NTNU – Trondheim
Norwegian University of
Science and Technology

Maximum Likelihood Analysis of IceCube Data:

Could Decaying Dark Matter explain the high
energy Neutrino Excess?

Jan Erik Leinaas

MSc in Physics
Submission date: May 2015
Supervisor: Michael Kachelriess, IFY

Norwegian University of Science and Technology
Department of Physics

Maximum likelihood analysis of IceCube data: Could decaying dark matter explain the high energy neutrino excess?

Jan Erik Leinaas

May 22, 2015

Sammendrag

I denne mastergradsoppgaven i teoretisk fysikk har jeg sett på ett sett med observasjoner fra et neutrino observatiorium i Antarktis kalt IceCube. Dette observatoriet har observert flere høyenergis neutrinoer enn forventet og jeg ønsket å undersøke hvor godt mørk materie ville passe som en forklaring på de uforventede neutrino observasjonene. Den synlige massen i universet er ikke stor nok til å forklare dynamikken vi ser, så man har konkludert med at mesteparten av massen i universet ikke er synlig for oss. Denne massen kalles mørk materie. Hvis mørk materie kan henfalle så er neutrinoer ett av de mulige resultatene.

Jeg har sett på retningen og energien til neutrino observasjonene til IceCube og sett statistisk på hvor godt de passer med en model for mørk materie i forhold til andre modeller for deres opphav. Oppgaven har i hovedsak vært numerisk og programmet jeg brukte til å finne svar på dette er vedlagt i appendiksene i slutten av oppgaven. Resultatet var at det kunne passe med at en liten del av neutrinoene kom fra henfall av mørk materie, men en modell uten noe kontribusjon fra mørk materie kunne forklare observasjonene nesten like godt.

Abstract

In this theoretical physics master thesis I've looked at a set of observations from a neutrino observatory in Antarctica called IceCube. This observatory has observed more high energy neutrinos than expected. I wanted to investigate whether dark matter would fit as an explanation of this neutrino excess. The visible mass in the universe isn't great enough to explain the dynamics we observe. This has led many to the conclusion that most of the mass in the universe is not visible to us. This masses is called Dark Matter. If Dark Matter decays then neutrinos are one of the possible end products.

I have looked at the direction and energy of the neutrino excess at IceCube and investigated statistically how well they fit a model of Dark Matter compared to other models for their origin. The thesis work has been primarily numerical and the program I wrote to answer the main question is found in the appendices at the end. The result was that a small amount of the neutrinos being from Dark Matter decay would fit with observations, but model with no Dark Matter could explain the data almost equally well.

Acknowledgment

I would like to thank my supervisor, Michael Kachelriess, for all the work put into making this thesis happen.

Contents

Chapter 1

Introduction

One of the outstanding mysteries in astrophysics is the apparent anomaly in mass distribution in galaxies; the inwards acceleration of orbiting objects (measured by measuring the radial velocity at a known radius by the Doppler effect) do not match the gravitational attraction that should be caused by the visible matter below those radii. While it is possible that the law of gravity works differently than previously thought in the very weak field limit it seems more likely that there is some additional massive matter that we cannot observe directly. Because the extra matter is not directly observable it must not emit electromagnetic radiation and we therefore call it Dark Matter.

Since Dark Matter still makes up the majority of the mass in the universe 13 billion years after the big bang it's usually been assumed that whatever particles it's made of are stable. For the last several decades the most widely favored candidate for the make up of Dark Matter are the lightest of the so called super-symmetric particles. That is to say; there is a theory that posits that each type of boson has a fermionic opposite and each type of fermion has a bosonic opposite, their so called super-symmetric partners. If there is a conserved super-symmetric charge then the lightest super-symmetric particle would be stable. Just like conservation of baryon number makes the proton stable since it's the lightest baryon. This lightest super-symmetric particle would be one possible candidate for making up stable dark matter.

However it's not required that Dark Matter is completely stable. As long as it has a life time significantly longer than the age of the universe it is possible that Dark Matter decays. Neutrinos are among the possible products of such decay. The IceCube neutrino detector in Antarctica has detected several neutrinos at a much higher energy than expected from atmospheric background, but smaller than that expected from astrophysical sources, which motivates us to investigate whether they come from the decay of slightly unstable Dark Matter. In this thesis I will use a statistical method called maximum likelihood estimation to determine how well the energy and spatial distribution of the high energy IceCube neutrinos fit with a Dark Matter model compared several other possible neutrino sources.

In the next sub-chapters of this introduction I'll give a brief overview of the astrophysics and statistics used. The next chapter will talk in more detail about Dark Matter, and possible Dark Matter candidates. Chapter 3 will talk briefly about neutrino detection, the history of neutrino astronomy and give and overview of the IceCube detector. Chapter 4 deals with how I approached the problem. It gives a more detailed overview of the statistical methods used, the astrophysical models and how the program that ran it all was programmed. Finally, in chapter 5, I present the results with some analysis and commentary. At the very end of the thesis there's a number of appendices that contain the program I used for the main calculation.

1.1 Dark Matter

The orbital velocity of an object is determined by the gravitational force exerted on it. In most galaxies the majority of the luminous mass is found in a concentrated area with only limited light sources outside this radius. For those more distantly orbiting objects we ought to be able to find their velocity by viewing them as test masses orbiting a point mass with the overall mass of the Galaxy. Assuming their eccentricity is low enough to let us view their orbits as circular we get:

$$G\frac{M}{r^2} = \frac{v^2}{r} \tag{1.1}$$

which implies $v \propto 1/\sqrt{r}$. Actual observations of the outer parts of galaxies however show a rotational velocity that's constant, or even increasing, out to the furthest radii ever measured. This suggests that the mass density of most galaxies doesn't fall of, the way their density of luminous matter does.

Astronomers have measured the rotational speed of other galaxies using the Doppler effect. When a radiating body is moving relative to an observer the frequency of the light seen by the observer will be different from the frequency seen in the objects frame of reference. Of course to tell how much a signal is red or blue shifted you would need to know what it's original frequency was. In astrophysics a specific line, the 21cm line in the spectrum of hydrogen, is usually used. Since we know the wavelength of this spectral line in the rest frame of the source we can find the velocity of the source relative to us at the time the light was sent. By measuring the velocity relative to us, on both sides of a Galaxy (assuming it's axis of rotation is not pointing at or too near us) and looking at the difference we can find the rotational speed of said Galaxy.

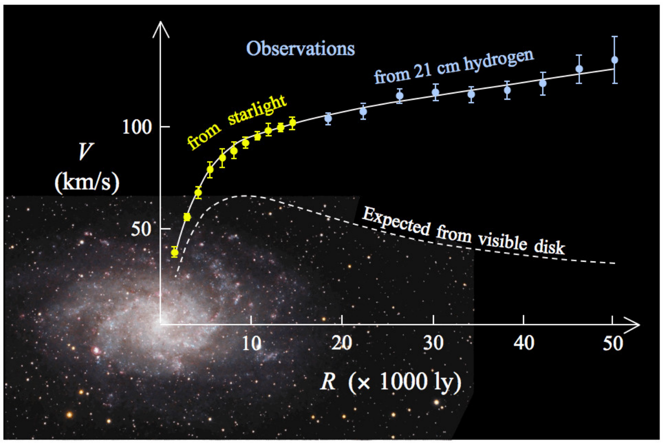


Figure 1.1: Rotation curve of a Galaxy (M33) compared to expected curve based on luminous matter[1]

1.2 Statistics

In order to look at how likely it is that some of the unexpected neutrino flux is from decaying Dark Matter we will analyze the direction of the incoming neutrinos using a statistical method known as maximum likelihood estimation. The core of this technique is to choose some class of theories and see which of them assigns a greater probability to the observed events being observed.

In addition to this analysis of the data we will additionally look at how well observations are capable of distinguishing between the possible scenarios. We will look at the degree of overlap between the distributions and also at the angular resolution of the IceCube observatory.

Chapter 2

Dark Matter: Theory

2.1 Distribution of Dark Matter

As mentioned previously the rotational velocity of a Galaxy at a radius r is approximately proportional to $\sqrt{M(r)/r}$. Where the $M(r)$ is the enclosed mass below that radius. That means that for the rotation curve to be flat, $\frac{dv}{dr} = 0$, $M(r)$ needs to scale $\propto r$ as the radius increases, or $\frac{dM}{dr} = const$. Of course $\frac{dM}{dr} = 4\pi\rho(r)r^2$ which implies $\rho_{flat} \propto r^{-2}$. A Galaxy with a rotation curve increasing at higher radii must then have a mass density falling off slower than r^{-2}. At some large radius the falloff in the halo density must become greater than this, or the galactic mass would diverge.

One common model for the density of the Dark Matter halo is the Navarro-Frenk-White profile,

$$\rho = \rho_0[r(1 + r^2/r_0^2)]^{-1} \tag{2.1}$$

This density function was proposed by Navarro, Frenk and White in the mid 90s based on N-body simulations of Galaxy formation[3]. There exist several variations on this kind of distribution. The Navarro-Frenk-White distribution, which goes as r^{-1} at small radii and as r^{-3} at large radii, is itself a modification of an earlier model that decreased as r^{-4} at large radii. And modifications have been suggested to it which would go as $r^{-1.5}$, or similar, at low radii.

Another type of distribution that is starting to see more use is the Einasto distribution. While actually far older than the Navarro-Frenk-White profile it didn't see a lot of use until the 2000s. The distribution, $\rho \propto \exp(-Ar^{\alpha})$, fits N-body simulations of Galaxy formation somewhat better than the Navarro-Frenk-White profile. However with the parameters that are reasonable for the Milkyway this distribution is a lot harder to distinguish from an isotropic background. Since we are severely limited in the number of observations we have available we will focus on the Navarro-Frenk-White distribution.

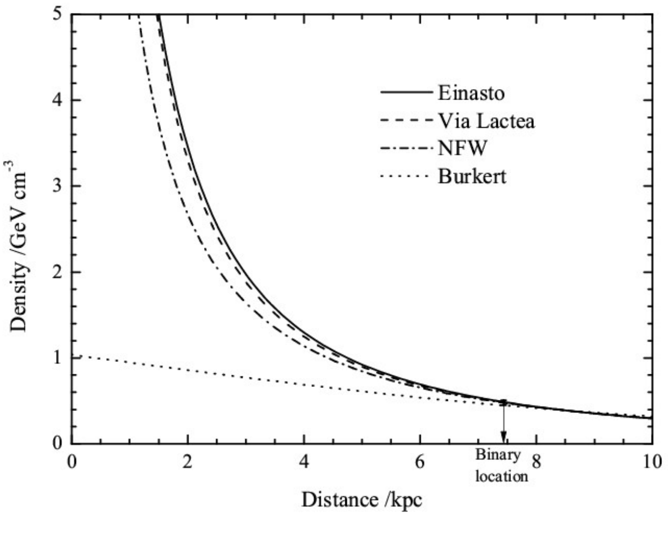


Figure 2.1: Comparrison of the Einasto and Navarro- Frenk-White (NFW) distributions[2]

2.2 Properties of Dark Matter

2.2.1 Interactions

The most obvious requirement for a Dark Matter candidate is that it must have electric charge zero. First of all because for any object that couples to electromagnetism we'd expect to see electromagnetic radiation emitted by it. The only way to avoid that would be by it being concentrated in dense, cold objects. But such objects can be seen by the way they bend the light from objects behind them and astronomers have searched extensively for this kind of lensing (looking for black holes for instance) and the ones seen have a combined mass far to small to represent all Dark Matter.

The second reason comes from big bang cosmology: today the universe is very inhomogeneous, with mass densities ranging from stars and even black holes to intergalactic space. This is the natural result of matter accumulating in the gravitational potential created by small starting inhomogeneities. The cosmic microwave background (CMB) was remains from shortly after the big bang when protons and electrons came together to create charge-neutral atoms. At that point the matter's cross section with photons dropped dramatically and space became transparent. Examining the CMB allows us to measure the inhomogeneity of the baryonic matter at that time. The homogeneity so measured is too low to result in the current structure of the universe over the life time of the universe however. In order to explain the inhomogeneity we see today there must have been some non-charged, cold matter that being uncoupled from the photon bath was able to begin accumulating before recombination. The newly created atoms, decoupled from the photon bath, settling into already existing gravitational potentials would explain the inhomogeneity of the current universe assuming the total mass of non-baryonic matter was five times that of the baryonic[4].

Furthermore there is an additional requirement, beyond the limitations on interactions with non-Dark Matter, that Dark Matter cannot interact strongly with other Dark Matter. In theory it would be possible to have a type of particles, or multiple types of particles, that interacted very weakly with other matter but have strong interactions among themselves. However there are several reasons why Dark Matter cannot be like this.

Normal matter, because of the interactions it has, experiences friction. This leads to the luminous matter in most galaxies becoming concentrated in the galactic plane as the energy from the movement normal to the plane of rotation dissipates into radiation. Observations indicate that the Dark Matter halos of galaxies are generally spherical. This means that there cannot be any dissipative interactions between Dark Matter on a scale similar to the ones between luminous matter.

Another strong piece of evidence against strong interactions between Dark Matter comes from colliding Galaxy clusters. The bullet cluster is a cluster made up of two colliding Galaxy clusters. Observations of the bullet cluster revealed that the gas cloud of the smaller cluster was lagging behind it's galaxies, due to being slowed down by scatterings as it passed through the larger cluster. When this was discovered it was quickly realized that if the mass distributions of the clusters could be mapped this would be an excellent measure of the strenght of interactions between Dark Matter. Since the Dark Matter makes up the largest portion of the mass in a Galaxy cluster being able to locate the majority of the mass in the smaller Galaxy cluster would pinpoint its Dark Matter halo. Like intergalactic gas the Dark Matter halo of a Galaxy cluster is very spread out. The galaxies on the other hand are extremely spread out compared to their size. This means that the galaxies of two colliding clusters are unlikely to encounter any areas of comparable density which is why they were not slowed down in the same way as the intergalactic gas as the clusters past through each other. If the Dark Matter halo of the smaller cluster lagged behind it in a similar way as the gas cloud it would imply interactions between Dark Matter particles. The larger the lag the stronger the interactions. The mass distribution of the cluster was successfully mapped and the result showed the Dark Matter halo to be coincident with the galaxies. This placed a strong limit on interactions between Dark Matter[5].

2.2.2 Dynamics

Secondly, Dark Matter must be cold. In order to match the universe we observe Dark Matter must cluster in gravity potentials. Relativistic (ie high kinetic energy to rest mass ratio) particles would pass through small gravitational perturbations too fast to be much effected by them. This means that very light Dark Matter would result in mass clustering only on a much higher scale than what is observed. Since most creation mechanisms posited by Dark Matter theories are thermal. Current observations and numerical simulations place a lower limit for the mass of thermal Dark Matter at 2keV[6].

2.3 Dark Matter Candidates

2.3.1 WIMPs

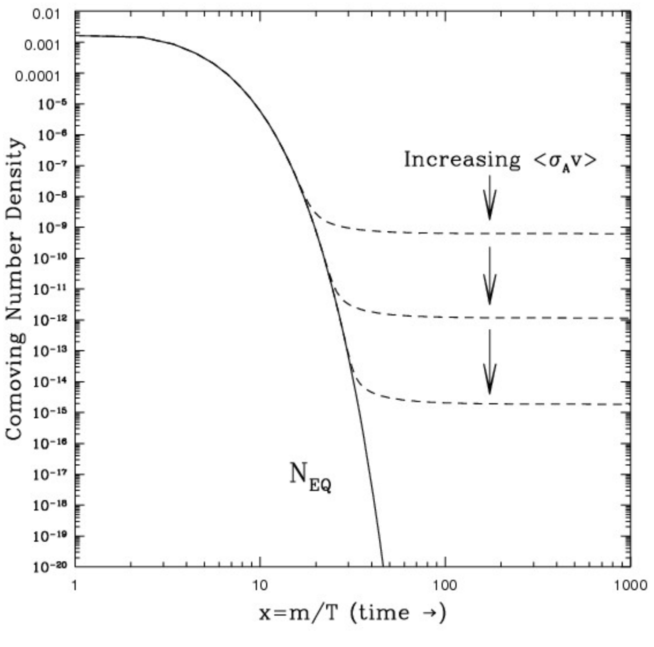
Weakly interacting massive particles (WIMPs) are the kind of particle most widely believed to make up Dark Matter so they will be discussed first. As the name suggests WIMPs interact only through weak interactions. They are also massive, generally WIMP posit masses in the order of 100GeV. These particles are either stable, or so long lived that decays have a negligible effect on their abundance over the lifetime of the universe. They can be created from, or annihilated into, ordinary particles through pair production/annihilation. In the early universe when matter was dense and hot enough for scatterings with $> M_{DM}$ energy to be common they were in thermal equilibrium with the baryonic matter.

At some point the temperature of the universe fell below the Dark Matter mass. At that point Standard Model (SM)-particle scatterings with high enough energy to create a Dark Matter pair become rare. In order to remain in equilibrium the density of Dark Matter will fall until the number of annihilations is as small as the number of pair creations, settling in an exponentially suppressed density,

$$n_{DM}^{eq} = (2\pi mT)^{\frac{3}{2}} \exp\left(\frac{-m_x c^2}{kT}\right). \tag{2.2}$$

However, as the universe cools and expands, the likelihood of Dark Matter pairs impacting each other decreases. This increases the relaxation time, how fast the system moves towards equilibrium. Once the annihilation rate becomes slower than the expansion of the universe, the system will quickly leave equilibrium and there will be an almost, but not completely, stable abundance of Dark Matter particles[7].

Figure 2.2: WIMP freeze-out scenarios depending on annihilation cross section and average velocity[23].



Super-symmetry is a class of theories that posit that for each standard model particle there is a super-symmetric opposite: a fermion for every boson and a boson for every fermion. Because the Higgs mass depends strongly on the mass of all particles, even ones that don't couple directly to it if they couple to particles that do, introducing new physics at higher energy scales becomes difficult. Since the observed Higgs mass is so low compared to those scales, any new particle would create corrections to the Higgs mass that are individually much larger than the overall Higgs mass. This would require some miraculous amount of cancellation of these corrections to produce the Higgs physics we observe. Because bosons and fermions make contributions to the Higgs mass with opposite sign a symmetry relating fermions and bosons would be a good explanation for this kind of cancellation.

If this is the case then the lightest super-symmetric particle would make an excellent WIMP candidate. In super-symmetric models baryon and lepton numbers are no longer conserved quantities. However, no violation of baryon number have been observed despite extensive observations. This implies that any baryon number violations must be extremely small. R-parity is a new symmetry introduced in super-symmetric theories to suppress baryon and lepton number violating interactions. If R-parity is an exact symmetry then it will have an associated conserved quantum number which would make the lightest super-symmetric particle stable[8].

2.3.2 Axions

Another Dark Matter candidate is a kind of particle know as an axion. The main point in their favor is that they solve another outstanding mystery in physics. The standard model Lagrangian contains a term:

$$\mathcal{L} = ... + \frac{\theta g^2}{32\pi^2} G^a_{\mu\nu} \tilde{G}^{a\mu\nu} \tag{2.3}$$

where G is the gluon field-strenght tensor and θ is a parameter. If $\bar{\theta} = \theta -$ arg det m_q, where m_q is the quark mass matrix, is different from zero then parity and CP-symmetry is violated in strong interactions. However measurements of the neutron dipole moment sets a limit of $|\bar{\theta}| < 10^{-9}$.

The fact that a symmetry that is not required is nevertheless respected to such a high degree would be a huge coincidence and strongly suggests that there is some unknown physics suppressing or forbidding the symmetry-violating interactions. One solution is to introduce an additional field $a(x)$ that determines θ as

$$\theta = \frac{a(x)}{f_a} \tag{2.4}$$

where f_a is a constant with unit of energy. The minimum of the effective potential for this field will be at $\bar{\theta} = 0$, so as the universe cools it will naturally settle at this value, solving the naturalness problem of strong CP-symmetry.

However the $aG\tilde{G}$ term in the Lagrangian is not renormalizable. That is to say it produces infinities that cannot be ”reduced” to finite quantities using our normal methods for avoiding infinite quantities. In order to get around this we can instead posit a renormalizable theory which has that term as the low energy limit. The proposed solution for this is a broken U(1) symmetry, called Peccei-Quinn symmetry.

Axions are not produced thermally, which makes them one of the exceptions mentioned in the argument placing a lower limit on the Dark Matter mass based on the temperature of the early universe. Particle physics experiments, stellar evolution and the duration of neutrino pulses in supernovas have ruled out overlapping areas of axion mass, giving an upper limit of $m_a = 3 \cdot 10^{-3}$eV[9].

2.3.3 Sterile neutrinos

Normal neutrinos ($\nu_{e/\mu/\tau}$) were briefly considered as a possible Dark Matter candidate, but were quickly rejected as while they have the correct interactions they are too light. The heaviest neutrino is lighter than 1eV while as argued in section 2.2 thermally created Dark Matter particles must have mass greater than 2keV. When the idea of WIMPs was created a much heavier fourth generation of neutrino was considered the most likely. However that too was eventually rejected, as such a heavy normal neutrino interact too strongly. Some manner of neutrino would make a good Dark

Matter candidate as they are the only known particles with the right kind of interactions, but this shows that Dark Matter cannot be made of normal neutrinos of any kind.

One possible solution is to posit a kind of neutrino that doesn't interact weakly. The best motivated example would be right handed neutrinos. All the fermions in the standard model come in two chiralities: right handed and left handed. The weak force is the only force (other than gravity) that neutrinos interact with, and it couples only to left handed particles. Because of this only left handed neutrinos are observed, and right handed neutrinos have been mostly ignored. Of course since right handed neutrinos only interact by gravity the fact that he haven't observed them is neither evidence for or against their existance. And since they only interact by gravity they would, if they exist, have many of the qualities that are requried of a dark matter candidate.

Lacking electroweak and strong interactions leaves open only self interactions and lepton/Higgs/right handed neutrino interaction, giving a general Lagrangian of

$$\mathcal{L} = \mathcal{L}_{SM} + i\bar{\nu}_R \partial\!\!\!/ \nu_R - \bar{l}_L F \nu_R \tilde{\Phi} - \bar{\nu}_R F^\dagger l_L \tilde{\Phi}^\dagger - \frac{1}{2}(\bar{\nu}_R^c M_M \nu_R + \bar{\nu}_R M_M^\dagger \nu_R^c) \quad (2.5)$$

Here the first non-standard model term is the kinetic energy and the second and third are interactions with the standard model; It describes an interaction between a right handed neutrino, a standard model lepton and a Higgs particle. The l is a standard model lepton doublet and F is a matrix of coupling constants where the indices (suppressed to save space) are the flavors of the lepton doublet and the flavor of the right handed neutrino (ν_R). The final terms are Majorana mass terms. M_M is a Majorana mass matrix, where the two flavor indices have again been suppressed. Usually an explicit mass term of this kind would be forbidden by gauge symmetry, but since right handed neutrinos transform as a singlet under every gauge transformation that doesn't apply here.

There are two kinds of terms in the Lagrangian that can give a fermion mass. One is a Dirac mass term, of the form $\bar{\phi}_L M_D \phi_R$, where ϕ is some fermionic field. The other is a Majorana mass term, of the form $\bar{\phi}_L M_M \phi_L$ or

$\bar{\phi_R} M_M \phi_R$. All the fermions we know about have Dirac masses, except the neutrino where we don't know. There are some experimentally measurable differences between the Dirac and Majorana mass neutrino scenarios. One of them is that a Majorana neutrino would allow for neutrinoless double beta decay. Extensive searches have been made for this with no success. This combined with the fact that we know neutrinos have mass and that a Dirac mass term for neutrinos is impossible without the existence of right handed neutrinos is a strong argument for their existence.

If we look more closely at the non-self interactions, $\bar{l}_L F \nu_R \tilde{\Phi}$ and its opposite, focusing first on the fact that in the low temperature limit we can replace the Higgs field with its vacuum expectation value, $\binom{0}{v}$. The first thing we see is that since the Higgs vacuum expectation value is purely in the neutral part of the doublet any interaction with a charged lepton becomes zero. Second, by actually making the substitution we get:

$$\bar{l}_L F \nu_R \tilde{\Phi} = \bar{\nu}_L F v \nu_R \tag{2.6}$$

Which is a Dirac mass term for the neutrinos, with a Dirac mass matrix $M_D = Fv$.

Unlike Dirac mass terms Majorana mass terms are independent for right and left handed particles. This means that even if the mass of the left handed neutrinos is solely from a Dirac mass term the right handed neutrinos can still have Majorana masses. This is vital for the possibility of right handed neutrinos explaining Dark Matter, as left handed neutrinos are far too light to be Dark Matter candidates. In the context of Dark Matter we are only interested in right handed neutrinos with a Majorana mass term and $M_M \gg M_D$[10]

2.4 Alternative to Dark Matter

Several alternatives to Dark Matter have been proposed for explaining the difference between galactic dynamics and apparent masses based on altering the laws governing the dynamics rather than the distribution of mass. Of these the most successful by far has been the theory called Modified

Newtonian Dynamics (MOND). It posits that for very weak gravitational fields (accelerations below $a_0 = 1.2 \cdot 10^8$ cm s^{-2}) the actual acceleration of gravity deviates from the one predicted by general relativity, decreasing linearly rather than quadratically with distance.

The greatest advantage of MOND is that it's a simpler theory than Dark Matter. Because it lacks the free parameters of the later (the distribution of Dark Matter) it cannot be fitted to individual galaxies. A single modification to the gravitational attraction must be able to explain the rotation curve of every Galaxy or the theory fails. The strongest empirical evidence for MOND is the fact that across many different galaxies the effect of Dark Matter fails to appear at accelerations above a_0 and increases with decreasing acceleration. This emerges naturally in MOND, but in any Dark Matter theory it is an enormous coincidence.

While MOND describes galaxies very well it fails to explain larger structures like Galaxy clusters, or cosmology. Further more MOND is in itself not a relativistic theory, which for many years made it little more than a curiosity. There have eventually been developed relativistic theories that produce MOND in the classical limit, but each of these introduce new difficulties. On the other hand the greatest advantage of cold Dark Matter (CDM) is that the standard model does not claim to describe all physics at all energies and we expect to find new particles outside the standard model while we have no prior reason to expect a modification of the laws of gravity[4].

Chapter 3

Overview of neutrino experiments and IceCube

3.1 Neutrino Astronomy

Neutrinos have several properties that make them useful for astronomical purposes. Since they only interact weakly they can pass through almost any medium without significant absorption. This is the reason that neutrino observations of the Sun have had such an important place in neutrino astronomy. Neutrinos, being the only particle created in the nuclear reactions in the Sun's core that can escape through to the surface, provide the only view of those reactions unaffected by the layers of the Sun above it.

Furthermore, because they don't interact electromagnetically they won't interact with the background microwave radiation or be deflected by magnetic fields and will only interact very weakly with interstellar gas. This is more valuable than it first seems; in some cases even open space isn't unobstructed, even for photons. When a photon hits another photon they can, if their total energy is above a certain threshold, create a particle anti-particle pair (an electron and a positron). This has a large cross section and photons energetic enough to experience it with cosmic background photons will not have a long mean free path on a cosmological scale[11]. Of course while its weak interactions allows it to travel to us unhindered it also makes neutrinos very difficult to detect.

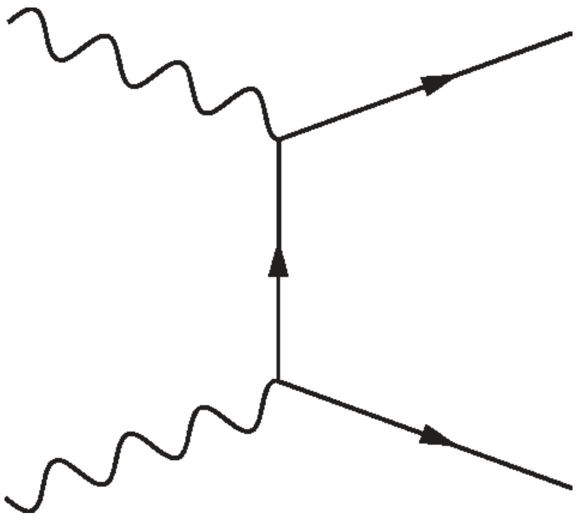


Figure 3.1: A high energy photon scattering off of a cosmic microwave background photon and creating an electron positron pair

Because of the small interaction cross sections of neutrinos the rate of neutrino interactions in the experiment will always be low. This means that any background of non-neutrino particles entering the experiment, usually high energy muons created by the scattering of cosmic ray protons in the atmosphere, will be large relative to the signal. Shielding of neutrino detectors is therefore extremely important. For this reason it's common to construct detectors deep underground. Old mines have been used for many neutrino detectors. Other detectors have been made deep under water, or in the case of IceCube deep in the antarctic ice.

Since detecting neutrinos is so difficult only large fluxes of neutrinos can be successfully detected. In 1987 a super-nova was detected in the Large Magellianic Cloud that was much closer to us than most super-novas. The neutrino burst from the super-nova was detected in several neutrino detectors, but at most 11 individual neutrinos were observed by a single detector. This number should give some perspective on the difficulty of neutrino detection. That super-nova is the only extra terrestrial object outside the Sun that has been observed by neutrino detection. Because of the very small chance of successfully detecting neutrinos neutrino astronomy is generally about diffuse fluxes. Sources are thought of in terms of detecting neutrinos from, for example, "extra galactic sources" rather than in terms of detecting neutrinos from some specific Galaxy, or cluster.

3.2 History of Neutrino detection

The first type of neutrino detectors built were based on induced beta decay. A large tank would be created and filled with a liquid containing a large amount of chlorine (for example perchloroethylene). When a neutrino hits the nucleus of a chlorine atom it could lead to one of the neutrons undergoing beta decay. This would transmute the chlorine to argon. By measuring the amount of argon in the tank after a certain time the number of neutrino events could be found. The Homestake, or Davis, experiment in the US first discovered solar neutrinos using this technique in the late 60s. The fact that the number of solar neutrino events detected was only a third of the theoretically predicted number lead to what was called the solar neutrino problem, and eventually to the discovery of neutrino oscillations.

At around the same time atmospheric neutrinos were first detected almost simultaneously at the Kolar Gold Fields detector in India and the East Rand Mine detector in South Africa. Both experiments used a similar method of neutrino detection. They were based on detecting muons created by neutrino scatterings in the rock surrounding the mines. Because the direction of cosmic ray muons was concentrated around the zenith the experiments looked at muons with incoming directions near the horizontal plane. The Kolar Gold Field (KFG) experiment was originally intended to study muons created from cosmic ray scattering in the atmosphere. However, when the deepest part of the detector failed to detect any muon events it was realized that this area would be a good place to study very low cross section events like neutrino interactions. Both detectors detected atmospheric neutrinos for the first time within months of each other.

The idea of neutrino detection by Cherenkov radiation in a volume of water was first suggested by Markov in 1960. The actual implementation of the idea however came somewhat later than the chemical detectors mentioned previously. The first generation of detectors based on this principle (most notably Kamiokande in Japan and Irvine-Michigan-Brookhaven in the US) were built around 1980, using tanks of water with photo-detectors along the tank walls. Originally these were not intended primarily as neutrino detectors. Their ability to serve as neutrino detectors was a side effect of their intended purpose of searching for proton decays. Grand Unified Theory (GUT) proposed in the seventies predicted that baryon number would not be perfectly conserved and as a result protons would decay with a life time of between 10^{30} and 10^{34} years. Existent neutrino detectors were able to set a lower limit on the life time of the proton only at 10^{30} years, so a series of specialized proton decay experiments were commissioned to probe the range of theoretically predicted life times. These experiments were not able to observe proton decays, but they did produce very useful data on atmospheric neutrinos[13].

The success of these experiments allowed a scaled up second generation of water Cherenkov detectors to be built. This included Super-Kamiokande which used a tank holding 50,000 tons of water, compared to the 3,000 tons used by the original Kamiokande. In 1998 the Super-Kamiokande collaboration published a definite proof of neutrino oscillations[14].

It was known from the very envisioning of neutrino astronomy that because of the low interaction cross sections of neutrinos the success of neutrino observatories would be limited by their size. Because larger water tanks imply increased pressure and weight, and larger engineering challenges the idea of placing photo detectors into existent bodies of water in order to convert some volume of a lake or ocean into a neutrino detector was proposed early on. The DUMAND project, which developed during the 70s, was the first attempt at constructing such a detector. The goal of the project was to create a neutrino detector at almost 5.000 meters depth

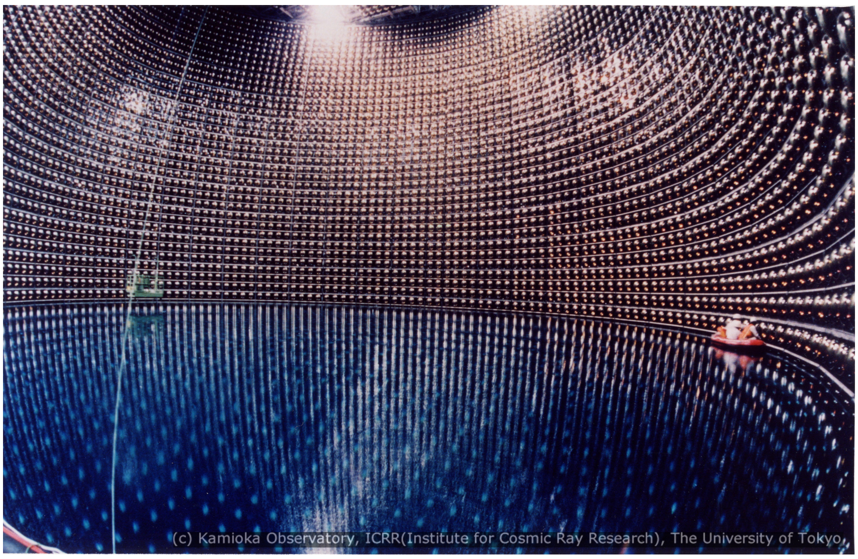

(c) Kamioka Observatory, ICRR(Institute for Cosmic Ray Research), The University of Tokyo

Figure 3.2: The inside of the Super-Kamiokande detector tank during maintenance. Of course there can be no photo of it in operation as the tank must be completely dark to allow Cherenkov radiation to be detected. The light spheres covering the tank walls are photo detectors[17].

in the ocean near Hawaii. A large amount of work was put into creating the technology required to operate the required measuring apparatuses at the pressures involved, and to deal with the challenges of ocean currents etc. Funding for the project was canceled in '95 after a number of sensors were destroyed by water pressure, but the knowledge the project had created by then was used to create several follow up projects.

Another notable and more successful attempt at an underwater neutrino detector is the Baikal Neutrino Telescope, in lake Baikal, Siberia. It was built in considerably shallower water, at around 1,200 meters depth. That means a larger background, but lessens the engineering challenges somewhat. A more direct successor to DUMAND is the ANTARES detector built at around 2.5 kilometers depth in the Mediterranean near Toulon.

3.3 IceCube

The IceCube neutrino observatory turns a cubic kilometer of the antarctic ice cap into a huge neutrino detector. It is the first detector that really showcases the ability to scale up detectors that don't rely on artificial bodies of water, being 20,000 times larger than Super-Kamiokande. The detector has a total of 5160 optical sensors, divided among 86 vertical strings that are placed in a hexagonal pattern. IceCube is the first observatory to use Cherenkov radiation in ice to detect neutrinos.

IceCube is a Cherenkov radiation based detector. Cherenkov radiation is an effect similar to a sonic boom. It happens when a charged particle moves through a medium at a higher speed than the speed of light in that medium.(figure: 3.3) When that happens the electromagnetic fields in front of the particle are unable to adjust to the particle's movement fast enough and a wave front of electric field is created. This results in a large amount of radiation being emitted. The optical sensors distributed throughout the ice are intended to detect this radiation.

Since the neutrinos themselves are electrically neutral they don't produce any Cherenkov radiation. The detector relies on neutrinos scattering off of atomic nuclei in the ice creating charged leptons. Additionally the neutrino can deposit enough energy in the nuclei to create a hadronic shower. Neutral current scatterings can only be detected by the hadronic shower as these results in an outgoing neutrino rather than charged lepton.

The IceCube detector has some ability to determine the flavor of a detected neutrino. A muon neutrino can create a muon that travels a considerable distance and can be seen as a track in the detector. On the other hand electron and tau neutrinos will appear as point-like events in the detector; electrons lose their energy very quickly, while the taus decay similarly fast. Hadronic showers created by energy deposited in the nucleus the neutrino scatters on will likewise appear as a point. It is much easier for the detector to accurately determine the direction of a muon track, but even for point-like events the direction can be determined to a degree as the light emitted is not isotropic, but is determined by the direction of motion.

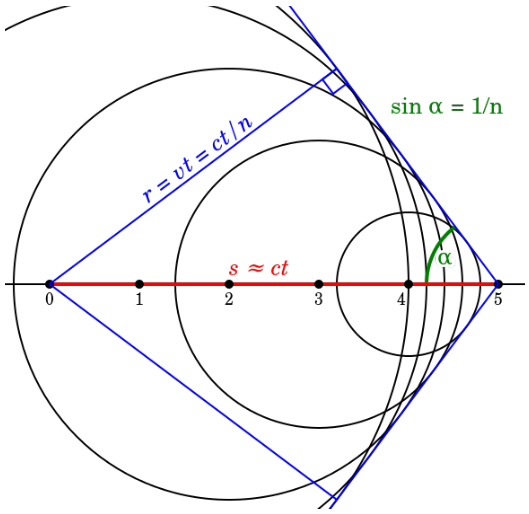


Figure 3.3: The ”wake” of light following a charged particle through a medium faster than the speed of light in that medium[15].

Previous detectors based on similar ideas as IceCube have all used water rather than ice. There are several advantages and disadvantages to both. Ice has the advantages of not having a background of light from biolumineсence or radioactive trace elements, and of not having to account for detectors being nudged by currents.

IceCube is the first ice based neutrino observatory. It is more common to use water rather than ice and IceCube builds on the experience from several water based observatories. Water offers better optical conditions than ice, like shorter scattering length, however for the IceCube experiment it was decided that the benefits of ice outweighed the downsides: reduced background due to absence of bioluminescence and not having to account for sensors being nudged by currents.

Because of the long tracks created by the muons the IceCube experiment has by far best angular resolution for muon-neutrino detection. Unfortunately this is also the flavor with the largest atmospheric background.

Of muon tracks in the experiment muons from showers created by cosmic rays in the atmosphere outnumber neutrino produced muons by a factor of 500,000:1. The outer most strings of optical sensors, as well as the Ice-Top and DeepCore experiments, respectively above and below IceCube are used to attempt to discard external muon events. However, even among neutrino events there's a large background of atmospheric muon neutrinos that makes neutrino astronomy with this flavor difficult. The best solution is to look only at events with energy above the cut off energy of atmospheric neutrinos.

In hadronic shower events the most Cherenkov light is created at the Cherenkov angle (ca 41 degrees), but due to scattering in the ice much of this directional information is lost before detection.

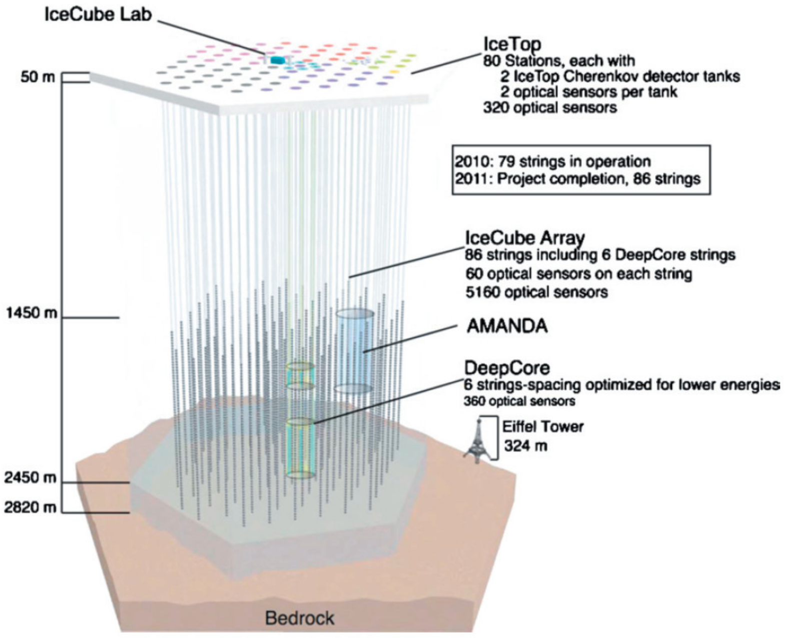

IceCube Lab
IceTop
80 Stations, each with
2 IceTop Cherenkov detector tanks
2 optical sensors per tank
320 optical sensors
2010: 79 strings in operation
2011: Project completion, 86 strings
IceCube Array
86 strings including 6 DeepCore strings
60 optical sensors on each string
5160 optical sensors
AMANDA
DeepCore
6 strings-spacing optimized for lower energies
360 optical sensors
Eiffel Tower
324 m
50 m
1450 m
2450 m
2820 m
Bedrock

Figure 3.4: Diagram of the IceCube experiment

Chapter 4

Method

4.1 Maximum Likelihood Estimation

When given a set of data and a model with free parameters one might want to find which values of the parameters best fit the observed data. One method for finding this is called Maximum Likelihood Estimation. Likelihood can be seen as probability looked at backwards. When talking about probability we have a fixed model and want to find out how probable different outcomes are under that model. On the other hand, when talking about likelihood we have a fixed outcome and want to find how likely that outcome was under different models. The likelihood of the outcome under some model is the same as the probability the model would assign to that specific outcome in advance.

In maximum likelihood estimation we have a field of models defined by a parameter, or a set of parameters, θ. We assume that the data is produced by the model described by some true value of θ, which we call θ_0. For some value of θ we define $f(x_1, x_2, x_3 ... x_n|\theta)$ as the likelihood of a specific set of n observations, $x_1, x_2, x_3 ... x_n$, according to the model described by θ. What we want to find is $P(\theta|x_1, x_2, x_3 ... x_n)$, the probability of θ being θ_0 given our set of observations. According to Bayes' Rule P and f are related by

$$P(\theta|x_1, x_2, x_3...x_n) = \frac{f(x_1, x_2, x_3...x_n|\theta)P(\theta)}{P(x_1, x_2, x_3...x_n)} \tag{4.1}$$

where $P(\theta)$ is the prior probability of θ being the correct value of the parameter(s) and $P(x_1, x_2, x_3...x_n)$ is the prior probability of seeing that specific set of observations. The later is independent of θ so if one merely wishes to find the most likely set of parameters given the data, rather than how likely those parameters are, then the denominator can be ignored. And if one further assumes that $P(\theta)$ is uniform (that you have no prior information pointing towards any value of θ) then

$$P(\theta|x_1, x_2, x_3...x_n) \propto f(x_1, x_2, x_3...x_n|\theta), \tag{4.2}$$

and it's merely necessary to find the value of θ that maximizes $f(x_1, x_2, x_3...x_n|\theta)$.

4.2 Coordinate systems

We will be making use of two coordinate systems here: equatorial coordinates, based on the sky as seen from Earth, but fixed relative to the stars rather than rotating with the Earth, and the galactic coordinate system based on the Suns position in the Galaxy.

The equatorial coordinate system has the center of the Earth as the origin. The coordinate system was created to have a system where the stars have fixed coordinates. So from the point of view of any observer on Earth the system appears to be rotating as the Earth rotates around its axis. The system is a spherical coordinate system: coordinates are given in terms of two angles, right ascension and declination. The celestial equator is found by taking the projection of the Earths equator onto the sky. A point's declination gives its angle north or south of this celestial equator (so the zenith above the north pole has declination $90°$ and the zenith at the south pole has declination $-90°$). The right ascension meanwhile gives a position along the celestial equator. Zero right ascension is the point where

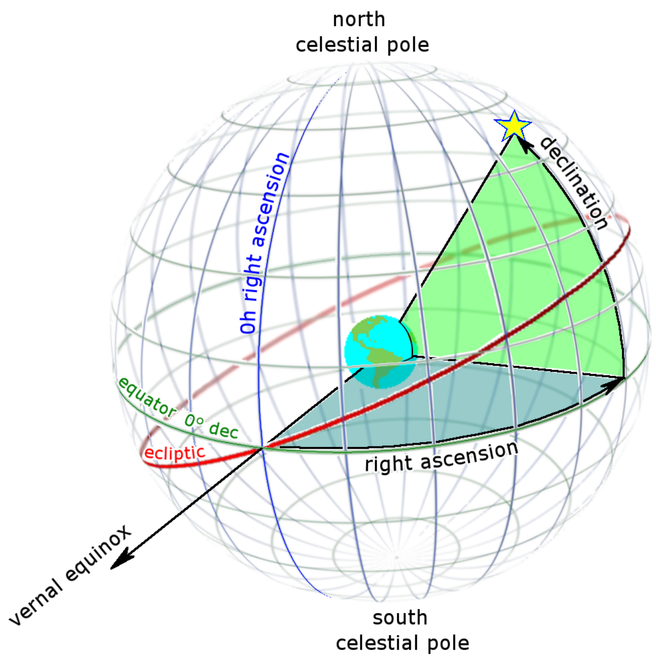


Figure 4.1: Diagram explaining the equatorial coordinate system. The red line is the Earth's orbital plane around the Sun[19].

the Sun crosses the celestial equator during the spring equinox and positive direction is towards the east.

By comparison to the equatorial system the galactic coordinate system is much simpler. It has its center in the Sun rather than the Earth, although the distance is too short on a galactic scale to make any difference at all to us. The system uses two coordinates, galactic longitude and galactic latitude. Galactic latitude gives a direction in the galactic plane, with longitude 0 being in the direction of the galactic center. Galactic longitude gives an angle above or below of the galactic plane. Positive direction is in the direction of the galactic north, as defined by the Galaxy's rotation.

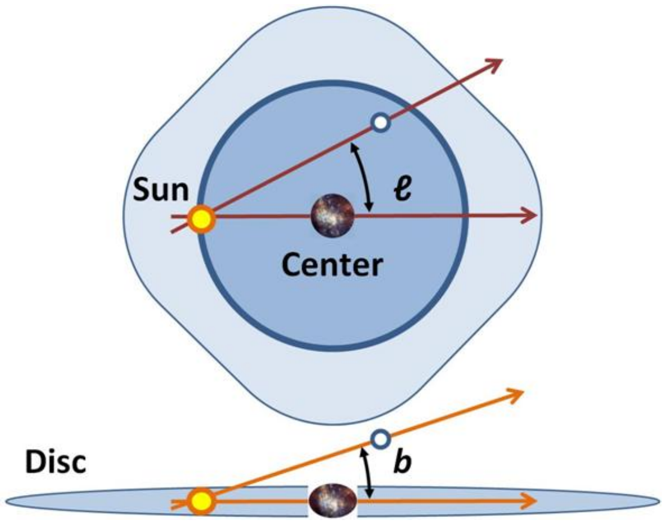


Figure 4.2: Diagram showing the meaning of galactic longitude l and galactic latitude b in the galactic coordinate system[18].

4.3 Model

The core of the model we use is the idea that the extra-stellar neutrino flux can be broken into three contributions, those from luminous matter in the Galaxy, those from Dark Matter in the Galaxy and those from all extragalactic sources. Because of the isotropy of space at large distances the last is assumed to be isotropic. The Dark Matter contribution is assumed to be spherically symmetric around the Galactic center. And the contribution from luminous matter in our Galaxy (from here on called the Galactic contribution) is assumed to be concentrated around the Galactic plane.

More specifically the density of neutrino sources in the Galactic contribution is assumed to be normally distributed around the Galactic plane when moving normal to that plane, with a density in the Galactic plane determined by the distance from the Galactic center.

$$\rho_{\mathrm{gal}} = \rho_0 (r/R_0)^{0.7} e^{-3.5\frac{r-R_0}{R_0}} e^{-(\frac{Z}{Z_0})^2} \tag{4.3}$$

where r is the distance from the galactic center in the galactic plane, $R_0 = 8.5\text{kpc}$ is the distance of the Sun from the galactic center and Z_0 is a characteristic length of the model, assumed to be $Z_0 = 210\text{pc}$ in our case. The use of the Suns orbital distance doesn't mean, obviously, that the Suns orbital distance has some special significance, it is merely a convenient scale constant. The basis for this density distribution is the approximate density of supernova in the Galaxy.

For the Dark Matter contribution we assumed the Navarro-Frenk-White distribution,

$$\rho_{\text{DM}} = \rho_0[r(1 + r/R_{DM})^2]^{-1}, \tag{4.4}$$

where r is the distance from the galactic center and R_{DM} is a characteristic scale of the model, set at $R_{\text{DM}} = 15\text{kpc}$ in our case.

However while our models describe the density of sources in space, the observations are the directions of incoming neutrinos. In order to find the likelihood of an observation under a model we must project the density predicted by the model onto the Earth's sky. This is where we benefit from the equatorial coordinate system being fixed against the sky. For a point in the sky the likelihood of an incoming neutrino coming from that direction will be proportional to the integral of the density of sources along a line extending from the Earth in that direction. Doing this allows us to transform the model of the galactic and Dark Matter contribution from distributions in space to distributions on the sky, the uniform extra-galactic contribution is of course trivial.

The theoretical expression for the projection of the density ρ onto the sky is:

$$D(\alpha, \delta) = \int_0^\infty \rho(\alpha, \delta, s)\mathrm{d}s, \tag{4.5}$$

where s is the distance from the earth. In order to calculate it we need to approximate it as a finite sum. First we choose a number of integration points N and then a cutoff point, in our case 85kpc. That's roughly 4 times the radius of the galaxy and the integral will cover the entire galaxy. The expression then becomes:

$$D(\alpha, \delta) = \sum_{n=0}^{N} \rho(\alpha, \delta, \frac{n}{N} 85kpc)/N, \tag{4.6}$$

which is what the program actually evaluates.

In addition to this spatial information we also want to make use of the energy information given by IceCube. Firstly we must define the energy spectrum, $\frac{dN}{dE}(E)$, that we expect to see from each of our sources. For the Galactic and extra galactic contributions we use simple power laws, $\frac{dN}{dE}_{\text{gal}}(E) \propto E^{-2.5}$, based on the spectrum of γ-ray observations, and $\frac{dN}{dE}_{\text{extgal}}(E) \propto E^{-2.2}$, based on what's typical for cosmic ray acceleration. For the Dark Matter decay spectrum I used first a power law with , $\frac{dN}{dE}_{\text{DM}}(E) \propto E^{-1.8}$. Later I changed to a more detailed model for the Dark Matter based on the modeling of the decay of super heavy particles into leptons in [22].

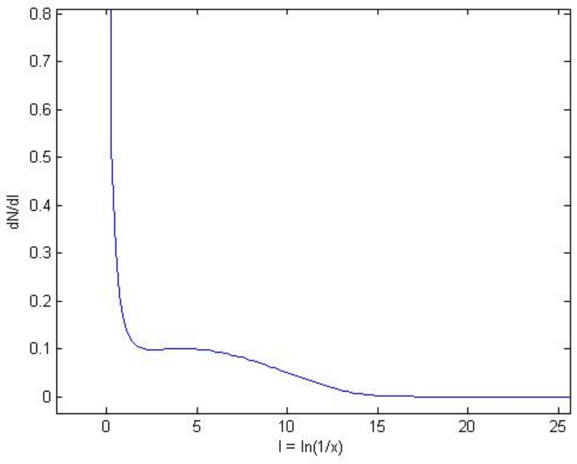


Figure 4.3: Graph showing the model used for the energy spectrum of decaying Dark Matter.[22]

While this describes our model for the neutrino flux from remote sources we also have to keep in mind that the methods used at IceCube to eliminate background events is not infallible. So we want our model to account for the possibility that some fraction of the events are atmospheric neutrinos, created by cosmic ray particles hitting the atmosphere. These we expect to be isotropically created and we have to use the energy dependence to seperate this from the extra-galactic contribution.

4.4 Implementation

Given the three distributions (D) on the sky, $D_{DM}(x)$, $D_{\text{gal}}(x)$ and $D_{\text{extgal}}(x)$, we have a model of three parameters $\theta = [\theta_1, \theta_2, \theta_3]$, limited by $\sum_i \theta_i = 1$, where the likelihood of an incoming neutrino coming from a given direction $x = (RA = \alpha, dec = \delta)$ is given by

$$f_{\text{sp}}(x|\theta) = \theta_1 \cdot D_{\text{DM}}(x) + \theta_2 \cdot D_{\text{gal}}(x) + \theta_3 \cdot D_{\text{extgal}}(x) \tag{4.7}$$

If we also include energy then the likelihood of an incoming neutrino from a specific direction with a specific energy, $x = (\alpha, \delta, E)$ is given by

$$\begin{aligned} f_{\text{model}}(x|\theta) = \theta_1 \cdot D_{\text{DM}}(x) N_{\text{DM}}(E) + \theta_2 \cdot D_{\text{gal}}(x) N_{\text{gal}}(E) \\ + \theta_3 \cdot D_{\text{extgal}}(x) N_{\text{extgal}}(E) \end{aligned} \tag{4.8}$$

Adding finally the possibility that some arbitrary fraction of the neutrino observations being due to a uniform atmospheric background with a spectrum $N_{bg}(E)$:

$$f(x|\theta) = \int_0^1 d\lambda(\lambda \cdot N_{\text{bg}}(E) + (1-\lambda) f(x|\theta)) \tag{4.9}$$

In our case each observation is independent and assumed to be taken from the same distribution. That means that $f(x_1, x_2, x_3 ... x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta)$.

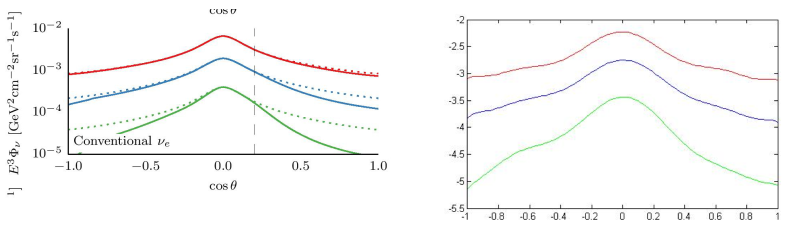


Figure 4.4: The sensitivity of IceCube to electron neutrinos at 1, 10 and 100TeV (red, blue and green respectively). On the left is the original data from [21]. On the right is my function evaluated at the same energy and the resulting polynomials plotted against $\cos(\delta)$.

Each observation consists of five facts, right angle and declination giving the direction of the neutrino, an error estimate for the former, an energy measurement and the flavor of neutrino (track or shower).

The sensitivity of IceCube, it's probability of detecting a given neutrino, is dependent on the declination (δ), energy (E), and flavor (ν_α) of the neutrino, $S(\delta, E, \nu_\alpha)$. In order to create this function I looked at the plot of IceCube sensitivity in [21], which shows a plot of the IceCube sensitivity against declination at 1TeV, 10TeV, and 100TeV for both electron and muon neutrinos, on a logarithmic scale. I choose a number of points from each graph and used interpolation to create a 10th degree polynomial in the cosine of the declination that matched each graph quite well. To find a sensitivity at an arbitrary energy I interpolated the three values for the coefficients to create a second order polynomial in the log energy for each coefficient. The coefficients thus determined were then used to find the sensitivity as a function of declination. This is the only place where flavor is used in our calculations.

The experimental uncertainty in the neutrino measurements is another factor to consider. If each observation was completely certain then the like-

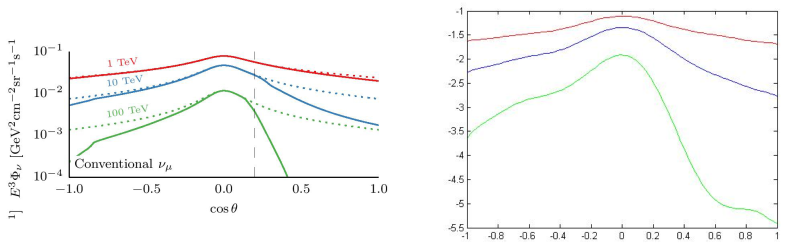


Figure 4.5: The sensitivity of IceCube to muon neutrinos at 1, 10 and 100TeV (red, blue and green respectively). On the left is the original data from [21]. On the right is my function evaluated at the same energy and the resulting polynomials plotted against cos(δ).

	e_i^2	e_i^1	e_i^0
c^{10}	1.4982	-1.2834	-16.848
c^9	-0.65172	2.1804	-0.1432
c^8	-2.1716	0.97143	48.638
c^7	1.1493	-5.2452	0.7870
c^6	-0.15176	1.6071	-52.264
c^5	-0.48987	4.464	-1.0083
c^4	1.3081	-1.6791	26.01
c^3	0.064428	-1.6051	0.42918
c^2	-0.64372	0.3115	-6.4204
c^1	-0.025337	0.13563	-0.075782
c^0	-0.082567	-0.4379	-2.2301

Table 4.1: Approximate icecube sensitivity to electron neutrinos $S(E,\delta)$:
$c^i(E) = e_i^2 \log_{10}^2(E) + e_i^1 \log_{10}(E) + e_i^0$
$S(E,\delta) = \Sigma_{i=0}^{10} c^i(E) \cos^i(\delta)$

	$log_{10}^2(E)$	$log_{10}(E)$	1
$cos^{10}(\delta)$	4.6898	-8.6509	-6.1085
$cos^{9}(\delta)$	5.2739	-2.9573	-0.17973
$cos^{8}(\delta)$	-11.15	23.276	17.835
$cos^{7}(\delta)$	-14.665	8.5211	0.33336
$cos^{6}(\delta)$	7.1089	-21.459	-19.653
$cos^{5}(\delta)$	14.047	-8.1772	-0.083802
$cos^{4}(\delta)$	0.74176	7.6195	10.313
$cos^{3}(\delta)$	-4.8159	2.554	-0.11426
$cos^{2}(\delta)$	-1.7929	-1.0146	-2.934
$cos^{1}(\delta)$	-0.057557	0.068357	0.014327
1	-0.16983	-0.063039	-1.1108

Table 4.2: Approximate icecube sensitivity to muon neutrinos

lihood of an events would simply be the likelihood of an event at that precise point given the distribution. When each observation is uncertain we have to sum the likelihood of a particle coming from every direction weighted by the likelihood that a neutrino observed in the observed direction was really coming from this other direction, based on the error estimate. That is to say, for $x = [\alpha, \delta, \sigma, E, \nu]$ and $\theta = [\theta_{DM}, \theta_{\text{gal}}, \theta_{\text{extgal}}]$:

$$f(x|\theta) = \int\int d\alpha' d\delta' S(\delta', E, \nu_\alpha) \cdot \exp\left(-\frac{[\Omega(\alpha, \delta, \alpha', \delta')]^2}{2\sigma^2}\right) \cdot D(\alpha', \delta') \quad (4.10)$$

where $\Omega(\alpha, \delta, \alpha', \delta')$ is the angle between (α, δ) and (α', δ'),

$$\Omega(\alpha, \delta, \alpha', \delta') = \cos^{-1}(\sin(\alpha)\sin(\alpha') + \cos(\alpha)\cos(\alpha)\cos(\delta - \delta')). \quad (4.11)$$

4.5 Programming

4.5.1 Overview

The program I wrote to implement this model was written in C++. A great deal of attention was placed on making it easy to reuse the program with different resolution, integration step length, models, etc with as little work required as possible. This was done both so as to be able to easily deal with changes to the project during development and in the hope that the program could be useful beyond the scope of this thesis. Most of the computations carried out in the program are done by various functions grouped in different files by topic; one file containing all functions relating to the coordinate systems used and transformations between them, and so on. This is both to make the program more readable, and easier to maintain, modify and use. The one thing that is ”hard coded” is the number of different contribution models considered. Changing that would take significant rewriting of how the program works. Other changes, like what the models for the different sources are like, require only that you change some easily accessible functions or constants. The program consists roughly of three parts, one part that reads in information and calculates a number of things that will be used later, a main loop that does the actual calculation and a final part that saves the result in a MATLAB readable format. The main loop takes the vast majority of the computational resources. The preliminary section contains the most code (as there are so many tasks that are done here), but is executed fairly quickly. The final section is light in both code and computation time.

4.5.2 Preliminary calculations

Because the problem being solved is much more demanding in terms of computing power than memory I have tried to move as many calculations as possible out of the main program loop. Computing and saving certain results ahead of the main loop prevents unnecessary repetition of calculations. In a different computation where memory is more of a bottleneck it might be better to avoid keeping large amounts of data stored in memory, but that has definitely not been the case here.

At the very start of the program a couple of parameters are set, one that defines the resolution used to divide the sky into a grid and one that defines the number of steps in parameter space (the granularity when varying the contributions from the different sources). Also set here is the name of the file that the events are read from.

After these parameters are set the program begins reading in the event data. The functions used to read the event data assume that the data follows a certain format; a number of lines, each of which describes a single observation, followed by at least one blank line and then an explanation of the data format. The first file reading function is intended to find the number of events described by the event file. It is given the file name set as a variable previously and attempts to open any file with that name in the same file location as the program is run from. It creates a counter and attempts to read the file line by line. If the line read is not empty the counter is increased by one and the function moves on to the next line. If the line is empty the function stops and returns the final value of the counter to the main program where it's saved in a variable. This variable will be used to allocate the correct amount of memory when the rest of the event data is read in and to loop over all the events in the main computation loop.

Each line in the event file, that is each set of observation data, is assumed to follow a format where energy (in TeV), declination (in degrees), right ascension (in degrees), median angular resolution (degrees), and whether the event was a track or shower event, are given in that order with spaces between. The first four are given as numbers while the last is given as a letter, t for track or s for shower. The next function called in the program reads and saves the numerical (that is all but the last) data. It is called with the file name and the number of events found before.

The first thing it does is to create a vector of vectors of floating point numbers. This vector of vectors contains four vectors, one for each of the numerical attributes of an observation, that each has a length equal to the number of events. Note that a vector of vectors is how a matrix is represented in C++ and I'll use the term matrix interchangably with this. The program then reads the file line by line for a number of lines equal to the number of events. At the nth line it divides the line into substrings based on the spaces, with the string from the start to the first space assumed to correspond to the energy, the substring from the first non-space character after the first space to the first space after that assumed to be the declination and so on. It then attempts to parse these substrings as numbers and sets the nth value of the corresponding vector to that number. Once it has read the number of lines instructed it returns the vector of vectors to the main program, where it is saved for later use.

A second similar function creates a vector of characters, then reads each line, takes the last character on the line and saves it in the vector. In retrospect it would probably have been a better idea to assign a numerical value to "track" and "shower". This would have make it possible to have all the information about the events in a single matrix. Also note that while this system for reading the data can handle an arbitrary number of spaces between the data it will fail if there are spaces before or after the data in a line. The reason for having vectors for each attribute (energy, declination, etc.) rather than vectors for each event is twofold: When the same quantity is calculated for all the events using this set up means less jumping around and therefore faster execution, and secondly because if some function is only interested in one or two attributes of the events then it's trivial to just give it those specific vectors as input. This simplifies some of the programming significantly.

The sensitivity (the chance of an incoming neutrino being successfully detected), is as previously mentioned determined by the energy, the declination, and the flavor of the neutrino. Since the number of events is so low there's no reason at all to consider the possibility of repeat energies, so the sensitivity has to be computed for every declination value for every event. So the next step in the program calls a function, giving it the vector of energies, the vector of flavors and the number of declination steps, which creates a matrix with dimensions of number of events by number of declination steps. Then it looks at each event, and sends its energy to one of two functions depending on its flavor. These functions take the log10 of the energy and create a vector of coefficients, as described in table 4.1 and table 4.2. The outer function then takes the vector of coefficients and loops over the declination values, taking the cosine, evaluating the resulting polynomial and saving it in the appropriate place in the matrix. Once this has been done for every event the matrix is returned to the main program, which saves it for later use.

After this the program calculates the relative likelihoods of neutrinos from each of the sources having each of the energies observed. This is saved in a matrix of size number of events by number of sources. Atmospheric background is considered as a source here for four total.

Next the program creates a matrix for each of the sources (Dark Matter, Galactic, extra-galactic) and populates it with the projection of the distribution of sources onto the sky. The extra-galactic term is merely uniform, so that function just creates a normalized uniform matrix. The other two functions take the indices of a matrix element, find the coordinates in equatorial coordinates that this corresponds to and then translates those coordinates into galactic coordinates. Each then starts incrementing a variable representing the distance from Earth, with the size of the steps set at the start of the program, for each value of the distance from Earth it evaluates the source density as a function of direction in the sky and distance. This is then multiplied by the step length in distance and summed up to some maximum distance much greater than the size of the Galaxy. Once this has been done for every entry in the matrix the functions returns the matrices to the main program.

4.5.3 Main loop

After the preliminary calculations the program enters the main loop. First a matrix is created to store the result, with the size determined by the number of steps in the parameter space (p), as set at the beginning of the program. While there are three parameters (the contribution of exragalactic background, the contribution from galactic luminous sources and the contribution from Dark Matter) the requirement of normalization means there are only two independent parameters and a two dimensional matrix is large enough to accomodate the result. I chose to use the Dark Matter parameter (i) and galactic parameter (j) as the variable parameters and the extra-galactic parameter as the derived parameter, but this is arbitrary. The main program loop is a nested loop over the Dark Matter and galactic parameters. The first should go from zero to one and the second from zero to one minus the first, but in order to get integer values I multiplied them by p and then divide by the same for the actual calculations. That means i goes from zero to p. In the inner loop, for each step in the outer loop, j loops over the integers from 0 to the difference between p and i.

For each set of possible parameters the program loops over all the events. For each event the program calls a function which is given all the matrices describing the source distributions, the parameters, the parameter resolution and the relative likelihood of each of the sources sending a neutrino at the energy observed for that event($n_h(E)$, where $h = gal, extgal, dm$). This function creates a matrix where element m,n is given by:

$$\begin{aligned} D_{i,j}(m,n) = \frac{i}{p} n_{DM}(E)\ D_{DM}(m,n) + \frac{j}{p} n_{gal}(E) D_{gal}(m,n) \\ + \frac{p-i-j}{p} n_{extgal}(E) D_{extgal}(m,n) \end{aligned} \tag{4.12}$$

Then, once the final matrix describing the sources has been made a further loop is run, looping over the background parameter b. This runs from 0 to p. At each step a final matrix $D_{final}(m,n)$ is computed, where

$$D_{final}(m,n) = (p-b)/p \cdot D_{i,j}(m,n) + b/p \cdot n_{bg} \cdot 1/l^2 \cdot S(n), \tag{4.13}$$

and l is the size of the matrix and $S(n)$ is the sensitivity. For each step likelihood of the event is calculated. For every position in the matrix a function is called which translates the indices into coordinates. Then a second function is called which computes the angular distance between those coordinates and those recorded for the event. We'll call this distance O. The likelihood of the event given this specific level of background is then

$$\sum_{m=0}^{l} \sum_{n=0}^{l} D_{final}(m,n) \frac{1}{2\pi\sigma^2 l^2} \exp(-\frac{O^2}{\sigma^2}) \tag{4.14}$$

where σ is the standard error reported by IceCube. This is summed for all the steps in the loop to give the likelihood for any level of background. Once that is done the logarithm is taken and the result is added to (i,j) in the result matrix. The reason is that the joint likelihood of the events is the product of the individual event likelihoods. Which is equivalent to summing their logarithms. Then the program moves on to the next event in the loop and so on.

4.5.4 Output

Once the main loop of the program has created the matrix of likelihoods a final function is called to print that result. It creates a MATLAB script file that when run will recreate the same matrix in MATLAB. First it creates a file called ”result.m”. Then it writes to that file ”X = [”. After that it starts looping through the matrix. For each row it loops over all the columns and for each column writes into the file the value in that position in the matrix followed by a space. After all the columns have been looped over it writes a semi colon before moving on to the next column. Once the last row is finished it writes ”];”, closes the file and the program ends.

Chapter 5

Results

During the project a number of different versions of the calculation were done as more details were added to the model. I will give a very brief overview of the earlier results before looking in more depth at the final result.

The first version of the computation I did looked only at the spacial distribution of the neutrinos. This computation found that a fully extra-galactic (that is to say uniform) distribution was favored. Adding a Dark Matter component would see the likelihood slowly decline while for any amount of Galactic contribution the likelihood fell rapidly. Because the likelihood of a set of events is the product of the individual likelihoods I decided that the geometric mean was the best way to define the average event. That means multiplying the likelihoods of the events and taking the n-th root, where n is the number of events. In the first calculation the likelihood of the average event under the assumption of a pure Dark Matter contribution was 76% of the likelihood of the average event under the assumption of a purely extra-galactic contribution.

The results of my computation strongly favored both the extra galactic distribution and the Dark Matter ansatz distribution over the galactic distribution. Overall the isotropic, extra galactic, distribution was favored,

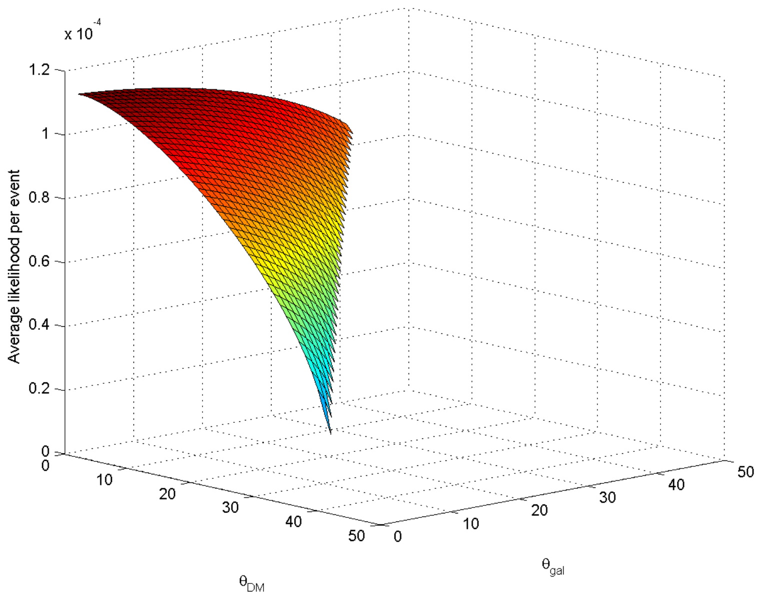


Figure 5.1: Likelihood as a function of distribution parameters, considering only the spatial distribution.

but the difference with the Dark Matter distribution was much smaller. I considered the geometric mean of the likelihood of the events the most useful way to define the average event. So I computed and compared this under the different parameter scenarios. I found that the average event was about 76% as likely under the assumption of a purely Dark Matter distribution as it was under a purely isotropic one. For 32 events, as were in our sample, this means the overall event series was about 60 times likelier under the assumption of isotropy than under the pure Dark Matter distribution.

Adding an energy dependence in the likelihoods happened in two stages. First a power law was assumed for each of the three source contributions. Very little effect. There was no qualitative change in the shape of the distribution, but the relative change near the Galactic corner was significant

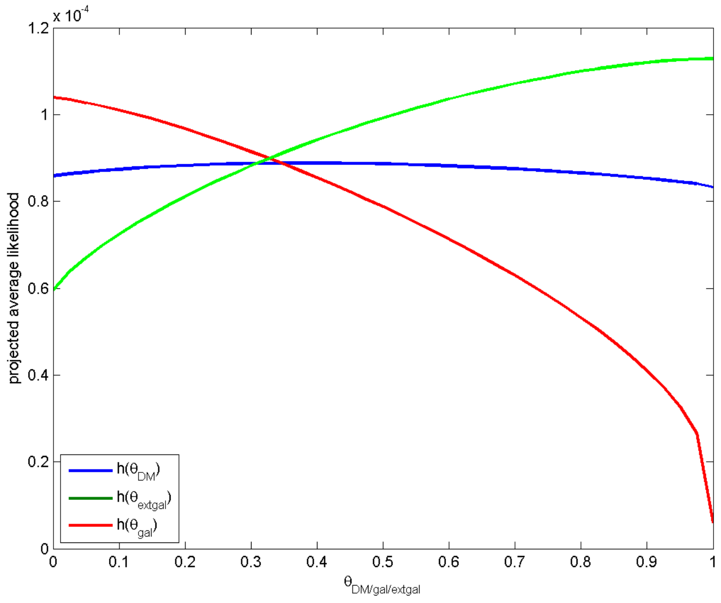


Figure 5.2: Likelihood as a function of distribution parameters, projected onto each axis in turn, considering only the spatial distribution. If $L(\theta_{\text{extgal}}, \theta_{\text{gal}}, \theta_{\text{DM}})$ is the likelihood of the average event in the set given those parameters, then the green line is $h_{extgal} = \int_0^{1-\theta_{\text{extgal}}} d\theta_{\text{gal}} L(\theta_{\text{extgal}}, \theta_{\text{gal}}, 1 - \theta_{\text{extgal}} - \theta_{\text{gal}})$ plotted against θ_{extgal}, while the blue line is similar for the Dark Matter contribution and the red line for the galactic contribution.

(between a factor 5 and 6). However the value in this area was very low in both cases, so a large relative change isn't necessarily important. On the extra-galactic/dark matter axis the largest increase was in the dark matter corner, where the average likelihood was increased by 13%. The purely extra-galactic corner was unchanged. Figure 5.5 shows the changes from

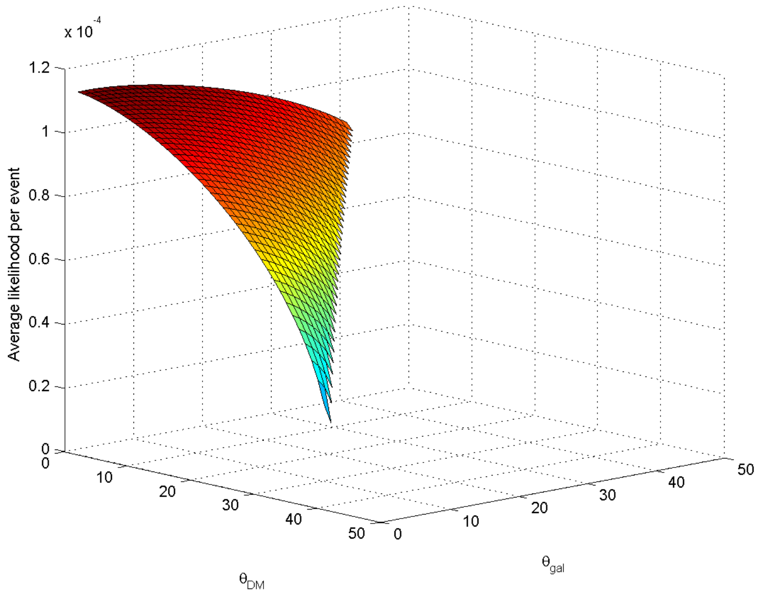


Figure 5.3: Likelihood as a function of distribution parameters, considering the energy dependence via a power law for each source.

adding the energy dependence.

The next step was to replace the power law for the dark energy source with a more detailed model. This didn't cause huge changes either, but it did move the maximum of the likelihood slightly away from a purely extra-galactic one. The movement happened along the extra-galactic-Dark Matter axis and any Galactic contribution at all still decreased the likelihood. The maximum of the likelihood was now at 92.5% extra-galactic, 7.5% Dark Matter. The likelihood here, per average event was 0.9% higher than for the case of purely extra-galactic.That would mean that the entire set of 36 events would have about 39% higher likelihood than under the

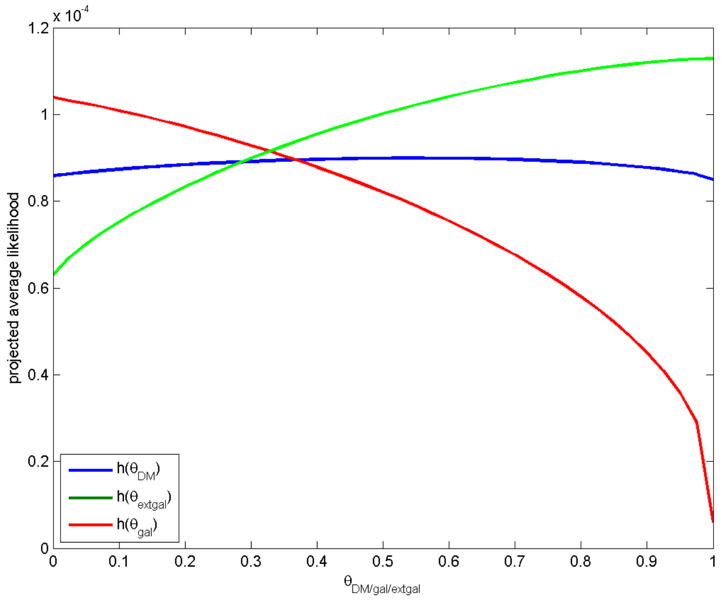


Figure 5.4: Likelihood as a function of distribution parameters, considering the energy dependence as a simple power law for all the sources. If $L(\theta_{\mathrm{extgal}}, \theta_{\mathrm{gal}}, \theta_{\mathrm{DM}})$ is the likelihood of the average event in the set given those parameters, then the green line is $h_{extgal} = \int_0^{1-\theta_{\mathrm{extgal}}} d\theta_{\mathrm{gal}} L(\theta_{\mathrm{extgal}}, \theta_{\mathrm{gal}}, 1 - \theta_{\mathrm{extgal}} - \theta_{\mathrm{gal}})$ plotted against θ_{extgal}, while the blue line is similar for the Dark Matter contribution and the red line for the galactic contribution.

purely extra-galactic theory.

The two ways of presenting the data that I have chosen (3 surface plot and 2d graph showing projection onto each axis) each has different strenghts. The 2d graph is easier to read, but the averaging over states would erase some of the data. This is especially a problem for the graphs of the Dark Matter and extra galactic contributions on the low contribu-

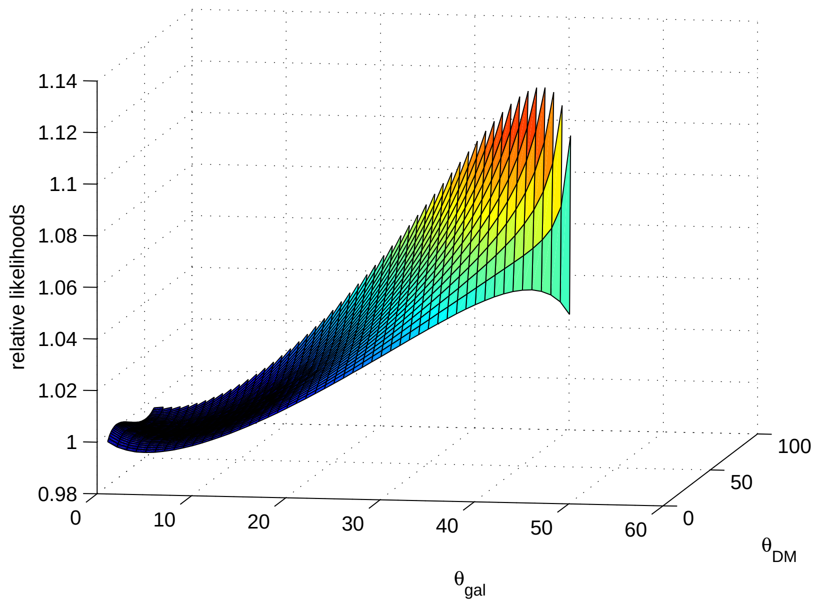


Figure 5.5: The ratio of the likelihoods including a power law energy dependence and the likelihoods with no energy dependence.

tion side. Here high Dark Matter/extra galactic states (high likelihood) are averaged with high galactic contribution states (low likelihood). In order to provide a clearer picture figure 5.11 plots the edges of the plot. The extra-galactic/Dark Matter axis (blue) is the most interesting.

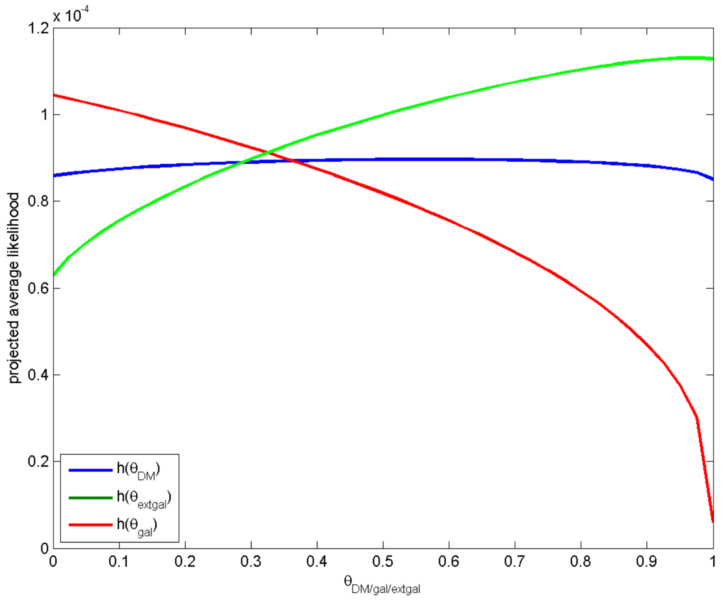


Figure 5.6: Likelihood as a function of distribution parameters, using the more complicated model for the Dark Matter energy dependence. If $L(\theta_{\text{extgal}}, \theta_{\text{gal}}, \theta_{\text{DM}})$ is the likelihood of the average event in the set given those parameters, then the green line is $h_{extgal} = \int_0^{1-\theta_{\text{extgal}}} d\theta_{\text{gal}} L(\theta_{\text{extgal}}, \theta_{\text{gal}}, 1 - \theta_{\text{extgal}} - \theta_{\text{gal}})$ plotted against θ_{extgal}, while the blue line is similar for the Dark Matter contribution and the red line for the galactic contribution.

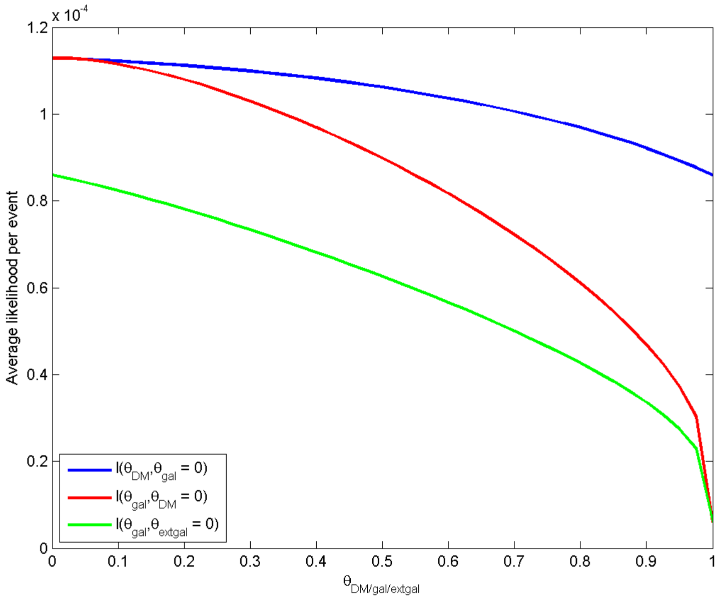


Figure 5.7: The likelihoods for one $\theta = 0$ as a function of the remaining parameters. This result is from looking only at the spatial distribution.

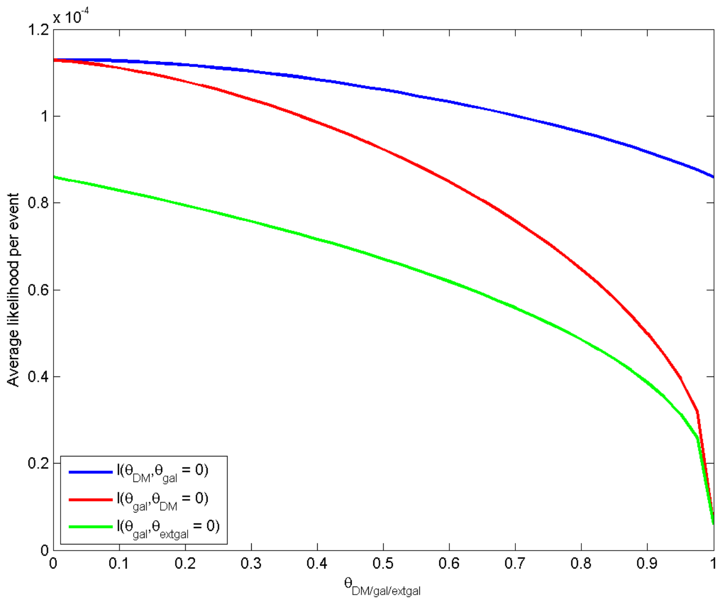


Figure 5.8: The likelihoods for one $\theta = 0$ as a function of the remaining parameters. This result is for the power law energy model.

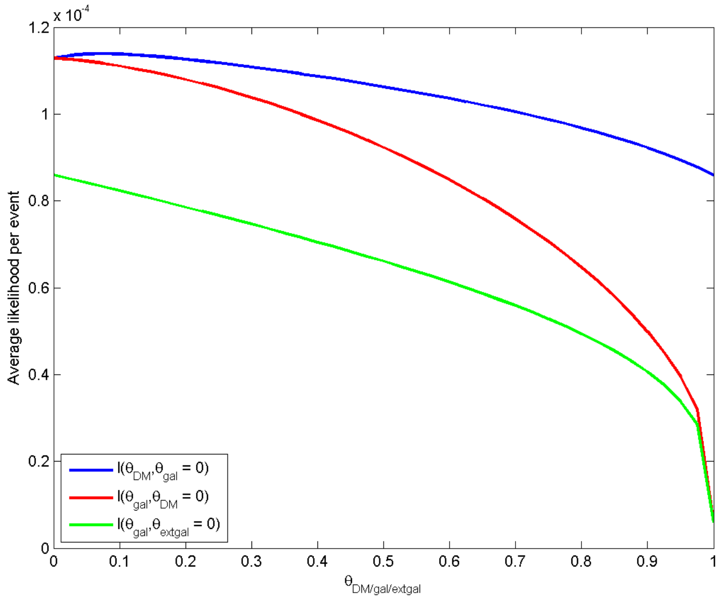


Figure 5.9: The likelihoods for one $\theta = 0$ as a function of the remaining parameters. This result is for the more detailed energy model.

5.1 Comparison

After finding that the data favored the isotropic contribution I tried to calculate how much evidence an observation would on average give for one distribution over the other. Here I looked a simpler model of just the extra-galactic and Dark Matter contributions. I calculated how likely a set of observations distributed according to the purely extra-galactic would be according to distributions with different values of θ_{extgal} and θ_{DM}. Here I used a simplified model though which disregards the possibility of background and which disregards the sensitivity information of IceCube. Essentially what I found was how easy the extra-galactic and the Dark Matter distribution are to tell apart. The result showed that the observed events fit better with distributions with a large θ_{DM} than events distributed according to the purely extra-galactic distribution would for both versions of the model. However it showed that for the model with the most detailed energy dependence the the model fit better than the purely-extra galactic one even for small values of θ_{DM} and much better for large ones.

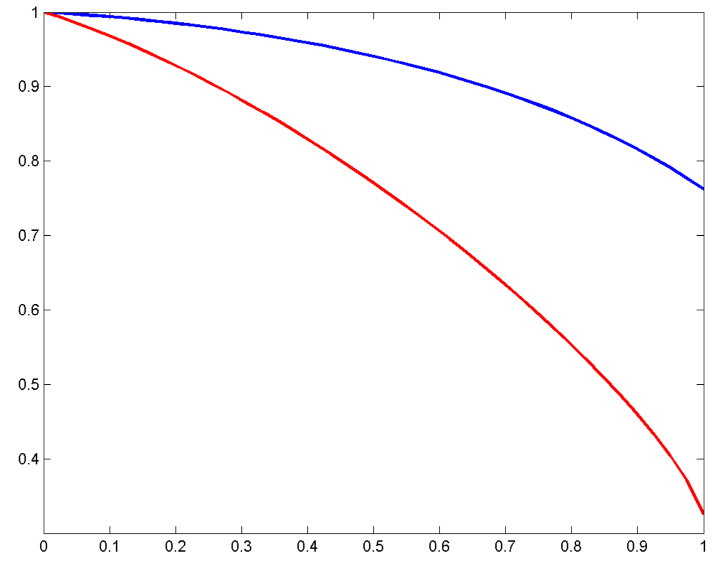


Figure 5.10: The likelihoods, for the model sans energy dependence, of the observed events $l(\theta_{DM})$ and $e(\theta_{DM}) = l(0) \cdot \langle \frac{l(\theta_{DM})}{l(0)} \rangle |_{extgal}$

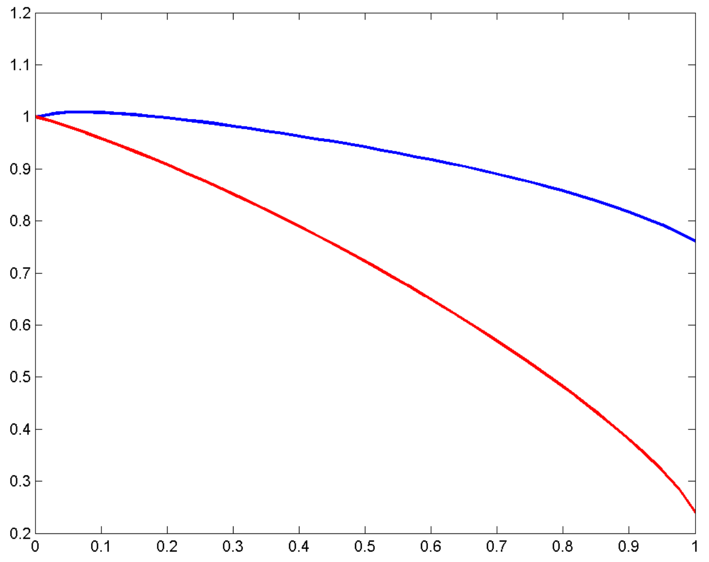


Figure 5.11: The likelihoods, for the model with the most detailed energy dependence, of the observed events $l(\theta_{DM})$ and $e(\theta_{DM}) = l(0) \cdot \langle \frac{l(\theta_{DM})}{l(0)} \rangle|_{extgal}$

5.2 Other analysis

While I was working on this thesis a paper was published [12] which also analyses the IceCube data with respect to the possibility that some of the neutrino flux could be generated by the decay of long lived Dark Matter. Their focus was different however, looking primarily at the energy distribution rather than the spatial distribution of the neutrinos. Non Dark Matter neutrinos were assumed to follow a single power law,

$$\frac{d\Phi_\nu}{dE_\nu} = \frac{C_0}{10^8} \cdot \frac{1}{E_\nu^2} \cdot \left(\frac{E_\nu}{100\text{TeV}}\right)^{2-s} \tag{5.1}$$

where C_0 and s are parameters. The flux of neutrinos from the decay of long lived particles meanwhile was assumed to peak at half the mass of the particle.

The article looked at the three possibilities that some, all, or none of the neutrino observations were caused by the decay of long lived particles for several sets of possible parameters and decay paths. They were unable to exclude any of the three scenarios. They found that if none of the neutrinos were from the decay of long lived particles they would be able to reject the hypothesis that all the neutrinos were caused by long lived particle decays with twice as many observations than are currently available. Excluding a mixed origin hypothesis would be much harder, not only because such an hypothesis would predict data more similar to an extreme scenario than the opposite extreme scenario, but also because a mixed origin hypothesis has more unconstrained variables. For some sets of parameters they concluded that it would not realistically possible to seperate a mixed and a pure scenario case.

In February 2015 the IceCube collaboration released an analysis of the flavor composition of events [20]. The flavor composition of neutrinos emitted by astrophysical sources can range from $(f_e, f_\mu, f_\tau) = (1, 0, 0)$ to (0, 1 ,0). However the most likely case is considered to be (1, 2 ,0). As a result of neutrino oscillations such neutrinos would arrive at Earth with a com-

	H:I	H:IIa	H:IIb	H:IIIb	H:IVb	H:V
H:I	-	40	9	7	8	4
H:IIa	50	-	10	6	12	4
H:IIb	50	35	-	80	20	50
H:IIIb	2.1	3	10	-	20	50
H:IVb	2.1	4.1	9.8	40	-	50
H:V	1.1	2.1	6	9	7	-

Table 5.1: How many times the current amount of data it would on average take to exclude an hypothesis given some other hypothesis being true. H:I is a purely power law signal, H:V is a purely long lived particle decay signal, H:II is a mixed of power law and decay into neutrino-lepton, H:III is power law mixed with decay into two Higgs particles, and H:IV is a power law mixed with decay into four Higgs particles. In all cases a refers to a decaying particle with mass 2.2 PeV and b refers to a mass of 4PeV[12].

position very near (1, 1, 1), and more extreme source compositions would still be in that neighborhood, (1,0,0) at the source would result in (1.6, 0.6, 0.8) at Earth and (0,1,0) at the source would result in (0.6, 1.3, 1.1) at Earth. This means a flavor ratio radically different from (1,1,1) would imply new physics relating to neutrinos, such as sterile neutrinos. Their result was that the favor composition observed at IceCube was consistent with the (1,1,1) ratio expected from astrophysical sources.

Chapter 6

Conclusion

In the end it's clear that the result has been that the IceCube data doesn't favor our Dark Matter model. Which, at least, is consistent with other analysis of the data. It's important to keep in mind, however, how small the data set is. By the time I'm writing this conclusion the largest data set I've used was only 36 events long, and some of those were likely background. IceCube will keep producing data and the results would definitely become clearer with more data.

I'm aware that adding refinements to the model underway in the way that I have done is, to an extent, bad form, and that both overfitting and motivated stopping are things to be concerned about. However all the numeric parameters have the first value I've used for them.

I still think the idea that Dark Matter may be responsible for some of the excess in high energy neutrinos and it would be interesting if someone were to do similar analysis for other Dark Matter models. One possible follow up would be to keep the Dark Matter distribution, but to look at annihilating rather than decaying Dark Matter. In that case the rate at which neutrinos are emitted would be proportional to the square of the Dark Matter density, rather than being proportional to the density.

One thing I always wanted to do was to parallelize the program. Execution time was always a limiting factor and the problem is extremely well suited for parallel programing: Since the joint log likelihood for a parameter set is the sum of the individual log likelihoods the set of events can simply be divided up, the program run separately for each event sublist and the resulting result matrices added together element-wise at the end. Unfortunately parallel programming in C++ is somewhat complicated though and I had no prior experience with it. It was something I was thinking about doing for most of the project, but I never found the time to learn it.

Appendices

Appendix A

Code overview

The appendices include the program I used. In order to increase readability I made extensive use of sub-functions that I divided between several files based on their topic. The main program is found in Appendix B, and the various function files are found in the appendices after that. celestial.hpp, found in Appendix C, contains all the functions relating to the celestial coordinate systems used, the transformations between them, and so on. model.hpp, found in Appendix D, contains the functions related to creating the probability distribution for the model. Functions for calculating density of sources at some point in space, for projecting those spatial distributions unto the sky and so on. energy.hpp, found in Appendix E, contains the functions relating to the energy dependence. IceCube.hpp, found in Appendix F, contains the functions relating to IceCube's sensitivity at different energies and angles. Finally inputoutput.hpp, found in Appendix G, contains the functions related to reading information from files or writing the result to file.

Appendix B

main.cpp

```
//Standard C++ libraries that include functions like reading
    from file , writing to file , sine , cosine , exponential , etc .
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <math.h>
#include <cmath>

using namespace std;

static float pi = 3.14159265359;

//Approximate distance that the sun orbits the galactic center.
static float sundist = 8.7; //[kpc]

//Characteristic length of the extra-galactic distribution.
static float r0 = 15;//[kpc]

//Characteristic length of the galactic distribution.
static float z0 = 0.21; //[kpc]

//number of integration points used when integrating some
    density of sources along our line of sight.
static int m = 1000;
```

```
//Header file containing functions relating to reading from and
    writing to files.
#include "inputoutput.hpp"

//Header file containing functions relating to celestial
    coordinate systems, transformations between them and so
    forth.
#include "celestial.hpp"

//Header file containing functions relating to our model of the
    neutrino sources, other that those that have to do with
    energy dependence.
#include "model.hpp"

//Header file containing functions relating to our model's
    energy dependence.
#include "energy.hpp"

//Header file containing functions that directly relates to the
    IceCube experiment, like sensitivity.
#include "IceCube.hpp"


float** unbinned(int, int, float**, int, float**, float**, float
    **, float**, float**);
float computelikelyhood(float**, float, float, float, int);

int main(){
    //The name of the file containing the events.
    char* file = "ICevents";

    //granularity of the parameter space
    int paraRes = 40;

    //Number of integration points, in all three dimensions
    int nInt = 100;
```

```
//Finds the number of lines (and thus number of events) in
    the event file
int len = linecounter(file);

//Creates a len times 4 matrix storing RA, declination,
    energy and experimental error in that order
float** events = eventsreader(file, len);

//"s" (shower) if an electron neutrino or "t", track, if a
    muon neutrino
char* flavors = flavorreader(file, len);

//Creates a vector of vectors containing one vector for each
     event that holds IceCube's sensitivity, at different
    declinations, for that event's energy
float** sensitivity = sensitivitycalc(len, events[2],
    flavors, nInt);

//Creates three vectors, one for each of the potential
    sources considered. Each of the vectors contain a number
     of elements equal to the number of events, the nth
    element describing the relative likelihood of a neutrino
     from that source having the energy of the nth event in
    the set.
float** energylikelihoods = energylikelihoodscalc(len,
    events[2]);

//Creates a nInt*nInt matrix giving the neutrino flux
    observed at the earth from luminous sources in our
    galaxy.
float** gal = galcalc(nInt);

//Creates a uniform nInt*nInt matrix, representing extra-
    galactic sources.
float** extgal = extgalcalc(nInt);

//Creates a nInt*nInt matrix giving the neutrino flux
    observed at the earth from dark matter sources in the
    galaxy.
float** ansatz = dmansatzcalc(nInt);
```

```
    //Main function in the program that loops over the different
        parameter sets. It returns a matrix that holds the
       likelihoods of the event set at different parameter sets
        at each point in the matrix
    float** probs = unbinned(len, nInt, events, paraRes,
       sensitivity, energylikelihoods, gal, extgal, ansatz);

    //Writes the matrix created above to a file in a matlab
       format.
    matlabfilewriter(probs, paraRes);

    return 1;
}

float** unbinned(int len, int nInt, float** events, int paraRes,
     float** sensitivity, float** energylikelihoods, float** gal
    , float** extgal, float** ansatz){

    //creates a matrix to store the results. Each position in
       the matrix corresponds to one parameter set
    float** probabilities = new float*[paraRes+1];
    for (int i = 0; i<paraRes+1; i++){
        probabilities[i] = new float[paraRes+1];
        for (int j = 0; j<paraRes+1; j++){
            probabilities[i][j] = 0;
        }
    }

    //loops over all combinations of the three parameters. i, j
       and 200 − i − j should cover every combination of
       parameters that sum to 200.
    for (int i = 0; i<paraRes+1; i++){
        for (int j = paraRes-i; j>=0; j--){
            //print to command line instruction that allows you
               to track the program's progress.
            cout << "i␣=␣" << i << "␣j␣=␣" << j << endl;

            probabilities[i][j] = 0;
            float loglikelihood = 0;
```

```
//loops over all the events, and calculates the log
    likelihood of that event given the parameter set
     given by the outer loop.
for (int k = 0; k<len; k++){

    //Calculates the final distribution of
        astrophysical neutrinos predicted by the
        model for a specific event. Taking into
        account both the parameters, the model for
        the sources, the energy dependence and the
        how the sensitivity varies by angle at that
        energy.
    //float** distribution =  distributioncalc(i,j,
        extgal, gal, ansatz, nInt, paraRes,
        energylikelihoods[0][k],energylikelihoods
        [1][k],energylikelihoods[2][k], sensitivity[
        k]);

    //Replacing the above function with the one
        below will result in looking only at the
        spatial distribution and ignoring the energy
         information
    float** distribution =  distributioncalc(i,j,
        extgal, gal, ansatz, nInt, paraRes, 1,1,1,
        sensitivity[k]);

    float likelihood = 0;
    float normconst = 0;

    //Background distribution is recalculated for
        each event because of the energy dependence
        in the sensitivity.
    float** background = new float*[nInt];
    for (int x = 0; x<nInt; x++){
        background[x] = new float[nInt];
        for (int y = 0; y<nInt; y++){
            background[x][y] = sin((1 - y/((float)
                nInt))*pi)/(nInt*nInt)*sensitivity[k
                ][y];
            normconst += sin((1 - y/((float)nInt))*
                pi)*background[x][y];
        }
    }
    for (int x = 0; x<nInt; x++){
        for (int y = 0; y<nInt; y++){
            background[x][y] = background[x][y]*
                normconst;
        }
    }

    //loops over different amounts of background.
    for (int l = 0; l<paraRes;l++){
        float** finaldist = new float*[nInt];
        for (int x = 0; x<nInt; x++){
            finaldist[x] = new float[nInt];
            for (int y = 0; y<nInt;y++){
                //For a certain amount of background
                     the final likelihood of seeing
                    a neutrino from a given
                    direction is a linear
                    combination of the likelihood of
                     seeing an astrophysical
                    neutrino from that direction and
                     of seeing a background neutrino
                     from that direction.
                finaldist[x][y] = l/((float)paraRes)
                    *distribution[x][y] + (paraRes -
                     l)/((float)paraRes)*background[
                    x][y];

                //Since we're working with floating
                    point numbers we can get
                    rounding errors that put a very
                    small number on the wrong side
                    of 0. Since we'll be doing
                    logarithms later it's best to
                    nip the problem in the bud.
                if(finaldist[x][y] < -1e-5){
                    cout << "ERROR:␣negative␣
                        probability␣distribution."
                       << endl;
                }
```

```
                            if(finaldist[x][y] < 0){
                                finaldist[x][y] = 0;
                            }
                        }
                    }

                    //computelikelyhood is a function that takes
                         the probability distribution of a model
                        , the coordinates of a neutrino
                        observation and it's standard error.
                        Based on this it integrates over all
                        directions the probability of the
                        observed neutrino coming from that
                        direction times the likelihood the model
                         describes for a neutrino being observed
                         from that direction.*/
                    likelihood += computelikelyhood(finaldist,
                        events[0][k],events[1][k], events[3][k],
                         nInt);
                    for (int x = 0; x<nInt; x++){
                        delete finaldist[x];
                    }
                    delete finaldist;
                }
                loglikelihood += log(likelihood);

                //memory clearing
                for (int m = 0; m<nInt; m++){
                    delete distribution[m];
                    delete background[m];
                }
                delete distribution;
                delete background;
            }
            float result = loglikelihood/len;
            probabilities[i][j] = exp(result);
        }
    }
    return probabilities;
}
```

```
//For a matrix giving the probability distribution of events
    according to some model, the direction of an observed event,
     and it's error, this function will find the likelihood of
    the observed event under the given model.
float computelikelyhood(float** distribution, float ra, float
    dec, float err, int n){

    float sum = 0;
    for (int j = 0; j<n; j++){
        for (int i = 0; i<n; i++){
            int* indices = new int[2];
            indices[0] = i;
            indices[1] = j;
            float* coords = indicestocoordinates(n, indices);

            //great circle distance between first and second set
                 of coordinates.
            float dist = angdist(ra, dec, coords[0], coords[1]);

            //every event is viewed as a normal distribution on
                the great circle with standard deviation equal
                to the experimental error.
            sum += sin(pi/2 - coords[1])*errfunc(n, err, dist)*
                distribution[i][j];

            delete coords;
            delete indices;
        }
    }
    return sum;
}
```

Appendix C

celestial.hpp

This header file contains the functions relating to the celestial coordinate systems used.

```
//Functions that have to do with celestial coordinate systems
    and transformations between them


//Translates a set of indices in a square matrix into two angles
    , going from -pi/2 to pi/2 in the vertical and 0 to 2pi in
    the horizontal.
float* indicestocoordinates(int n, int* indices){

    float* coords = new float[2];
    coords[0] = indices[0]*2*pi/n;
    coords[1] = (indices[1]*1.0/n-0.5)*pi;
    return coords;
}

//Translates a set of coordinates given as right angle and
    declination into galactic longitude and latitude
float* eqtogal(float ra, float declination){
    float longitude = 5.2883 - atan(sin(3.3554 - ra)/(cos(3.3554
        - ra)*sin(0.4782) - tan(declination)*cos(0.4782)));
    float latitude = asin(sin(declination)*sin(0.4782) + cos(
        declination)*cos(0.4782)*cos(3.3554 - ra));
    float* result = new float[2];
    result[0] = longitude;
    result[1] = latitude;

    return result;
}

//Computes the great circle distance between two directions
float angdist(float long1, float lat1, float long2, float lat2){
    return acos(sin(long1)*sin(long2) + cos(long1)*cos(long2)*
        cos(lat1 - lat2));
}

//Given a galactic longitude and galactic langitude and a
    distance from the sun/earth it computes the distance of the
    point specified from the galactic center.
float orbdist(float dfe, float lon, float lat){
    //lon is galactic longitude, lat is galactic latitude, dfe
        is distance from earth.
    float orbd = sqrt(abs(dfe*dfe + sundist*sundist - 2*sundist*
        dfe*cos(lon)*cos(lat)));
    if (orbd < 0.0000001){
        orbd = 0.0000001;
    }
    return orbd;
}
```

Appendix D

model.hpp

This header file contains the functions relating to our model of the source distributions, and the creation of probability distributions for a given model.

```
//Methods relating to our specific model


//Takes the distance from the galactic center to the projection
    of a point onto the galactic plane (r) and the height of the
     point above or below the galactic plane. Then it returns
    the density of luminous neutrino sources at the point
    described by those numbers.
float galdensity(float r, float z){
    if (r<0.00001){r = 0.00001;}

    //The density in the galactic plane at a distance r from the
         center.
    float density = pow(r/sundist, 0.7)*exp(-3.5*(r-sundist)/
        sundist);

    //The density at a point z distance above or below the
        galactic plane.
    density = density*exp(-(z/z0)*(z/z0));
    return density;
}
```

```
//Takes a galactic longitude, a galactic latitude and a distance
     from earth (dfe) and calculates the density of dark matter
    at that point according to the Navarro-Frenk-White profile.
float dmdensity(float longitude, float latitude, float dfe){
    float r = orbdist(dfe, longitude, latitude);
    if (r<1.0/10000){
        r = 1.0/10000;
    }
    float temp = 1.0 + r/r0;
    float rho = 1/(r*temp*temp);
    return rho;
}

//Creates a matrix where each entry represents a direction in
    the sky. For each point the matrix contains the Navarro-
    Frenk-White profile integrated over our line of sight in
    that direction.
float** dmansatzcalc(int nInt){
    float** ansatz = new float*[nInt];
    for (int i = 0; i<nInt; i++){
        ansatz[i] = new float[nInt];
    }
    float normconst = 0;

    //Loops over the matrix indices in the matrix being created
        For each set of indices it finds what direction in the
        sky this corresponds to and numerically integrates the
        density of sources along our line of sight in that
        direction.
    for (int i = 0; i<nInt; i++){
        for (int j = 0; j<nInt; j++){
            ansatz[i][j] = 0;
            int* indices = new int[2];
            indices[0] = i;
            indices[1] = j;
            float* coord = indicestocoordinates(nInt,indices);
            coord = eqtogal(coord[0], coord[1]);

            //Iterates over a series of points at increasing
                distance from us and sums up the density at
                those points (discrete approximation of line
                integral).
            for (int k = 0; k<m; k++){
                float s = (10*sundist*k)/m;
                ansatz[i][j] += dmdensity(coord[0], coord[1], s)
                    ;
            }

            //Adjusting for the fact that a grid in longitude
                and latitude has decreasing distance between
                nodes closer to the poles.
            ansatz[i][j] = sin(pi/2 - coord[1])*ansatz[i][j];
            normconst += sin(pi/2 - coord[1])*ansatz[i][j];

            //Memory clearing.
            delete coord;
            delete indices;
        }
    }
    for (int i = 0; i<nInt; i++){
        for (int j = 0; j<nInt; j++){
            ansatz[i][j] = ansatz[i][j]/normconst;
        }
    }
    return ansatz;
}

//Creates a matrix describing the galactic distribution.
float** galcalc(int nInt){
    float** gal = new float*[nInt];
    for (int i = 0; i<nInt; i++){
        gal[i] = new float[nInt];
    }
    float normconst = 0;

    //Loops over the matrix indices in the matrix being created.
         For each set of indices it finds what direction in the
        sky this corresponds to and numerically integrates the
        density of sources along our line of sight in that
        direction.
    for (int i = 0; i<nInt; i++){
        for (int j = 0; j<nInt; j++){
            gal[i][j] = 0;
            int* indices = new int[2];
            indices[0] = i;
            indices[1] = j;
            float* coord = indicestocoordinates(nInt,indices);
            coord = eqtogal(coord[0], coord[1]);

            //Iterates over a series of points at increasing
                distance from us and sums up the density at
                those points (discrete approximation of line
                integral).
            for (int k = 0; k<m; k++){
                float s = (10*sundist*k)/m;
                if (s != s){
                    cout << "s!=s\n";
                }
                float r = orbdist(s, coord[0],0);
                if (r!=r){
                    cout << "r!=r\n";
                }
                float z = r*sin(coord[1]);
                if (z != z){
                    cout << "z!=z\n";
                }
                gal[i][j] += galdensity(r, z);
            }

            //Adjusting for the fact that a grid in longitude
                and latitude has decreasing distance between
                nodes closer to the poles.
            gal[i][j] = sin(pi/2 - coord[1])*gal[i][j];
            normconst += sin(pi/2 - coord[1])*gal[i][j];

            //Memory clearing.
            delete coord;
            delete indices;
        }
    }
    for (int i = 0; i<nInt; i++){
        for (int j = 0; j<nInt; j++){
            gal[i][j] = gal[i][j]/normconst;
        }
    }
    return gal;
}

//Creates a normalized matrix of constant probability density.
   As the coordinate points become denser closer to the poles
   the value at each point is angle dependent.
float** extgalcalc(int nInt){
    float** extgal = new float*[nInt];
    float normconst = 0;
    for (int i = 0; i<nInt; i++){
        extgal[i] = new float[nInt];
        for (int j = 0; j<nInt; j++){
            float coord = (1 - j*1.0/nInt)*pi;
            extgal[i][j] = sin(coord)*1.0/(nInt*nInt);
            normconst += sin(coord)*extgal[i][j];
        }
    }
    for (int i = 0; i<nInt; i++){
        for (int j = 0; j<nInt; j++){
            extgal[i][j] = extgal[i][j]/normconst;
        }
    }
    return extgal;
}

float** distributioncalc(int k, int l, float** extgal, float**
   gal, float** ansatz, int n, int paraRes, float dmelike,
   float galelike, float extelike, float* sensitivity){

    //Turns the variables k and l used to loop over the
       parameters (which are integers in the range 0 to paraRes
        rather than floating point numbers in the range 0 to 1)
        into the actual paramters.
    float dmpara = k/((float)paraRes);
    float galpara = l/((float)paraRes);
    float extgalpara = (paraRes - k - l)/((float)paraRes);

    float** dist = new float*[n];
    float normconst = 0;
    for (int i = 0; i<n;i++){
        dist[i] = new float[n];
        for (int j = 0; j<n; j++){
            dist[i][j] = sensitivity[j]*(dmelike*dmpara*ansatz[i
                ][j] + galelike*galpara*gal[i][j] + extelike*
                extgalpara*extgal[i][j]);
            float coord = (1 - j*1.0/n)*pi;
            normconst += sin(coord)*dist[i][j];
        }
    }
    for (int i = 0; i<n; i++){
        for (int j = 0; j<n; j++){
            dist[j][i] = dist[j][i]/normconst;
            if(!(dist[i][j]==dist[i][j])){
                cout << "dist␣=␣" <<dist[i][j] <<endl;
            }
        }
    }
    return dist;
}

//Gaussian function.
float errfunc(int n, float err, float dist){
    float error = exp(-(dist*dist)/(2*err*err));
    error = error/(2*pi*err*err*n*n);
    return error;
}
```

Appendix E

energy.hpp

This header file contains the functions relating to the energy dependence of our model.

```
//Functions pertaining to the effect of energy on the likelihood

float dmenergylikelihood(float energy, float** energytable,
    float massparam){
    float param = log(massparam/(2*energy));
    int i = 0;
    while (i<2004){
        if (energytable[0][i] > param){
            return energytable[1][i]/energy;
        }
        else{
            i++;
        }
    }
    cout << "ERROR energy range";
    return 0;
}

/*
//Replace the above with n the below to use the simplified
    energy dependence. It's correct that two of the arguments
    are unused. That's so it takes the same arguments as the
    more detailed function, and any code calling it doesn't have
     to be changed.
float dmenergylikelihood(float energy, float** energytable,
    float massparam){
     float like = pow(energy, −1.8);
     return like;
}
*/

float galenergylikelihood(float energy){
     float like = pow(energy, −2.5);
     return like;
}

float extgalenergylikelihood(float energy){
     float like = pow(energy, −2.2);
     return like;
}

float backgroundenergylikelihood(float energy){
     float like = pow(energy, −3.7);
     return like;
}

float** energylikelihoodscalc(int len, float* energies){
     //Name of the file containing a table which describes the
         more detailed energy model.
     char* energyfile = "log_e_M8_ff";

     //The more detailed energy model depends on the energy
         relative to the assumed dark matter mass. The article
         the model is taken from was calculated for masses around
          10^GeV
     float massparam = 1000; //[TeV]

     //Reads the table in the file specified by the given
         filename and saves it as a 2x2004 matrix. One column is
         ln(2E/M). And the other is the associated likelihoods.
     float** energytable = energyreader(energyfile);

     float** energylikelihoods = new float*[4];
    for (int i = 0; i<4; i++){
        energylikelihoods[i] = new float[len];
    }
    float normconst1 = 0;
    float normconst2 = 0;
    float normconst3 = 0;
    float normconst4 = 0;
    for (int j = 0; j<len; j++){
        energylikelihoods[0][j] = dmenergylikelihood(energies[j
            ],energytable, massparam);
        normconst1 += energylikelihoods[0][j];
        energylikelihoods[1][j] = galenergylikelihood(energies[j
            ]);
        normconst2 += energylikelihoods[1][j];
        energylikelihoods[2][j] = extgalenergylikelihood(
            energies[j]);
        normconst3 += energylikelihoods[2][j];
        energylikelihoods[3][j] = backgroundenergylikelihood(
            energies[j]);
        normconst4 += energylikelihoods[3][j];
    }
    for (int j = 0; j<len; j++){
        energylikelihoods[0][j] = energylikelihoods[0][j]/
            normconst1;
        energylikelihoods[1][j] = energylikelihoods[1][j]/
            normconst2;
        energylikelihoods[2][j] = energylikelihoods[2][j]/
            normconst3;
        energylikelihoods[3][j] = energylikelihoods[3][j]/
            normconst4;
    }
    return energylikelihoods;
}
```

Appendix F

IceCube.hpp

This header file contains the functions relating to IceCube's sensitivity at different neutrino energies, flavors and angles.

```
//Functions relating to IceCube and its properties

/*
Given an energy this function computes the coefficients of the
    polynomial that will describe IceCube's sensitivity to
    electron neutrinos as a function of the cosine of the
    declination.
*/
float* coeffselcalc(float energy){
    //coefficients c_i in S(dec) = sum over i (c_i * cos(dec))
    float lng = log10(energy);
    float* coeffs = new float[11];
    coeffs[10] =  1.4982*lng*lng    -1.2834*lng   -16.848;
    coeffs[9] =  -0.65172*lng*lng   +2.1804*lng    -0.1432;
    coeffs[8] =  -2.1716*lng*lng    +0.97143*lng +48.638;
    coeffs[7] =   1.1493*lng*lng    -5.2452*lng    +0.7870;
    coeffs[6] =  -0.15176*lng*lng   +1.6071*lng   -52.264;
    coeffs[5] =  -0.48987*lng*lng   +4.464*lng      -1.0083;
    coeffs[4] =   1.3081*lng*lng    -1.6791*lng   +26.01;
    coeffs[3] =   0.064428*lng*lng  -1.6051*lng    +0.42918;
    coeffs[2] =  -0.64372*lng*lng   +0.3115*lng    -6.4204;
    coeffs[1] =  -0.025337*lng*lng +0.13563*lng   -0.075782;
    coeffs[0] =  -0.082567*lng*lng -0.4379*lng    -2.2301;
    return coeffs;
}

/*
Given an energy this function computes the coefficients of the
   polynomial that will describe IceCube's sensitivity to mu
   neutrinos as a function of the cosine of the declination.
*/
float* coeffsmucalc(float energy){
    //coefficients c_i in S(dec) = sum over i (c_i * cos(dec))
    float lng = log10(energy);
    float* coeffs = new float[11];
    coeffs[10] =  4.6898*lng*lng    -8.6509*lng    -6.1085;
    coeffs[9] =   5.2739*lng*lng    -2.9573*lng    -0.17973;
    coeffs[8] = -11.15*lng*lng     +23.276*lng    +17.835;
    coeffs[7] = -14.665*lng*lng     +8.5211*lng    +0.33336;
    coeffs[6] =   7.1089*lng*lng   -21.459*lng    -19.653;
    coeffs[5] =  14.047*lng*lng     -8.1772*lng    -0.083802;
    coeffs[4] =   0.74176*lng*lng   +7.6195*lng   +10.313;
    coeffs[3] =  -4.8159*lng*lng    +2.554*lng     -0.11426;
    coeffs[2] =  -1.7929*lng*lng    -1.0146*lng    -2.934;
    coeffs[1] =  -0.057557*lng*lng +0.068357*lng +0.014327;
    coeffs[0] =  -0.16983*lng*lng  -0.063039*lng -1.1108;
    return coeffs;
}

//takes a vector and interprets it as the coefficients of a
   polynomial and evaluates that polynomial for a given value (
   cos(dec)).
float detfactor(float* coeffs, float dec){
    float factor = 0;
    float cosdec = cos(dec);
    for (int i = 0; i<11; i++){
        factor += coeffs[i]*pow(cosdec, i);
    }
    return pow(10, factor);
}

//The sensitivity of IceCube, it's probability of detecting an
   incoming neutrino, is a function of the energy, flavor and
```

```
    declination. This function creates a vector of vectors. One
    vector for each energy value with each element in that
    vector specifying a sensitivity at one declination for that
    energy.
float** sensitivitycalc(int len, float* energies, char* flavor,
    int nInt){
     float** sensitivity = new float*[len];
     for (int i = 0; i<len; i++){
          sensitivity[i] = new float[nInt];
          float* coeffs;
          if (flavor[i] == 's'){
               coeffs = coeffselcalc(energies[i]);
          }
          else if (flavor[i] == 't'){
               coeffs = coeffsmucalc(energies[i]);
          }
          else{cout << "Error:␣Event␣is␣neither␣shower␣nor␣track\n
              ";}

          for(int j = 0; j<nInt; j++){
               float dec = (j - nInt/2.0)*2*pi/nInt;
               if (dec < -pi){cout << "dec␣=␣" << dec << "\n";}
               if (dec > pi){cout << "dec␣=␣" << dec << "\n";}
               sensitivity[i][j] = detfactor(coeffs, dec);
          }
          delete coeffs;
     }
     return sensitivity;
}
```

Appendix G

inputoutput.hpp

This header file contains the functions relating to reading from, and writing to, files.

```
//Takes the filename of the file containing the IceCube data as
    input and finds and returns the number of events it
    describes.
int linecounter(char* file){
    int len = 0;
    ifstream linecounter;
    linecounter.open(file);
    string wastebasket;
    while(linecounter.peek() != EOF){
        getline(linecounter, wastebasket);
        if (wastebasket == ""){
            linecounter.close();
            return len;
        }
        else{
            len++;
        }
    }
    linecounter.close();
    return len;
}

//Function for reading the observation data (other than flavor)
   from file and saving it in a matrix.
float** eventsreader(char* file, int len){
    ifstream infile;
    infile.open(file);

    float** events = new float*[4];
    for (int i = 0; i<4; i++){
       events[i] = new float[len];
    }

    //Reads the table and saves values converted into radians
    string STRING;
    for (int i = 0; i<len; i++){
        getline (infile,STRING);
        int endenergy = STRING.find("␣");
        string energy = STRING.substr(0,endenergy);
        int beginfirst = STRING.find_first_not_of("␣", endenergy
            + 1);
        int endfirst = STRING.find("␣", beginfirst);
        string first = STRING.substr(beginfirst, endfirst);
        int beginsecond = STRING.find_first_not_of("␣", endfirst
           );
        int lensecond = STRING.find("␣", beginsecond) -
           beginsecond;
        string second = STRING.substr(beginsecond, lensecond);
        int beginthird = STRING.find_first_not_of("␣",
           beginsecond + lensecond);
        int lenthird = STRING.find("␣", beginthird) - beginthird
           ;
        string third = STRING.substr(beginthird, lenthird);
        events[0][i] = atof(second.c_str())*pi/180;
        events[1][i] = atof(first.c_str())*pi/180;
        events[2][i] = atof(energy.c_str());  //energy[TeV]
        events[3][i] = atof(third.c_str())*pi/180;  //median
           angular resolution[radians]
    }
    infile.close();
    return events;
}

//Takes the file name with the IceCube data and the number of
    observations as input. Creates an array of characters of
    length equal to the number of observations, then fills it
    with either the letter 's' or 't' depending on whether that
    event is a shower or track event.
char* flavorreader(char* file, int len){
    ifstream infile;
    infile.open(file);
    char* flavors = new char[len];
    string STRING;
    for (int i = 0; i<len; i++){
        getline (infile,STRING);
        flavors [i] = STRING.at(STRING.length()-1);
        cout << flavors[i];
    }
    return flavors;
}

//Function for reading the table describing the more detailed
    energy model.
float** energyreader(char* filename){
    int length = 0;
    ifstream linecounter;
    linecounter.open(filename);
    string wastebasket;
    while(linecounter.peek() != EOF){
        getline(linecounter, wastebasket);
        if (wastebasket == ""){break;}
        length++;
    }
    linecounter.close();
    ifstream energyreader;
    energyreader.open(filename);
    string line;
    float** result = new float*[2];
    result[0] = new float[length];
    result[1] = new float[length];
    int counter = 0;
    while (counter < length){
        getline(energyreader, line);
        int start = line.find_first_not_of('␣');
        int last = line.find_last_not_of('␣');
        line = line.substr(start, last);
        last = line.find_first_of('␣');
        string a = line.substr(0, last);
        start = line.find_last_of('␣');
        string b = line.substr(start, -1);
        result[0][counter] = atof(a.c_str());
        result[1][counter] = atof(b.c_str());
        counter++;
    }
    return result;
}

//Takes an integer n, and an nxn matrix and saves the matrix as
   a MATLAB script file called result.m. Running this file in
   MATLAB will leave you with an identical matrix that will be
   called "X".
void matlabfilewriter(float** probs, int paraRes){
    ofstream prob;

    prob.open("result.m");
    prob << "X␣=␣[␣";

    for (int i = 0; i<paraRes+1; i++){
        for (int j = 0; j<paraRes+1; j++){
            prob << "␣" << probs[i][j];
        }
        prob << ";";
    }

    prob << "␣];";
    prob.close();
}
```

Bibliography

[1] http://upload.wikimedia.org/wikipedia/commons/e/ec/M33_rotation_curve_HI.gif

[2] A. Li, F. Huang and R. X. Xu, "Too massive neutron stars: The role of dark matter?," Astropart. Phys. **37** (2012) 70 [arXiv:1208.3722 [astro-ph.SR]]. http://inspirehep.net/record/1128196/plots

[3] J. F. Navarro, C. S. Frenk and S. D. M. White, The Structure of cold Dark Matter halos, Astrophys. J. **462** (1996) 563 [astro-ph/9508025].

[4] S. S. McGaugh, A tale of two paradigms: the mutual incommensurability of ΛCDM and MOND, Can. J. Phys. **93** (2015) 2, 250 [arXiv:1404.7525 [astro-ph.CO]].

[5] M. Markevitch, A. H. Gonzalez, D. Clowe, A. Vikhlinin, L. David, W. Forman, C. Jones and S. Murray *et al.*, Direct constraints on the Dark Matter self-interaction cross-section from the merging galaxy cluster 1E0657-56, Astrophys. J. **606** (2004) 819 [astro-ph/0309303].

[6] Annika H. G. Peter, Dark Matter: A Brief Review, arXiv:1201.3942v1 [astro-ph.CO]

[7] V. Zacek, Dark Matter, arXiv:0707.0472 [astro-ph].

[8] S. P. Martin, A Supersymmetry primer, Adv. Ser. Direct. High Energy Phys. **21** (2010) 1 [hep-ph/9709356].

[9] Pierre Sikivie, Axion Theory, http://www-library.desy.de/preparch/desy/proc/proc08-02/sikivie_pierre.pdf

[10] M. Drewes, The Phenomenology of Right Handed Neutrinos, Int. J. Mod. Phys. E **22** (2013) 1330019 [arXiv:1303.6912 [hep-ph]].

[11] M. Kachelriess, S. Ostapchenko, R. Tomas, ELMAG: A Monte Carlo simulation of electromagnetic cascades on the extragalactic background light and in magnetic fields ,arXiv:1106.5508v2 [astro-ph.HE] Computer Physics Communications, Volume 183, Issue 4, p. 1036-1043.

[12] C. S. Fong, H. Minakata, B. Panes and R. Z. Funchal, arXiv:1411.5318 [hep-ph].

[13] V. S. Narasimham, Perspectives of Experimental Neutrino Physics in India, http://www.imsc.res.in/~ino/OpenReports/Insa/naras.pdf

[14] Y. Fukuda *et al.* [Super-Kamiokande Collaboration], Phys. Rev. Lett. **81** (1998) 1562 [hep-ex/9807003].

[15] http://upload.wikimedia.org/wikipedia/commons/thumb/3/3e/Cherenkov_Wavefront.svg/580px-Cherenkov_Wavefront.svg.png

[16] http://galacticinteractions.scientopia.org/wp-content/uploads/sites/13/2012/10/pair_production.png

[17] http://www-sk.icrr.u-tokyo.ac.jp/sk/gallery/wme/sk_01h-wm.jpg

[18] http://en.wikipedia.org/wiki/Galactic_coordinate_system#mediaviewer/File:Galactic_coordinates.JPG

[19] http://en.wikipedia.org/wiki/Right_ascension#mediaviewer/File:Ra_and_dec_on_celestial_sphere.png

[20] M. G. Aartsen *et al.* [IceCube Collaboration], arXiv:1502.03376 [astro-ph.HE].

[21] https://indico.cern.ch/event/287474/session/7/contribution/85/material/slides/0.pdf

[22] V. Berezinsky, M. Kachelriess and S. Ostapchenko, Phys. Rev. Lett. **89** (2002) 171802 [hep-ph/0205218].

[23] J. L. Feng, Class. Quant. Grav. **25** (2008) 114003 [arXiv:0801.1334 [gr-qc]].