

$$\textbf{if } \text{rand}(0,1) < p(\text{step})$$

$$\quad \textbf{then } \vec{\alpha}_{i+1} = \vec{\alpha}_{i+1}^{prop}$$

$$\quad \textbf{else } \vec{\alpha}_{i+1} = \vec{\alpha}_i$$

$$\vec{\sigma}_{\vec{\alpha},i+1} = \vec{\sigma}_{\vec{\alpha},i} + \vec{\sigma}_{\vec{\alpha},0}(p(\text{step}) - 0.234)i^{-\beta}$$

where $\beta$ is a real number controlling how quickly the adjustments reduce in size, usually set somewhere between 0.5 and 1. This is a very simple change: we now have a variable step size, which we adjust at every step, decreasing if the probability to take a step was smaller than 23.4% and increasing if it was larger. This serves to force the algorithm to take % step = 23.4%. This algorithm, as presented in the paper, has an unacceptable drawback: it changes all the step sizes together. This means that their relative proportions never change. This was the case for the example in the paper, but is not the case for us. We do not know beforehand what the ratios should be, since some parameters affect the likelihood more than others. To correct this, we adjust this algorithm to only alter one parameter at a time. The algorithm behaves identically, save that the last line now reads

$$(\vec{\sigma}_{\vec{\alpha},i+1})_j = (\vec{\sigma}_{\vec{\alpha},i})_j + (\vec{\sigma}_{\vec{\alpha},0})_j(p(\text{step}) - 0.234)i^{-\beta}$$

and j is incremented at each step, modulo the number of parameters, so it walks through the parameter list from top to bottom repeatedly.

When we initially did this, we found that the first parameter was being altered too much. The size of the changes in $\vec{\sigma}_{\vec{\alpha},i}$ is falling like $i^{-\beta}$, where we used $\beta = 0.66$, which is 1 for the first step, and approximately 0.08 by the end of the first pass through our roughly 50 parameter long list. To correct for this, we instead keep the size fixed for the first pass through the list: we have the first $m$ steps (where $m$ is the number of parameters) replace $i^{-\beta}$ with $m^{-\beta}$.

This altered algorithm does amazingly well at causing the chain to settle in to a set of step sizes where % step is 23.4%. Unfortunately, there are many such sets of step sizes - this is a single (albeit complicated) constraint in a high-dimensional space of possible step sizes. This adaptive algorithm did not produce useful results, as it always found a region where some of the autocorrelations were very small and some were very large. We wound up discarding it after many trials and reverting to the original Metropolis algorithm with the step size finding procedure outlined above.

## 7.6 Benefits and drawbacks

The most obvious strengths to this method are computational. In particular, it is an "embarrasingly parallel" algorithm - to run it across multiple computers, we run an instance of the algorithm on each machine. They do not need to communicate. We then simply take each machine's output for $\{\vec{\alpha}\}$ (which will be different, as this is an inherently random process) and concatenate them in to a larger set. This is in stark contrast to a deterministic Minuit-style minimizer, which is extremely difficult to parallelize. The only catch to this is the burn-in period. Each computer will individually go through this process, so the first few events of each computer's output chain must be removed before concatenation. Even with that being true, this easily allows much longer chains to be run. For instance, in this analysis we had access to a cluster of approximately 1500 nodes, called Tier2, of which we could reasonably expect to use 100 or so at a given time. In the final analysis, we could run approximately 7000 algorithm steps in a 24 hour period. With a burn-in period of 2000 steps, we were able to use a chain of 305000 events in a 24-hour run (not all 100 jobs ran due to computer cluster problems), more than sufficient for good results.

In addition, the MCMC algorithm deals reasonably well with large numbers of parameters. The additional cost of an additional parameter is very small, one random draw from a Gaussian per step. The real additional costs come from two places: adding an additional parameter usually implies adding complexity to $p(\vec{x}|\vec{\alpha})$, which can be expensive, and more parameters generally require a longer burn-in period and chain length. The longer burn-in period is from adding a dimension to our parameter space, which slows the random walk process. The chain length increases because of the step size problem: as more parameters are added, the probability of taking a step goes down (assuming the additional parameters have a reasonably strong influence on the ELL), requiring smaller step sizes.

Another, less obvious benefit is that the MCMC deals well with systems where the likelihood is very complicated. If the likelihood is not smooth near the maximum, as is the case for our SigEx, a typical Minuit-type minimizer can easily get caught in a local maximum. In addition, since these fitters assume Gaussian-like behavior near the minimum (corresponding to our maximum), they can get incorrect uncertainties. The MCMC, however, simply maps out the likelihood space and is immune to these problems. They will, instead, show up as non-smoothness is $p(\vec{\alpha}|\{\vec{x}\})$, which is appropriate. How to interpret this is a separate issue. We found all of our parameters to have posterior distributions sufficiently close to a two-sided Gaussian near the maximum that we were comfortable fitting that function to them and report a value $\alpha_j\,{}^{+\sigma_+}_{-\sigma_-}$ for each.

From a statistical standpoint, a major benefit is that we get a random sample of $p(\vec{\alpha}|\{\vec{x}\})$. We can now compute any quantity we like, such as the correlation between any pair of variables, by simply computing that quantity on those two elements of $\{\vec{\alpha}\}$. In addition, when we histogram a single parameter's value, we integrate across all other parameters. This is in a rather literal sense: when we do this, we get

$$p(\alpha_j|\{\vec{x}\}) = \int \cdots \int p(\vec{\alpha}|\{\vec{x}\})\mathrm{d}\alpha_1 \ldots \mathrm{d}\alpha_{j-1}\mathrm{d}\alpha_{j+1} \ldots \mathrm{d}\alpha_m \tag{7.9}$$

This is the correct way to treat nuissance parameters such as systematics, as it correctly folds their contributions to the likelihood in to the width of the distribution of the parameter, including any correlations. This then avoids complications associated with correlated parameters, though if we want to give the point of maximum likelihood rather than the most probable value for each parameter, we need to fit a multi-dimensional function to the joint distributions of those parameters we are concerned about.

The MCMC is not without drawbacks. The most glaring is the issue of finding step sizes. This adds greatly to the complexity of running an MCMC, though we hope that researchers will one day perfect an adaptive MCMC that does not have this problem. Another problem related to its complexity is that, as a random process, it tends to be slower than a deterministic fitter on simpler systems. In addition, it is not always clear when the burn-in period has passed, as the algorithm does occasionally fall in to local minima. It will wander out of them eventually, but that can be much longer than is reasonable to wait. This is unusual, thankfully, and can be tested by either starting many parallel chains with different initial conditions or by running one very long chain. The last drawback we will mention is that since this algorithm is less well understood than some other fitters, interpreting the results is more difficult. It is impractical to report the full $p(\vec{\alpha}|\{\vec{x}\})$, except possibly for a few parameters. In addition, the expectation is that a single number plus an uncertainty will be reported.

This is fine when $p(\alpha_j|\{\vec{x}\})$ is well approximated by a Gaussian or two-sided Gaussian near the maximum, but that is not always the case.

# Chapter 8

# Signal Extraction

Thus far, we have looked at all the pieces that make up the analysis presented in this thesis, the three phase Day/Night signal extraction. Now we put all of these pieces together to actually perform the analysis. We look at the underlying methodology, the computer code used to implement it, and the tests done to ensure it was working as expected.

We begin by summarizing the method: The analysis starts with a set of Monte Carlos of the SNO detector, simulating the principle fluxes (CC, ES, $\mathrm{ES}_{\mu\tau}$ and NC) and backgrounds. These are summed together, varying the relative amounts of each and applying a set of distortions representing possible inaccuracies and uncertainties in the simulation (the systematics). The parameters describing the relative amounts and distortions are collectively refered to as $\vec{\alpha}$. This summed, distorted MC is compared to the data, giving a log likelihood. We then vary this $\vec{\alpha}$ via a Markov Chain Monte Carlo method (the Metropolis algorithm) to extract the probability distributions for values of our parameters given our data - this is the posterior distribution $p(\vec{\alpha}|\{\vec{x}\})$ described in Chapter 7.

To understand how this process is carried out, we first explain how $N\,p(\vec{x}|\vec{\alpha})$ is calculated. We then explain how the results of the two external analyses LETA and PSA are included. We then explain the implementation itself and how our computer code is structured. Finally, we show the results of our tests to confirm the code is working as expected.

## 8.1 PDF

Throughout our discussion of the log likelihood, we left $N\,p(\vec{x}|\vec{\alpha})$ abstract. To actually build a functional MCMC, we must specify how it is evaluated. There are many ways to do this in principle, though the complexity of any modern experiment leaves us with basically one choice: Monte Carlo. A detailed MC of our experiment gives us a set of MC simulated events $\{\vec{y}\}$. Each of these events will consist of a set of values containing those in the data, $\{\vec{x}\}$, plus additional information from the MC not available in the data, such as the "true" values that went in to the generation of the event (as opposed to the measured values). As described in Chapter 4, the analysis only uses three data variables, $E$, $\rho^3$ and $\cos(\theta_{sun})$, while the Monte Carlo uses $E$, $E_{true}$, $x$, $y$, $z$, $x_{true}$, $y_{true}$, $z_{true}$, $\cos(\theta_{sun})$, and $E_\nu$.[1]

To turn this set of points in to a PDF, we histogram $\{\vec{y}\}$ and interpret the value in each bin as $p(\vec{x}|\vec{\alpha})$ for any point in that bin. This potentially introduces a systematic uncertainty - we are effectively integrating the "true" $p(\vec{x}|\vec{\alpha})$ in each bin and only looking at the average. If the underlying $p(\vec{x}|\vec{\alpha})$ varies quickly, we may wash out features that we are interested in and lose discriminating power. This can be tested by looking at the one dimensional projections of the MC in each of the principle variables ($E$, $\rho^3$ and $\cos(\theta_{sun})$), reducing the bin size significantly and looking for any regions of rapid change. The one dimension projections are used only to increase statistics and allow finer bins. This was done for the NCD phase analysis presented in [57], though not discussed there. Additionally, we repeated this simple test and saw no such regions that would be averaged out by the binning process.

We define a set of bins in each of the data values (a 3-D histogram here) and fill with the MC events. As explained in Section 4.3, we restrict to a particular range in each value, so we only create histogram bins over that range. Many of our MC events are outside these ranges and thus do not appear in the histogram at this stage, but we retain them as the systematics may alter their values enough to move them inside our cuts. In this analysis, we used the following bins: 10 equal bins in $\rho^3$ from 0 to

[1] $\rho^3$ is then computed from $x$, $y$ and $z$

0.77025, 25 equal bins in $\cos(\theta_{sun})$ from $-1$ to 1 and a set of bins in $E$ consisting of 0.5 MeV spacing between 6 and 13 Mev, a larger bin from 13 to 20 MeV (13 total bins). Since the Pdf is three dimensional, this it has $10 \cdot 25 \cdot 13 = 3250$ bins. These are the same bins used in the NCD phase analysis [57].

Several different physical processes, our primary fluxes, contribute to our signal. Each of these has a separate set of MC events. We also expect to have several backgrounds producing events, again each with a separate set of MC events. These backgrounds are treated identically to the fluxes. We take each set of MC events separately, apply the relevant subset of the systematics to each event in that set, and fill the resulting "new" event values in to the histogram. Each event is filled with a weight determined both by the weight systematics and a conversion factor from the number of MC events to the expected number of data events. In this manner, we are able to vary the relative proportions of each flux in the resulting histogram, as well as the total number of events $N$. This makes $N$ dependent on both the intensities of each flux as well as the systematics.

This leads us to our construction method for $N\,p(\vec{x}|\vec{\alpha})$. We have $m$ sets of MC, one for each flux (or background), and a set of parameters $\vec{\alpha}$. This $\vec{\alpha}$ contains values for our systematic parameters and parameters defining the size of each flux. The latter took several related forms over the course of testing the code, but for sake of exposition we'll use the final form: each flux has a parameter with nominal value 1 that represents the "scale", i.e. how much it deviates from expectation (with expectation normalized to 1). This necessitates a fixed conversion factor from MC events to data events representing how many "experiments worth" of MC are in the set, called `TimesExpected`, defined as the number of events expected for this flux (in the data) divided by the number of MC events in the analysis window. The systematics can change the number of events, but this ratio assumes systematics at their nominal values. Since not all systematics are applied to each flux, each flux is assigned a `FluxNumber` and each systematic knows which `FluxNumber`'s it acts on. The process is then to go through the fluxes one at a time, taking each individual event, computing the value changes for the relevant systematics and filling the histogram with the

| Flux | MC events | Data events | `TimesExpected` |
|---|---|---|---|
| CC | 6198.589 | 5628150 | 907.9728 |
| NC | 266.051 | 244751 | 919.9367 |
| ES | 532.611 | 970822 | 1822.762 |
| $\text{ES}_{\mu\tau}$ | 82.7599 | 1508525 | 18227.725 |
| ex | 20.60133 | 119417 | 5796.528 |
| d2opd | 8.2987 | 244751 | 29492.692 |
| ncdpd | 5.7265 | 70175 | 12254.431 |
| k2pd | 9.36112 | 112710 | 12040.226 |
| k5pd | 11.557 | 118464 | 10250.411 |
| atmos | 24.66224 | 244751 | 9924.119 |

Table 8.1: The total number of MC events in the analysis window, expected number of data events and `TimesExpected` for each flux and background. `TimesExpected` can be computed by (expected data events)/(MC events). Note that the expected data events are for the PMTs, the amounts in Table 4.2 are for the NCDs.

resulting event values, with weight given by

$$W_{final} = \frac{\text{Scale}}{\text{TimesExpected}} \prod_k W_{sys,k} \tag{8.1}$$

where the survival probability from Section 4.5 is treated as a systematic. The number of MC events, expected number of data events and `TimesExpected` for each flux and background are shown in Table 8.1.

The resulting histogram is *almost* $N\,p(\vec{x}|\vec{\alpha})$. If the bin volumes in our histogram are all equal, then it is and we stop. However, if they are unequal, we must compensate for that. The problem arises in that we are interpreting this as an (unnormalized) PDF. A simple example to illustrate is the case of a constant probability over some region, say $[0, 1]$. A sequence of 1000 random draws from this distribution (as our MC) will be uniformly distributed across $[0, 1]$, as we expect. Suppose our histogram has two bins. If they are $[0, 0.5)$ and $[0.5, 1]$, then there is no problem - each has (on average) 500 events. If the two bins are $[0, 0.3)$ and $[0.3, 1]$, however, this doesn't work - one has 300 events and the other 700. If we then look at these as our $Np(x)$, we see that this will incorrectly over-weight the larger bin. To correct for this, we divide each bin by its width. This results in each bin having a "size" (it is no longer simply counts) of 1000.0, again equal. This naturally raises the question of whether we should then re-scale the histogram to get back to some semblance of counts. Interestingly, this actually does that for us - we have $\int(Np(x))\,\mathrm{d}x = N$, in this case 1000. Of course, the next question is: does not dividing by the bin width in the uniform case cause a problem? Since we are using the log likelihood, the answer is no. If we have a constant scaling factor $\gamma$ applied to all bins, this is equivalent to having $N\gamma p(\vec{x}|\vec{\alpha})$. Ignoring constraint terms (which are unaffected), using this in our ELL gives

$$\mathrm{ELL}(\vec{\alpha})_\gamma = \sum_i^n \log(N\gamma p(\vec{x}_i|\vec{\alpha})) = \sum_i^n (\log(Np(\vec{x}_i|\vec{\alpha})) + \log(\gamma)) = \mathrm{ELL}(\vec{\alpha}) + n\log(\gamma) \tag{8.2}$$

Since $\gamma$ is fixed during the maximization process (it is a property of the binning, which doesn't change), it has no effect and we can neglect it.

## 8.2 LETA Constraint

As has been mentioned, this signal extraction sets out to incorporate both the NCD phase data and LETA in a "three phase" analysis using all of SNO's data. In theory this could be done by creating a very large signal extraction that fits $P_{ee}$ and $A_{P_{ee}}$ from the NCD phase and LETA simultaneously, reproducing the results of [51] in the process. This is decidely impractical. In [77], a better method was outlined. The key insight was that very few of the systematics and backgrounds are correlated between the NCD phase and LETA, thanks to the large number of independent calibrations done and the changes in the detector. This vastly reduces the amount of information from LETA needed for the NCD phase - we need only the likelihood function for those correlated values.

As explained in [77], we can look at the problem as follows. Denote our signal parameters ($^8$B flux, $P_{ee}$, $A_{P_{ee}}$) by $\vec{\phi}$, the systematics and backgrounds correlated between NCD and LETA by $\vec{\beta}$, those specific to LETA as $\vec{\gamma}$ and those specific to NCD as $\vec{\eta}$. We can write our log likelihood (equivalent to our ELL($\vec{\alpha}$)) as

$$\ln\mathcal{L}(\vec{\phi},\vec{\beta},\vec{\gamma},\vec{\eta}) = \ln\mathcal{L}_{LETA}(\vec{\phi},\vec{\beta},\vec{\gamma}) + \ln\mathcal{L}_{NCD}(\vec{\phi},\vec{\beta},\vec{\eta}) \tag{8.3}$$

where we recognize that our $\vec{\alpha}$ from before is the combination of $\vec{\phi}$, $\vec{\beta}$ and $\vec{\eta}$, and $\ln \mathcal{L}_{NCD}$ as our ELL($\vec{\alpha}$). But we aren't interested in the values of the "nuisance" parameters $\vec{\beta}$, $\vec{\gamma}$, $\vec{\eta}$, so we want to marginalize over them, i.e. we wish to integrate over these parameters so that we can ignore them. Ideally, we would compute

$$\ln \mathcal{L}(\vec{\phi}) = \iiint \ln \mathcal{L}(\vec{\phi}, \vec{\beta}, \vec{\gamma}, \vec{\eta})\, \mathrm{d}\vec{\beta}\, \mathrm{d}\vec{\gamma}\, \mathrm{d}\vec{\eta} \tag{8.4}$$

However, we do not yet know $\vec{\eta}$ or $\vec{\beta}$, since they involve information from the NCD phase. So instead we compute

$$\ln \mathcal{L}(\vec{\phi}, \vec{\beta}, \vec{\eta}) = \int \ln \mathcal{L}(\vec{\phi}, \vec{\beta}, \vec{\gamma}, \vec{\eta})\, \mathrm{d}\vec{\gamma} = \int \ln \mathcal{L}_{LETA}(\vec{\phi}, \vec{\beta}, \vec{\gamma})\, \mathrm{d}\vec{\gamma} + \ln \mathcal{L}_{NCD}(\vec{\phi}, \vec{\beta}, \vec{\eta}) \tag{8.5}$$

Something very nice has happened - the full, three phase $\ln \mathcal{L}$ is simply the sum of $\ln \mathcal{L}_{NCD}$, which we have been computing all along, and $\int \ln \mathcal{L}_{LETA}(\vec{\phi}, \vec{\beta}, \vec{\gamma})\, \mathrm{d}\vec{\gamma}$ - the LETA likelihood with uncorrelated parameters marginalized, i.e their analysis results. We make the approximation that the maximum of their likelihood, $\ln \mathcal{L}_{LETA}$, is a high-dimensional Gaussian, so we can describe it with a covariance matrix (and a mean). If $\vec{\phi}$ and $\vec{\beta}$ have small numbers of elements, this matrix will be manageable. We can now treat our NCD phase analysis as a full three phase analysis, simply by adding an extra term to our likelihood.

A slight complication comes in that the LETA group reports a correlation rather than a covariance matrix. This is simple to convert, however. If we call our set of parameters $\vec{\mu}$, then given a mean $\bar{\vec{\mu}}$, a set of uncertainties $\vec{\sigma}$ and a correlation matrix $\rho$, the covariance matrix is

$$\mathbf{\Sigma}_{ij} = \rho_{ij} \sigma_i \sigma_j \tag{8.6}$$

where there is not an implied sum. Once we have the covariance matrix, the log likelihood contribution is the logarithm of the corresponding Gaussian probability distribution, giving

$$\ln \mathcal{L}_{LETA} = -\frac{1}{2} \left(\vec{\mu} - \bar{\vec{\mu}}\right)^T \mathbf{\Sigma}^{-1} \left(\vec{\mu} - \bar{\vec{\mu}}\right) \tag{8.7}$$

| Parameter | Value | Uncertainty |
|---|---|---|
| $^8$B Scale | 0.931651 | 0.0380756 |
| $P_0$ | 0.319267 | 0.0217643 |
| $P_1$ | 0.00737952 | 0.0093339 |
| $P_2$ | -0.000617788 | 0.00363615 |
| $A_0$ | 0.02664 | 0.0425076 |
| $A_1$ | -0.0220252 | 0.0318916 |
| $a^E_{0,c}$ | -0.00162829 | 0.0030292 |
| $c^E_0$ | 0 | 0.0069 |
| Winter | 0 | 1 |

Table 8.2: LETA best fit values and uncertainties. The uncertainties here are the statistical and systematic uncertainties added in quadrature. See text for details. Note that the values are presented at the precision input to the program; the excessive number of digits in the values and uncertainties is intentional.

| | $^8$B Scale | $P_0$ | $P_1$ | $P_2$ | $A_0$ | $A_1$ | $a^E_{0,c}$ | $c^E_0$ | Winter |
|---|---|---|---|---|---|---|---|---|---|
| $^8$B Scale | 1.000 | -0.704 | 0.292 | -0.120 | 0.031 | -0.013 | -0.138 | -0.029 | 0.095 |
| $P_0$ | -0.704 | 1.000 | -0.318 | -0.422 | -0.386 | 0.114 | 0.066 | 0.202 | 0.008 |
| $P_1$ | 0.292 | -0.318 | 1.000 | -0.061 | 0.169 | -0.686 | -0.443 | -0.212 | 0.053 |
| $P_2$ | -0.120 | -0.422 | -0.061 | 1.000 | 0.016 | -0.061 | -0.066 | -0.334 | 0.005 |
| $A_0$ | 0.031 | -0.386 | 0.169 | 0.016 | 1.000 | -0.231 | -0.012 | -0.012 | 0.000 |
| $A_1$ | -0.013 | 0.114 | -0.686 | -0.061 | -0.231 | 1.000 | 0.010 | 0.025 | 0.000 |
| $a^E_{0,c}$ | -0.138 | 0.066 | -0.443 | -0.066 | -0.012 | 0.010 | 1.000 | 0.000 | 0.000 |
| $c^E_0$ | -0.029 | 0.202 | -0.212 | -0.334 | -0.012 | 0.025 | 0.000 | 1.000 | 0.000 |
| Winter | 0.095 | 0.008 | 0.053 | 0.005 | 0.000 | 0.000 | 0.000 | 0.000 | 1.000 |

Table 8.3: The correlation matrix from LETA. The entries are actually double precision, but truncated to three digits for readability. See text for details.

just as in the case for a constraint term with a covariance matrix, as this is the same situation.

SNO determined that the set of correlated parameter was quite small, consisting of only the signal parameters ($P_0$, $P_1$, $P_2$, $A_0$ and $A_1$) and three systematics (the correlated energy scale $a^E_{0,c}$, the energy nonlinearity $c^E_0$ and the Winter spectral shape uncertainty). For the full data set, the LETA group reports the mean and uncertainties shown in Table 8.2 and a correlation matrix shown in Table 8.3.

## 8.3 PSA

The process for computing ELL($\vec{\alpha}$) so far has only included information from the PMTs. In the NCD phase analysis [57], the NCDs were included with a likelihood computation directly analogous to that described here for the PMTs. The energy for each NCD event, $E_{ADC}$, was used as the data $\{x\}$ in a one dimensional Pdf, compared against a set of MC events for both the expected neutron spectrum and a set of backgrounds. These backgrounds contained both those neutron backgrounds described in Section 4.6 and NCD-specific backgrounds. Since only the event energy was used and neutrons have thermalized before they capture in the NCDs, the background neutrons are indistinguishable from signal neutrons. The NCD-specific backgrounds are dominated by alpha particles emitted from radioactive decays in the nickel bodies of the NCDs, but also include some instrumental events, just called “other” backgrounds.

This solution was considered less than satisfactory, as it only used the energy information from the NCDs and ignored the full digitized pulse also recorded. Additionally, it combined all of the NCD string events together (except for six strings completely excluded from the data set due to instrumental problems), but it was known that the alpha and “other” backgrounds varied from string to string. To remedy this, the Pulse Shape Analysis (PSA) analysis was created. A much more detailed simulation of the NCDs was generated [48] and PSA used this in three separate methods to increase differentiation between neutrons and alphas. The results of these three methods were ultimately combined to improve both rejection of alpha events, to over 98%, and acceptance of neutron events, to 74.8%. Details of the three analyses are available in [49, 78].

The NCDs only have one purpose: detecting neutrons. PSA has limited ability to distinguish the spatial location of an event using which NCD the event occured in and the vertical position along the NCD. The latter can be crudely determined by the timing distribution of the signal pulse from the NCD, as the pulse is read off of the central wire at the top, but reflects off the bottom, creating two pulses a few nanoseconds apart. This suggests that the PSA may be able to distinguish the NC signal from some of the neutron backgrounds. Unfortunately, this turned out to not be the case. Instead, the PSA only reports the total number of neutrons detected as $N_{PSA} \pm \sigma_{PSA}$. To integrate this in to the analysis, we compute the number of neutrons expected in the NCDs given the current value of $\vec{\alpha}_i$, which we call $N_{exp}$. In Section 4.6, we saw that there is a direct relationship between the number of events in the PMTs and those in the NCDs, in that the number of events expected in the PMTs is given by $f_x N_{x,NCD}$ where $x$ is one of our backgrounds and $f_x$ is the NCD-to-PMT conversion factor. We choose to vary the number of events in the PMTs in actual analysis for computational ease. Our MC of the detector also provides us with the expected number of neutrons in the NCDs from the NC flux (labeled $N_{NC,NCD}$). We combine these to give

$$N_{exp} = \frac{\epsilon_{NCD}}{\bar{\epsilon}_{NCD}} B8 N_{NC,NCD} + \sum_x \frac{N_{x,PMT}}{f_x} \tag{8.8}$$

where $B8$ is the $^8$B flux, normalized to 1, $\epsilon_{NCD}$ is the NCD efficiency of capturing neutrons and $\bar{\epsilon}_{NCD}$ is the mean value of $\epsilon_{NCD}$. The backgrounds are given in expected numbers of events, so they are not altered when the efficiency is floated. The efficiency is normalized since $N_{NC,NCD}$ already accounts for it.

$N_{exp}$ is computed at each step in the MCMC and compared to $N_{PSA}$ to give a contribution to the log likelihood given by

$$\ln \mathcal{L}_{NCDs}(\vec{\alpha}) = -\frac{(N_{exp} - N_{PSA})^2}{2\sigma_{PSA}^2} \tag{8.9}$$

which is added to the total ELL($\vec{\alpha}$) in the same manner as the LETA constraint.

## 8.4 Implementation

To actually run the signal extraction, the abstract method we have been describing must be made concrete in the form of a computer program. We wanted this program to be written in C++ using ROOT, the current standard in particle physics, as C++ gives reasonably fast, portable code and ROOT allows direct interaction with ROOT's powerful analysis utilities. We wanted it to be object oriented to take advantage of C++'s object-oriented nature and object-oriented code's flexibility and understandability. We wanted it to be flexible, so that it could be used for another experiment with minor or no changes. Finally, we wanted as much as possible to be controlled by a configuration file that exists outside of the code, so that we could run various tests on different configurations of data, systematics and MC without having to change the underlying code.

In the following discussion, we will attempt to distinguish between the C++ computer code (the code), the compiled executable that runs (the program), the configuration file that directs the program as to what to do (the config file or the configuration file) and the files containing the data events (the data file) and the MC events (the MC file). Often the difference between something in the code and something in the program is ambiguous, as the program is generated from the code. In those cases, it is simplest to assume that the thing in question is part of both, or rather that it is part of the abstract conception underlying both rather than these two concrete expressions of this conception. Additionally, the expression "structure" here does not mean a traditional C/C++ `struct`, rather it refers to any sort of logical organization within the code. Anything appearing in `typewriter` font refers to a component of the code or program, for example to distinguish between the `Pdf` class and the mathematical Pdf $N\,p(\vec{x}|\vec{\alpha})$.

### 8.4.1 Overview

A run of the program consisted of several stages. First, the config file was read and checked for errors (usually typos). Next, the information in the config file was used to set up the necessary internal structures, telling the program how many fluxes of what type, how many systematics and what their action should be, the initial state and step sizes for each parameter, likelihood contributions from any constrained parameters, etc. Then the MCMC chain ran, walking the parameters across configuration space and recording the appropriate parameter values. Finally, the results were output to files and the program performed clean up actions (closing files, clearing memory, etc).

The setup phase is the most complicated. A hierarchy of structures is created. At the highest level, the MCMC itself runs. It contains $\vec{\alpha}$ as a list of parameters, with an initial starting point, step size, current value and possible constraint for each extracted from the config file. It also extracts from the config file how many steps the chain should take and how it should record its output, i.e. to what file, which parameters to write and how often to print status messages to the screen. To run the MCMC, it varies the current values of the parameters, evaluates $\text{ELL}(\vec{\alpha})$, decides whether to keep or reject the step and records the result. Computing $\text{ELL}(\vec{\alpha})$ is not trivial. There are two contributions: from the data and from the constraints. With a few exceptions noted in the class descriptions below, the constraints are dealt with by the MCMC via the mechanism of the `LogLikelihoodFormula` or parameter constraints. The former is a function defined in the config file that is explicitly added to the log likelihood. The latter adds directly to the log likelihood as well, but is a separate, simpler function that only deals with Gaussian constraints, which are by far the most common kind. The data portion is handled by the `Pdf`'s, one for each data set. For this analysis we have two three dimensional `Pdf`'s, one for the day data and one for the night.

Each `Pdf` has a simple task: it takes a the current $\vec{\alpha}$ as input and returns the corresponding log likelihood. To this end, it contains the data set $\{\vec{x}\}$ and a number of `Flux`'s and `Sys`. Each `Flux` corresponds to a flux as we have been referring to it so far: it is a repository of the MC events $\{\vec{y}\}$ for a single signal or background. Each `Sys` takes a single MC event $\vec{y}_i$ as input and returns how the vector has changed, the $\Delta\vec{y}_i$ from Chapter 4. The Pdf thus takes one flux at a time, applies the relevant systematics (not all systematics are applied to all fluxes, see Section 4.7) to it one event at a time, then fills that event in to a histogram with the appropriate weight, all of which are calculated by the `Sys`. Once all the fluxes have been processed, the histogram is renormalized to create the binned Pdf. The log likelihood is then computed by finding the value of our Pdf corresponding to each data event and summing, then adding the extended log likelihood component at the end to give the $\text{ELL}(\vec{\alpha})$ described in Section 7.3.

### 8.4.2 Classes

In object-oriented programming, the majority of the code and actual computation is handled through classes. The code has ten classes, three of which are "helper" classes that aren't directly involved in the signal extraction, each of which handles a particular aspect of the process. To explain how the code as a whole works, we must first describe how each class works. Thankfully, the nature of object-oriented programming allows us to ignore most of the internal details of a class when it interacts with another class, only the inputs and outputs matter. In this explanation, we will strive to separate the descriptions in to a short explanation of how the "outside world" sees the class, then a longer explanation of what happens inside the class. We will start from the most basic classes and build up to the most complicated, as the topmost class (the `MCMC` class) contains within it instances of almost all the other classes.

#### ConfigFile

`ConfigFile` is the first helper class, and the only code not written expressly for this project. It is an open-source config file reading utility, written by Richard J. Wagner at the University of Michigan [79]. This is the only code we inherited from and have in common with [57]; that analysis downloaded this class from an online repository. This rather simple class reads in a file that consists of a list of lines of the form `key = value` and creates an internal structure allows one to either traverse the list of keys or reports the value associated with a particular key. It is very similar to the `map` class in the C++ standard library. We will refer to this library by its customary acronym STL for brevity.

#### Tools

`Tools` is the next helper class. As the title suggests, it contains a set of tools that were useful across most of the other classes. The first of these utilities is `SearchStringVector`, which looked through an STL vector (effectively a list) of strings for a particular string. This was used a great deal during set up, as the various program components were kept track of by a string (their "name"). The next is `ParInfoToString`, which converted internal descriptions of components to the external, more human-readable ones used in the config files. The third is `DoublesAreCloseEnough`, which compares to doubles to see if they only differ in the last few bits - this allows checking if two doubles are "equal" within machine rounding. The final tool is `VectorScramble`, which randomizes the order of entries in an STL vector. This was used mostly for debugging.

**Errors**

The final helper class is `Errors`. It is a very simple class that contains an STL vector of strings. This is a global list, so that any time any piece of code adds to it, all of the program can see it. This was used for error handling - any time a piece of code encountered an error, for example a malformed configuration file entry or a non-existant file, it reported this to `Errors`. This allowed the program to process things as best as possible instead of immediately exiting, allowing multiple problems to be seen at once. At the end of the setup phase or when an error that could not be overcome was encountered, the `Errors` class printed all of the accumulated error messages to the screen and exited the program cleanly, rather than crashing. This was extremely helpful any time the config file was changed, as it often caught all the typos in one run.

**RealFunction**

`RealFunction` is a simple class that takes an arbitrary number of double (real number) inputs and returns a single double, i.e. a code implementation of a simple multivariable function $f(x, y, z, ...)$. Another class can set the values of each input with `SetParameter` and ask for the computed value by calling `Eval`. It is actually a virtual parent class, with each daughter class being specified to perform one computation. For example, one daughter class takes two inputs and returns their product, and another takes four inputs ($a$, $b$, $c$ and $x$) and returns $a + bx + cx^2$. So what function `RealFunction` performs is decided at set up time by choosing one of the available daughter functions, which have the property that the code can't distinguish between them and `RealFunction` in terms of use.

Internally, `RealFunction` contains an array of doubles, one element for each input. The `Eval` function is different in each daughter class, but it is almost always just the real function being performed. Unfortunately, that means that if a function not already available is needed, a new daughter class must be written and the code recompiled (these daughter classes are very simple and are all defined in `FunctionDefs.h`. This naturally raises the question of why this was done this way, rather than writing a general purpose class that can take any arbitrary function without needing to specify beforehand. The answer is simple: speed. This general purpose function exists in ROOT as `TF1`, but is much, much too slow to be useful. This implementation is over ten times faster, and this function gets called billions of times during a typical run, meaning that speed is critical. In fact, evaluation of RealFunction is more than 30% of the total running time of the program.

**Flux**

The `Flux` class keeps track of the MC events for a single flux or background. It is the first "named" class - it has a unique string value that identifies it, so each instance must have a name that is unique in the program (not shared with any other named structure). This allows for it to be looked up and called in other classes by name rather than some more abstract way, which is necessary as part of the setup process. During setup, it reads the MC events from a config file specified file. It also reads from the config file the `TimesExpected` (described in Section 8.1) and a quantity called `FluxNumber`; the former is a scaling factor that determines the nominal number of data events for this flux, the latter is used by `MCMC` to decide whether a given `Sys` should act on this flux. During running, this simple class returns an individual MC event $\vec{y}_i$ when called. It can return events in any order.

Internally, the MC events are stored in memory as a large array, for speed. They are returned as a vector of doubles of length equal to the length of $\vec{y}_i$ plus one, with the extra element being the weight $W$, set to 1 and altered by the `Sys`. `Flux` has member functions to return its name, number of MC events, `TimesExpected` and `FluxType`

### Sys

The `Sys` class applies a function to a set of values. It is another named class, again its name must be unique within the program. At its heart, an instance of Sys contains either a `RealFunction` or a `TF1` that returns a single double given a set of parameters. The `Sys` keeps track of which parameters out of $\vec{\alpha}$ it needs, as well as any parts of $\vec{y}_i$ it needs, and hands these to the underlying function. It then records the results in a separate vector with the same size as $\vec{y}_i$, which it returns. This allows for all `Sys` to have access to $\vec{y}_i$ as it comes from the `Flux`. Each `Sys` also contains a list of `FluxNumber`'s that it affects, and returns either true or false when asked if it affects a particular `FluxNumber`. Most instances are within a `Pdf`, but the class also serves double duty in that it is used to implement both the `LogLikelihoodFunction`'s and the `AsymmFunc`'s, described in the `MCMC` description.

As mentioned, `Sys` contains either a `RealFunction` or a `TF1`. These are nearly identical in function, save that `RealFunction` has a very short list of functions it can perform but is fast, while `TF1` can perform an arbitrary function but is slow. During setup, the `Sys` is given a list of all the parameters in $\vec{\alpha}$ (called the mcmcPars) and all of the components of the MC events (called the branchPars or branches). The config file tells the `Sys` what function to use and what values it needs in what order, out of these two lists. Since this is a bit error prone, many of these are hard-coded into the `Sys` class and automatically selected by the `Sys`'s name, though this can be overridden. `Sys` keeps track of which values it needs to pass to the underlying function, in the sense that it stores these values as class members. Each `Sys` only alters one component of $\vec{y}_i$, called the “target”, just as each systematic does. This is necessary, both to follow the logical structure of the analysis and because the underlying function only provides a single output. The config file (or the automatic selection) sets whether the computation is done using the $\vec{y}_i$ from `Flux` or $\vec{y}_i + \Delta\vec{y}_i$, i.e. the “new” value incorporating all `Sys` applied thus far. It also sets whether the output value should be added to the target element (most systematics) or multiply the target element (the weight altering systematics). Finally, it reads from the config file (or the automatic selection) a list of `FluxNumber`'s, which it stores. During a run, a `Sys` takes the list of parameters $\vec{\alpha}$ and updates its internal values to correspond to the new list. A member function takes a `FluxNumber` as input and returns either true or false, corresponding to whether it acts on that `FluxNumber` or not. This happens once per `Flux`. Finally, the `Sys` is given an event $\vec{y}_i$ and the changes made thus far to its value $\Delta\vec{y}_i$ and updates those changes.

### Pdf

This actually encompasses three classes: `PdfParent`, `Pdf1D` and `Pdf3D`. `Pdf1D` and `Pdf3D` implement a 1D and a 3D histogram/pdf, respectively, while `PdfParent` handles tasks that are common to the two. We will ignore the division of labor and dimensionality differences and speak of a `Pdf` class. From the viewpoint of outside entities, it only has two tasks: it reads from the config file, data file and MC file the information it needs (and sets itself up) and it returns its contribution to $\text{ELL}(\vec{\alpha})$ when given $\vec{\alpha}$.

Internally, this class coordinates the activities of the other classes (except `MCMC`). It stores the data events $\{\vec{x}\}$ in the same way that `Flux` stores the MC events. Also, it contains the histogram that is to be filled with the MC data to create $N\,p(\vec{x}|\vec{\alpha})$, a `TH1D` for `Pdf1D` and a `TH3D` for `Pdf3D`. During setup, it reads from the config file which fluxes it should use when building its Pdf and which systematics (including things treated as systematics, such as $P_{ee}$), creates instances of `Flux` and `Sys` as appropriate, and gives them the information they need to construct themselves. In particular, each element in the MC event vector $\vec{y}_i$ has a name, given in the config file, which must be consistent across all MC sets. The config file specifies what file the `Pdf`'s data is in, which it reads in to an internal array, with the element names also specified in the config file as for the MC events. During a run, the `Pdf` is given a new value for $\vec{\alpha}$, which is passes along to the `Sys`. The histogram is then emptied. Then the first `Flux` is asked for its `FluxNumber` and each `Sys` is asked if it acts on this flux, creating a list of `Sys` that act on this `Flux`. The `Flux` is then asked for its elements one at a time, with each acted upon by the appropriate `Sys` to create $\Delta\vec{y}_i$. The histogram is then filled with $\vec{y}_i + \Delta\vec{y}_i$, with weight given by equation 8.1. The Scale term is special, in that it is a parameter with the same name as the `Flux` that is automatically created by `MCMC` when the `Flux` is created. This is then repeated for each `Flux` and the histogram is rescaled as explained in Section 8.1. Then the `Pdf` takes each data event, finds the corresponding bin in the histogram, and adds the logarithm of the histogram's entry to log likelihood (which is reset to zero when this process starts, so only this `Pdf` and this step's values are used). The extended log likelihood is computed by subtracting the total number of events. This value is returned.

### MCMC

The `MCMC` is the highest level class. There is only one instance per program. In fact, other than some input processing and a timer, the main program consists of having `MCMC` read the config file then calling `MCMC.Run()`. From the outside point of view, this class is a black box that takes a config file as input, then performs the analysis (including reading the files and writing the output).

As most of the work is actually done by `Pdf` and the classes inside it, the `MCMC` class mostly acts to coordinate. During setup, it reads from the config file those parameters directly governing the MCMC chain behavior, such as the number of steps in the chain, how often to print to the screen (every $n^{th}$ step it prints the current parameter values) and which extra information about the parameters to save (such as the proposed values $\vec{\alpha}_i^{prop}$ that are rejected). It reads the name of the output file and creates both that file and the `TTree` (a ROOT data storage class) that will be written to the file. It reads the number and dimension of the `Pdf`'s and creates the appropriate `Pdf`'s, then passes the config file to each to allow it to construct itself. It then reads out of the config file all of the parameters that make up $\vec{\alpha}$, most of which are explicitly parameters, but one is automatically created and shares a name with each `Flux` and `Sys`. For each of these parameters, it reads in the initial value, maximum and minimum value (if defined), step size and (if applicable and Gaussian) the mean and width of the parameter's constraint. Finally, it reads from the config file the setup information for the `LogLikelihoodFunction`'s and the `AsymmFunc`'s. These are special instances of the `Sys` class that do not act on the MC events. Instead, the `LogLikelihoodFunction` uses the `Sys` machinery to create a function that takes some number of the parameters as an input and outputs a number that is directly added to the log likelihood at each step in the MCMC. This is used to compute values for non-Gaussian constraints, which can be arbitrary functions, and for direct contributions to the likelihood such as LETA and PSA. The `AsymmFunc`'s are used to alter the values of parameters based on other parameters. This allows a parameter such as the energy scale to be computed - it is actually the sum of two other parameters. It is also used to compute the values for the day and night versions of a parameter from $\bar{\alpha}$ and $A_\alpha$, as the name might suggest. Once this setup is complete, the `MCMC` unsurprisingly run the Metropolis algorithm. At each step, it varies the values of the parameters according to their step sizes (which can be zero) and adjusts them according to any `AsymmFunc` to create the $\vec{\alpha}_i^{prop}$. If any parameter is above its max or below its min, the step is rejected. Otherwise, the log likelihood is reset to zero and the contribution from each constraint, each `LogLikelihoodFunction` and each `Pdf` is computed and added. Then the Metropolis algorithm decides whether to keep or reject the step. If it is kept, the proposed parameters replace the old parameters, if not the proposed parameters are rejected. The values of the parameters are then added to the `TTree`. When the whole algorithm has finished (i.e. it has taken the prescribed number of steps), the `TTree` is written to the output file, stray memory is cleaned up and the program exits.

Details about how to interact with the various aspects of the program, in particular how to write a config file to use the program, are given in Appendix A.

## 8.5 Three independent methods

This analysis was one of three competing analyses. All three took the NCD phase data with the goal of extracting a three phase, day/night result. We all shared the same data, Monte Carlo, background and systematic numbers and some simulated data for testing. The major differences were in implementation. One of the two, QSigEx by Pierre Luc Drouin, uses a minuit-type minimizer and limits the number of floated systematics. The other, UASigEx by Shahnoor Habib, also uses an MCMC method. However, it reuses the code used in [63, 57] with heavy modification, while the code used here was entirely written from scratch, except for the open-source code that reads in configuration files. The other two analyses are described in detail in [49].

## 8.6 Fake data tests

This section describes the series of ensemble tests done to test to see if the machinery of the SigEx was working correctly. Many sets of fake data events were generated, each one equivalent to the real data in size and scope. Each was then fit using the SigEx as if they were real data, and results were compared against the known input values.

As this was a common task for all three SigEx's, we share a set of fake data sets, created by Pierre Luc Drouin by sampling the Monte Carlo. To help with terminology, we will refer to one experiment-equivalent set of fake data events as a "set" and a group of these sets sharing common parameters an "ensemble".

For each ensemble, values of $P_0$, $P_1$, $P_2$, $A_0$, $A_1$ and systematics values (most of $\vec{\alpha}$ from the previous section) were chosen. A mean value (matching that of the real data) was chosen for each flux and background for each ensemble. For individual sets within an ensemble, the number of events for each flux was randomly drawn from a Poisson distribution with mean equal to the ensemble's mean value for that flux. This is important for getting the statistics to work out correctly, see [80] for details. Also of note is that the individual fluxes were kept separate at the time of the ensemble creation so that which fluxes and backgrounds were used in a particular test could be varied at will. This was extremely helpful, as it made it possible to remove the backgrounds when they were distracting from testing other components.

Comparing the SigEx output over the many of sets in an ensemble, certain characteristics of the results are expected. In particular the distributions of the Pull and Bias (defined below) are expected to be Gaussian distributed, with centriods consistent with zero and, in the case of Pull, a standard deviation consistent with 1. The test of the SigEx was to see if this was indeed the case for its output.

### 8.6.1 Pull and Bias

For each fake data set signal extraction, the MCMC procedure produces a posterior distribution for each parameter. The "best fit" value for each parameter is then retrieved by either taking the mean of the posterior distribution or by fitting a function (either a Gaussian or a two-sided Gaussian) to the distribution. Since we will be comparing our results against a Minuit-type fitter, we fit our results to an interated Gaussian fit and use the centriod ($\mu$) of the Gaussian as the value of the parameter, and the $\sigma$ of the Gaussian as the uncertainty on that parameter. Figure 8-1 is a sample fit.

The Pull and Bias are defined for each parameter and are viewed in terms of the fit results of that parameter for each set in an ensemble. For the rest of this discussion, we will be looking at things in terms of an individual ensemble and will assume that context. For each data set, we get a set of parameter values $\vec{\alpha}$. Since we now have a collection of vectors, let the first subscript define which parameter we are referring to, and the second subscript define which fake data set it is extracted from. For example, the centriod of the $i$th parameter (say, $P_0$) from the $j$th data set (the ensemble is, in some sense the set $\{j\}$) will be denoted $\mu_{i,j}$. Then the pull for a given parameter and data set is defined as

$$P_{i,j} = \frac{\mu_{i,j} - \bar{\mu}_{i,j}}{\sigma_{i,j}} \tag{8.10}$$

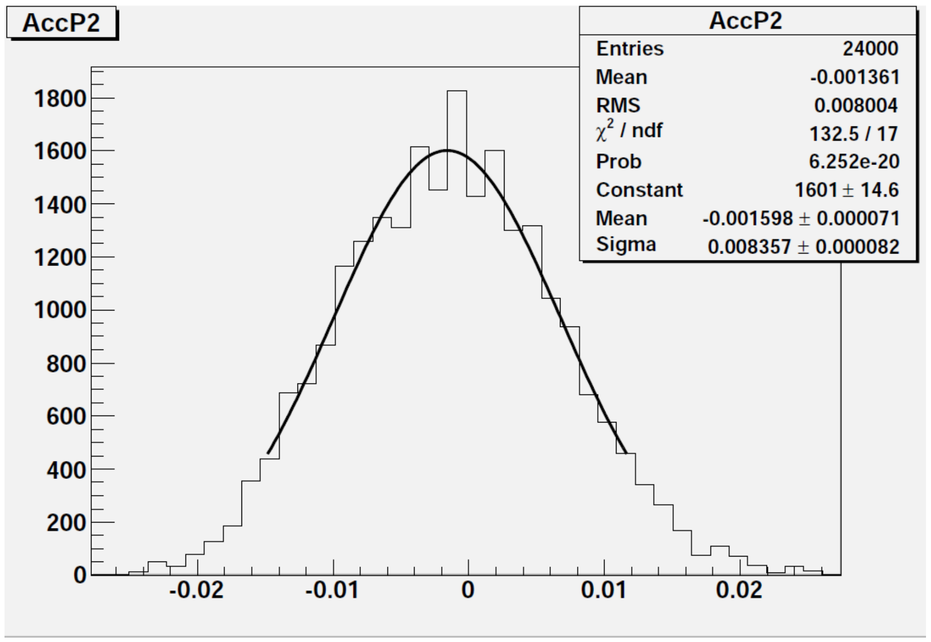


Figure 8-1: A sample iterated Gaussian fit. This is from the Signal Only Day/Night pull/bias test, for parameter $P_2$

and the Bias is defined as

$$B_{i,j} = \frac{\mu_{i,j} - \bar{\mu}_{i,j}}{\bar{\mu}_{i,j}} \tag{8.11}$$

where $\bar{\mu}_{i,j}$ is the expected value, i.e. that used to generate the fake data set (the value in the case of a systematic or other parameter, the mean in the case of a number of events in a flux).

For each parameter, a histogram is made of $P_{i,j}$ over all $j$'s. A Gaussian is then fit to this distribution (Figure 8-2 is a sample fit). The centriod of this Gaussian gives the value of the Pull for that parameter for the ensemble and is the quantity we are interested in. We expect it to be consistent with zero. We expect the distribution to have a "width" (as defined by the $\sigma$ of the Gaussian) of 1. The Bias $B_{i,j}$ is treated identically save that there is no expectation for the width of the Bias distribution - its width can be anything and is not particularly interesting.

One note of caution is necessary when dealing with parameters with constraints. If run naively, using the same constraint term for all sets, the distribution of pulls will be much too narrow. To deal with this, the mean value of the constraint needs to be varied - the idea being that if this really were an ensemble of experiments run

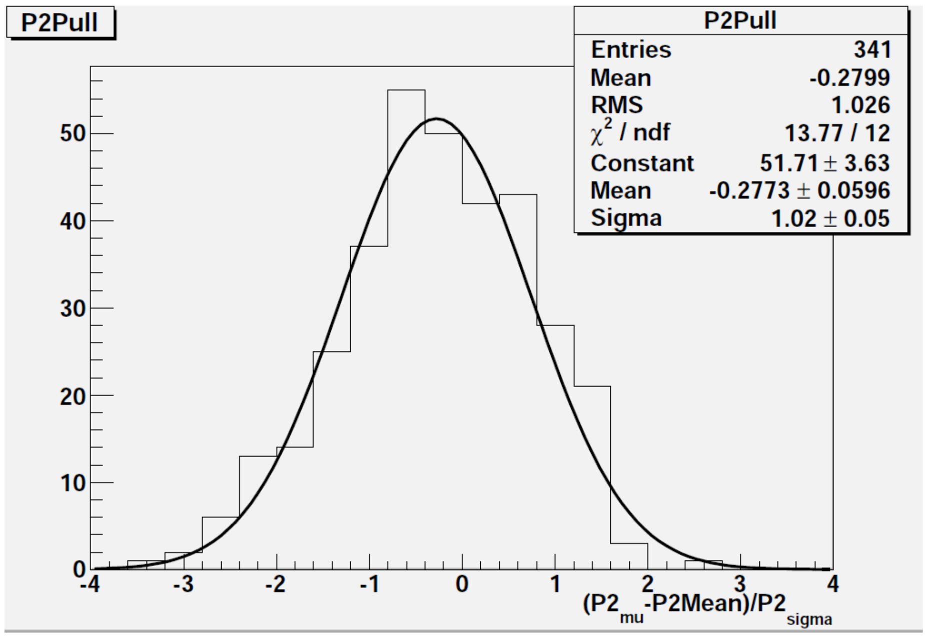


Figure 8-2: A sample Gaussian fit of a set of Pulls. This is from the Signal Only Day/Night pull/bias test, for parameter $P_2$

independently, each would have a different measured value of the mean of the external constraint (due to the limits of our ability to measure the constraint value as reflected in its uncertainty). To simulate this, the value of each constraint is drawn from a Gaussian with mean and width equal to the measured constraint value. See [80] for more information.

### 8.6.2 Signal only, no Day/Night

The ensemble tests were done in order of increasing complexity. The simplest fake data set to use was one where there were no systematics, no backgrounds, and no Day/Night difference - Pierre Luc's pee_lma_highstats data sets were used here. Hence, in this first round of tests, only NC, CC, ES and ES$_{\mu\tau}$ were included on the PMT side, and only the $^8$B signal on the NCD side. This was before PSA was implemented, so the NCDs were treated very similarly to how they were treated in [63]. The parameters varied in this set of data sets were $P_0$, $P_1$, $P_2$, the $^8$B scale (labeled BoronFlux in the plots), NCD Efficiency (labeled EffNCD in the plots) and PMT Efficiency (labeled EffPMT in the plots). One point of concern is that the pee_lma_highstats data sets were generated with a third-order polynomial survival probability (so it had a $P_3$ value), but these fits were done assuming a second-order polynomial. A second-order polynomail was used since all subsequent fake data sets used second order, and this was decided by the 3-Phase group to be the appropriate order to use in the actual extraction. This does not seem to have been a problem with the results, as we will see below.

Due to limits of computing resources available, each run (a full SigEx for a single fake data set) was limited to 25000 steps, with the first 1000 steps rejected as burn-in. Two concerns immediately present themselves: if the burn-in period is sufficiently long, and if enough steps have been taken. This is especially a concern when the other MCMC-type method uses much longer runs, often 10 times a many steps. Thankfully, there are simple methods for determining if these are sufficient.

For the burn-in period, three checks are needed. The most primitive is to look at the (log) likelihood as a function of step number and make sure that it has converged, indicating that the equilibrium (best-fit) value of the parameters has been reached. Convergence is indicated by the value becoming relatively stable. It will still fluctuates, but to a much smaller degree. Figure 8-3 shows the expected behavior. The second is to do the same for each parameter - occasionally a parameter will be able to vary without noticably changing the likelihood (for example, if two parameters are strongly correlated and are changing together in a non-random way). The third is to arrange an extremely long run and make sure that it converges to the same equilibrium value. Additionally, the burn-in period can be artificially shortened by simply starting the MCMC near the equilibrium value, which is very easy when we know the values used to construct the fake data sets.

For determining if a sufficient number of steps have been taken, it is important to realize that it is *independent* steps that are important. The Autocorrelation, described in section 7.5, is a measure of how many steps need to be taken before the parameter values at a step are not correlated with the values at an earlier step.

The resulting values of Pull are plotted in Figure 8-4, with the widths of the Pull distributions in Figure 8-5. As noted, the ideal situation is to have the Pull values

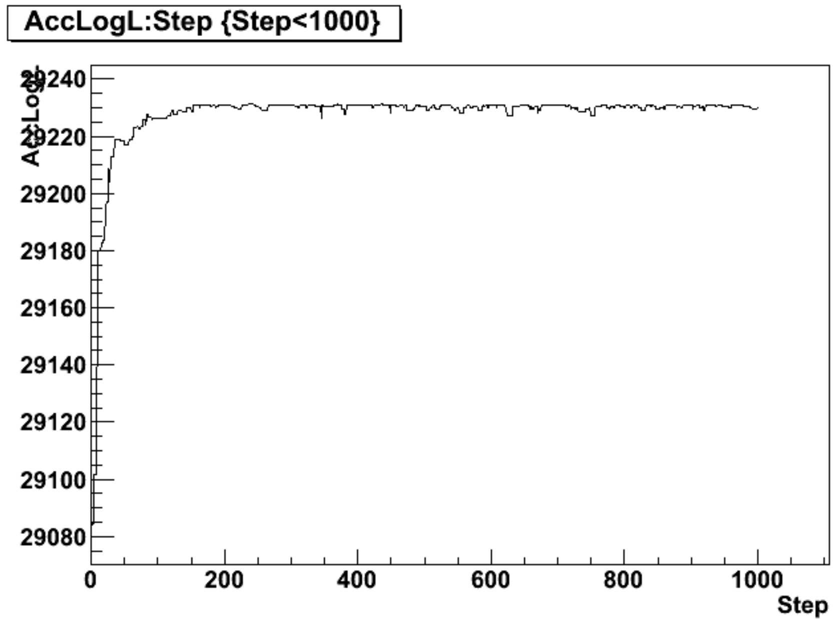


Figure 8-3: Sample log likelihood v. step plot showing burn-in

consistent with zero and the Pull widths consistent with 1. Here, all of the pulls are within two standard deviations of zero, and all of the widths are consistent with 1, indicating a successful test. The Biases are a bit less clear, though they are all acceptable. The "raw" Bias (as defined above) is shown in Figure 8-6. The parameter $P_2$ is not as well determined as many of the other parameters, as indicated by the very large error bar on its Bias (again, this error is the uncertainty in the centriod of the Gaussian fit). Its Bias is still consistent with zero, however. Unfortunately, it makes the other Biases hard to see, so Figure 8-7 shows the Biases scaled by the width of the distribution of the Bias, i.e. $\frac{\mu}{\sigma}$ for each Bias Gaussian fit, with uncertainties scaled appropriately. From this we see that all Biases are consistent with zero.

### 8.6.3 Signal only, with Day/Night

For the rest of the ensemble tests, Pierre Luc's data set pee_lma_highstats_v4 was used. This includes a Day/Night difference and a second order $P_{ee}$ polynomial. In this section, as in the previous, no systematics and no backgrounds are included, so only NC, CC, ES and $ES_{\mu\tau}$ are included on the PMT side and the $^8$B signal on the NCD side. The varied parameters now include the $P_{ee}$ asymmetry terms, bringing the

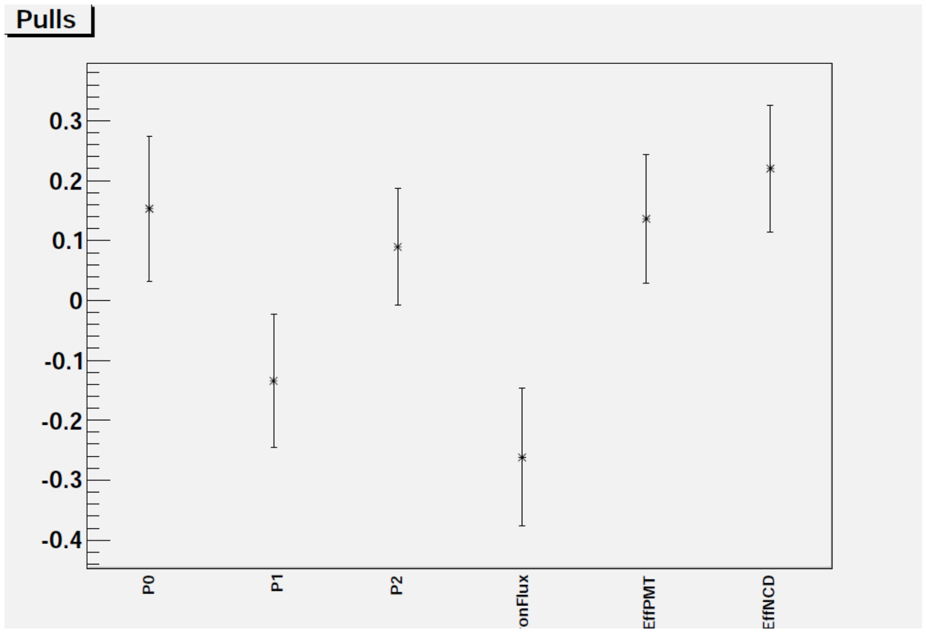


Figure 8-4: Centriod (with errors from Gaussian fit) of Pulls for parameters in Signal Only, no Day/Night studies

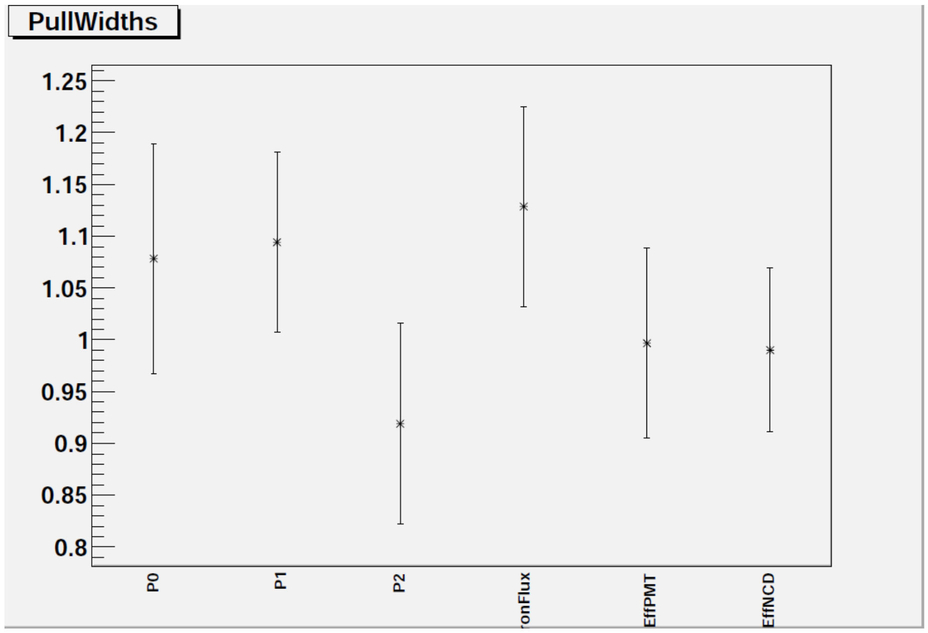


Figure 8-5: Distribution width (with errors from Gaussian fit) of Pulls for parameters in Signal Only, no Day/Night studies

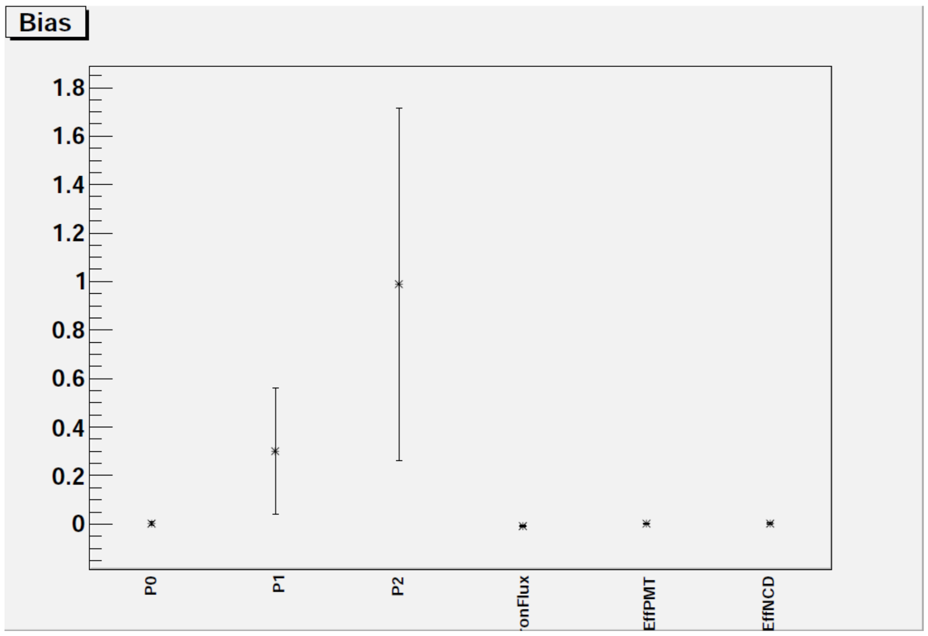


Figure 8-6: Centriod (with errors from Gaussian fit) of Bias for parameters in Signal Only, no Day/Night studies

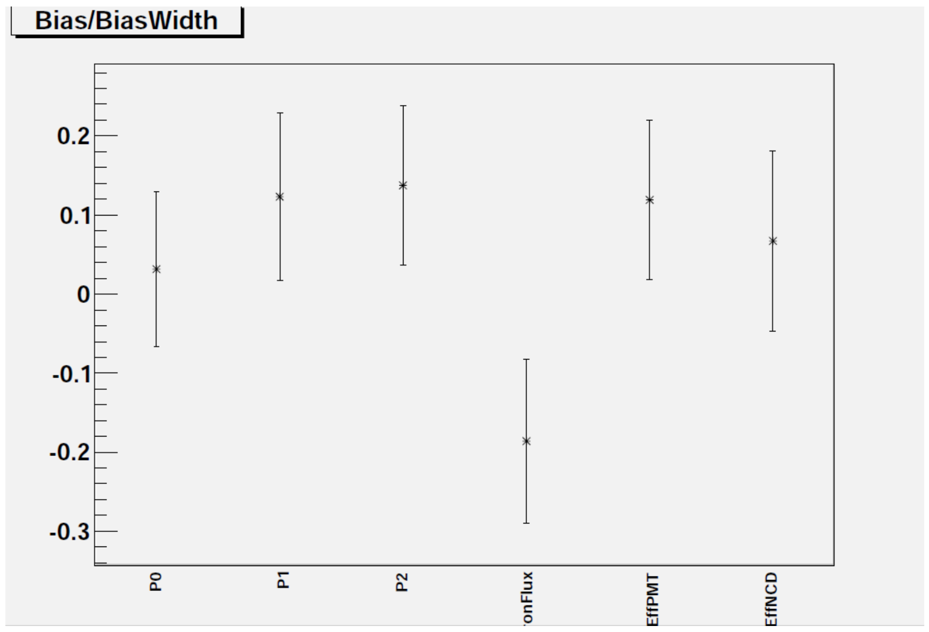


Figure 8-7: Centriod (with errors from Gaussian fit) of Bias, divided by the width of the Bias distribution, for Signal Only, no Day/Night studies

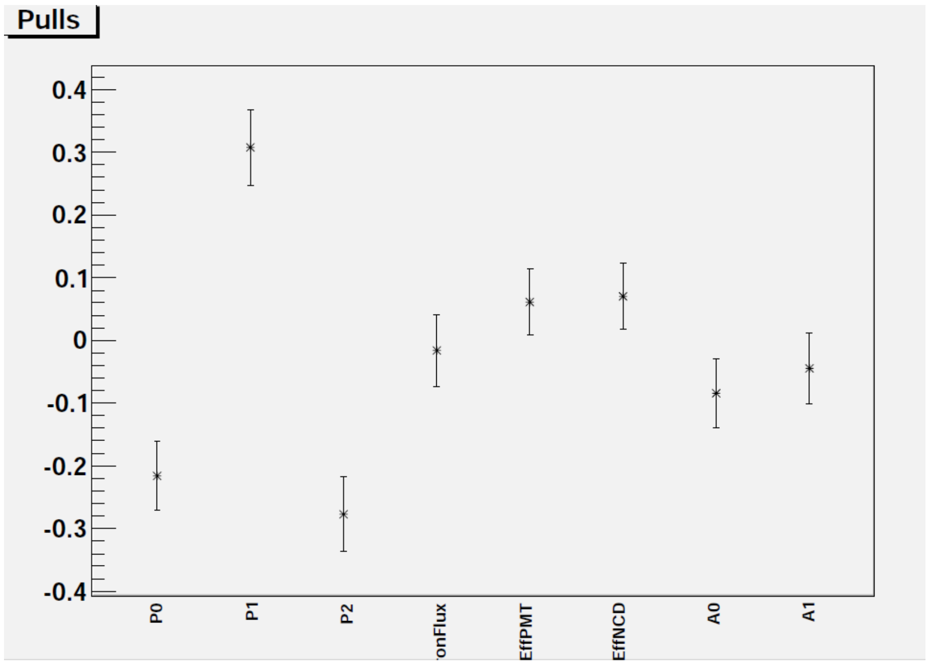


Figure 8-8: Centriod (with errors from Gaussian fit) of Pulls for parameters in Signal Only, Day/Night studies

total parameters to $P_0$, $P_1$, $P_2$, $A_0$, $A_1$, the $^8$B scale (labeled BoronFlux in the graphs), the PMT Efficiency (labeled EffPMT) and the NCD Efficiency (labeled EffNCD). The results for Pull values are shown in Figure 8-8, Pull widths are shown in Figure 8-9, Bias values are shown in Figure 8-10 and Bias divided by Bias width is shown in Figure 8-11. Here we see that the results aren't quite as ideal as before, especially for $P_0$, $P_1$ and $P_2$. The Pull widths are all consistent with 1, but both the Bias and Pull values for the $P$'s are not consistent with zero, or at least are marginally so. While this seems like a problem, much of this was resolved in the next section, when Backgrounds were included. In the process of including the backgrounds, it was discovered that a small adjustment to the expected number of events in the CC, ES and ES$_{\mu\tau}$ was missing - the presence of $P_{ee}$ slightly adjusts the relative amounts of the these three fluxes, which we had not accounted for when determining the correct MC to Data ratio.

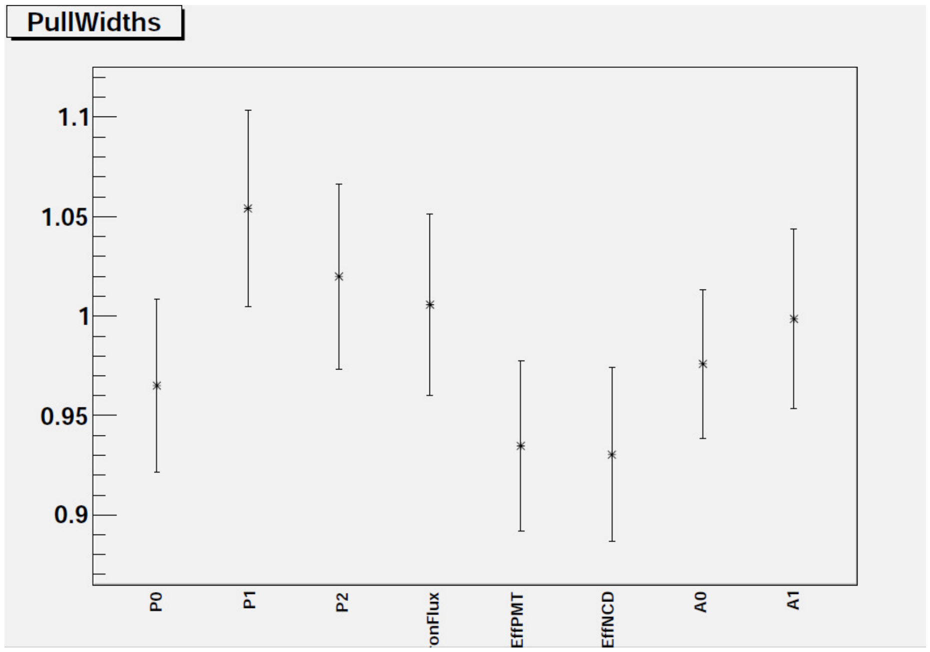


Figure 8-9: Distribution width (with errors from Gaussian fit) of Pulls for parameters in Signal Only, Day/Night studies

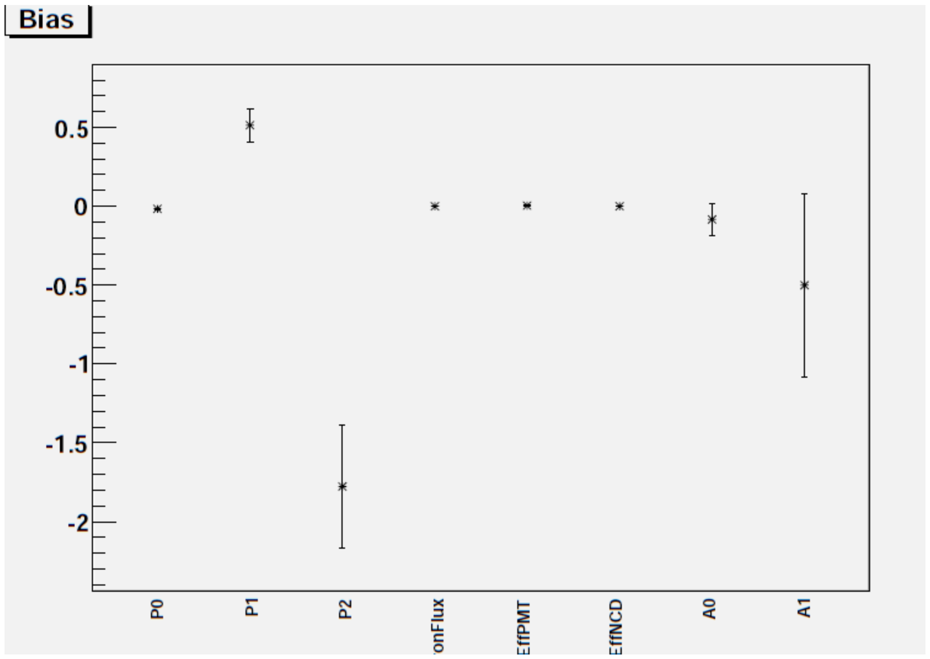


Figure 8-10: Centriod (with errors from Gaussian fit) of Biases for parameters in Signal Only, Day/Night studies

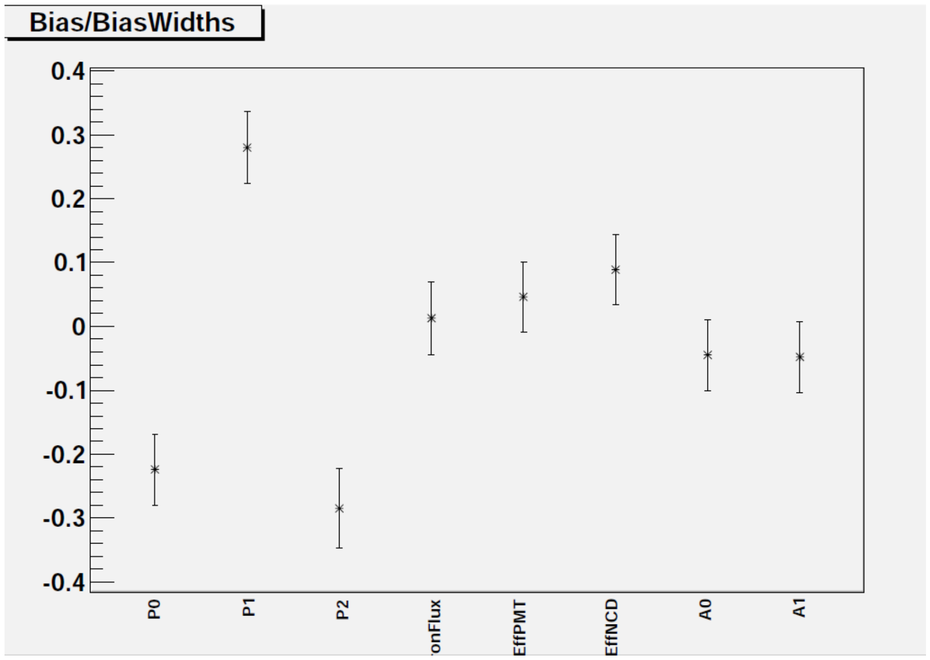


Figure 8-11: Centriod (with errors from Gaussian fit) of Biases, divided by width of the Bias distribution, in Signal Only, Day/Night studies

### 8.6.4 Backgrounds, no Systematics

The next set of Pull/Bias tests included everything in the previous section as well as the backgrounds available in Pierre Luc's fake data: external neutrons (labeled Ex), NCD photodisintigration (labeled ncdpd), $D_2O$ photodisintigration (labeled d2opd), K2 hotspot photodisintigration (labeled k2pd), K5 hotspot photodisintigration (labeled k5pd) and atmospheric events (labeled atmos). Cable events were not included. Each background had a very small number of events relative to the main fluxes (CC, ES, NC) and thus had a small impact on the fits. They were almost entirely constrained by their measured constraints, which became apparent when the widths of the posterior distributions was the same as the width of the constraint. The Pull values are shown in Figure 8-12, the Pull widths are shown in Figure 8-13, and the Bias divided by the width of the bias distribution is shown in Figure 8-14. The results shown here are from a set of tests performed after the next section (the systematics) and include several corrections discovered after the initial set of pull/bias tests, and include the PSA contraint. These results are in excellent agreement with

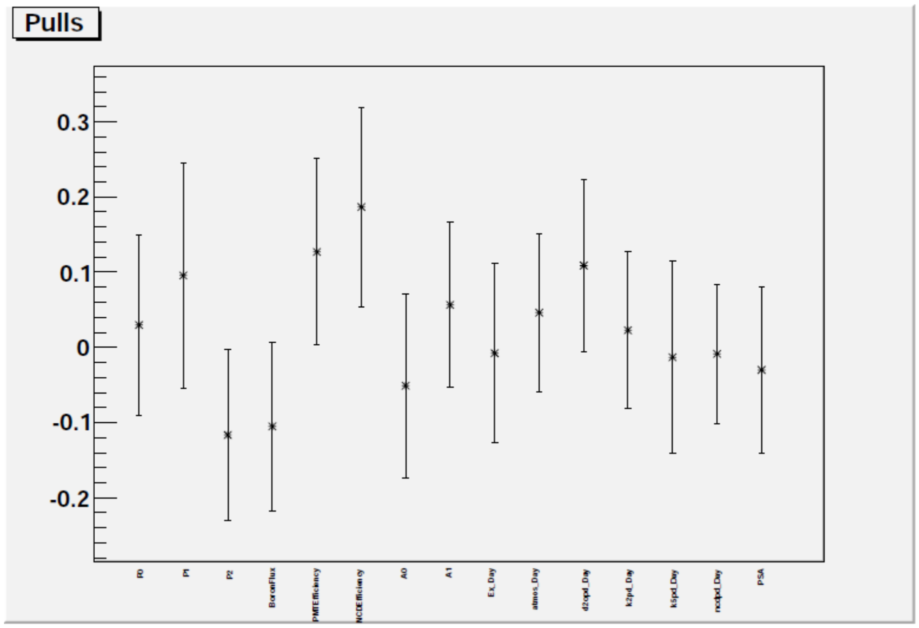


Figure 8-12: Centriod (with errors from Gaussian fit) of Pulls for parameters in Day/Night studies with backgrounds but no systematics

expectations.

### 8.6.5 Systematics, no Backgrounds

The next component of the signal extraction tested was the implementation of the systematics. These were subdivided into groups called Energy, XY and CosTheta. Due to an oversight, a few Z direction position systematics were not put through this test, but should behave identically to the X and Y position systematics. The Winter spectrum uncertainty was also not tested. These tests were all performed with backgrounds turned off, to both speed up and simplify the tests. Note that these tests were performed before the ones in the previous section. They do not include all of the final corrections or the PSA constraint. We decided not to rerun them due to time constraints, as each test took several weeks, but are confident that their results would show improvement similar to that between sections 8.6.3 and 8.6.4.

The Energy tests included the systematics Energy Scale (this was before the distinction between the energy scale and correlated energy scale was made), the diurnal asymmetry on the energy scale, the directional asymmetry on the energy scale, the

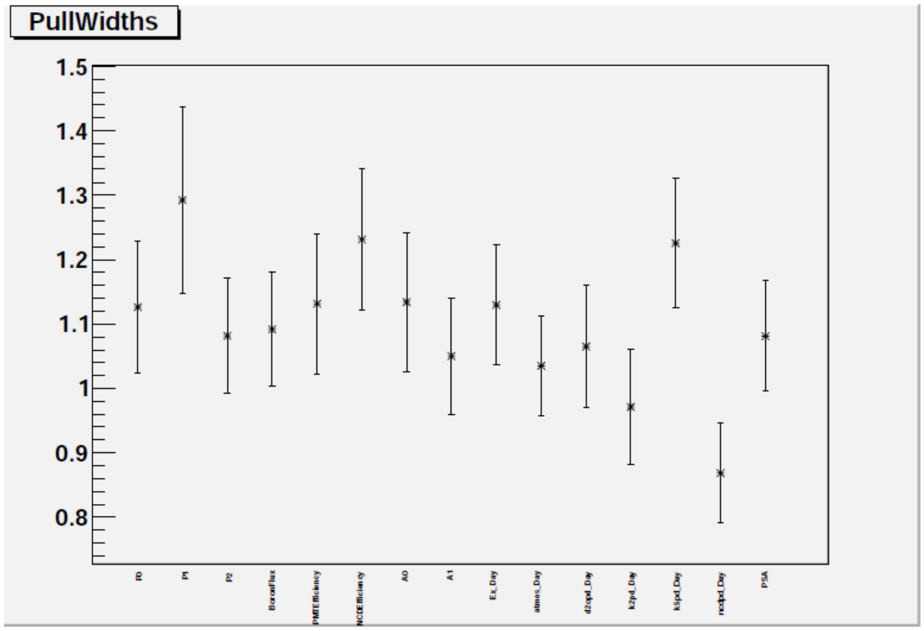


Figure 8-13: Distribution width (with errors from Gaussian fit) of Pulls for parameters in Day/Night studies with backgrounds but no systematics

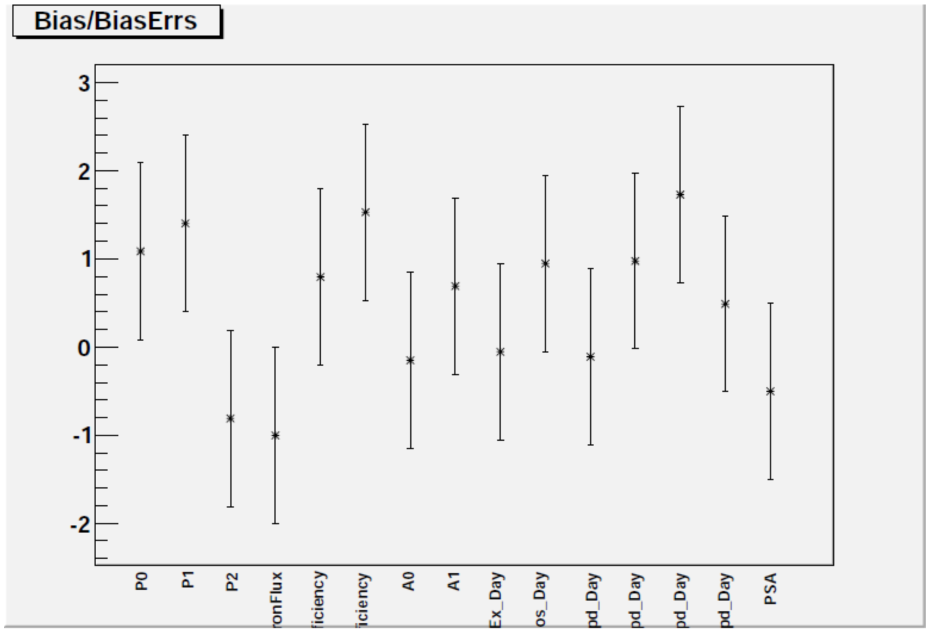


Figure 8-14: Centriod (with errors from Gaussian fit) of Biases, divided by width of the Bias distribution, for parameters in Day/Night studies with backgrounds but no systematics

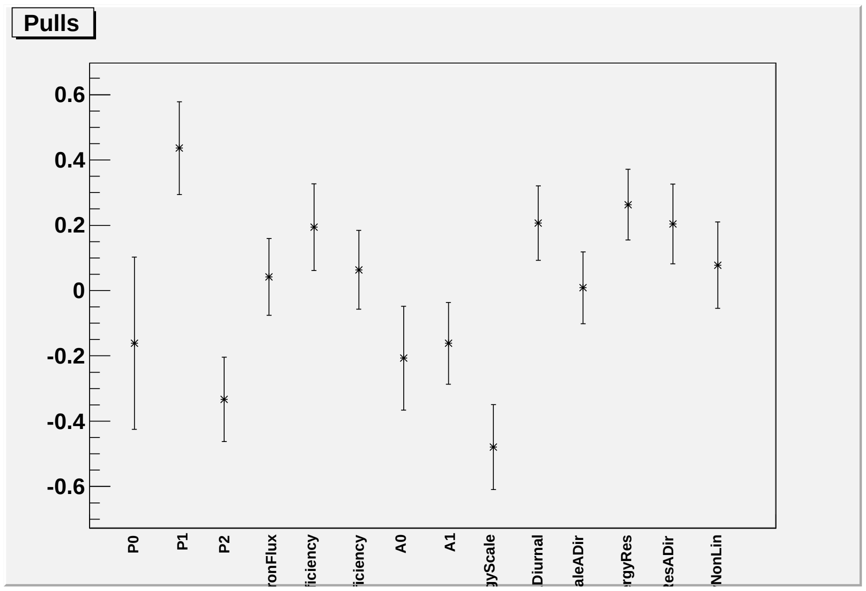


Figure 8-15: Centriod (with errors from Gaussian fit) of Pulls for parameters in Energy systematics set

energy resolution, the directional asymmetry on the energy resolution and the energy non linearity. The pulls are shown in Figure 8-15, the pull widths in Figure 8-16, and the biases divided by the width of the bias distribution in Figure 8-17. The offsets for the energy scale and $P_2$ are large, but at a bit more than 2 $\sigma$ were considered marginally acceptable.

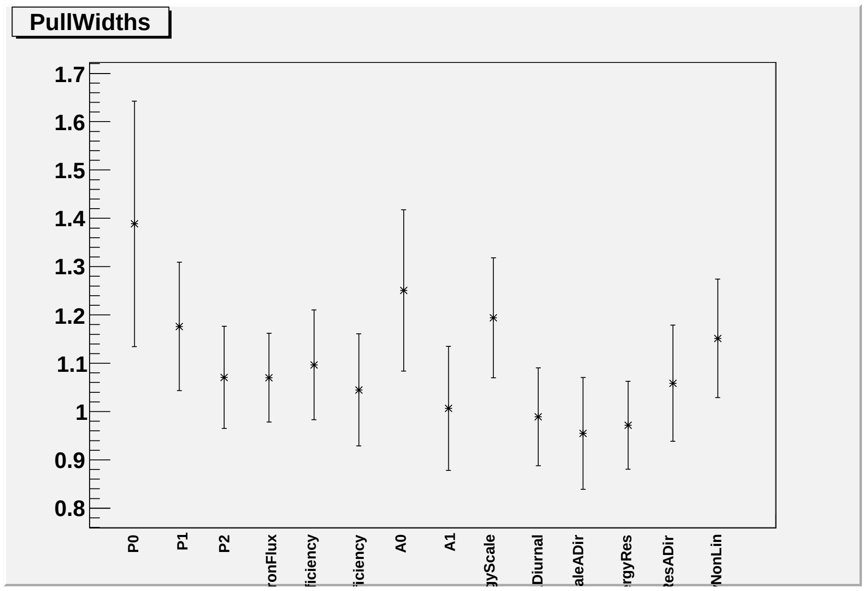


Figure 8-16: Distribution width (with errors from Gaussian fit) of Pulls for parameters in Energy systematics set

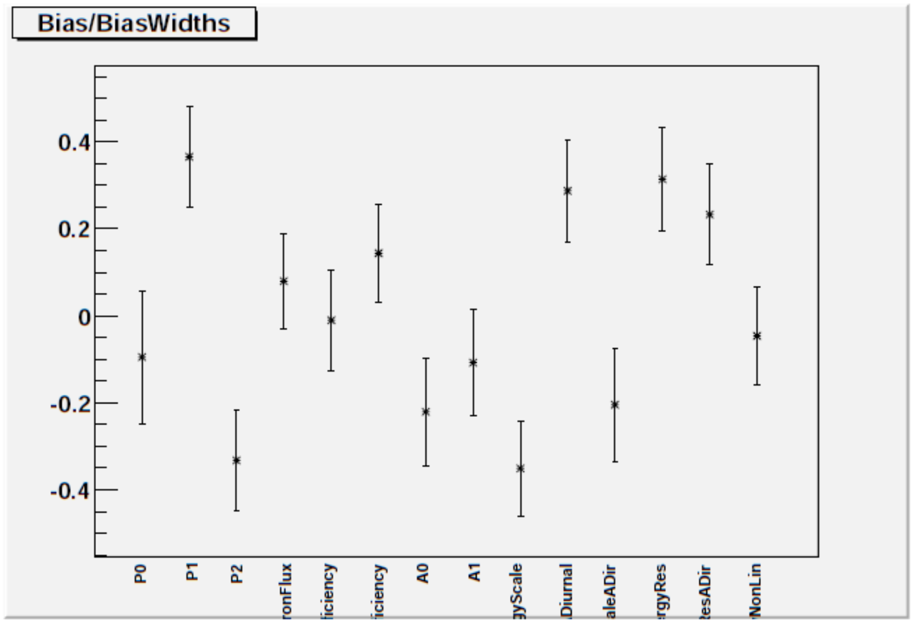


Figure 8-17: Centriod (with errors from Gaussian fit) of Biases, divided by width of the Bias distribution, for parameters in Energy systematics set

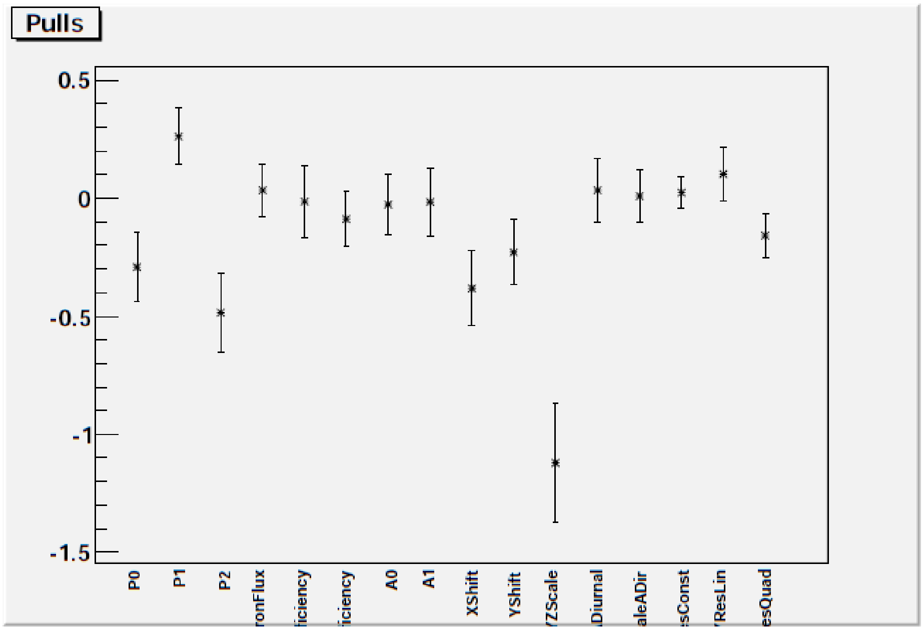


Figure 8-18: Centriod (with errors from Gaussian fit) of Pulls for parameters in the XY systematics set

The XY tests included the systematics x shift, y shift, the xyz scale, the diurnal asymmetry on the xyz scale, the directional asymmetry on the xyz scale, and the three xy resolutions (constant, linear and quadratic). The pulls are shown in Figure 8-18, the pull widths in Figure 8-19 and the biases in Figure 8-20. The large pull for the xyz scale is not a concern, it merely reflects that the xyz scale variable has an asymmetric uncertainty with more weight in the negative direction.

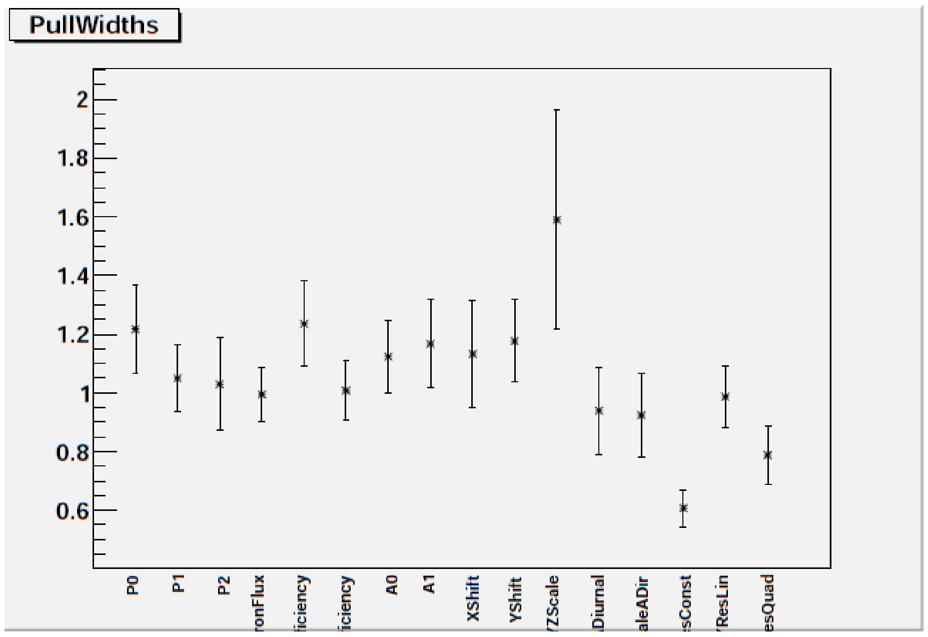


Figure 8-19: Distribution width (with errors from Gaussian fit) of Pulls for parameters in the XY systematics set

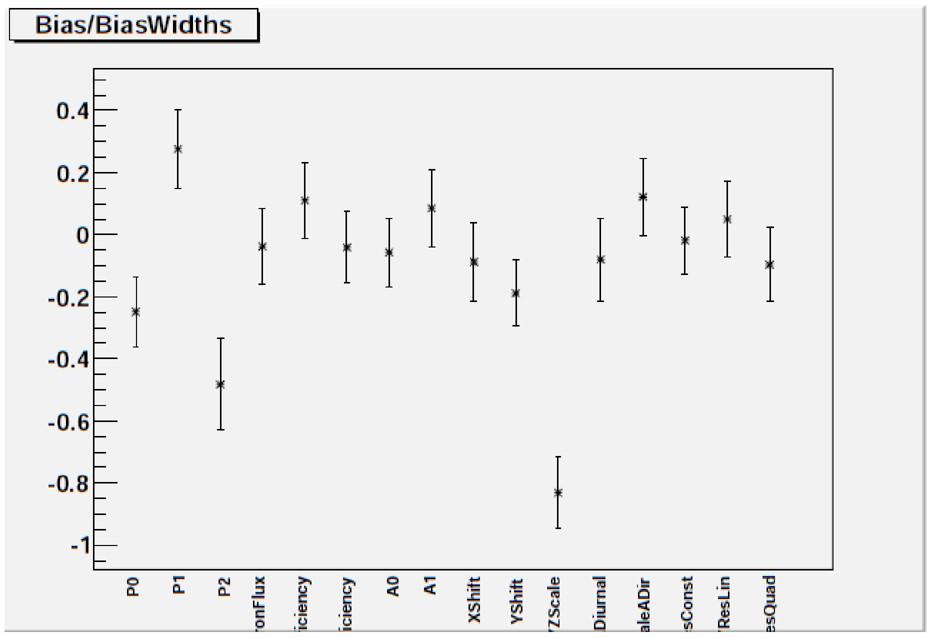


Figure 8-20: Centriod (with errors from Gaussian fit) of Biases, divided by width of the Bias distribution, for parameters in the XY systematics set

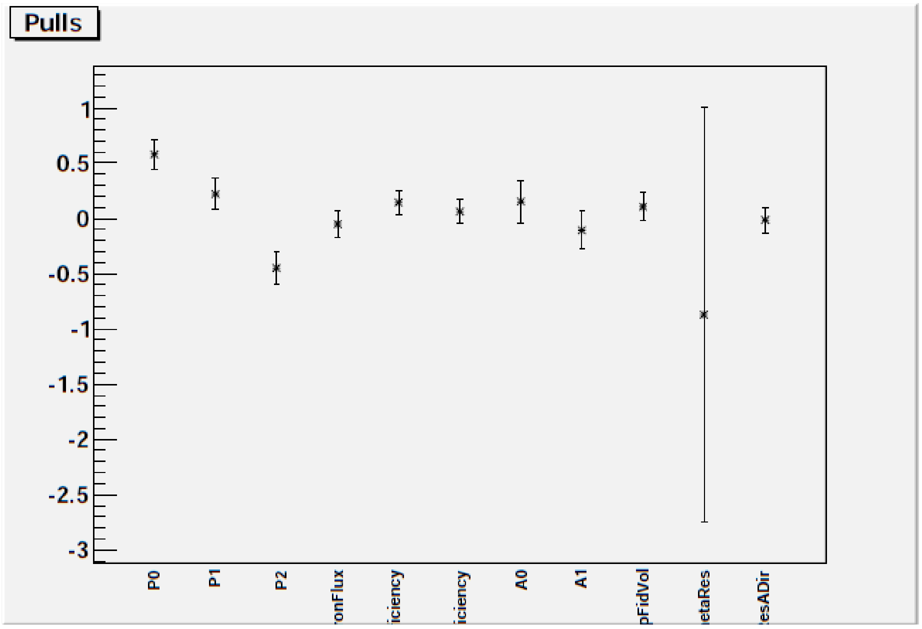


Figure 8-21: Centriod (with errors from Gaussian fit) of Pulls for parameters in CosTheta systematics set

The final set of systematics tests, the CosTheta tests, included the systematics $\cos(\theta_{sun})$ resolution, the directional asymmetry on $\cos(\theta_{sun})$ resolution and the energy dependent fiducial volume. The pulls are shown in Figure 8-21, the pull widths in Figure 8-22 and the biases in Figure 8-23. The large uncertainty on the $\cos(\theta_{sun})$ resolution and the large bias on $P_2$ are due to a fitting error. Unfortunately, the raw data is no longer available to correct this.

## 8.7 Comparison between methods

### 8.7.1 $\frac{1}{3}$ fake data comparison

The major comparison tests between the three SigEx methods was a $\frac{1}{3}$ fit. This is a fit over a fake data set with one third of the statistics of the real data set, so a third of the statistics used thus far in fake data sets. The reason for this change was preparation for the next stage - a test on a third-statistics real data set. For testing and process development purposes, SNO creates a data set with one third of the full statistics with some small changes to the data. This allows code to be tested

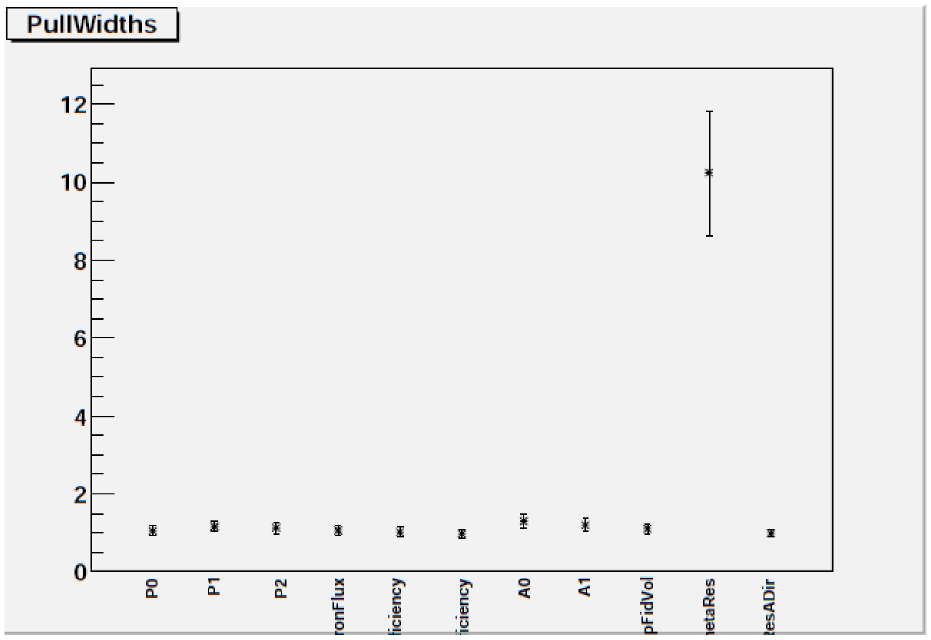


Figure 8-22: Distribution width (with errors from Gaussian fit) of Pulls for parameters in CosTheta systematics set

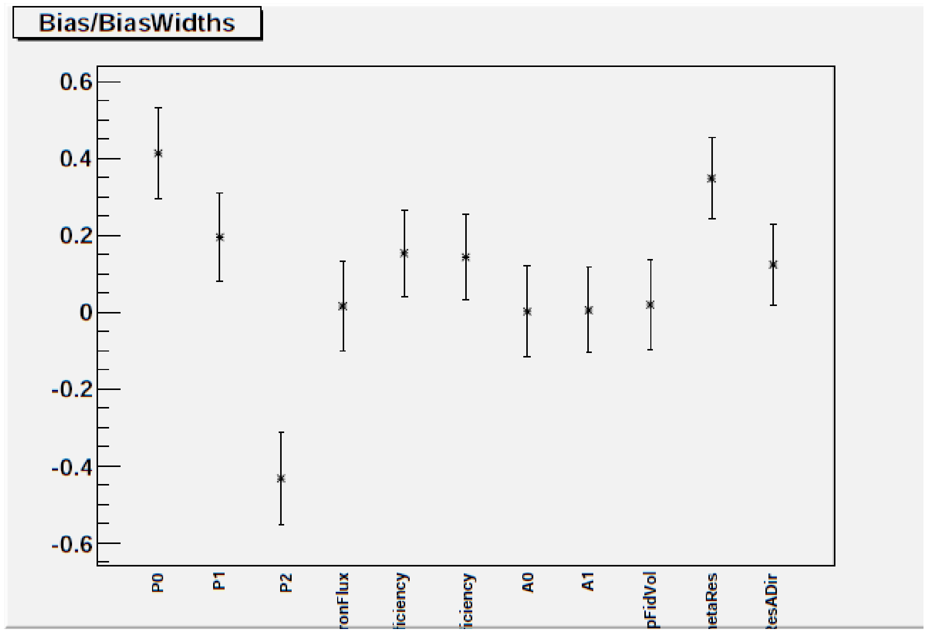


Figure 8-23: Centriod (with errors from Gaussian fit) of Biases, divided by width of the Bias distribution, for parameters in the CosTheta systematics set

on realistic data without breaking blindness, in the hopes of catching any errors that may still be lurking.

Since this is an ensemble test, many different fake data sets were fit. In this case, 45 data sets in one ensemble and 45 in a different, nearly identical ensemble. The second ensemble, called the "Alternate Seed", was exactly that - the same input parameters with the random number generator reseeded, resulting in slightly different data sets. Since so many data sets were fit, a reduced set of systematics and backgrounds were used to reduce computation time. Also, some of the systematics used different values than those listed in Section 4.6. The full list of systematics and backgrounds used appears in Table 8.4. As always, the four main fluxes (CC, ES, $\text{ES}_{\mu\tau}$ and NC), $P_0$, $P_1$, $P_2$ and $A_0$, $A_1$ were present.

As per usual, the output of the MCMC was fit with Gaussians and the mean ($\mu$) and standard deviation ($\sigma$) of the Gaussian fit were used as the fit value and uncertainty, respectively. These values (labelled RO in the graph) are compared to QSigEx (Pierre-Luc Drouin's SigEx, labelled PL) and UASigEx (Shahnoor Habib's SigEx, labelled SH). Each graph is for a single parameter, showing the three SigEx's values together for each data set. Only the "primary" values are compared: $^8$B Scale (called BoronFlux, Figure 8-24), $P_0$ (Figure 8-25), $P_1$ (Figure 8-26), $P_2$ (Figure 8-27), $A_0$ (Figure 8-28) and $A_1$ (Figure 8-29). Similar graphs were made for the alternate seed and showed very similar results.

The comparison results are summed up in two tables. Table 8.5 is the "average" value, and it is simply that - an unweighted average. The uncertainty on that value is a little misleading, as it is the uncertainty treating the uncertainties ($\sigma$'s) from the fits as the uncertainties on the values, rather than the fit uncertainties. It is most likely a significant overestimation, though the fit uncertainties would produce an underestimation, as the fitted value for a particular set is known better than the size of the fluctuations between sets. Unfortunately, this information was not available for all three methods.

| Systematic | Value |
|---|---|
| NCD NC Efficiency | $1.000000 \pm 0.012905$ |
| PMT NC Efficiency | $1.000000 \pm 0.023697$ |
| Energy Scale Correlated ($a_{0\ c}^{E}$) | $0.0000 \pm 0.0041$ |
| Energy Scale ($a_0^E$) | $0.0000 \pm 0.0081$ |
| Energy Scale Diurnal Asymmetry ($A_{diurnal}^{NCD}$) | $0.0000 \pm 0.0038$ |
| Energy Scale Directional Asymmetry ($A_{dir}^{NCD,ES}$) | $0.0000 \pm 0.0099$ |
| Energy Resolution, e$^-$ ($b_0(e^-)$) | $0.000000 \pm 0.016184^*$ |
| Energy Resolution, n ($b_0(n)$) | $0.0000 \pm 0.0119^*$ |
| Position Scale, xyz ($a_1^{x,y,z}$) | $0.000 \pm 0.009$ |
| $\cos(\theta_{sun})$ Resolution ($a_0^\theta$) | $0.00 \pm 0.12$ |
| $\cos(\theta_{sun})$ Resolution Directional Asymmetry ($A_{dir}^{ES}$) | $0.000 \pm 0.069$ |
| Background | Number of Events[1] |
| ex | $6.918 \pm 3.484$ |
| d2opd | $2.768 \pm 0.429$ |
| atmos | $8.227 \pm 1.633$ |
| ncdpd | $1.979 \pm 0.678$ |
| k2pd | $3.134 \pm 0.497$ |
| k5pd | $2.793 \pm 0.327$ |
| hepcc | $4.281 \pm 0.000$ |
| hepes | $0.356 \pm 0.000$ |
| hepnc | $0.385 \pm 0.000$ |

Table 8.4: Systematics and backgrounds used for $\frac{1}{3}$ comparison. Symbols in parenthesis correspond to those in Section 4.6
$^*$ these values are 100% correlated and vary as the same parameter multiplied by a constant
[1] These are events in the PMTs. In the actual fit, the number of events is incorporated in to the `TimesExpected` quantity and the floated parameter is a scaling factor of mean 1

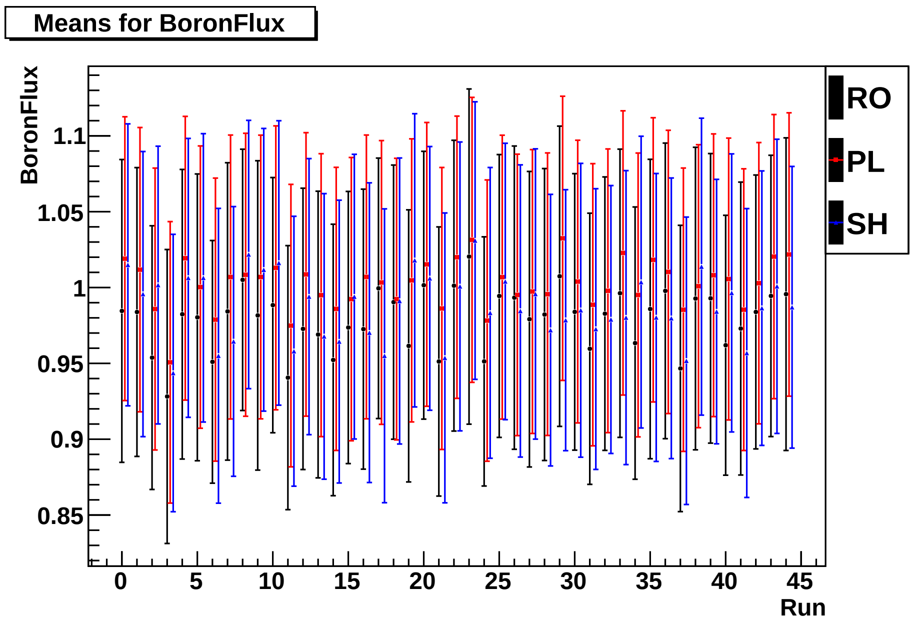


Figure 8-24: Comparison between the three SigEx's, for BoronFlux

Table 8.6 shows the "average difference", defined as

$$\sum_j 2\frac{|\mu_{j,RO} - \mu_{j,SH}|}{\sigma_{j,RO} + \sigma_{j,SH}}$$

Where $j$ represents a set in the ensemble, and the RO v. SH subscripts represent different fitter results. This is the average "number of $\sigma$'s" that the fitter results differ by (with the $\sigma$'s averaged). The absolute value was chosen over the square to prevent over-penalizing excursions. Since the 3-Phase group agreed that $0.5\sigma$ was the largest acceptable difference, this test was considered successful.

It is interesting to note that there is a systematic difference between the methods, particularly for BoronFlux. This difference is thought to occur due to the MCMC method integrating out parameters, as $P_0$ and BoronFlux have a very strong correlation. This tends to give a difference between a Minuit-type fitter, which returns the values at the point of maximum likelihood, and an MCMC-type process, which returns the posterior distribution.

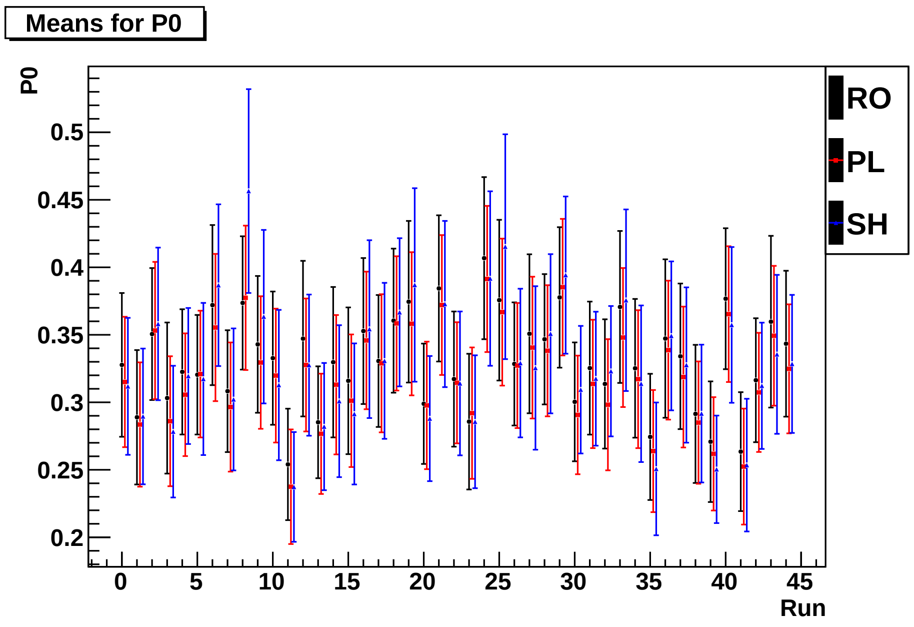


Figure 8-25: Comparison between the three SigEx's, for P0

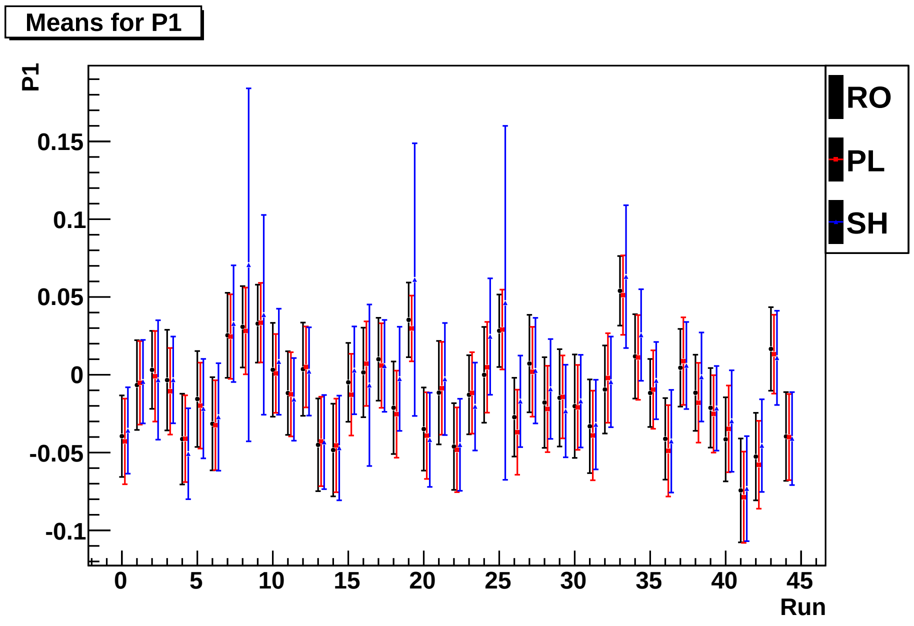


Figure 8-26: Comparison between the three SigEx's, for P1

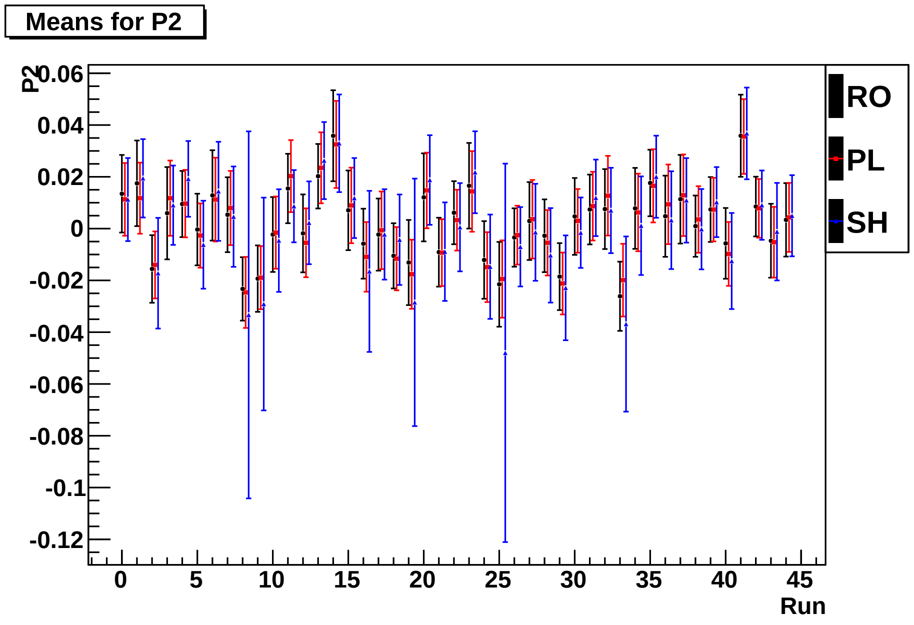


Figure 8-27: Comparison between the three SigEx's, for P2

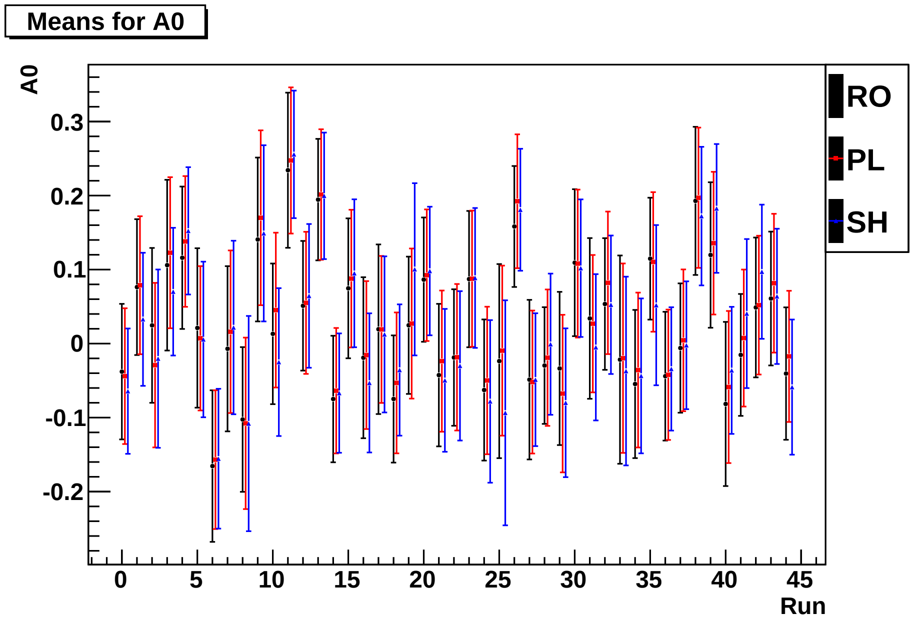


Figure 8-28: Comparison between the three SigEx's, for A0

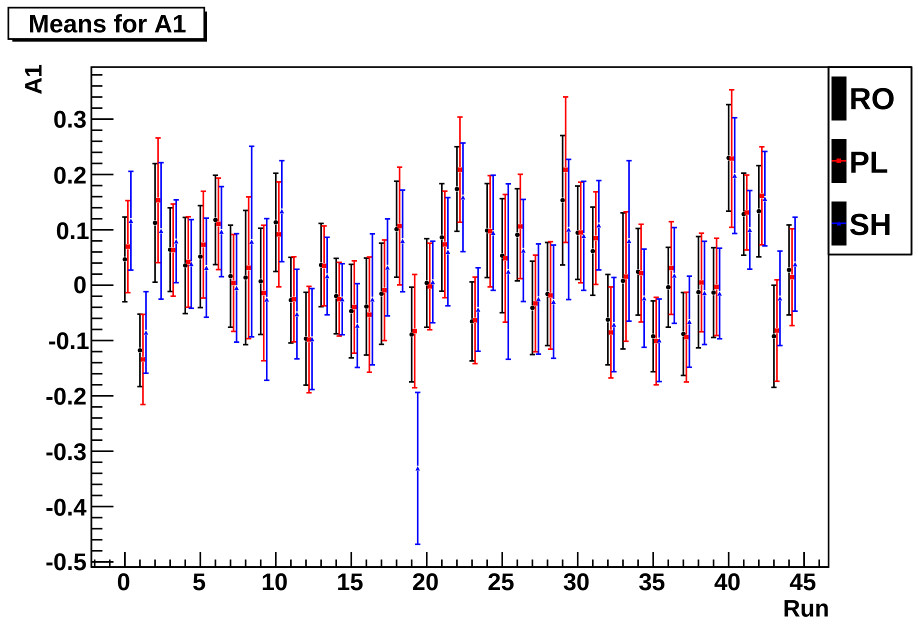


Figure 8-29: Comparison between the three SigEx's, for A1

| Normal seed | | | |
|---|---|---|---|
| Parameter | RO | PL | SH |
| BoronFlux | $0.979 \pm 0.093$ | $1.002 \pm 0.093$ | $0.987 \pm 0.093$ |
| P0 | $0.331 \pm 0.052$ | $0.321 \pm 0.049$ | $0.327 \pm 0.056$ |
| P1 | $-0.012 \pm 0.028$ | $-0.013 \pm 0.027$ | $-0.007 \pm 0.042$ |
| P2 | $0.002 \pm 0.014$ | $0.002 \pm 0.014$ | $0.000 \pm 0.025$ |
| A0 | $0.026 \pm 0.099$ | $0.034 \pm 0.099$ | $0.03 \pm 0.10$ |
| A1 | $0.025 \pm 0.087$ | $0.030 \pm 0.094$ | $0.021 \pm 0.099$ |
| Alternate seed | | | |
| BoronFlux | $0.973 \pm 0.092$ | $1.002 \pm 0.093$ | $0.982 \pm 0.094$ |
| P0 | $0.328 \pm 0.051$ | $0.328 \pm 0.050$ | $0.333 \pm 0.059$ |
| P1 | $-0.011 \pm 0.028$ | $-0.010 \pm 0.028$ | $-0.005 \pm 0.042$ |
| P2 | $-0.001 \pm 0.015$ | $-0.000 \pm 0.014$ | $-0.002 \pm 0.028$ |
| A0 | $-0.012 \pm 0.097$ | $0.014 \pm 0.098$ | $0.00 \pm 0.10$ |
| A1 | $0.003 \pm 0.092$ | $0.006 \pm 0.096$ | $0.00 \pm 0.10$ |

Table 8.5: Average value of each measured parameter for each SigEx. Uncertainties are from fitted uncertainties ($\sigma$ of fit Gaussian) rather than fitting uncertainties.

| Normal seed | | | |
|---|---|---|---|
| Parameter | RO v. PL | RO v. SH | PL v. SH |
| BoronFlux | 0.249 | 0.181 | 0.201 |
| P0 | 0.207 | 0.246 | 0.216 |
| P1 | 0.13 | 0.219 | 0.254 |
| P2 | 0.166 | 0.26 | 0.275 |
| A0 | 0.139 | 0.256 | 0.225 |
| A1 | 0.139 | 0.309 | 0.319 |
| Alternate seed | | | |
| BoronFlux | 0.312 | 0.189 | 0.221 |
| P0 | 0.198 | 0.229 | 0.244 |
| P1 | 0.281 | 0.241 | 0.303 |
| P2 | 0.244 | 0.218 | 0.28 |
| A0 | 0.345 | 0.256 | 0.273 |
| A1 | 0.275 | 0.186 | 0.249 |

Table 8.6: Average difference of each measured parameter between pairs of SigEx's

### 8.7.2 Conclusion

The analysis code has now been tested on many fronts. It was shown to not have any significant pull or bias, with or without backgrounds and systematics. It was also shown to produce the same fit results as the other two analyses performing signal extraction on our shared data sets. Since all analyses were developed independently, this indicates the extraction is being performed correctly - it is extremely unlikely all three analyses made the same mistake. Thus we decided that the analysis was ready to run on the real data.

# Chapter 9

# Results

## 9.1 1/3 Data set

SNO analyses are all blinded. The full data set is not available until the analysis is in its final state, at which point no more changes are allowed. This is a common practice in particle physics, with the aim of preventing analysts from unconsciously biasing their results. Instead, SNO relies on both an extensive amount of MC simulations and a "$\frac{1}{3}$ data set". The latter is a random selection of real data events, with one third the statistics of the full data set. This allows for a test of the analysis machinery on real data, without the danger of the final result being revealed, as the statistical uncertainties are significantly larger. Table 9.1 shows the results from running this analysis on the $\frac{1}{3}$ data set. Results both including the LETA constraint, giving the full three phase analysis, and not including this constraint, giving only the NCD phase results, are shown. The LETA results improve the uncertainties dramatically for the parameters correlated with LETA (explained in Section 8.2). Ideally, these results would be compared against the other two competing analyses. However, that would not be an apples-to-apples comparison in this case, as the other two analyses looked at the $\frac{1}{3}$ data set earlier in the process than we did. In the intervening time, several of the systematics values changed as the result of other analyses wrapping up, and it was not practical for any of the analyses to be rerun to extract comparable results. Instead, we relied on the agreement in the simulated data sets of Section 8.7.

| Parameter | with LETA | without LETA |
|---|---|---|
| BoronFlux | $0.930^{+0.046}_{-0.052}$ | $0.985^{+0.098}_{-0.099}$ |
| PMTEfficiency | $0.999^{+0.030}_{-0.028}$ | $1.001^{+0.026}_{-0.027}$ |
| NCDEfficiency | $1.010^{+0.020}_{-0.028}$ | $1.000^{+0.024}_{-0.022}$ |
| P0 | $0.303^{+0.028}_{-0.028}$ | $0.275^{+0.047}_{-0.040}$ |
| P1 | $-0.002^{+0.012}_{-0.013}$ | $-0.010^{+0.024}_{-0.026}$ |
| P2 | $(-2.1^{+4.6}_{-5.1}) \times 10^{-3}$ | $-0.005^{+0.012}_{-0.012}$ |
| A0 | $0.064^{+0.056}_{-0.055}$ | $0.086^{+0.097}_{-0.100}$ |
| A1 | $0.025^{+0.042}_{-0.048}$ | $0.068^{+0.103}_{-0.089}$ |
| ExFlux | $1.13^{+0.50}_{-0.48}$ | $1.06^{+0.48}_{-0.50}$ |
| AEx | $-0.000^{+0.011}_{-0.010}$ | $-0.001^{+0.011}_{-0.011}$ |
| D2OFlux | $1.03^{+0.14}_{-0.16}$ | $1.00^{+0.16}_{-0.15}$ |
| AD2O | $-0.01^{+0.11}_{-0.11}$ | $-0.01^{+0.12}_{-0.11}$ |
| k2pd | $1.02^{+0.15}_{-0.16}$ | $1.00^{+0.17}_{-0.17}$ |
| k5pd | $0.99^{+0.18}_{-0.17}$ | $1.00^{+0.17}_{-0.16}$ |
| ncdpd | $1.05^{+0.32}_{-0.31}$ | $1.00^{+0.37}_{-0.33}$ |
| atmos | $0.97^{+0.21}_{-0.18}$ | $0.99^{+0.20}_{-0.22}$ |
| EnergyScaleCorr | $(2.7^{+2.6}_{-2.7}) \times 10^{-3}$ | $(-0.2^{+4.5}_{-3.9}) \times 10^{-3}$ |
| EnergyScale | $(-1.9^{+8.4}_{-7.0}) \times 10^{-3}$ | $(2.7^{+8.2}_{-9.2}) \times 10^{-3}$ |
| EnergyScaleADiurnal | $(0.4^{+3.8}_{-4.0}) \times 10^{-3}$ | $(0.1^{+3.7}_{-4.0}) \times 10^{-3}$ |
| EnergyScaleADir | $(0.7^{+9.6}_{-10.}) \times 10^{-3}$ | $(0.5^{+10.}_{-9.8}) \times 10^{-3}$ |
| EnergyRes | $0.019^{+0.012}_{-0.016}$ | $0.019^{+0.014}_{-0.014}$ |
| EnergyResADir | $0.000^{+0.012}_{-0.011}$ | $0.001^{+0.011}_{-0.012}$ |
| EnergyNonLin | $(0.6^{+4.6}_{-5.0}) \times 10^{-3}$ | $(-0.2^{+7.5}_{-7.0}) \times 10^{-3}$ |
| EDepFidVol | $(-1.7^{+7.9}_{-6.0}) \times 10^{-3}$ | $(-0.6^{+8.7}_{-7.0}) \times 10^{-3}$ |
| CosThetaRes | $0.017^{+0.109}_{-0.099}$ | $0.02^{+0.12}_{-0.11}$ |
| CosThetaResADir | $0.016^{+0.052}_{-0.083}$ | $-0.011^{+0.068}_{-0.073}$ |
| XShift | $0.5^{+3.6}_{-4.3}$ | $0.7^{+3.9}_{-4.4}$ |
| YShift | $-0.7^{+4.1}_{-3.7}$ | $-0.6^{+3.9}_{-3.7}$ |
| ZShift | $4.7^{+4.5}_{-4.2}$ | $5.1^{+3.8}_{-4.2}$ |
| XYZScale | $(0.9^{+2.5}_{-6.9}) \times 10^{-3}$ | $(0.6^{+2.8}_{-7.0}) \times 10^{-3}$ |
| XYZScaleADiurnal | $(0.1^{+1.4}_{-1.6}) \times 10^{-3}$ | $(-0.2^{+1.5}_{-1.4}) \times 10^{-3}$ |
| XYZScaleADir | $(0.1^{+1.8}_{-1.9}) \times 10^{-3}$ | $(-0.1^{+1.8}_{-1.8}) \times 10^{-3}$ |
| XYResConst | $0.067^{+0.027}_{-0.030}$ | $0.065^{+0.028}_{-0.029}$ |
| XYResLin | $(-4.7^{+6.0}_{-6.3}) \times 10^{-5}$ | $(-5.0^{+5.8}_{-6.1}) \times 10^{-5}$ |
| XYResQuad | $(3.8^{+2.1}_{-1.9}) \times 10^{-7}$ | $(3.9^{+2.1}_{-1.9}) \times 10^{-7}$ |
| ZScale | $(-0.1^{+1.7}_{-1.1}) \times 10^{-3}$ | $(-0.1^{+1.7}_{-1.2}) \times 10^{-3}$ |
| ZResConst | $0.077^{+0.022}_{-0.034}$ | $0.072^{+0.027}_{-0.031}$ |
| ZResLin | $(1.09^{+0.90}_{-0.76}) \times 10^{-4}$ | $(1.10^{+0.87}_{-0.79}) \times 10^{-4}$ |
| Winter | $-0.02^{+0.73}_{-0.67}$ | $0.01^{+0.99}_{-0.97}$ |

Table 9.1: Results from $\frac{1}{3}$ data set. The "with LETA" column are results including the LETA constraint, the "without LETA" are results with no LETA constraint, i.e. results from just the NCD phase data.

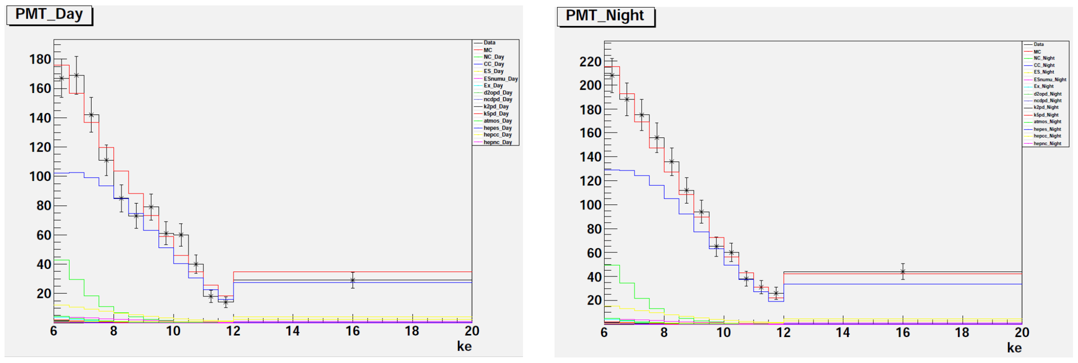


Figure 9-1: The data (shown in black) compared to the best fit results (distorted MC, in red). The components of the MC are shown, most notably CC in blue, NC in green and ES in yellow. The figure on the left is the Day portion of the fit, the right is the Night. This is the distribution in energy $E$, labeled "ke" in the graph.

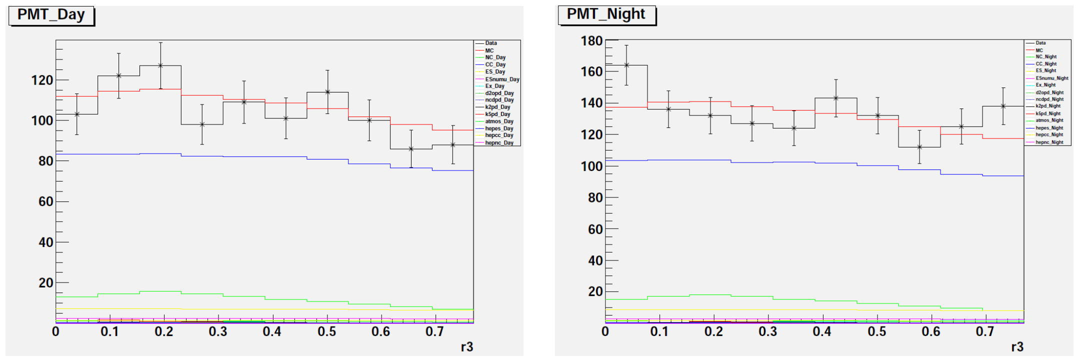


Figure 9-2: The data (shown in black) compared to the best fit results (distorted MC, in red). The components of the MC are shown, most notably CC in blue, NC in green and ES in yellow. The figure on the left is the Day portion of the fit, the right is the Night. This is the distribution in $\rho^3$, labeled "r3" in the graph.

## 9.2 Full data set

Once we were confident that the simulated data and $\frac{1}{3}$ data results made sense, we were ready to run our analysis on the final data. Due to the blind analysis, our analysis was "frozen" at this stage, so no changes to code or configuration files was allowed. Step size tuning was done on simulated data before this point. The results of the final signal extraction are shown in Table 9.2. The best fit results are compared to the data in three 1-D projections: $E$ in Figure 9-1, $\rho^3$ in Figure 9-2 and $\cos(\theta_{sun})$ in Figure 9-3. These figures are shown as an indication that the fit behaved reasonably, but as 1-D projections do not accurately reflect the fitting process.

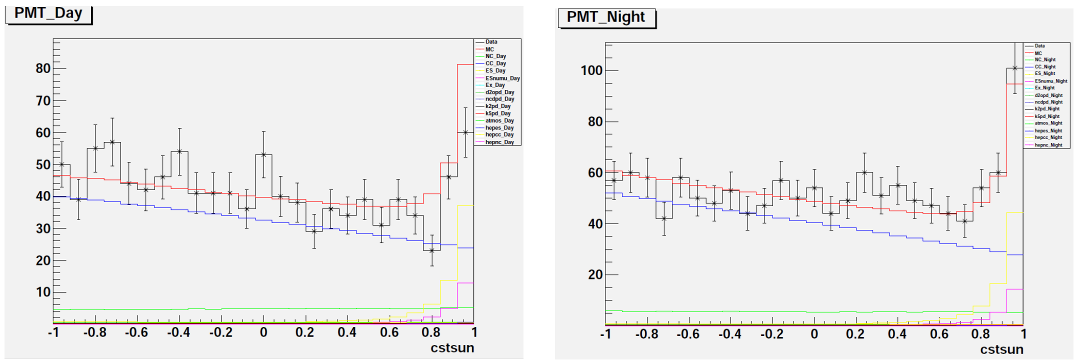


Figure 9-3: The data (shown in black) compared to the best fit results (distorted MC, in red). The components of the MC are shown, most notably CC in blue, NC in green and ES in yellow. The figure on the left is the Day portion of the fit, the right is the Night. This is the distribution in $\cos(\theta_{sun})$, labeled "cstsun" in the graph.

At this stage, the three analysis results were compared again, shown in Table 9.3. As is apparent, the agreement between the three methods is nearly perfect. Since the coordination between analyses was mostly limited to shared input parameters and Monte Carlo, we take this to mean that the analyses correctly extracted the physics parameters.

To convert the results in to more physical units, we multiply by the input $^8$B neutrino flux of $5.69 \times 10^6\ cm^{-2}s^{-1}$. This gives a measured total $\nu_x$ neutrino flux of $(5.26 \pm 0.20) \times 10^6\ cm^{-2}s^{-1}$. The $\nu_e$ flux is given by multiplying this by $P_0$, giving approximately $(1.65 \pm 0.12) \times 10^6\ cm^{-2}s^{-1}$. These are consistent with the NCD phase results in [57]. The day/night asymmetry is measured to be $0.044^{+0.037}_{-0.031} + (-0.018 \pm 0.028)(E_\nu - 10)$, where $E_\nu$ is in MeV. This is consistent with zero, and all earlier measurements, but is an improvement in uncertainty over earlier phases of SNO. The cumulative results of the measurements of the total neutrino flux and the Day/Night asymmetry (in the $\nu_e$ flux) are shown in Table 9.4 and graphically in Figures 9-4 and 9-5. The values for asymmetry in table and graph are only the constant term $A_0$, as only LETA and this analysis allowed for energy dependence.

| Parameter | Value |
|---|---|
| $^8$B Scale | $0.925 \pm 0.035$ |
| $P_0$ | $0.314^{+0.020}_{-0.018}$ |
| $P_1$ | $(5.9 \pm 8.4) \times 10^{-3}$ |
| $P_2$ | $(-1.0 \pm 3.3) \times 10^{-3}$ |
| $A_0$ | $0.044^{+0.037}_{-0.031}$ |
| $A_1$ | $-0.018 \pm 0.028$ |
| PMTEfficiency | $0.997 \pm 0.027$ |
| NCDEfficiency | $0.999 \pm 0.023$ |
| ExFlux | $0.89 \pm 0.50$ |
| AEx | $-0.000 \pm 0.011$ |
| D2OFlux | $1.02 \pm 0.17$ |
| AD2O | $0.01 \pm 0.12$ |
| k2pd Scale | $0.99 \pm 0.16$ |
| k5pd Scale | $1.00 \pm 0.17$ |
| ncdpd Scale | $0.99^{+0.34}_{-0.31}$ |
| atmos Scale | $0.97 \pm 0.19$ |
| EnergyScaleCorr | $(-0.5 \pm 2.7) \times 10^{-3}$ |
| EnergyScale | $(3.5 \pm 9.0) \times 10^{-3}$ |
| EnergyScaleADiurnal | $(1.1 \pm 3.8) \times 10^{-3}$ |
| EnergyScaleADir | $(2.6^{+9.9}_{-8.9}) \times 10^{-3}$ |
| EnergyRes | $0.016 \pm 0.013$ |
| EnergyResADir | $0.000 \pm 0.012$ |
| EnergyNonLin | $(0.6 \pm 5.0) \times 10^{-3}$ |
| EDepFidVol | $(-3.8^{+6.4}_{-5.5}) \times 10^{-3}$ |
| CosThetaRes | $0.06 \pm 0.11$ |
| CosThetaResADir | $-0.014 \pm 0.069$ |
| XShift | $0.9 \pm 4.2$ |
| YShift | $-1.6^{+3.9}_{-3.3}$ |
| ZShift | $4.9^{+4.2}_{-3.7}$ |
| XYZScale | $(0.7 \pm 5.5) \times 10^{-3}$ |
| XYZScaleADiurnal | $(-0.1 \pm 1.4) \times 10^{-3}$ |
| XYZScaleADir | $(-0.1 \pm 1.8) \times 10^{-3}$ |
| XYResConst | $0.072 \pm 0.031$ |
| XYResLin | $(-5.4 \pm 6.3) \times 10^{-5}$ |
| XYResQuad | $(3.6 \pm 2.1) \times 10^{-7}$ |
| ZScale | $(0.0^{+1.6}_{-1.2}) \times 10^{-3}$ |
| ZResConst | $0.073 \pm 0.028$ |
| ZResLin | $(1.13^{+0.88}_{-0.74}) \times 10^{-4}$ |
| Winter | $0.04 \pm 0.77$ |

Table 9.2: Results from the signal extraction for the full, final data from SNO.

| Parameter | RO | PL | SH |
|---|---|---|---|
| $^8$B Scale | $0.925 \pm 0.035$ | $0.921 \pm 0.035$ | $0.921 \pm 0.036$ |
| $P_0$ | $0.314^{+0.020}_{-0.018}$ | $0.319 \pm 0.018$ | $0.321 \pm 0.020$ |
| $P_1$ | $(5.9 \pm 8.4) \times 10^{-3}$ | $0.002 \pm 0.008$ | $0.005 \pm 0.008$ |
| $P_2$ | $(-1.0 \pm 3.3) \times 10^{-3}$ | $-0.001 \pm 0.003$ | $-0.002 \pm 0.003$ |
| $A_0$ | $0.044^{+0.037}_{-0.031}$ | $0.044 \pm 0.034$ | $0.048 \pm 0.035$ |
| $A_1$ | $-0.018 \pm 0.028$ | $-0.017 \pm 0.027$ | $-0.015 \pm 0.028$ |

Table 9.3: Comparison between final results for all three signal extraction methods. "RO" is this analysis, "PL" is QSigEx by Pierre-Luc Drouin and "SH" is UASigEx by Shahnoor Habib. Details of the other analyses are available in [49]. Only the primary signal results are shown. The agreement between the analyses is excellent. Note that the values for PL and SH are as of July 1, 2011, after box opening.

| Phase | $^8$B | $A_0$ |
|---|---|---|
| $D_2O$ | $6.42^{+1.57}_{-1.57}{}^{+0.55}_{-0.58}$ | $0.070 \pm 0.049^{+0.013}_{-0.012}$ |
| Salt | $4.94^{+0.21}_{-0.21}{}^{+0.38}_{-0.34}$ | $0.037 \pm 0.040$ |
| LETA | $5.046^{+0.159}_{-0.152}{}^{+0.107}_{-0.123}$ | $0.027 \pm 0.043$ |
| NCD | $5.54^{+0.33}_{-0.31}{}^{+0.36}_{-0.34}$ | N/A |
| 3-Phase | $5.26 \pm 0.20$ | $0.044^{+0.037}_{-0.031}$ |

Table 9.4: The extracted $^8$B total neutrino flux and Day/Night asymmetry in $\nu_e$ flux for each phase of SNO. $\nu_e$ flux is not shown due to differences introduced by spectral distortions making numbers not directly comparable. Note that only the constant part of the asymmetry is shown, and that the asymmetry for NCD was subsumed into the 3-Phase analysis. Units are $10^6$ cm$^{-2}$ s$^{-1}$ for $^8$B, none for asymmetry (it is a ratio).

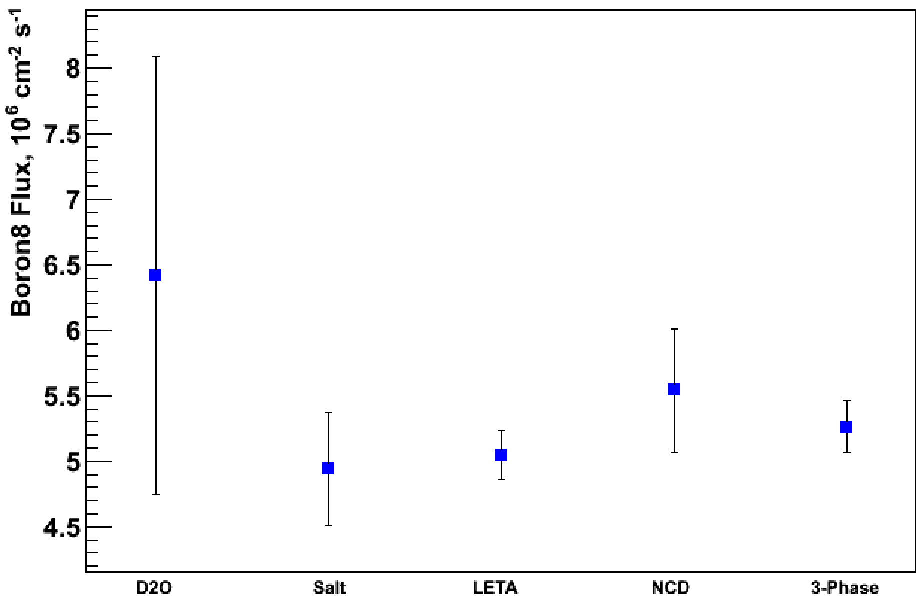


Figure 9-4: The measured total (all flavor) $^8$B neutrino flux from the sun for each phase of SNO.

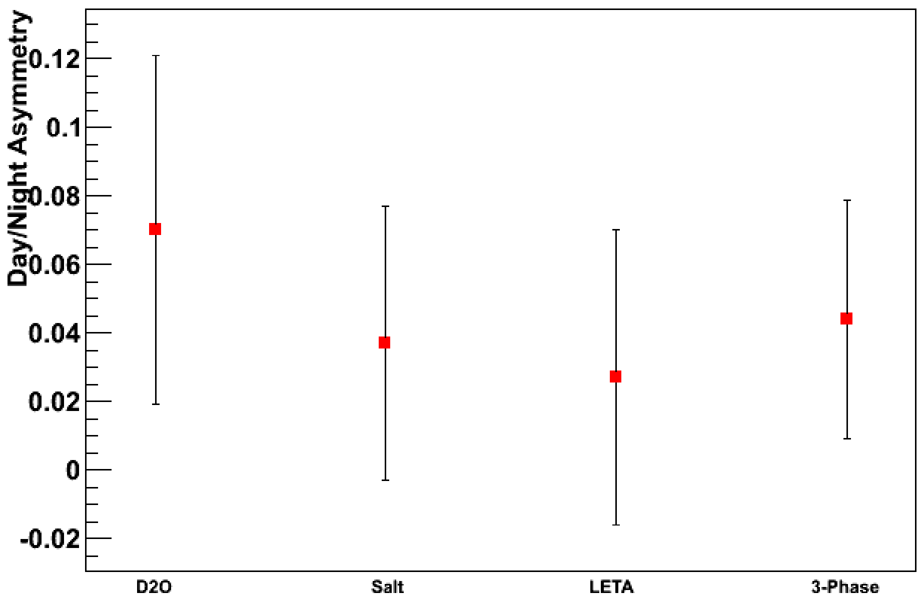
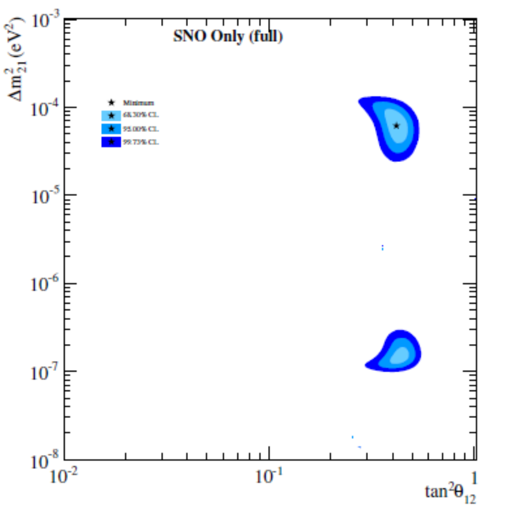
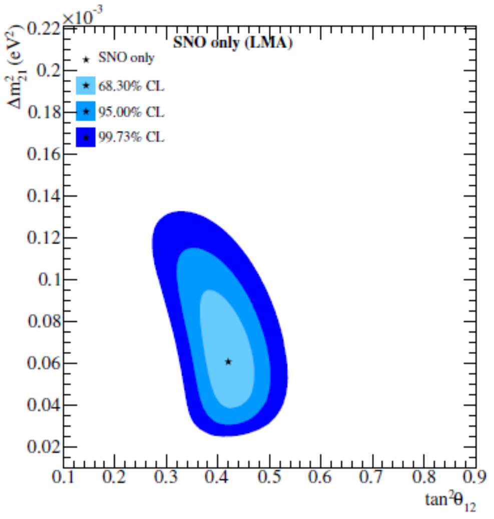


Figure 9-5: The measured day/night asymmetry for each phase of SNO. Note that $A_0$ is used for the LETA and 3-Phase measurements, as the previous phases did not allow for energy dependence.



(a) SNO-only two-flavour oscillation parameter analysis



(b) Detail of SNO-only LMA result.

Figure 9-6: $\tan^2(\theta_{21})$ v. $\Delta m^2_{21}$ contours for SNO 3-Phase results. This does not include external measurements from other experiments, which remove the lower contours on the left graph. Taken from [81].

## 9.3 Physics Interpretation

For the reasons outlined in Chapter 2, neutrino oscillations and the MSW effect convert $\nu_e$ generated in the sun in to the measured proportions of $\nu_e$ and $\nu_{\mu,\tau}$. We can approximate the oscillations with the two neutrino model, so that the physical parameters that drive this process are the mass difference $\Delta m^2_{21}$ and the mixing angle $\theta_{21}$. These are the physical parameters that SNO is designed to measure. Converting the extracted signal parameters ($^8$B flux, $P_{ee}$ and $A_{P_{ee}}$) to these physics parameters is not simple, however. The MSW effect aspect requires modeling the neutrino's passage both through the sun and the Earth, and must be done numerically. As this is so complicated, an entire subgroup of SNO, called PhysInt, is dedicated to this task. We do not attempt to reproduce this analysis, instead we quote their results [81]. The graph of $\tan^2(\theta_{21})$ v. $\Delta m^2_{21}$, using only the results from SNO's 3-Phase analysis, is shown in Figure 9-6. The best fit values are $\tan^2(\theta_{21}) = 0.420^{+0.034}_{-0.0.36}$ and $\Delta m^2_{21} = 6.06^{+2.20}_{-1.76} \times 10^{-5}\text{eV}^2$.

# Chapter 10

# Conclusion

We have measured the Day/Night effect at SNO, using the data from all three phases. This is, in many ways, the final result of the entire SNO project, as it uses all of the solar neutrino data measured by SNO. In doing this, we have reduced the uncertainties on the measured values of the total neutrino flux from the sun and the day/night asymmetry. In addition, we have attempted to measure the energy dependence of both the survival probabilty and the day/night asymmetry, but have found both to be consistent with zero. We are confident that we have extracted these values correctly, as two other analyses within SNO perform the same extraction independently and arrive at the same results.

From these refined measurements, SNO has improved its contribution to the measurements of the neutrino mixing parameters $\Delta m^2_{21}$ and $\tan^2(\theta_{12})$. Understanding the values of these parameters is very important to many upcoming experiments, as most less well understood neutrino properties can only be measured if the oscillation parameters are well understood. In addition, by improving our understanding of the neutrino we press forward in our quest to understand the particles that make up our universe and how they behave.

# Appendix A

# Code Usage for rottSigEx

## A.1 Introduction

For the NCD phase of SNO, the first signal extraction used a Markov Chain Monte Carlo (MCMC) method [63], then new to the SNO project. For the Day/Night aspect of the NCD phase, we decided to use this method again, as it has many benefits. It properly integrates "nuisance parameters" (systematics), is able to handle a large number of parameters and is "embarrassingly parallel", i.e. very easy to run across multiple computers simultaneously. With so many parameters (more than 50 for the full data set with all systematics and backgrounds), this is a crucial feature.

While we considered just taking the code used in [63] and updating it to suit the needs of this project, we instead decided to re-write everything from scratch. Two main reasons drove us to this decision: we wanted code that would be more object-oriented and flexible, and we wanted code that was independent of the other, competing signal extractions for the NCD day/night (one of which did take the code from [63] and update it).

At its core, rottSigEx runs the Metropolis algorithm, discussed in Chapters 7 and 8. The likelihood function evaluated is the Extended Likelihood, with the "parent" pdf that the data is compared against generated by binning MC simulation events. Some of the fit parameters are systematics, which are applied to the MC events at the time of binning, so the resultant histogram is rebuilt at each step in the Metropolis algorithm. The output is the posterior distribution for each of the parameters. This can then be fit if one chooses; a program to fit gaussians to the results is supplied with the main program.

## A.2 What's in the box

In its current incarnation, rottSigEx compiles five executables:

- `rottSigEx.exe` is the principal program. This actually runs the MCMC
- `metaConfig.exe` is a conversion program to translate from the "meta" config file format (more human readable) to the config file format used by `rottSigEx.exe`
- `autoFit.exe` fits Gaussians (with some options) to the results of a run of `rottSigEx.exe`
- `getAutoCorr.exe` computes the autocorrelation of each parameter in the MCMC results, useful for finding optimal step sizes
- `drawResults.exe` creates a set of plots showing binned data v. binned MC for the parameters its given, useful for checking reasonableness of results

Of these, only `rottSigEx.exe` is actually *necessary*. We highly recommend using `metaConfig.exe` as well, as the "native" config file format used by `rottSigEx.exe` is very error-prone when written by hand. The other three are post-processing utilities, included for convenience.

The code itself is C++ and compiles against CERN's Root, available at `http://root.cern.ch` [82]. If Root is installed properly, simply running `make` should be sufficient to build the program. Unfortunately, due to speed requirements it may be necessary for the user to add systematic functions, which must be present at compile time (i.e. the code itself must be edited). This should be a minimal change, and is detailed later.

## A.3 rottSigEx.exe

This section details the usage of the main program. This description assumes that you are writing meta files rather than directly writing config files. In addition, it assumes a basic working knowledge of the Metropolis algorithm and the MC events $\rightarrow$ pdf process. Since the code is still a work-in-progress, I'll strive to highlight potential pitfalls and known places where the error handling may not catch problems.

### A.3.1 Objects

This term has two meanings here: objects in the sense of "Object-Oriented" (i.e. C++ classes) and in the sense of things in the program you can manipulate. Here we discuss the latter. The code has several "layers" of objects.

At the top is the MCMC itself. It keeps track of the MCMC status (step, parameter values, likelihood value), the "pure" parameters (numbers that are varied, but not associated with any other object; they are independent objects in this description), and computes the Likelihood functions not associated with another object (these are called LogLikelihoodFormulas, and are also independent objects). The MCMC also handles I/O. Options can be passed to the MCMC directly to control its behavior, called directives.

The MCMC contains a number of Pdfs, each of which serves one function: it takes a list of the current parameter values, and hands back an Extended Likelihood. These are then summed and added to those computed by the MCMC. Each Pdf corresponds to and keeps track of a data set $\{\vec{x}\}$; the dimension of events $\vec{x}_i$ determines the dimension of the Pdf. Currently, 1-D and 3-D Pdfs are available. The binned pdf created from the MC data is also stored here and has the same number of dimensions as the data. Each dimesion has an Axis, which is an object in its own right. In addition, the Pdf needs to know what branches are present in the MC events. If multiple MC sets are used, they all need to have those branches. This may require the creation of dummy branches.

Each Pdf contains a number of Fluxes and Systematics, and a number of Axes equal to Pdfs dimension. Each Flux keeps track of a set of MC events and its normalization. Each Systematic is a function applied to a subset of the Fluxes on an event-by-event basis. These are the "lowest level", containing no further (user-accessible) parts.

### A.3.2 Parameters, Names and Keys

At each step in the MCMC, each element of the set of parameters $\vec{\alpha}$ is incremented. This is still true for the code implementation, though a step width of zero (no step) is allowed. However, $\vec{\alpha}$ can be more complicated here - it is, in some sense, just a collection of real numbers that are varied, that the user can combine in almost any way he or she sees fit.

Parameters can be created explicitly as "pure" parameters (see section A.3.7), i.e. parameters not associated with anything. Additionally, whenever a Flux, Systematic or Background is created, a parameter with the same name is created automatically. For Fluxes these automatic parameters have a special meaning (see sections A.3.9 for details). These parameters can be combined using the AsymmFunc system described in section A.3.11. Since the program keeps track of everything by its Name, each parameter must have a unique Name, including those that are automatically generated. So each Flux, Sys, Bkgd and Parameter must be uniquely named.

In the meta file, everything is described by key pairs (which become unique when expanded in to the Config file). Except for `new`, every command in the meta file must be of the form `Key=Value` or `Key = Value` (these spaces are ignored), where `Value` is either an integer, a floating point number (read in as a `Double_t`) or a string (which cannot have spaces in it); which one it is depends on the `Key` being used. See sec A.3.4 examples of use, and the appropriate section below for valid Keys for a particular object.

### A.3.3 Program behavior

The program begins by reading in the config file and using that information to create and set up all the appropriate objects, and set up the output file and TTree. It then initializes all parameters and computes the Log Likelhood (LL) for these initial parameter values, to create the "step zero" starting point for the MCMC.

The code then runs the Metropolis algorithm. At a given step, the code:

1. Creates a proposed value for each parameter by adding to it a draw from a Gaussian of mean zero and width specified by the user (constant throughout the program), giving $\tilde{\vec{\alpha}}$

2. Computes any parameters whose values are altered by an AsymmFunc

3. Computes the LL contribution from any LogLikelihoodFormulas and checks to see if any pure Parameters are outside their minimum or maximum (if specified), and computes their contraints' contributions to the LL

4. Gives a Pdf the current list of parameter values and asks it for its contribution to the LL. It does this for each Pdf in order of appearance in the meta file (this order shouldn't matter). Each Pdf:

   (a) Checks to see if any parameters it "owns" are out of their bounds

   (b) Rebuilds the MC pdf by:

      i. Emptying the histogram

      ii. Taking an MC event from a Flux

      iii. Passing this MC event through each Sys that affects it, in the order they appear in the meta file (order may matter here), giving the new value for the event

      iv. Filling this event into the histogram (only using those values which correspond to Pdf dimensions), with a weight of `InternalUseWeight*FluxName/TimesExpected` (see A.3.9)

      v. Repeating this for each event in the Flux

vi. Repeating this for each Flux (in the order of appearance, though order shouldn't matter)

vii. Checking that no bins are zero or negative, and setting any that are to a nominal value (currently 1e-10)

viii. Dividing each histogram bin by that bin's volume (width, area, etc) to "normalize" it to a proper un-normalized pdf

(c) Takes each data point $\vec{x}_i$, asks the MC pdf for its value at that point to get $p(\vec{x}_i|\tilde{\vec{\alpha}})$ and computes the ELL

(d) Computes the LL contributions from constraints on any parameters it "owns"

(e) Returns this LL

5. Compares computed LL with LL from previous step to decide if to accept step (see Chapter 7)
6. If step is accepted, updates parameter list, if not, discards proposed parameters
7. Records current parameter list in TTree
8. Every 100th step, AutoSaves TTree
9. After all steps are taken, saves TTree to output file and exits

### A.3.4 Basic meta file

The only control we have over the process in section A.3.3 is in setting up the system. Via the mechanism of the config file, we can create parameters, set step sizes, control the behvior of the MCMC to a certain degree, set up AsymmFuncs and LogLikelihoodFormulas, command the creation of any combination of objects, etc. But then the code runs its prescribed process. The config files themselves must have unique keys for each command, accomplished via a numbering system. This turned out to be difficult to keep track of, since mis-numbering caused odd behavior, so I created the much more readable meta files. Running `metaConfig.exe` converts from meta file to config file. Currently, the code only accepts config files.

Every meta file consists of some number of directives to the MCMC, to control how many steps to take, where to save files, etc; the creation of at least one Pdf, and at least on Flux inside that Pdf, plus a number of Axis objects equal to the dimension of the Pdf. Most also have some Systematics as well, and a few pure Parameters. An example of a simple metafile is:

```
MCMC_ChainLength=5000
MCMC_PrintFrequency=100
OutputFilename=results_test.root

new Pdf
Name=SimpleTest
Dimension=1
MCBranch=Energy
DataFile=fakeData.root
DataTree=data

new Axis
Name=Energy
Bin=0
Bin=0.1
Bin=0.3
Bin=0.7
Bin=1

new Sys
Name=AddConst
Init=0.1
Width=0.01
```

```
Mean=0.1
Sigma=0.05

new Flux
Name=Sim
File=fakeMC.root
Tree=mc
TimesExpected=5
Min=0
Init=1
Width=0.01
```

In general, a new object is created with the `new` command. `metaConfig.exe` knows about the heirarchy of objects, so any Axis, Flux, etc. created will go in the Pdf most recently created. Between one invocation of `new` and the next, commands apply to the "active" object. So `Name=AddConst` above gives the Systematic the name "AddConst". Note that commands to the MCMC directly and the creation of pure Parameters can occur anywhere and ignore heirarchy rules.

Working through the sample file one line at a time (skipping repeats):

`MCMC_ChainLength=5000` tells the MCMC to take 5000 total steps

`MCMC_PrintFrequency=100` tells the MCMC to print its status to stdout at every 100th step, useful if something goes wrong

`OutputFilename=results_test.root` sets the output file. This defaults to the current directory. This file will contain a single TTree named Tmcmc, this can be overridden (see A.3.6)

`new Pdf` creates a new Pdf, at this stage with undefined characteristics

`Name=SimpleTest` gives the Pdf the name "SimpleTest"

`Dimension=1` sets SimpleTest to be a 1-D Pdf

`MCBranch=Energy` tells SimpleTest that its Fluxes will contain the branch Energy. If we wanted more branches (for example, if there was a branch “TrueEnergy” that was the input simulation energy, useful for altering resolutions), we just add an additional line `MCBranch=TrueEnergy`

`DataFile=fakeData.root` tells SimpleTest that its data is located in the file fakeData.root

`DataTree=data` tells SimpleTest to look for a TTree named data for its data

`Bin=0` Tells the Energy Axis the location of bin boundaries. In this case, it has 4 bins, $[0, 0.1)$, $[0.1, 0.3)$, $[0.3, 0.7)$ and $[0.7, 1]$

`Init=0.1` tells the MCMC that the parameter AddConst takes the value 0.1 initially, before stepping starts

`Width=0.01` tells the MCMC that the parameter AddConst should be incremented by a Gaussian draw with $\sigma = 0.01$ at each step

`Mean=0.1` and `Sigma=0.05` tell the pdf SimpleTest that the parameter AddConst has a Gaussian constraint of mean 0.1 and $\sigma = 0.05$, which is added as a penalty term to the Likelihood computed by SimpleTest

`File=fakeMC.root` tells the Flux Sim that the MC events for it are in the file fakeMC.root

`Tree=mc` tells Sim to look for a TTree named mc for its events

`TimesExpected=5` tells Sim that it has 5 times as many MC events as it expects for data events, i.e. its simulation had “five experiments” worth of simulated events. This acts as a normalization, more on that in A.3.9

`Min=0` tells the MCMC that the parameter Sim has minimum 0. If it is ever below this value, it returns Likelihood zero (i.e. it cannot take this step). Since we’re using Log Likelihoods here, it actually returns a very large negative value (but not negative infinity)

### A.3.5 new

As mentioned, the `new` command creates a new object. Valid objects to create are: `Pdf`, `Sys`, `Flux`, `Axis`, `Parameter` and `LogLikelihoodFormula`. They must obey the heirarchy: to create a Flux, Sys, or Axis, a Pdf must already exist for it to "live in". At any given time, only the most recently created object is "active" and any commands given apply to it. MCMC Directives ignore this rule and can be anywhere in the file, but I've found it easiest to just include all of them at the beginning of the file to prevent confusion later.

### A.3.6 MCMC Directives

A number of commands can be given to the MCMC to change its behavior. A few of these are necessary and showed up in the sample file. If a command isn't present, a default value is assumed, listed with that command below. A full listing of these commands and their consequences is:

`MCMC_ChainLength=integer` sets the total length of the chain. DynamicSteps are taken first, then normal steps. If this number is less than `MCMC_DynamicSteps`, it is ignored. If it isn't present, the MCMC defaults to 1000 steps.

`MCMC_DynamicSteps=integer` tells the MCMC to take steps where the step size is allowed to vary, with the goal that the chain take 23.5% steps. This was implemented for testing and is described in Section 7.5. Default is 0.

`MCMC_PrintFrequency=integer` controls how often the MCMC prints its status to the screen, setting it to print every $n^{\text{th}}$ step. Useful for making sure the chain is still going and checking on its status as it is running. Default is 1 (i.e. every step).

`MCMC_SkipSteps=integer` originally was going to control how many "burn-in" period steps to skip when making graphs, but all of the graphing programs take that input separately and ignore this now. Defaults to 0.

`MCMC_UseAsymmetry=true/false` tells the MCMC whether to check for AsymmFuncs. It *must* be set to `true` if AsymmFuncs are being used. See section A.3.11 for more details. Default is `false`.

`MCMC_SaveProposed=true/false` tells the MCMC whether to save the proposed values for each step as well as the accepted. These are the values $\tilde{\vec{\alpha}}$ generated by varying the parameters. If the step is rejected, these values are lost if this set to `false`, but the output files are half the size. Useful for debugging. Default is `true`.

`MCMC_SaveUnvaried=true/false` tells the MCMC whether to save the value of parameters with width $\leq 0$, i.e. those that aren't varied by the MCMC. Often useful for debugging. Setting this to `false` gives smaller output files. Default is `true`.

`RandomSeed=integer` sets the seed for the TRandom3 that generates the random numbers needed throughout the MCMC process. Default is 0, i.e. a random seed based on the clock time, among other things.

`OutputDirectory=string` sets the directory to which MCMC writes output files. Default is `.` (current directory).

`OutputFilename=string` sets the output filename. It is concatenated with the OutputDirectory internally, so putting the directory string as part of the file name has the same result as setting `OutputDirectory`. Default is `results.root`.

`OutputTreeName=string` sets the name of the TTree created in the output file. Default is `Tmcmc`.

### A.3.7 Parameter

A "pure" parameter is one not associated with another object. It is simply a number that is varied when the parameters are varied by the MCMC. It can be altered with an AsymmFunc. To create one, use `new Parameter`. Required settings are `Name`, `Init` and `Width`. `Name` must be unique. If `Width` is $\leq 0$, this Parameter will not be varied (though AsymmFunc can still change its value). Valid Keys are:

`Name=string` sets the name. Must be unique. If this isn't present, an error message is generated and setup fails.

`Init=double` sets the initial value. Error and setup failure if not present.

`Width=double` sets $\sigma$ for the Gaussian drawn from at the incrementing step for this variable. If `Width` is $\leq 0$, parameter is not incremented. Error and setup failure if not present.

`Mean=double` creates a Gaussian constraint for this parameter. Sets $\mu$ of this constraint to `Mean`. If this is present and `Sigma` is not, an error is generated and setup fails. If neither is present, no constraint. See `Sigma` for more details.

`Sigma=double` creates a Gaussian constraint for this parameter. Sets $\sigma$ of this constraint to `Sigma`. This constraint gives LL contribution $-\frac{(\alpha-\mu)^2}{2\sigma^2}$. If this is present and `Mean` is not, an error is generated and setup fails. If neither is present, no constraint.

`Min=double` sets the minimum value for the parameter. If its value goes below `Min`, the returned LL is $-1e200$, effectively $-\inf$, so no step is taken. If this is not present, no minimum.

`Max=double` sets the maximum value for the parameter. If its value goes above `Max`, the returned LL is $-1e200$, effectively $-\inf$, so no step is taken. If this is not present, no maximum.

`AsymmFunc=string` creates an Asymm function for this parameter. See section A.3.11. If this is not present, no AsymmFunc.

`AsymmPar=string` sets a parameter for the Asymm Func. See section A.3.11. If this is present and `AsymmFunc` isn't, an error is returned and setup fails.

### A.3.8 Pdf

A Pdf is a basic building block of the system. It contains a data set and the corresponding histogram for the binned probability distribution for that data. Its dimension must be specified and a number of Axis objects equal to that dimension must be created. The data must be stored in a TTree in a `.root` file, and only the branches that have the same name as an Axis will be used. All events in the data set *must* be in the range of the bins defined in the Axis. If this isn't the case, the behavior is undefined (and is likely to be a program crash, or at least very strange behavior), since this will involve requesting binned pdf values outside the range where they are defined. Any number of `MCBranches` can be defined, these describe the branches that the MC events in the various Fluxes will have. This will cause all Fluxes to have the same branches, and only those branches will be used. Each Axis name must have a corresponding MCBranch and each set of MC events must have branches with the same names as the Axis objects. Valid Keys are:

`Name=string` sets the name. Must be unique. If this isn't present, an error message is generated and setup fails.

`Dimension=integer` sets the dimension of the Pdf. A number of Axis variables equal to this `Dimension` must be created in the Pdf. Only dimensions 1 and 3 are currently available. If this isn't present, an error message is generated and setup fails.

`DataFile=string` sets the location of the file containing the data events. Can include a directory string, assumes current working directory otherwise. The file must contain the TTree named in `DataTree`. If this isn't present, an error message is generated and setup fails.

`DataTree=string` sets the name of the TTree containing the data events, in the file `DataFile`. Must contain branches with the same names as the Axis objects. If this isn't present, an error message is generated and setup fails.

`MCBranch=string` tells the Pdf to look for a branch of this name in the TTrees containing the MC events in the Flux objects. To have the Pdf look for more than one branch (necessary for 3D Pdfs, useful for any Pdf), simply call this for each branch. They will be numbered (in the order called) internally. Must be called at least a number of times equal to the `Dimension` of the Pdf.

### A.3.9 Flux

A Flux object must live in a Pdf object, so a Pdf must be created (with `new`) before a Flux is created. Each Flux contains one set of MC events. When this is created, a parameter with the same `Name` as the Flux is created. During the filling of the MC binned pdf, the Flux hands events to the Pdf, and they are filled in to the Pdf with weight `Name`*`InternalUseWeight/TimesExpected`. `InternalUseWeight` is an internally created MC branch that is used for reweighting procedures, see section A.3.10 for more details. The `TimesExpected` nominally is used to keep track of how many "experiments worth" of MC you have, i.e. if the MC contains 500 times the events you expect for that particular signal in your data, a `TimesExpected` of 500 gives a nominal value of 1 to the automatic parameter. Another option is `TimesExpected` set to the number of events in your analysis window, then the nominal value of the automatic variable is the number of data events. These MC events are not restricted to being in the range specified by the Axis objects; events outside the range are ignored. This is imporant for any Systematic that changes the effective analysis window (say by adding a constant to the energy or position) - if this happens, there need to be MC events in the new region or the log likelihood will not give the correct value. Fluxes can also have a `FluxNumber` assigned, which is a distinguishing feature used by the Sys to decide which Flux to act on (so if you have a systematic that you want to act on this Flux and not others, give it a distinct `FluxNumber`). Any number of Pdfs can share the same `FluxNumber`. Valid Keys are:

`Name=string` sets the name. Must be unique. If this isn't present, an error message is generated and setup fails.

`Init=double` sets the initial value of the automatic parameter. Error and setup failure if not present.

`Width=double` sets $\sigma$ for the Gaussian drawn from at the incrementing step for the automatic parameter. If `Width` is $\leq 0$, parameter is not incremented. Error and setup failure if not present.

`Mean=double` creates a Gaussian constraint for the automatic parameter. Sets $\mu$ of this constraint to `Mean`. If this is present and `Sigma` is not, an error is generated and setup fails. If neither is present, no constraint. See `Sigma` for more details.

`Sigma=double` creates a Gaussian constraint for the automatic parameter. Sets $\sigma$ of this constraint to `Sigma`. This constraint gives LL contribution $-\frac{(\alpha-\mu)^2}{2\sigma^2}$. If this is present and `Mean` is not, an error is generated and setup fails. If neither is present, no constraint.

`Min=double` sets the minimum value for the automatic parameter. If its value goes below `Min`, the returned LL is $-1e200$, effectively $-\inf$, so no step is taken. If this is not present, no minimum.

`Max=double` sets the maximum value for the automatic parameter. If its value goes above `Max`, the returned LL is $-1e200$, effectively $-\inf$, so no step is taken. If this is not present, no maximum.

`AsymmFunc=string` creates an Asymm function for the automatic parameter. See section A.3.11. If this is not present, no AsymmFunc.

`AsymmPar=string` sets a parameter for the Asymm Func. See section A.3.11. If this is present and `AsymmFunc` isn't, an error is returned and setup fails. `File` sets the location of the file containing the MC events. Can include a directory string, assumes current working directory otherwise. The file must contain the TTree named in `Tree`. If this isn't present, an error message is generated and setup fails.

`Tree=string` sets the name of the TTree containing the MC events, in the file `File`. Must contain branches with the same names as the MCBranches of the Pdf this Flux is in. If this isn't present, an error message is generated and setup fails.

`TimesExpected=double` weights each MC event when the binned pdf is filled with (automatic variable)*`InternalUseWeight/TimesExpected`. If this isn't present, an error message is generated and setup fails.

`FluxNumber=integer` sets the `FluxNumber` of the Flux. This allows for Sys to be selective in which Fluxes they act on, as one of the settings for Sys is which FluxNumbers to affect. Fluxes can share the same Flux number. Default is -1.

### A.3.10 Sys

The Sys are functions that are applied to the MC events before they are filled in to the binned pdf. They can be selectively applied to some Fluxes and not others, and only apply to Fluxes in the Pdf in which the Sys object resides. A Sys can use any number of MC branches and parameters in its evaluation, but can only alter (target) one MC branch. So each Sys is a function taking some number of real values and returning a single real value. In addition to the MC Branches listed in Pdf, there is an additional, internal branch created named `InternalUseWeight`. It is set to 1, but can be targeted with the Sys to change this value. `InternalUseWeight` is directly multiplied by any other weighting for the MC events, so that it effectively acts a relative weight for MC events. Since multiple Sys may (and usually do) apply to the same Flux (and hence to the same MC event), the Sys function returns not the new value of the branch, but the change in that value due to this Sys's action, so that each Sys can see the values as they appear in the Flux. The option to look at the branch values including all changes up to that point (the Sys are applied in the order they appear in the meta file) is also available, but then the order matters. To make a Sys apply to only a select subset of the Fluxes, it can be set to only act on certain FluxNumbers. When a Sys is created, a parameter of the same name is automatically created. This parameter does not have to be used in the Sys, but must be unique as it identifies the Sys to the code.

Unfortunately, much of the Sys behavior must still be hard-coded in to the system. This is due to a combination of speed requirements (the Sys are evaluated for *every MC event*, so they are called *a lot*) and a code upgrade that never got implemented. In fact, the meta file can only control the behavior of the automatic parameter generated when Sys is created. All other control is done by directly altering the source files `Sys.cxx` and `FunctionDefs.h`. This must be done with care, as error checking inside the code is not very thorough.

First, `Sys.cxx`. In this file, there is a member function `Sys::LookupName`. In this function is a very long chain of else-if statements. This is what needs to be edited.

An example is:

```
} else if (nameToGet == "ZRemap_Day"){
  sysFunc = new ZRemap();
  found &= LookupNameChecker("ZShift",mcmcParNames,'p')
  found &= LookupNameChecker("ZScale_Day",mcmcParNames,'p');
  found &= LookupNameChecker("ZResConst",mcmcParNames,'p');
  found &= LookupNameChecker("ZResLin",mcmcParNames,'p');
  found &= LookupNameChecker("z",branchNames,'d');
  found &= LookupNameChecker("truez",branchNames,'d');
  dataTarget = SearchStringVector(branchNames,"z");
  AddFluxAffected(0);
  AddFluxAffected(1);
  AddFluxAffected(-1);
  useMultiply=false;
  useOriginalData = true;
}
```

Each one begins with `if (nameToGet == "string")`. The "string" here (`ZRemap_Day` in the example) is the name of a Sys. This is read in from the config file and matched, so only the Sys with this name follows the behavior set here, meaning that *each* Sys must have a corresponding entry in this else-if table. Part of The Upgrade That Never Was was to make this more flexible. The next line is `sysFunc = new FunctionName()`, where the FunctionName is the name of a class defined in `FunctionDefs.h`, see the next paragraph for details. The next set of lines,
`found &= LookupNameChecker("ParName",parNamesList,'type')`, sets the values that go in to the Sys function. Each invocation adds a parameter or MC branch (with name "ParName"), in the order they are called. The next entry, `mcmcParNames` or `branchNames`, selects parameters from the parameter list $\{\vec{\alpha}\}$ or from the list of MC branches, respectively. The `branchNames` list includes the automatically generated `InternalUseWeight`, used for re-weighting. The final argument, `'type'`, must

be `'p'` for mcmcParNames and `'d'` for branchNames. This doesn't affect the actual functioning of the Sys; it is used to help make the resultant error messages more clear. It is critically important that all the mcmcPars are listed before the MC branches, otherwise the Sys will behave in an ill-determined way. The next line, `dataTarget = SearchStringVector(branchNames,"name")`, selects which MC branch the Sys will alter. Only one selection can be made. Also, there is absolutely no error checking on this, so the name *must* be spelled correctly, otherwise there is likely to be a seg fault. Any branch, plus `InternalUseWeight`, can be targeted. The `AddFluxAffected` function tells the Sys which FluxNumbers it should affect. Each call adds a FluxNumber to affect to the Sys's list. If this isn't called at all, it defaults to affecting all FluxNumbers. Note that a flux with no FluxNumber specified has FluxNumber $-1$. The line `useMultiply` tells the Sys to add its value to the target branch (if `false`) or multiply the target branch by the Sys's value (if `true`). The latter is necessary when re-weighting, as the default weight is 1. The line `useOriginalData` tells the Sys to use the values the MC event's branches had before any other Sys has been applied (if `true`), or after all Sys *up to that point* have been applied (if `false`). This latter point is important: the order matters. Sys are called in the order they are listed in the meta file.

In `FunctionDefs.h`, the functional form of the Sys is specified. This is done by creating a class that inherits from the class `RealFunction` (defined in `RealFunction.h` and `RealFunction.cxx`). This function needs to know how many parameters it is looking for, which must match the number listed in the corresponding selection in `Sys.cxx`. Note that a function defined here is not restricted to a single Sys entry; if there are multiple Sys that use the same function form, it is best to use a single function for them. An example definition is:

```
class MultiplyConst : public RealFunction {
 public:
  MultiplyConst() {
    nPars = 2;
    parameters = new Double_t[nPars];
  }

  Double_t Eval(const Double_t x) {
    return (parameters[0]*parameters[1]);
  }
};
```

Only two things need to be defined: the constructor and the function `Eval`. The `nPars` term defines how many parameters the function takes. The function `Eval` actually defines the systematic function. This can call any function in `TMath` (from Root) and can use `TRandom3` (from Root), though random number draws are very slow and should be used only if necessary. Note that the way that Sys works needs this function to return the *change* in the target value, if `useMultiply=false`. In the above case, if this is the only Sys, the resulting value will be $x_{new} = x_{MC} + a_0 x_{MC}$ (where $a_0$ is parameters[0]). `Eval` takes an argument `x`, but this is a leftover from an earlier version and shouldn't be used. The `parameters` array has the values of the mcmcPars and the mcBranches from `Sys.cxx`. The order is the order selected in `Sys.cxx`, with mcmcPars first and mcBranches second (the order mcmcPars first and mcBranches second is hard-coded, hence why they need to be called in that order in Sys). A mis-match between `nPars` and the number of parameters in Sys will result in undefined behavior, and could cause a seg fault.

In the meta file, the behavior of the automatic parameter sharing the name of the Sys can be controlled, with the same controls as the Parameters of section A.3.7. Valid Keys are:

`Name=string` sets the name. Must be unique. If this isn't present, an error message is generated and setup fails.

`Init=double` sets the initial value. Error and setup failure if not present.

`Width=double` sets $\sigma$ for the Gaussian drawn from at the incrementing step for this variable. If `Width` is $\leq 0$, parameter is not incremented. Error and setup failure if not present.

`Mean=double` creates a Gaussian constraint for this parameter. Sets $\mu$ of this constraint to `Mean`. If this is present and `Sigma` is not, an error is generated and setup fails. If neither is present, no constraint. See `Sigma` for more details.

`Sigma=double` creates a Gaussian constraint for this parameter. Sets $\sigma$ of this constraint to `Sigma`. This constraint gives LL contribution $-\frac{(\alpha-\mu)^2}{2\sigma^2}$. If this is present and `Mean` is not, an error is generated and setup fails. If neither is present, no constraint.

`Min=double` sets the minimum value for the parameter. If its value goes below `Min`, the returned LL is $-1e200$, effectively $-\inf$, so no step is taken. If this is not present, no minimum.

`Max=double` sets the maximum value for the parameter. If its value goes above `Max`, the returned LL is $-1e200$, effectively $-\inf$, so no step is taken. If this is not present, no maximum.

`AsymmFunc=string` creates an Asymm function for this parameter. See section A.3.11. If this is not present, no AsymmFunc.

`AsymmPar=string` sets a parameter for the Asymm Func. See section A.3.11. If this is present and `AsymmFunc` isn't, an error is returned and setup fails.

### A.3.11 AsymmFunc

The AsymmFunc system is used to alter the values of parameters in user-specified ways, rather than just the random draw used to vary them. The basic idea is that `AsymmFunc` defines a function, using the notation of Root's `TF1`, which is *added* to the parameter's value. The reason for adding rather than replacing is an artifact of how this is implemented: it uses the machinery of Sys. The `AsymmPar` then specify what parameters to use in the function. An example use is:

```
new Parameter

Name=NCFlux

Init=0
```

```
Width=-1
AsymmFunc=[0]*[1]-[2]
AsymmPar=BoronFlux
AsymmPar=PMTEfficiency
AsymmPar=NCFlux
```

This takes the Parameter NCFlux and sets its value to be

NCFlux + BoronFlux*PMTEfficiency - NCFlux = BoronFlux*PMTEfficiency

In this case, the NCFlux parameter isn't varied because its value is irrelevant - it is entirely replaced by the combination. This doesn't have to be the case, of course. The notation of `AsymmFunc` is that each number in a set of square brackets is a parameter (of `TF1`); there cannot be spaces in the function definition (due to parsing issues). Any parameter defined in the meta file is valid to use, and the function itself can use any combination of basic C++ operation and functions defined in TMath. Random numbers are not available here, which should not be a strong constraint as the parameters are randomly varied. The `AsymmPar` correspond to the parameters `[i]`, numbered in order of appearance starting with 0. There is limited error checking on the parameter numbers: the system should give an error if too high of a parameter number is requested in the `AsymmFunc`, but will not if a number is missing. This can be problematic for long functions, especially when they are changed; exercise caution.

## A.4 metaConfig.exe

This section can be safely skipped by most users. It details what happens during the conversion process from the human-readable meta file to the less-readable (but still human readable, just much harder to follow) config file.

The config file format used by `rottSigEx.exe` is parsed and read-in by the class described by `ConfigFile.h`. This is the only code reused from [63], as it was downloaded from elsewhere to start with. The basic premise is that the config file consists of Key-Value pairs, similar to the meta file. The major differences are that each Key must be unique, the order that the keys appear in does not matter at all, and the existence of objects is kept track of by a numbering system.

The `metaConfig.exe` system is fairly simple: it keeps a running total of each kind of object present. When the first `Pdf` is created with `new`, the `Pdf` counter is set to 0. Until `new` is called again, the system then adds `Pdf_0_` to the beginning of each command, so that

```
Name = FirstPdf
```

is converted into

```
Pdf_0_Name=FirstPdf
```

A special case here is `MCBranch`, which has its own numbering system, so that every time it is called it is incremented. As an example

```
MCBranch= energy
MCBranch=x
```

is converted into

```
Pdf_0_MCBranch_00=energy
Pdf_0_MCBranch_01=x
```

When `new` is called again for an object inside a `Pdf`, such as `Axis`, that counter begins incrementing. Then a command is appended with `Pdf_0_Axis_0_`, assuming this is both the first `Pdf` and the first `Axis`. `Bin` is again a special case with its own counting system, but this time the format is slightly different, as

```
Bin=1
```

becomes

```
Pdf_0_Axis_0_Bin00=1
```

All of counters under a particular object are reset when `new` is called for that object, so if `new Pdf` is called, the first call of `MCBranch=y` will give

```
Pdf_1_MCBranch_00=y
```

This pattern is repeated for every object with sub-objects. Those that interact directly with the MCMC, however, are kept track of separately. The MCMC directives are just copied, as they take the same form in both the config and meta files. The `Parameter` counter is never reset, calling `new Parameter` anywhere in the meta file increments it.

One word of caution is that the `metaConfig.exe` program does very little error checking. It will return errors if it encounters something it can't translate into a config file, such as creating a `Axis` without first creating a `Pdf`. It will also return an error if it encounters a command it doesn't recognize - both in the sense of commands that don't conform to the pattern key=value, and in the sense of calling something like `new Banana`, as it does not recognize an `Banana` class. Anything else that fits the valid patterns will be allowed. It is thus essential to run a test run of any config file through `rottSigEx.exe` to make sure setup completes properly.

## A.5 Other Programs

The other three programs included, `autoFit.exe`, `drawResults.exe` and `getAutoCorr.exe`, are simple enough that the built-in help is sufficient. This can be accessed by running either the the program with no arguments or with the argument `--help`.

# Bibliography

[1] P. C. de Holanda and A. Yu Smirnov. Lma msw solution of the solar neutrino problem and first kamland results. *JCAP*, 0302:001, 2003.

[2] Wolfgang Pauli. On the earlier and more recent history of the neutrino. In Klaus Winter, editor, *Neutrino Physics*, pages 1–25. Cambridge University Press, 1991.

[3] J. Chadwick. *Verh. d. deutschen Phys. Ges.*, 16:383, 1914.

[4] C. D. Ellis and W. A. Wooster. *Proc. R. Soc. A*, 117:109, 1927.

[5] L. Meitner and W. Orthmann. *Z. Phys.*, 60:143, 1930.

[6] Frederick Reines and Clyde L. Cowan. The neutrino. *Nature*, 178:446–449, September 1956.

[7] Raymond Davis. Attempt to detect the antineutrinos from a nuclear reactor by the $cl37(\nu, e-)a37$ reaction. *Phys. Rev.*, 97(3):766–769, Feb 1955.

[8] G. Danby et al. Observation of high-energy neutrino reactions and the existence of two kinds of neutrinos. *Phys. Rev. Lett.*, 9(1):36–44, Jul 1962.

[9] K. Kodama et al. Final tau-neutrino results from the donut experiment. *Phys. Rev. D*, 78(5):052002, Sep 2008.

[10] A. Heister et al. Single and multi-photon production in $e^+e^-$ collisions at $\sqrt{s}$ up to 209 gev. *The European Physical Journal C*, 28:1–13, 2003. DOI:10.1140/epjc/s2002-01129-7.

[11] K. Nakamura et al. *JPG*, 37, 2010. `http://pdg.lbl.gov`.

[12] C. S. Wu, E. Ambler, R. W. Hayward, D. D. Hoppes, and R. P. Hudson. Experimental test of parity conservation in beta decay. *Phys. Rev.*, 105(4):1413–1415, Feb 1957.

[13] F.J. Hasert et al. Search for elastic muon-neutrino electron scattering. *Physics Letters B*, 46(1):121 – 124, 1973.

[14] Christian Iliadis. *Nuclear Physics of Stars*. Wiley-Vch, 2007.

[15] John N. Bahcall, M. H. Pinsonneault, and Sarbani Basu. Solar models: Current epoch and time dependencies, neutrinos and helioseismological properties. *Astrophysical Journal*, 555:990–1012, 2001.

[16] John N. Bahcall, Aldo M. Serenelli, and Sarbani Basu. New solar opacities, abundances, helioseismology, and neutrino fluxes. *Astrophysical Journal*, 621:L85–L88, 2005.

[17] H. A. Bethe. Energy production in stars. *Physical Review*, 55:434–456, 1939.

[18] L. C. Stonehill, J. A. Formaggio, and R. G. H. Robertson. Solar neutrinos from cno electron capture. *Phys. Rev. C*, 69(1):015801, Jan 2004.

[19] Raymond Davis, Don S. Harmer, and Kenneth C. Hoffman. Search for neutrinos from the sun. *Phys. Rev. Lett.*, 20(21):1205–1209, May 1968.

[20] Bruce T. Cleveland et al. Measurement of the solar electron neutrino flux with the homestake chlorine detector. *The Astrophysical Journal*, 496(1):505, 1998.

[21] J. N. Abdurashitov et al. Measurement of the solar neutrino capture rate with gallium metal. *Phys. Rev. C*, 60(5):055801, Oct 1999.

[22] W. Hampel et al. Gallex solar neutrino observations: results for gallex iv. *Physics Letters B*, 447(1-2):127 – 133, 1999.

[23] M. Altmann et al. Gno solar neutrino observations: results for gno i. *Physics Letters B*, 490(1-2):16 – 26, 2000.

[24] S. Fukuda et al. Solar $b8$ and hep neutrino measurements from 1258 days of super-kamiokande data. *Phys. Rev. Lett.*, 86(25):5651–5655, Jun 2001.

[25] John N. Bahcall. Solar models and solar neutrinos: Current status. *Phys. Scripta*, T121:46–50, Dec 2004. arXiv:hep-ph/0412068v1.

[26] B. Pontecorvo. Neutrino experiments and the question of lepton charge conservation. *Zhurnal Eksperimental'noi i Teoreticheskoi Fiziki*, 53(5):1717–1725, 1967.

[27] V. Gribov and B. Pontecorvo. Neutrino astronomy and lepton charge. *Physics Letters B*, 28(7):493 – 496, 1969.

[28] Ziro Maki, Masami Nakagawa, and Shoichi Sakata. Remarks on the unified model of elementary particles. *Progress of Theoretical Physics*, 28(5):870–880, 1962.

[29] David J. Griffiths. *Introduction to Elementary Particles.* Wiley-Vch, 2008.

[30] Francis Halzen and Alan D. Martin. *Quarks and Leptons: An Introductory Course in Modern Particle Physics.* John Wiley and Sons, 1984.

[31] Rabindra N. Mohapatra and Palash B. Pal. *Massive Neutrinos in Physics and Astrophysics.* World Scientific, 2004.

[32] Alan H. Guth, Lisa Randall, and Mario Serna. Day-night and energy dependence of MSW solar neutrinos for maximal mixing. *Journal of High Energy Physics*, 1999(08):018, 1999.

[33] L. Wolfenstein. Neutrino oscillations in matter. *Physical Review D*, 17(9):2369–2374, May 1978.

[34] S. P. Mikheyev and A. Yu. Smirnov. Resonant amplification of $\nu$ oscillations in matter and solar-neutrino spectroscopy. *Il Nuovo Cimento*, 9 C(1):17–26, 1986.

[35] Alessandro Strumia and Francesco Vissani. Neutrino masses and mixings and .... `http://arXiv.org/abs/hep-ph/0606054`.

[36] E. Kh. Akhmedov. Neutrino physics. Trieste Summer School in Particle Physics lectures, 1999. arXiv:hep-ph/0001264.

[37] Mattias Blennow, Tommy Ohlsson, and Håkan Snellman. Day-night effect in solar neutrino oscillations with three flavors. *Phys. Rev. D*, 69(7):073006, Apr 2004.

[38] Adam M. Dziewonski and Don L. Anderson. Preliminary reference earth model. *Physics of The Earth and Planetary Interiors*, 25(4):297 – 356, 1981.

[39] Kathryn Miknaitis. *A Search for Matter Enhanced Neutrino Oscillations through Measurements of Day and Night Solar Neutrino Fluxes at the Sudbury Neutrino Observatory.* PhD thesis, University of Washington, Seattle, Washington, 2005.

[40] Evgeny Kh. Akhmedov, Maria A. Tórtola, and José W. F. Valle. A simple analytic three-flavor description of the day-night effect in the solar neutrino flux. *Journal of High Energy Physics*, JHEP05, May 2004.

[41] Herbert H. Chen. Direct approach to resolve the solar-neutrino problem. *Phys. Rev. Lett.*, 55(14):1534–1536, Sep 1985.

[42] Q. R. Ahmad et al. Measurement of the rate of $\nu_e + d \to p + p + e-$ interactions produced by $B8$ solar neutrinos at the Sudbury Neutrino Observatory. *Phys. Rev. Lett.*, 87(7):071301, Jul 2001.

[43] Q. R. Ahmad et al. Direct evidence for neutrino flavor transformation from neutral-current interactions in the sudbury neutrino observatory. *Phys. Rev. Lett.*, 89(1):011301, Jun 2002.

[44] J. Boger et al. The sudbury neutrino observatory. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 449(1-2):172 – 207, 2000.

[45] B. Aharmim et al. Measurement of the $\nu_e$ and total $^8$B solar neutrino fluxes with the Sudbury Neutrino Observatory Phase-III data set. To Be Published, 2011.

[46] John N. Bahcall. Neutrino-electron scattering and solar neutrino experiments. *Reviews of Modern Physics*, 59:505–521, 1987.

[47] S. Nakamura, T. Sato, V. Gudkov, and K. Kubodera. Neutrino reactions on the deuteron. *Phys. Rev. C*, 63(3):034617, Feb 2001.

[48] B. Beltran et al. A Monte Carlo simulation of the Sudbury Neutrino Observatory proportional counters. *New J. Phys.*, To appear, 2011. arXiv:1104.2573v1.

[49] 3-Phase working group. 3-phase unidoc. 2011.

[50] B. Aharmim et al. Determination of the $\nu e$ and total $B8$ solar neutrino fluxes using the Sudbury Neutrino Observatory Phase I data set. *Phys. Rev. C*, 75(4):045502, Apr 2007.

[51] B. Aharmim et al. Low-energy-threshold analysis of the Phase I and Phase II data sets of the Sudbury Neutrino Observatory. *Phys. Rev. C*, 81(5):055504, May 2010.

[52] W. Winter et al. Measurement of the 8b neutrino spectrum. *Nuclear Physics A*, 721:C553 – C555, 2003.

[53] G. O. Gann et al. Leta unidoc. SNO Internal Document. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-7R5LC8`.

[54] Q. R. Ahmad et al. Measurement of the rate of $\nu e + d \rightarrow p + p + e-$ interactions produced by $b8$ solar neutrinos at the Sudbury Neutrino Observatory. *Phys. Rev. Lett.*, 87(7):071301, Jul 2001.

[55] Q. R. Ahmad et al. Direct evidence for neutrino flavor transformation from neutral-current interactions in the sudbury neutrino observatory. *Phys. Rev. Lett.*, 89(1):011301, Jun 2002.

[56] S. N. Ahmed et al. Measurement of the total active $b8$ solar neutrino flux at the sudbury neutrino observatory with enhanced neutral current sensitivity. *Phys. Rev. Lett.*, 92(18):181301, May 2004.

[57] B. Aharmim et al. Independent Measurement of the Total Active $B8$ Solar Neutrino Flux Using an Array of $He3$ Proportional Counters at the Sudbury Neutrino Observatory. *Phys. Rev. Lett.*, 101(11):111301, Sep 2008.

[58] Q. R. Ahmad et al. Measurement of day and night neutrino energy spectra at SNO and constraints on neutrino mixing parameters. *Phys. Rev. Lett.*, 89(1):011302, Jun 2002.

[59] B. Aharmim et al. Electron energy spectra, fluxes, and day-night asymmetries of 8b solar neutrinos from measurements with nacl dissolved in the heavy-water detector at the sudbury neutrino observatory. *Phys. Rev. C*, 72(5):055502, Nov 2005.

[60] Scott Oser. Summary of ncd-phase day-night systematics. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-7UZ56U`, 2009.

[61] Jeff Secrest. Energy systematics for the phase III. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-7CPS49`, 2008.

[62] Alain Bellerive. Energy Non-Linearity - NCD Phase. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-87E4K5`, 2010.

[63] Blair Jamieson. SNO NCD phase signal extraction on unblinded data with integration over systematic nuisance parameters by Markov-chain Monte Carlo. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-7NCU9S`, 2008.

[64] Alain Bellerive. PMT Radial Scale - NCD Phase. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-8573HE`, 2010.

[65] Pierre-Luc Drouin. Implementation of FTN systematic uncertainties in NCD phase signal extraction. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-7CLTE5`, 2008.

[66] Blair Jamieson. NCD phase day-night systematic uncertainties from $^{16}$N data. SNO Internal Document, September 2007. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-76XT65`.

[67] Alan Poon. Day-night asymmetry study of the Berkeley Blob position in phase III, April 2009. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-7R7NEX`.

[68] Helen O'Keefe. Results from diurnal studies of radioactive backgrounds during the NCD phase of SNO, April 2009. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-7QWHMF`.

[69] Kevin Graham. Directional uncertainties for NSP. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-5VUQEH`, April 2004.

[70] Blair Jamieson. N16 study data. Private communication, February 2008.

[71] F. James. MINUIT reference manual. `http://wwwasdoc.web.cern.ch/wwwasdoc/minuit/minmain.html`.

[72] Phil Gregory. *Bayesian Logical Data Analysis for the Physical Sciences*. Cambridge University Press, 2005.

[73] E. T. Jaynes and G. Larry Brelthorst. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[74] Radford M. Neal. Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

[75] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *Journal Of Chemical Physics*, 21:1087–1092, June 1953.

[76] Yves F. Atchad and Jeffrey S. Rosenthal. On adaptive markov chain monte carlo algorithms. *Bernoulli*, 11(5):pp. 815–828, 2005.

[77] Scott Oser. Implementing the ultimate 3-Phase SNO analysis as a meta-analysis. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-7PLUFN`, 2009.

[78] N S Martin R, Oblath and Tolich N. Pulse-Shape Analysis for the Third Phase of the Sudbury Neutrino Observatory. XXIV International Conference on Neutrino Physics and Astrophysics, Athens, Greece, 2010.

[79] Richard J. Wagner. http://www-personal.umich.edu/ wagnerr/ConfigFile.html.

[80] Stan Seibert. Constraints and correlations: Is your pull normal? SNO Internal Document, 2008. `http://manhattan.sno.laurentian.ca/sno/ananoteb.nsf/URL/MANN-7BRSWV`.

[81] Nuno Barros. Neutrino oscillation analysis documentation. SNO Internal Document, June 2011.

[82] CERN. Root homepage. `http://root.cern.ch`.