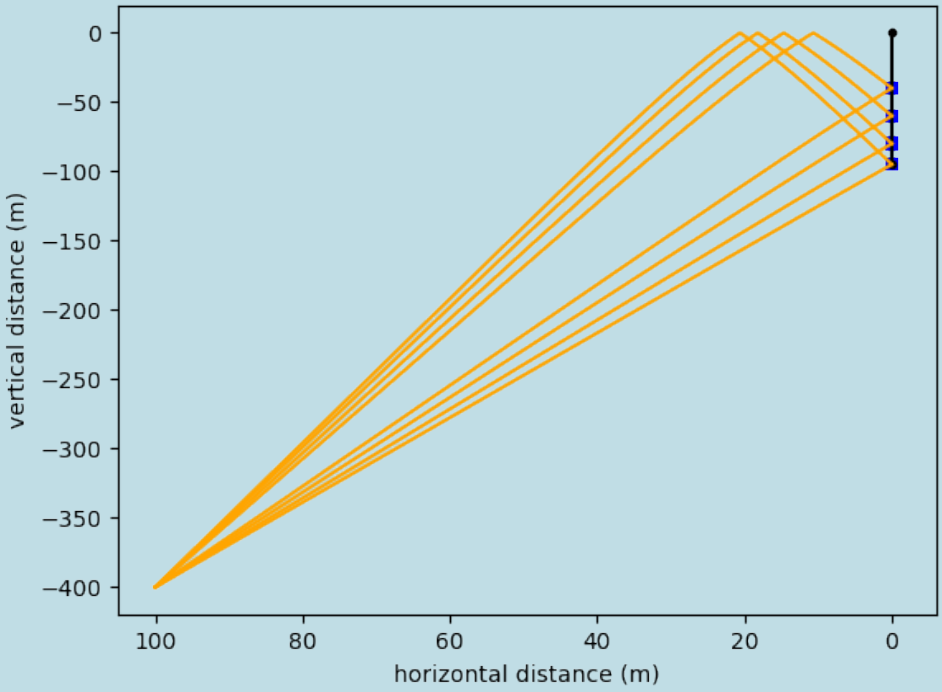
FACULTY OF SCIENCES

Radio detection of high energy neutrinos in the Greenland icecap

Arthur Adriaens



Department of Physics and Astronomy

Promotor: Prof. dr. Dirk Ryckbosch Dirk.Ryckbosch@ugent.be

Accompanist: Bob Oeyen Bob.Oeyen@ugent.be

Master's dissertation submitted in partial fulfilment of the requirements for the degree of master in Physics and Astronomy

GHENT UNIVERSITY

CONTENTS

1 Foreword **3**

2 Neutrinos **5**
2.1 Discovery . . . 5
2.2 Neutrino sources . . . 6
2.2.1 Cosmological/Primordial neutrinos . . . 6
2.2.2 Solar neutrinos . . . 7
2.2.3 Supernovae . . . 8
2.2.4 Terrestrial anti-neutrinos . . . 9
2.2.5 Reactor anti-neutrinos . . . 9
2.2.6 Background from old supernovae . . . 10
2.2.7 Cosmic rays . . . 10
2.2.8 Atmospheric neutrinos . . . 10
2.2.9 How do they fit into the full detector spectrum? . . . 10
2.2.10 astrophysical neutrinos: general . . . 10
2.3 Do neutrinos have mass? . . . 10
2.3.1 See-saw mechanism . . . 11
2.3.2 Majorana fermions . . . 13

3 Radio detection of neutrinos **15**
3.1 Neutrino interactions in ice . . . 15
3.2 Askaryan effect . . . 16
3.3 Wave propagation . . . 17
3.4 Ice model . . . 18
3.5 Reconstruction . . . 18

4 The Detector **19**

5 Hybrid Ray tracer **24**
5.1 Shortcomings of the exponential ice model . . . 24
5.2 How it works . . . 24

5.3 Performance Optimalisation 26
5.3.1 Length of the normal vector 26
5.3.2 ztol 26
5.3.3 Sphere Size & Step Size 27

6 Weather Balloon **31**
6.1 Plane Wave Reconstruction 31
6.2 Is the goal feasible? 34
6.3 Fitting the index 36

A List of abbreviations **42**

B Hybrid minimizer code **43**

CHAPTER

1 FOREWORD

English

The Radio Neutrino Observatory in Greenland - RNO-G - is under construction at Summit Station in Greenland to search for neutrinos of several PeV energy up to the Eev range. It's a mid-scale, discovery phase, extremely high-energy neutrino telescope that will probe the astrophysical neutrino flux at energies beyond the reach of IceCube. More particularly if will make it possible to reach the next major milestone in astroparticle physics: the discovery of cosmogenic neutrinos. All simulations carried out within this work were made with the three

programs

- NuRadioMC
- radiotools
- RadioPropa

whom are free to download on github. The projects built over the course of this thesis were the "Hybrid minimizer": a new algorithm for finding the path of a neutrino interaction to the detector in complex ice models and (...)

Nederlands

De *Radio Neutrino Observatory in Greenland* - RNO-G - is een detector die momenteel onder constructie is in Groenland. Deze detector heeft als doel het vinden van neutrinos met energieën van enkele PeV tot in het EeV gebied. Deze extreem hoge energy neutrino telescoop zal astrofysische neutrinos zoeken op energieën waar icecube te klein voor is. Het zal ook het vinden van cosmogenische neutrinos mogelijk maken. Alle simulaties werden mogelijk gemaakt door de volgende programmas:

- NuRadioMC

- radiotools
- RadioPropa

Dewelke te downloaden zijn op github De projecten die tot completie zijn gebracht over deze master thesis waren de *Hybrid minimizer*: een algoritme gemaakt met als doel het snel vinden van een pad vanuit de interactievertex tot de detector in complexe ijsmodellen en (...)

CHAPTER

2

NEUTRINOS

When looking outside into the night sky you might see various stars, but invisible to the naked eye various kinds of particles are also transversing light-years to end up where you stand.

When looking at phenomena outside our earth the astronomer will turn to electromagnetic radiation, as a lot of the most interesting events emit photons in some kind of way this is very useful but he's missing out on a big part of the full picture. There are regions in the universe from which photons can't escape. They can also be absorbed on their way to earth. That's where the astroparticle physicist comes in who takes a look at muons, nuclei,... and neutrinos.

Neutrinos are often called "ghost" particles because they very rarely interact with matter, on average 100 trillion neutrinos pass through your body per second, none of them having any effect. You'd even need a light year of lead to give you just a 50% chance of stopping a neutrino.

As they also have no charge they are not deflected by magnetic fields. This means that once a cosmic neutrino is detected and it's direction is inferred, it's origin can be found.

Because they can travel huge distances without getting distorted or sidetracked, neutrinos are ideal messengers to identify sources of ultra high energy (UHE) cosmic rays in the universe. Neutrinos can serve as unique clues about what's happening elsewhere in the universe including the cosmic collisions, galaxies, supernovae, Gamma-ray bursts (GRBs),... where they are created.

2.1 Discovery

When researching β^- decay, the decay of a neutron into a nucleus, researcher detected a proton and an electron coming from the neutron. However on closer inspection it became apparent that energy was lost somewhere in violation with the conservation law of energy, and angular momentum wasn't conserved. The solution postulated by Wolfgang Pauli was to introduce a new, really hard to detect particle: the neutrino. The neutrino comes in three flavours: electron, muon and tau neutrinos, each corresponding to their respective lepton denoted as

$$\nu_e \quad \nu_\mu \quad \nu_\tau \tag{2.1}$$

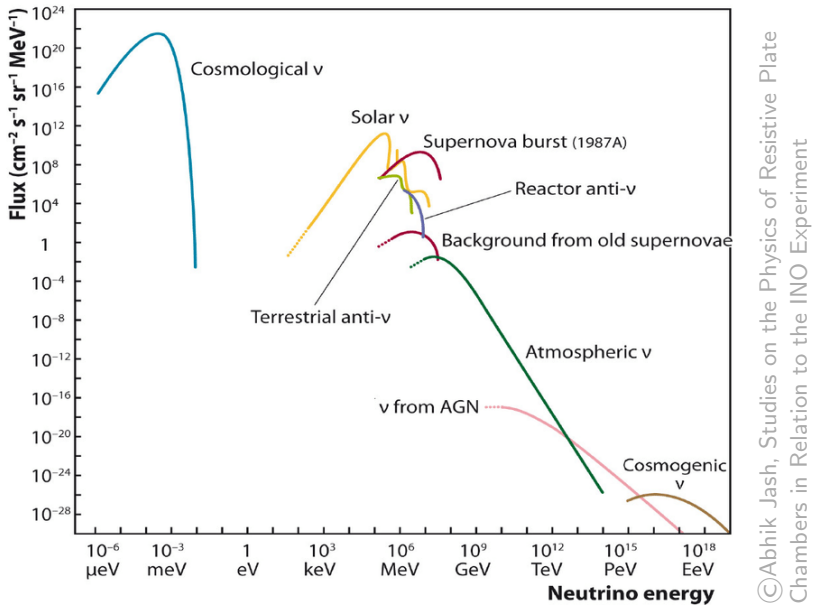


Figure 2.1: Predicted neutrino flux for various natural sources

and each also having an anti-particle.

$$\bar{\nu}_e \quad \bar{\nu}_\mu \quad \bar{\nu}_\tau \tag{2.2}$$

Now with the introduction of the neutrino the full β^- decay becomes

$$n \rightarrow p^+ + e^- + \nu_e \tag{2.3}$$

The inverse can then also be used to detect neutrinos:

$$n \rightarrow p^+ + e^- + \nu_e \tag{2.4}$$

Which is called beta capture and was first experimentally detected in 1956 [4] also making it the first experimental detection of a neutrino.

2.2 Neutrino sources

Neutrinos can be generated in 2 ways: either they're generated in interactions at the sources, termed *astrophysical neutrinos*. Or they're created through the interaction of ultra-high energy cosmic rays during propagation with the cosmic microwave or other photon backgrounds termed *cosmogenic neutrinos*. In figure 2.1 you see the neutrino flux theoretically expected in function of energy for various sources of neutrinos. I'll walk you through each of them seperately.

2.2.1 Cosmological/Primordial neutrinos

The first source of neutrinos we'll talk about is the one in blue to the left of the figure termed the *Cosmological neutrinos*: the neutrino version of the CMB. To understand this source we'll have to go back all the way to just after the big bang: The very early universe was hot and dense. As a result, interactions among particles occurred much more frequently than they do today. As an example, a photon today can travel across the observable universe without deflection or capture, so it has a mean free path greater than 10^26 m. When the universe was 1 second old,

though, the mean free path of a photon was about the size of an atom. Thus in the time it took the universe to expand by a factor of 2, a given photon interacted many, many times. These multiple interactions kept the consitituents in the universe in thermal equilibrium. But as the universe expanded there were times when reactions could not proceed rapidly enough to maintain equilibrium conditions, these particles then fall out of thermal equilibrium. This falling out of equilibrium is termed *decoupling*. And we're interested in when neutrinos decoupled. Neutrinos were kept in equilibrium through the interaction

$$\nu e \leftrightarrow \nu e \tag{2.5}$$

up until the universe cooled down to about 1MeV when they decoupled. To estimate the temperature of the neutrinos who decoupled at the start of the universe, we can take a look at conservation of entropy [5] from which we'll find that:

$$T_\nu = \left(\frac{4}{11}\right)^{1/3} T_\gamma \tag{2.6}$$

Note that they decoupled before the photons making them lower in temperature. As T_γ is the CMB temperature which, nowadays, is measured to be around 2.7K. These primordial neutrinos are very low in energy.

2.2.2 Solar neutrinos

The sun fuses elements to release energy and thus keeping itself from collapsing in on itself, with most of the various ways particles get fused neutrinos get released:
pp − cycle

$$\mathrm{p} + \mathrm{p} \rightarrow \mathrm{D} + e^+ + \nu_e \tag{2.7}$$
$$\mathrm{D} + \mathrm{p} \rightarrow {}^3\mathrm{He} + \gamma \tag{2.8}$$
$${}^3\mathrm{He} + {}^3\mathrm{He} \rightarrow {}^4\mathrm{He} + \mathrm{p} + \mathrm{p} \tag{2.9}$$

boron − cycle

$${}^4\mathrm{He} + {}^3\mathrm{He} \rightarrow {}^7\mathrm{Be} + \gamma \tag{2.10}$$
$${}^7\mathrm{Be} + p \rightarrow {}^8\mathrm{B} + \gamma \tag{2.11}$$
$${}^8\mathrm{B} \rightarrow {}^8\mathrm{Be}^* + e^+ + \nu_e \tag{2.12}$$
$${}^8\mathrm{Be}^* \rightarrow {}^4\mathrm{He} + {}^4\mathrm{He} \tag{2.13}$$

Be − capture

$${}^7\mathrm{Be} + e^- \rightarrow {}^7\mathrm{Li} + \nu_e \tag{2.14}$$

pep

$$\mathrm{p} + e^- + \mathrm{p} \rightarrow \mathrm{D} + \nu_e \tag{2.15}$$

Now with this and some information about the sun like the pressure and mass, the so-called "standard solar model" was made. This model predicted a certain amount of neutrinos to be hitting the earth from the previously mentioned thermonuclear fusion, it was however 3 times

higher than the observed amount of neutrinos back at our planet. This led to a little bit of hysteria as this could've meant that the sun was dying and we'd see the aftermath only in a couple of years. Through various experiments however, it became apparent that this was due to the various kind of neutrinos oscillating into each other on their way to earth, i.e 2/3 of the original electron neutrinos had oscillated into mu and tau neutrinos. But for them to oscillate into eachother, theoretically they not only require mass but each flavor also should have a different mass as the 2D transition probability is

$$P(\nu_e \to \nu_\mu) = |\langle \nu_\mu | \psi(L,T) \rangle|^2 = c_\mu c_\mu^* = \sin^2(2\theta)\sin^2\left(\frac{\Delta\phi_{12}}{2}\right) \tag{2.16}$$

with

$$\Delta\phi_{12} \approx \frac{m_1^2 - m_2^2}{2p} L \tag{2.17}$$

In general (3D)

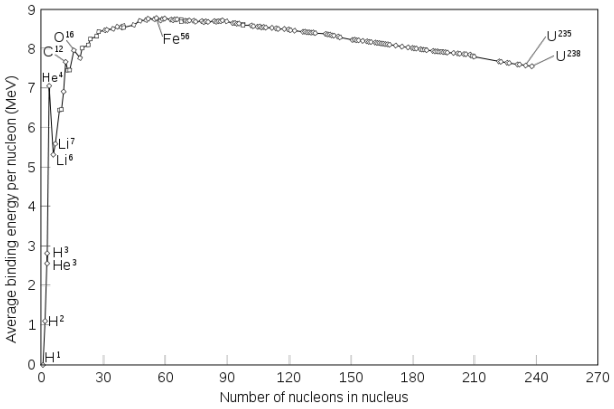
$$|\nu_\alpha\rangle = \sum_i U_{\alpha i} |\nu_i\rangle \tag{2.18}$$

With $U_{\alpha i}$ e.g the Pontecorvo-Maki-Nakagawa-Sakata (PMNS) matrix. This phenomenon has been observed e.g through the descrepency from the observed and expected neutrino events coming from a nuclear reactor [6].

2.2.3 Supernovae

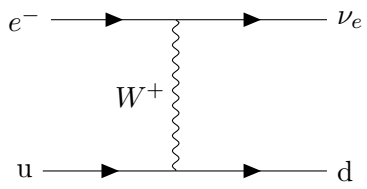
A star starts its life as a ball of pure hydrogen. At the core, due to the gravitational pressure of the outside plasma, fusion of hydrogen into deuterium and helium happens. Thus converting mass into energy. The pressure of this energy counteracts the pressure of gravity and the star is stable.

When the hydrogen at the core runs out no more hydrogen can be fused. For stars with masses between $8M_\odot$ and $30M_\odot$ the fusion of heavier elements starts, this can't keep going on however as at some point the star starts to form the most stable element: iron. It costs energy to both



make lighter elements than iron and heavier ones. As the iron core builds up the outside pressure from the core starts to decrease as no new energy is released. This goes on until the treshold of an iron core with a mass of $1.4M_\odot$ known as the Chandrasekhar limit and the the inwards pressure becomes too large, making the electrons surrounding the iron core fuse with the protons (uud), creating neutrons (udd) and neutrinos, diagramatically:
This last part happens in a split second as the collapse goes at 25% the speed of light, creating a very dense neutron star (3000km in diameter iron core to 30km in diameter neutron star) and up to 10^{52} ultra-relativistic neutrinos, carrying up to 99% of the released energy [12][1]. As the

[1] 1% is released as kinetic energy, only 0.001% as electromagnetic radiation



density has suddenly increased so much there's a huge distance of pure vaccuum between the plasma outer layer and the (now) neutron star, this plasma starts free-falling inwards, also at 25% the speed of light whilst the neutrinos carrying tremendous amounts of energy start going outwards from the neutron stars core.

The neutrinos then collide with the plasma resulting in what we observe as a "supernova", wrongly thought of by Kepler as being a "new (nova) star" rather being a violent death of an old star.

This is quite unexpected as neutrinos rarely interact, they only do becouse the incoming plasma is so dense and due to the tremendous amount of neutrinos that collissions happen at all. Some, however, escape and will be visible on earth in our neutrino detectors $\approx$ 18h before the light escapes the exploding star.

Neutrino observatories are thus useful to know where to point our various telescopes before the supernova is actually visible in the night sky.

2.2.4 Terrestrial anti-neutrinos

Terrestrial anti-neutrinos, also termed a *geoneutrinos* are neutrinos coming from the decay of radionuclides (unstable nuclides due to excess in nuclear energy) naturally occuring in our earth. Most geoneutrinos are electron antineutrinos originating in β^- decay modes of ^{40}K, ^{232}Th and ^{238}U. Together these decay chains account for more than 99% of the present-day radiogenic heat generated inside the Earth

2.2.5 Reactor anti-neutrinos

Our most important energy source today, both in terms of energy output and their impact on the enviroment are fission reactors. Looking back at our energy per nucleon vs number of nucleons in nucleus figure shown in the supernovae subsection we see that all the way to the right uranium is found, a lot of energy can thus be released by splitting in into nuclei with a lower amount of nucleons. Fission reactors operate by splitting this urantium by the introduction of a neutron:

$$n + {}^{235}U \rightarrow \text{Energy (radiation)} + 2 \times \text{Fission fragments} + 2.5 \times n \tag{2.19}$$

this 2.5 is of course an average. The surplus of neutrons means this is a self-sustaining interaction meaning that for a stable thermonuclear interaction carbon capture rods are needed. When this uranium nucleus fissions into two daughter nuclei fragments, about 0.1 percent of the mass of the uranium nucleus appears as the fission energy which is about 200 MeV. For uranium-235 about 169 MeV appears as the kinetic energy of the daughter nuclei , the neutrons carry a mean kinetic energy per neutron of 2 MeV (total of 4.8 MeV) and 7 MeV is released in the form of gamma ray photons. This all sums up to the *total prompt fission energy* which amounts to about 181 MeV, or 89% of the total energy which is eventually released by fission over time. The remaining 11% is released in beta decays where we get our reactor anti-neutrinos from.

2.2.6 Background from old supernovae

Also termed the *diffuse supernova neutrino background* (DSNB), as the universe is quite old various supernovae have happened over it's lifetime, each generating a lot of neutrinos as was discussed in section 2.2.3. This is postulated to have generated a continuous neutrino background.

2.2.7 Cosmic rays

Before we can talk about atmospheric neutrinos it's necessary to discuss *cosmic rays*. Cosmic rays are ionized nuclei of which 90% are protons, 9% are alpha particles and the rest are heavier nuclei. Almost all of them originate from outside the solar system but from within our galaxy, the few particles that do come from our solar system can be temporally linked to violent events on the sun. In contrast to this the particles coming from outside our solar system show an anti-correlation with the sun as they can more easily reach the earth if solar activity is low.

2.2.8 Atmospheric neutrinos

Cosmic rays hit the Earth's atmosphere at a rate of about 1000 per square meter per second. These cosmic rays interact with atomic nuclei in the Earth's atmosphere, creating showers of particles, many of which are unstable and produce neutrinos when they decay these neutrinos are what's called *Atmospheric neutrinos*.

2.2.9 How do they fit into the full detector spectrum?

The origin of the most energetic cosmic rays is still not conclusively identified. One approach to solving this problem is *multi-messenger astrophysics*, where several types of cosmic particles are used to identify the sources of these ultra-high energy cosmic rays (UHECRs). E.g we simultaneously measure gravitational waves (gravitons?) with the Einstein telescope, neutrinos with IceCube (or eventually RNO-G), photons with various telescopes and muons with a muon detector.

2.2.10 astrophysical neutrinos: general

As cosmic rays get accelerated they will occasionally interact with matter and photon fields near the source. These will then produce new particles, of which the most common is pions due to their low mass (a lot of available phase space). Pions can decay into neutrinos as follows:

$$\pi^- \rightarrow \mu^- + \bar{\nu}_\mu \rightarrow e^- + \bar{\nu}_e + \nu_\mu + \bar{\nu}_\mu \tag{2.20}$$

$$\pi^+ \rightarrow \mu^+ + \nu_\mu \rightarrow e^+ + \nu_e + \bar{\nu}_\mu + \nu_\mu \tag{2.21}$$

Another possibility is direct production of neutrinos at the source, of which Supernovae are a prime example:

2.3 Do neutrinos have mass?

Now the question remains, how do neutrinos get their mass (theoretically)?

2.3.1 See-saw mechanism

We can give neutrinos mass as with any other spinor throught Yukawa couplings and an extra term:

$$\mathcal{L} = i\bar{\nu}\partial\!\!\!/\nu - m\bar{\psi}\nu - \frac{M}{2}(\bar{\nu}_R\nu_{Rc} + \bar{\nu}_{Rc}\nu_R) \tag{2.22}$$

More general, given a spinor ψ with the following Lagrange density:

$$\mathcal{L} = i\bar{\psi}\partial\!\!\!/\psi - m\bar{\psi}\psi - \frac{M}{2}(\bar{\psi}_R\psi_{Rc} + \bar{\psi}_{Rc}\psi_R) \tag{2.23}$$

We can re-write this Lagrangian in terms of

$$\chi := \frac{1}{\sqrt{2}}(\psi_R + \psi_{Rc}) \tag{2.24}$$

$$\omega := \frac{1}{\sqrt{2}}(\psi_L + \psi_{Lc}) \tag{2.25}$$

by first identifying that

$$-\frac{M}{2}(\bar{\psi}_R\psi_{Rc} + \bar{\psi}_{Rc}\psi_R) \tag{2.26}$$

is equal to

$$-M\bar{\chi}\chi = -\frac{M}{2}(\bar{\psi}_R + \bar{\psi}_{Rc})(\psi_R + \psi_{Rc}) = -\frac{M}{2}(\bar{\psi}_R\psi_{Rc} + \bar{\psi}_{Rc}\psi_R) \tag{2.27}$$

Where we've used

$$\bar{\psi}_R\psi_R \equiv \psi_R^\dagger\gamma^0\psi_R \equiv P_R\psi^\dagger\gamma^0 P_R\psi = P_R P_L\psi^\dagger\gamma^0\psi = 0 \tag{2.28}$$

And next that in

$$i\bar{\psi}\partial\!\!\!/\psi - m\bar{\psi}\psi = i\bar{\psi}_R\partial\!\!\!/\psi_R + i\bar{\psi}_L\partial\!\!\!/\psi_L - m\bar{\psi}_R\psi_L - m\bar{\psi}_L\psi_R \tag{2.29}$$

We can identify that

$$\bar{\chi}\partial\!\!\!/\chi = \frac{1}{2}(\bar{\psi}_R\partial\!\!\!/\psi_R + \bar{\psi}_{Rc}\partial\!\!\!/\psi_{Rc}) = \bar{\psi}_R\partial\!\!\!/\psi_R \tag{2.30}$$

$$\bar{\omega}\partial\!\!\!/\omega = \bar{\psi}_L\partial\!\!\!/\psi_L \tag{2.31}$$

Where we've done an integration by parts and used $\bar{\psi}_c\gamma_\mu\chi_c = -\bar{\chi}\gamma_\mu\psi$. We thus get:

$$\mathcal{L} = i\bar{\chi}\partial\!\!\!/\chi + i\bar{\omega}\partial\!\!\!/\omega - m\bar{\psi}_R\psi_L - m\bar{\psi}_L\psi_R - M\bar{\chi}\chi \tag{2.32}$$

And finally we notice that

$$\bar{\chi}\omega = \frac{1}{2}\left(\bar{\psi}_R\psi_L + \bar{\psi}_R\psi_{Lc} + \bar{\psi}_{Rc}\psi_L + \bar{\psi}_{Rc}\psi_{Lc}\right) \tag{2.33}$$

$$= \frac{1}{2}\left(\bar{\psi}_R\psi_L + \bar{\psi}_R\psi_{Lc} + \bar{\psi}_{Rc}\psi_L + \bar{\psi}_L\psi_R\right) \tag{2.34}$$

$$\bar{\omega}\chi = \frac{1}{2}\left(\bar{\psi}_L\psi_R + \bar{\psi}_{Lc}\psi_R + \bar{\psi}_L\psi_{Rc} + \bar{\psi}_{Lc}\psi_{Rc}\right) \tag{2.35}$$

$$= \frac{1}{2}\left(\bar{\psi}_L\psi_R + \bar{\psi}_{Lc}\psi_R + \bar{\psi}_L\psi_{Rc} + \bar{\psi}_R\psi_L\right) \tag{2.36}$$

As $\bar{\psi}_c\chi_c = \bar{\chi}\psi$. Summing these, we get:

$$\bar{\chi}\omega + \bar{\omega}\chi = \bar{\psi}_L\psi_R + \bar{\psi}_R\psi_L \tag{2.37}$$

$$+ \frac{1}{2}\left(\bar{\psi}_R\psi_{Lc} + \bar{\psi}_{Rc}\psi_L + \bar{\psi}_{Lc}\psi_R + \bar{\psi}_L\psi_{Rc}\right) \tag{2.38}$$

$$= \bar{\psi}_L\psi_R + \bar{\psi}_R\psi_L \tag{2.39}$$

Where we've used the handedness flips a charge conjugation entails, at last we now have:

$$\mathcal{L} = i\bar{\chi}\partial\!\!\!/\chi + i\bar{\omega}\partial\!\!\!/\omega - m(\bar{\chi}\omega + \bar{\omega}\chi) - M\bar{\chi}\chi \tag{2.40}$$

Now to determine the mass eigenstates we'll re-write this equation into the form

$$\mathcal{L} = i\bar{\chi}\partial\!\!\!/\chi + i\bar{\omega}\partial\!\!\!/\omega - \begin{pmatrix} \bar{\chi} & \bar{\omega} \end{pmatrix} \cdot \begin{pmatrix} M & m \\ m & 0 \end{pmatrix} \cdot \begin{pmatrix} \chi \\ \omega \end{pmatrix} \tag{2.41}$$

lets now find the eigenvalues:

$$\begin{vmatrix} M-\lambda & m \\ m & -\lambda \end{vmatrix} = (\lambda - M)\lambda - m^2 = 0 \implies \lambda_\pm = \frac{M \pm \sqrt{M^2 + 4m^2}}{2} \tag{2.42}$$

i.e

$$\lambda_\pm = \frac{M}{2}\left(1 \pm \sqrt{1 + 4m^2/M^2}\right) \tag{2.43}$$

Now we'll determine the eigenvectors:

$$\begin{bmatrix} M-\lambda_\pm & m \\ m & -\lambda_\pm \end{bmatrix} \begin{pmatrix} x_\pm \\ y_\pm \end{pmatrix} = \vec{0} \tag{2.44}$$

this gives

$$Mx_\pm - \lambda_\pm x_\pm + my_\pm = 0 \tag{2.45}$$

$$mx_\pm - \lambda_\pm y_\pm = 0 \implies \frac{mx_\pm}{\lambda_\pm} = y_\pm \tag{2.46}$$

inserting the second equation in the first we get:

$$Mx_\pm - \lambda_\pm x_\pm + m\frac{mx_\pm}{\lambda_\pm} = 0 = M\lambda_\pm x_\pm - \lambda_\pm^2 x_\pm + m^2 x_\pm \tag{2.47}$$

$$= x_\pm(\lambda_\pm^2 - M\lambda_\pm + m^2) \tag{2.48}$$

which is true for every $x_\pm$ (see equation 2.42) i.e we can freely choose $x_\pm$ so let's opt for $x_\pm^2 + y_\pm^2 = 1$:

$$x_\pm^2(1 + \frac{m^2}{\lambda_\pm^2}) = 1 \implies x_\pm^2 = \frac{1}{1 + \frac{m^2}{\lambda_\pm^2}} \implies x_\pm = \frac{\lambda_\pm}{\sqrt{\lambda_\pm^2 + m^2}} \tag{2.49}$$

$$\& \quad y_\pm = \frac{m}{\sqrt{\lambda_\pm^2 + m^2}} \tag{2.50}$$

And thus, summing up, we get the eigenvectors $\phi_\pm$ with eigenvalues $m_\pm$:

$$\phi_\pm := \begin{pmatrix} x_\pm \\ y_\pm \end{pmatrix} = \begin{pmatrix} \frac{\lambda_\pm}{\sqrt{\lambda_\pm^2+m^2}} \\ \frac{m}{\sqrt{\lambda_\pm^2+m^2}} \end{pmatrix} \quad \text{and} \quad m_\pm := \lambda_\pm = \frac{M}{2}\left(1 \pm \sqrt{1 + 4m^2/M^2}\right) \tag{2.51}$$

Now we'll consider the limit $M \gg m$:

$$m_\pm = \frac{M}{2}\left(1 \pm \sqrt{1 + 4m^2/M^2}\right) \approx \frac{M}{2}\left(1 \pm 1 \pm \frac{2m^2}{M^2}\right) \tag{2.52}$$

Which gives:

$$m_+ \approx M \tag{2.53}$$

$$m_- \approx -\frac{m^2}{M} \tag{2.54}$$

and:

$$\phi_+ \approx \begin{pmatrix} \frac{M}{M} \\ \frac{m}{M} \end{pmatrix} = \begin{pmatrix} 1 \\ m/M \end{pmatrix} \tag{2.55}$$

$$\phi_- \approx \begin{pmatrix} -\frac{m}{M} \\ \frac{m}{m} \end{pmatrix} = \begin{pmatrix} -m/M \\ 1 \end{pmatrix} \tag{2.56}$$

i.e, in the χ, ω basis:

$$m_+ \approx M \quad \text{with eigenstate} \quad \chi + \frac{m}{M}\omega \tag{2.57}$$

$$m_- \approx -\frac{m^2}{M} \quad \text{with eigenstate} \quad \omega - \frac{m}{M}\chi \tag{2.58}$$

$$\tag{2.59}$$

This is what's called the See-Saw mechanism, it's both a theory of giving mass to neutrinos and explaining their small mass. For a certain m if we choose a big M we'll get a big m_+ and small m_- (and vice versa). We also see that there's only a very small mixing of states, i.e the m_- mass state is almost purely ω (and m_+ almost purely χ). The parameter m in the original matrix is forbidden by electroweak gauge symmetry, and can only appear after the symmetry has been spontaneous broken by a Higgs mechanism; for this reason a good estimate of the order of m is the vaccuum expectation energy: $m \approx v = 246 \approx 10^2$GeV. In grand unified theories it's theorised that $M \approx 10^{15}$GeV after symmetry breaking, using these values we get

$$m_- \approx 10^{-11}\text{GeV} \approx 10^{-2}\text{eV} \tag{2.60}$$

which seems [2] to be a reasonable order of magnitude estimate for the observed neutrino mass. This mechanism would also lead to supermassive neutrinos, which are a possible dark matter candidate.

2.3.2 Majorana fermions

A theory that's also quite interesting is that neutrinos are majorana, a dirac fermion has the following density:

$$\mathcal{L} = i\bar{\psi}\partial\!\!\!/\psi - m\bar{\psi}\psi \tag{2.61}$$

$$= i\bar{\psi}_L\partial\!\!\!/\psi_L + i\bar{\psi}_R\partial\!\!\!/\psi_R - m\bar{\psi}_R\psi_L - m\bar{\psi}_L\psi_R \tag{2.62}$$

Now assume that we only have right-handed particles:

$$\mathcal{L} = i\bar{\psi}_R\partial\!\!\!/\psi_R - \frac{M}{2}\bar{\psi}_R\psi_{Rc} - \frac{M}{2}\bar{\psi}_{Rc}\psi_R \tag{2.63}$$

A majorana fermion is a fermion which is it's own anti-particle, i.e:

$$\chi = \frac{1}{\sqrt{2}}\left(\psi_R + \psi_{Rc}\right) = \chi_c \tag{2.64}$$

With this we have that

$$\bar{\chi}\chi = \frac{1}{2}\left(\bar{\psi}_R\psi_{Rc} + \bar{\psi}_{Rc}\psi_R\right) \tag{2.65}$$

Which we can recognize in 2.63 and

$$\bar{\chi}\partial\!\!\!/\chi = \frac{1}{2}(\bar{\psi}_R + \bar{\psi}_{Rc})\partial\!\!\!/(\psi_R + \psi_{Rc}) \tag{2.66}$$

$$= \frac{1}{2}\bar{\psi}_R\partial\!\!\!/\psi_R + \frac{1}{2}\bar{\psi}_{Rc}\partial\!\!\!/\psi_{Rc} \tag{2.67}$$

$$= \bar{\psi}_R\partial\!\!\!/\psi_R \tag{2.68}$$

where we've used integration by parts at the end, with these we can re-write 2.63 as:

$$\mathcal{L} = i\bar{\chi}\partial\!\!\!/\chi - M\bar{\chi}\chi \tag{2.69}$$

Thus arriving at a density for Majorana fermions with mass M, a possible detection mechanism to find out if neutrinos are Majorana is "Neutrinoless double beta decay":

Figure 2.2: normal and neutrinoless double beta decay

This is (at the time of writing) actively investigated.

CHAPTER

3

RADIO DETECTION OF NEUTRINOS

3.1 Neutrino interactions in ice

As neutrinos propagate through ice they can interact with nuclei in the following ways [7]:

Figure 3.1: Most prominent ways of neutrino-nucleus interaction

With the produced leptons in the W boson mediated interaction being either electrons, resulting in an electromagnetic shower, muons wich typically go undetected as they live too long or tauons wich will decay via

$$\tau^- \rightarrow e^- + \bar{\nu}_e + \nu_\tau \tag{3.1}$$

or, less ideally

$$\tau^- \rightarrow \mu^- + \bar{\nu}_\mu + \nu_\tau \tag{3.2}$$

In both of the possible interactions the resulting nucleus will result in a hadronic shower, for the neutral current interaction (mediated by the Z boson) the fraction of the neutrino energy that gets transferred to the nucleon is described by the inelasticity y and is heavily shifted towards small values of y [1]. This causes a big, irreducable uncertainty when trying to estimate the original neutrino energy from these kinds of events. With the charged current interaction (mediated by the $W^\pm$ bosons) this isn't a problem however as the full neutrino energy ends up in the resulting cascades.

3.2 Askaryan effect

For a particle shower to emit strong radio signals, two conditions have to be met:

- There needs to be a speration of positive and negative charges in the shower front
- The signals produced over the length of the shower profile need to overlap coherently.

The *Askaryan* [3] effect, also known as Askaryan radiation describes the effect at radio frequencies which abides by both of these conditions, in general it's a quite difficult effect but we'll give a crude overview. The above described interactions create a shower of secondary charged particles containing a charge anisotropy, this charge imbalance is a result of medium electrons either Compton scattering into the advancing shower or annihilating with shower positrons. In the end you have a moving charge anisotropy, propagating faster than the speed of light in the medium, creating Cherenkov radiation. Cherenkov radiation is like the elektromagnetic equivalent of a sonic boom, a sonic boom happens when something goes faster than the sounds speed in the medium; A particle emits Cherenkov radiation if it goes faster than the light speed in the medium[1] . Choosing the particle trajectory to lie along the z axis we can approximately find an equation for $\frac{\mathrm{d}^2\mathscr{J}}{\mathrm{d}\omega\mathrm{d}\Omega}$: the energy radiated per elementary unit solid angle and per elementary unit frequency interval

$$\frac{\mathrm{d}^2\mathscr{J}(\omega)}{\mathrm{d}\omega\mathrm{d}\Omega} = \frac{q^2}{4\pi}\sqrt{\frac{\mu}{\epsilon}}\beta^2\omega^2\delta^2[\omega(1-\beta\mathbf{e}_r\cdot\mathbf{e}_z)]|\mathbf{e}_r\times\mathbf{e}_z|^2 \tag{3.3}$$

Now we can re-write this equation in spherical coordinates, which gives $1-\beta\mathbf{e}_r\cdot\mathbf{e}_z = 1-\beta\cos(\theta_c)$ in the delta function. We thus only expect radiation if

$$\cos(\theta_c) = \frac{1}{\beta} = \frac{c'}{u} = \frac{c}{n}\cdot\frac{1}{u} \tag{3.4}$$

I.e if $u > \frac{c}{n}$ with n the index of refraction, Cherenkov radiation will be emitted along a cone surface with half angle $\frac{\pi}{2}-\theta_c$ as illustrated in figure 3.3. Integrating equation 3.3 over the solid angle and formally deviding by the time interval we get:

$$\frac{\mathrm{d}^2\mathscr{J}}{\mathrm{d}\omega\mathrm{d}t} = \frac{q^2}{4\pi}\sqrt{\frac{\mu}{\epsilon}}\beta\omega\left(1-\frac{1}{\beta^2}\right) \tag{3.5}$$

We see that the energy is proportional to ω, so we expect that most radiation will be emitted "in blue", as seen in figure 3.2. For ice the index of refraction is roughly 1.78 in deep ice, so we expect an ultra-relativistic particle to produce the most radiation at around 56° opening as

$$\cos(\theta_c) \approx \frac{1}{n} \implies \cos^{-1}\left(\frac{1}{1.78}\right) \approx 56^\circ \tag{3.6}$$

Of course this is just an estimate, as the actual index of refraction is depth-dependent which we'll get to in section 3.4. Now this explains how the signals get generated but logically, from only knowing this we'd expect radio waves to almost be non-existent due to the "in blue" nature of Cherenkov radiation. This isn't the full story however as we'll need to talk about coherent overlap to fully understand the Askaryan effect. This can be intuitively explained as follows: generally the shower is of length $\mathcal{O}(10\text{cm})$ [11], over this length the radiation gets emitted, most frequencies decoherently interfering, but radio waves with wavelengths of $\approx$ 10cm coherently interfere, and it's these waves we then wish to detect.

[1] The reader who wants a thorough explanation and derivation is advised to check out *Chapter 14: Radiation by Moving Charges* from the book *Classical Electrodynamics* by Jackson.

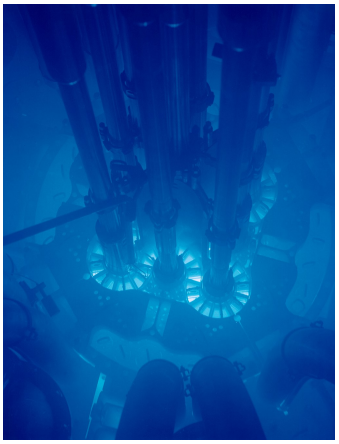


Figure 3.2: Cherenkov radiation in a nuclear reactor

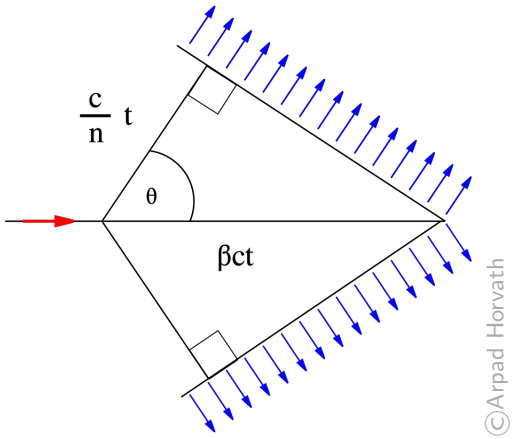
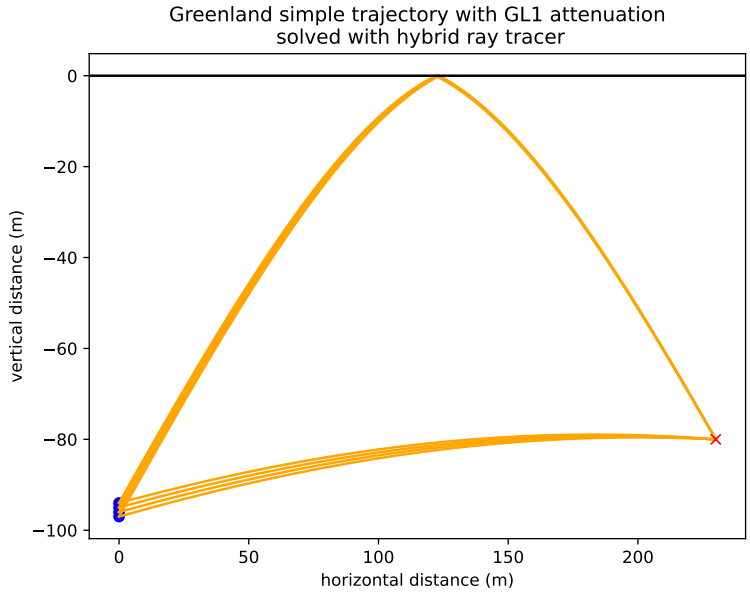


Figure 3.3: Diagrammatic representation of Cherenkov radiation

3.3 Wave propagation

Waves propagating through ice to a detector generally looks something like this:



Here the interaction of the neutrino happens somewhere in the neighbourhood of the red cross, the orange rays represent the resulting radio wave paths to the detectors which are signaled by the blue dot. Notice that reflection at the surface is also a possible path to the detector. In a homogeneous dielectric medium a ray propagates straight with it's signal wave-speed determined by the local index of refraction as $v = c/n$, the dependence of the index of refraction on density for ice is given by the Schytt equation:

$$n(x, y, z) \approx 1 + 0.78\rho(x, y, z)/\rho_0 \tag{3.7}$$

Where $\rho(x, y, z)$ is the local ice density and ρ_0 is the density for solid ice (917 kg/m^3). This isn't the only part of the index of refraction with which we'll need to concern ourselves however. If a ray propagates towards a boundary dividing 2 media with different indexes of refraction, the

ray will refract and the refracted angle can be found from Snell's law:

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \tag{3.8}$$

Where n is the index of refraction and θ is the angle with respect to the surface. The system we'll consider however, isn't homogeneous with some specified boundary, it's continous: ice in greenland has a continously varying density. Because of this we can't work with Snell's law but we'll have to take a look at the continuous version to figure out how these rays propagate. The "continuous version of Snell's law" is the eikonal equation: a path of a ray $\mathbf{r}(s)$ with path parameter s in a medium with index of refraction $\mathrm{n}(\mathbf{r})$ is described by:

$$\frac{d}{ds}\left(n(\mathbf{r})\frac{d\mathbf{r}}{ds}\right) = \nabla n \tag{3.9}$$

The software we'll be using for ray reconstruction is called "radiopropa" [16] and in radiopropa the local paraxial approximation is used, i.e. if we assume that in any individual step of the algorithm the change of the refractive index along the path ds is small it's possible to re-write the equation as:

$$n(\mathbf{r})\frac{d^2\mathbf{r}}{ds^2} \approx \nabla n \tag{3.10}$$

Which is then easily iteratively solved. If there are boundaries (such as defects or the surface) these are treated seperately using Snell's law.

3.4 Ice model

Equation 3.10, and thus the path, depends on the index of refraction on a given location. Purely from classical gravity and density considerations it can be derived that the index of refraction abides by

$$n(z) = n_{ice} - \Delta n e^{z/z_0} \tag{3.11}$$

with n_{ice} the refractive index of solid ice and $\Delta n = n_{ice} - n_s$ with n_s the index of refraction of snow. This is called "the exponential model".

From the measured density with depth variation this seems to roughly hold but it could be better (as will be explained in chapter (TBC)). This exponential model has a huge advantage as it's analytically solvable, meaning that we don't have to iteratively search for the path to the detector but know it instantly after the location of the neutrino interaction and the detector are specified, this "model" is called the *analytic ray tracer*.

3.5 Reconstruction

It can pose a challenge to reconstruct the radio signals produced by the Cherenkov radiation as they are often obscured by background noise. A solution used in RadioReco is Information Field Theory (IFT) implemented in RadioReco by Welling et al. [14] which uses Bayesian inference to calculate the most likely radio signal, given recorded data. The full reconstruction will then work as follows: events and their propagation are generated in a monte-carlo simulation using NuRadioMC [9] [8], how the detector signals would look like is then simulated with RadioReco and saved in a database. If we then detect a neutrino event in the real detector it can be compared to the database, thus finding the origin.

CHAPTER

4

THE DETECTOR

Both cosmic ray and neutrino detectors face the same main problem at the highest energies: the steeply falling flux (as can be seen on figure 4.1) requires large effective areas, which leads to the construction of neutrino detectors with volumes on the cubic kilometer scale: IceCube. As we wish to detect neutrinos with even higher energies we turn to look at an array of detectors spanning multiple cubic kilometers: RNO-G.

One such detector is illustrated in figure 4.2.

The deep component of the detector can be split up in three parts: Two *helper strings*, one *power string* and the surface components. The helper strings are the 2 vertical cables shown on the right of the figure and each one houses 2 vertically polarized antennas (Vpols), one quadslot antenna for the horizontal polarization component (Hpol) and one radio pulser on each helper string which can be used to generate calibration signals.

The power string (the leftmost vertical cable) is more densely instrumented: At the bottom it houses a set of four Vpol and two Hpol antennas with a spacing of 1m and further up the string, with a spacing of 20m are three more Vpol antennas.

The signal from each of these antennae are fed into a low-noise amplifier directly above it, from there the signal is send to the data acquisition (DAQ) system at the surface via a Radio Frequency over Fiber (RFoF) cable. There it's again amplified, digitized and saved onto an SD card. This data is then transmitted via a Long Term Evolution (LTE) telecommunications network to a local server[1], from where it is sent via a sattelite link.

There are solar panels as a power source who charge up battery banks, but as there is't enough light during the Greenland winters, there're plans to build wind turbines (with the problem being the possibly detectable RF noise the 'engine' produces)

[1]There is additionally a Long Range Wide Area Network (LoRaWAN) antenna as backup in case of problems with the LTE network

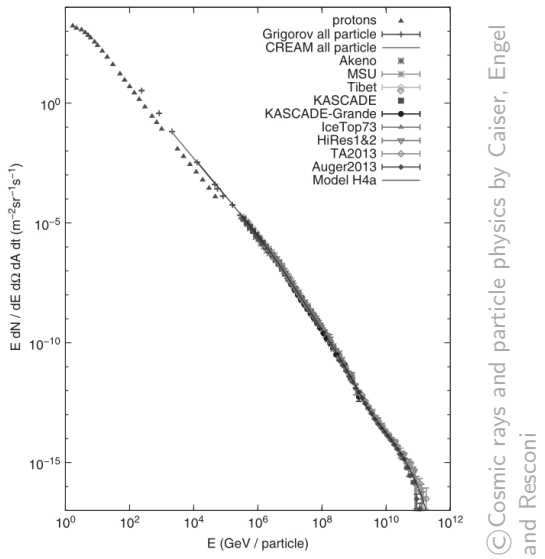


Figure 4.1: Falling cosmic ray flux with energy

The radio signal from a neutrino often travels along both direct and refracted paths (designated DnR) to the deep array, this happens because the upper ice layer is a non-uniform medium where the signal trajectory is bent, as illustrated by a simulation in figure 4.3.

This double pulse characteristic would be a smoking-gun signature of an in-ice source. The two helper strings are needed for a full direction reconstruction. Three independent measurements are needed for azimuthal information, which is provided by the Vpol (Vertical polarization) antennas and placing the Hpol (Horizontal polarization) antennas at different depths on every string, both zentih and azimuth information will be provided for those signals. The helper strings' calibration pulsers, as well as one on the surface, will ensure regular monitoring of the performance of the station and provide information useful for precise calibration of the antenna geometry.

The plan is to construct these detectors in an array as shown in figure 4.4, note that all the individual detectors are named after various species living in greenland (in the native tongue).

Christoph Welling did an investigation into energy reconstruction from the received signals [15] for air showers in one single station (as the RNO-G stations are so far apart this is the case here aswell) and he noticed that it is nescessary to know if the detector who observes an event falls inside or outside the Cherenkov cone to accurately reconstruct the primary particle energy as most over-estimated energies in his simulations are caused by events viewed from within the Cherenkov ring being mistaken for events outside of it. He went on to show that, if we somehow know if the shower was seen from inside or outside the ring from some extra source, that most outliers in the energy disappeared. It is shown by Hiller et al. [10] that the combination of a muon detector with the radio detector might make the issue of confusion between being within or outside of the Cherenkov-ring disappear. Because of this the RNO-G stations are fitted with surface Log Periodic Dipole Antennas (LPDA), capable of detecting muons. Note that this is for air showers, the radio signal from neutrinos show additional complexities.

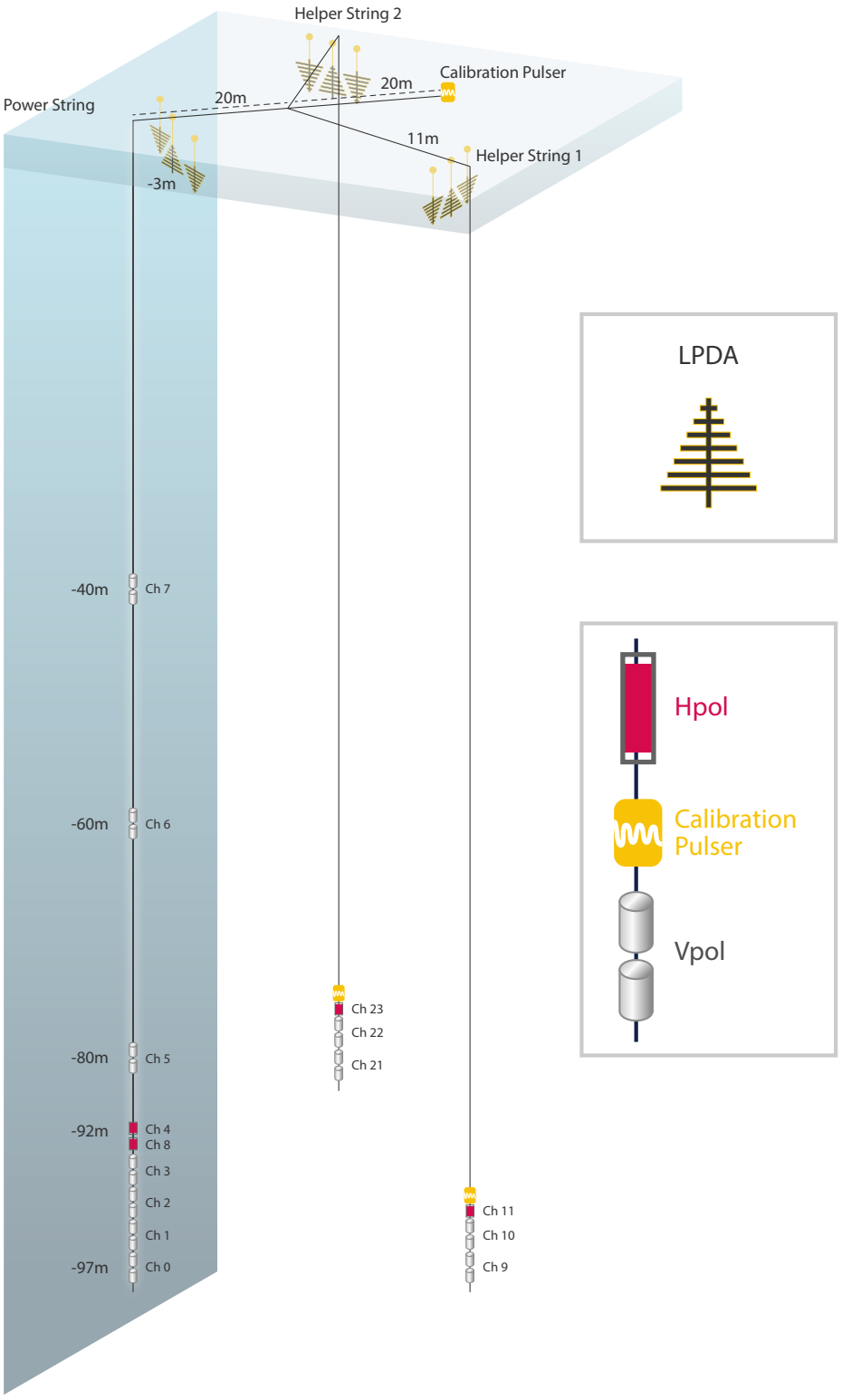


Figure 4.2: illustration of the detector

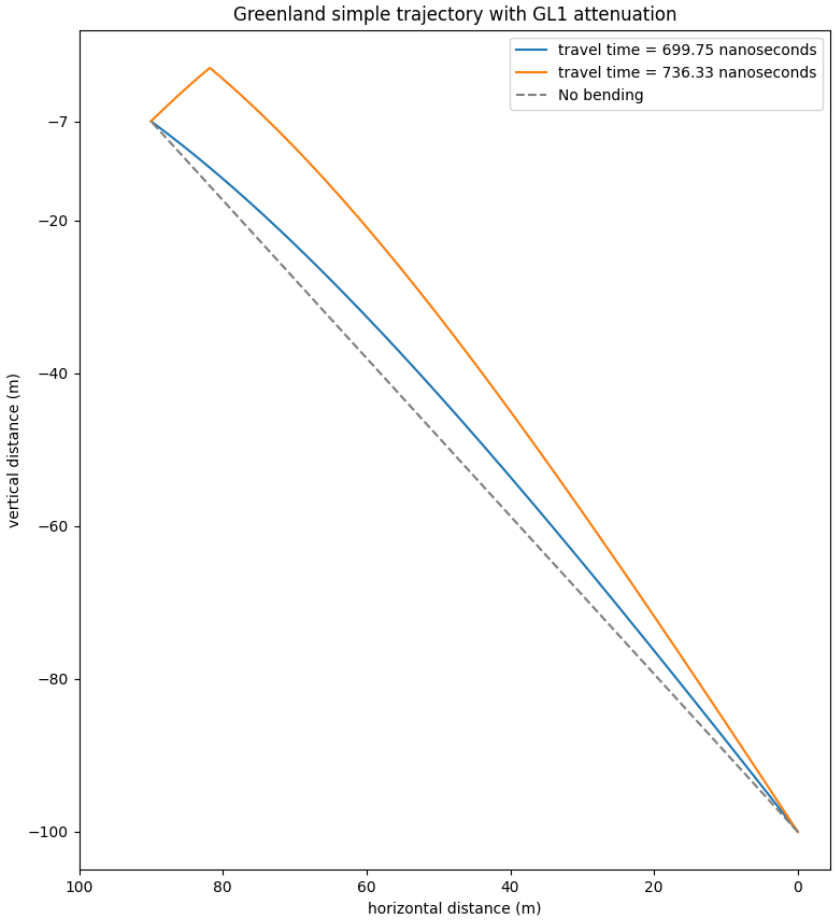


Figure 4.3: illustration of a neutrino signal path, the neutrino interaction being located at (0,-100) and the detector at (90,-7)

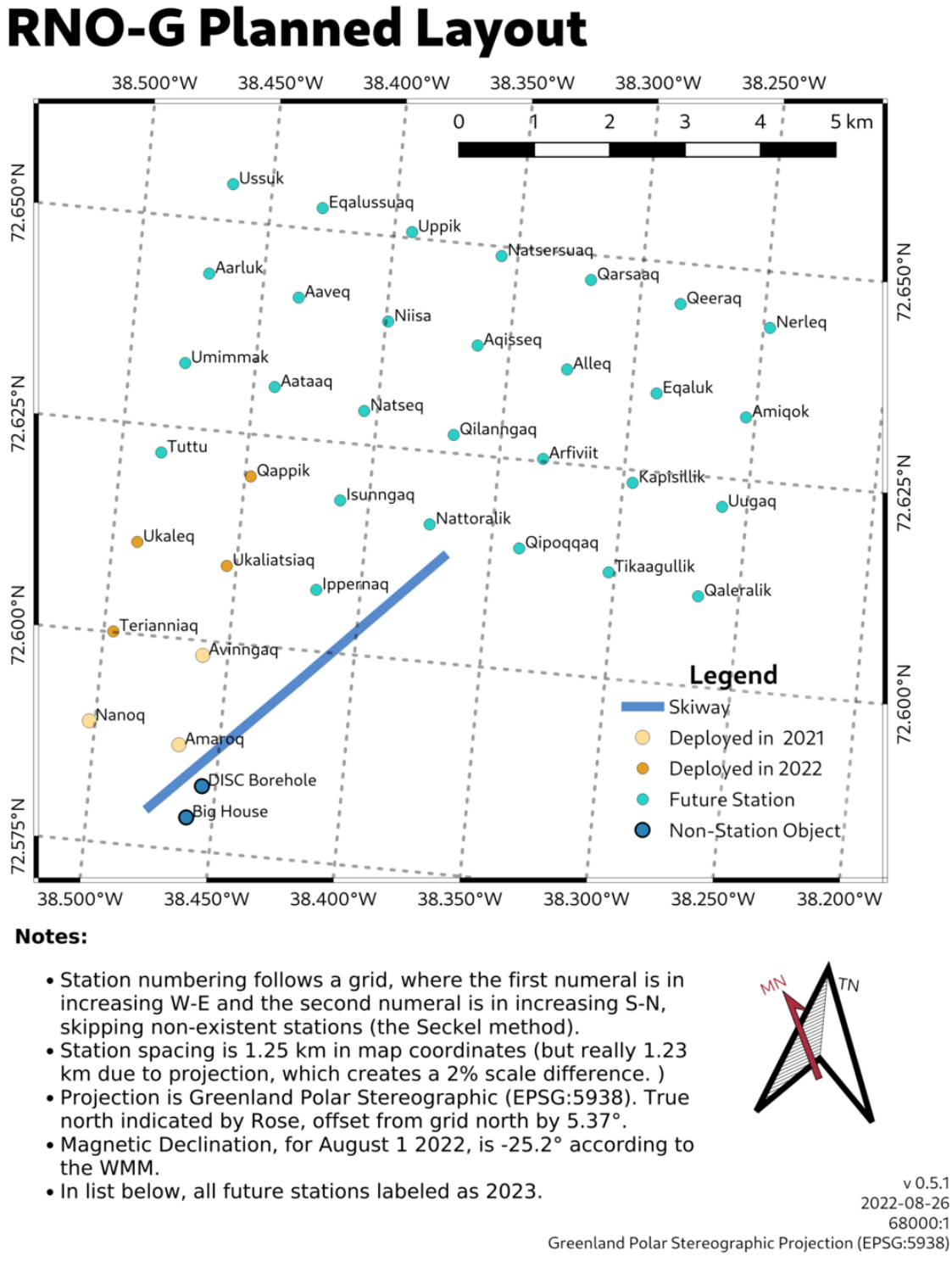


Figure 4.4: map of the station

CHAPTER

5

HYBRID RAY TRACER

5.1 Shortcomings of the exponential ice model

It has become apparent [citation needed] that complex ice models will be nescessary moving forward as the exponential ice model fails to fit the density curve. The ideal software for radio wave propagation through ice is radiopropa [16], but due to the way it works you'll have to know the start point , the end point and the launch angle of your ray to work out the path. This isn't difficult for the analytic model as it's exactly solvable but for a general ice model you'll somehow have to find where to shoot the ray. Work has been done on finding the launch angle in the case of complex ice models by B. Oeyen et al. [13], where they created a ray tracer which iteratively finds the solution, called the "iterative ray tracer". The full explanation of how their algorithm works can be found in the mentioned paper. This is however a sub-optimal solution in python as an optimalisation library will generally work faster, work had been done on trying to implement such an algorithm but this attempt failed. As we saw this work the idea came to mind to combine the iterative ray tracer and the code using the optimisation libraries (a so called "minimizer"), to come up with the algorithm that will be discussed in this chapter: The hybrid ray tracer, in the source code called the "hybrid minimizer".

It succeeds in more rapidly tracing the path from the event to the detector, is more accurate and also arrives closer to the detector as the final result is not limited by the final drawn sphere size but by a given tolerence.

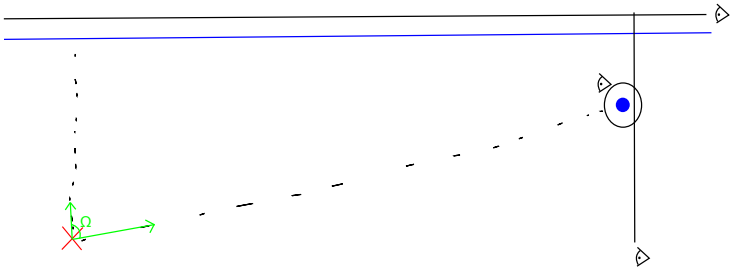
5.2 How it works

The full code can be found in appendix B, here we'll walk you through the algorithm. The hybrid minimizer can be seen as an extension of the iterative raytracer [13] as it starts out the same way: Say our source of radiation is at position $\mathbf{X}_1$ and our detector is located at position $\mathbf{X}_2$, we start by defining the vector $\mathbf{v} = \mathbf{X}_2 - \mathbf{X}_1$, then we clone it as a new vector u and set u's z coordinate to 0, making it a normal vector of a plane parallel to the z direction. we now wish to know where we'd actually be able to find possible paths, looking at figure 4.3 we see that no solutions below the direct path are possible as there would need to be upwards reflection,

so we convert our vector **v** representing the path from the source to the detector to spherical coordinates, giving us a polar angle (zenith angle) "theta-direct". With this we know that the, at the source, the ray should propagate with an initial zenith angle within the angle interval 0° to theta-direct° := Ω.

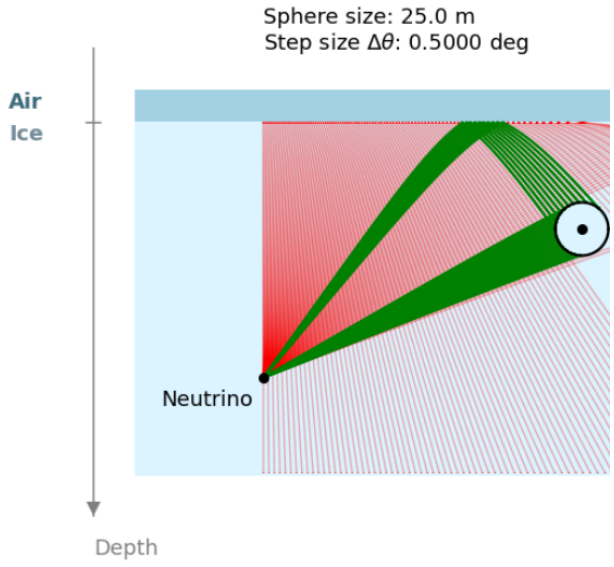
Next we need to define our "observers", if you shoot a ray with the radiopropa module from a certain point at a certain angle the ray path will get simulated until it interacts with this "observer". Ideally we would like to a priori know where to shoot our ray and have the detector be an infinitesimally small observer in our simulation, but as we'll be working with general ice models this can't be done.

The algorithm of finding the possible paths is then as follows: We define a spherical observer at the location of the detector, with a radius of fair size. We place an observer plane directly behind the detector (with normal vector **u**, no rays can propagate back after crossing) and an observer above the surface (as no rays could make it back after escaping the ice) our full setup is then what's illustrated in the figure below

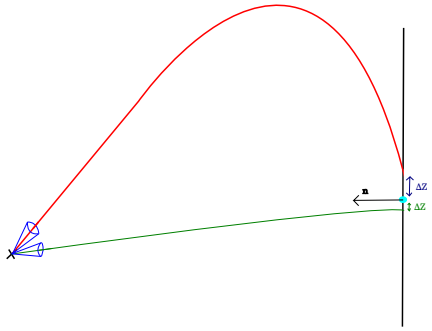


Here the red cross is the radio source, the blue horizontal line is the ice-air boundary surface, the blue dot to the right is the detector and the green Ω indicates the range over which solutions to the problem are possible.

We start off by just iteratively guessing: given a certain angle stepsize $\Delta\theta$ shoot rays at the angles $\{0, \Delta\theta, 2\Delta\theta, ..., \Omega\}$ And see which ones get detected at the sphere around the detector, this process is illustrated in a modified version of B. Oeyen et al. their figure below



if there are 2 distinct launch regions, it will start the so called "minimization", using scipy's module optimize.minimize. First we get rid of the spherical observer and place the vertical observer exactly at the detector, now to be able to use the minimize module we'll need a function to minimze, for this reason we define the function *delta_z* as, given a certain launch angle, returning the distance from the point where it lands on the plane to the detector as illustrated below



The function we'll minimize is then detla_z_squared which is just the square of delta_z as we wish it to be as close as possible to 0, it gets minimized within the angle boundaries found from the previous step. With this our algorithm is done, it does have a fail-safe as well for if the first step, finding the launch regions, doesn't work namely it reverts back to being the iterative ray tracer.

5.3 Performance Optimalisation

To test the hybrid minimizer the numpy random module was used to generate random coördinates, the considered square (as there is only a z component to the ice model the 3D problem is essentially only a 2D problem) is x:0.1km,4km and z:-0.1km,-3km.[1] Every test point shown in the following subsections consists of at least 500 random initial positions. As the speed of the algorithm is computer dependent the algorithm's speed is always plotted relative to the iterative ray tracer's speed, simulated with the same coordinates at the same time.

5.3.1 Length of the normal vector

As visually explained in figure 5.1, the size of the normal vector seems to influence how big the ray tracer's step size is taken close to the detector. This thus influences the convergence and time taken. The results of varying this are shown in figures 5.4 and 5.5. Looking at these figures the first optimization conclusion is as expected: take the normal vector length to be 1 meter.

5.3.2 ztol

We'll now change the tolerence on the vertical distance away from the detector which is deemed accepted i.e in figure 5.1 if Δz is below this threshold it's accepted. The results are shown in figures 5.6 and 5.7. From which we can conclude the second optimization conclusion: take ztol to be 0.05 m.

[1] This start at 100m depth was to get around issues concerning events that won't even trigger in a full simulation

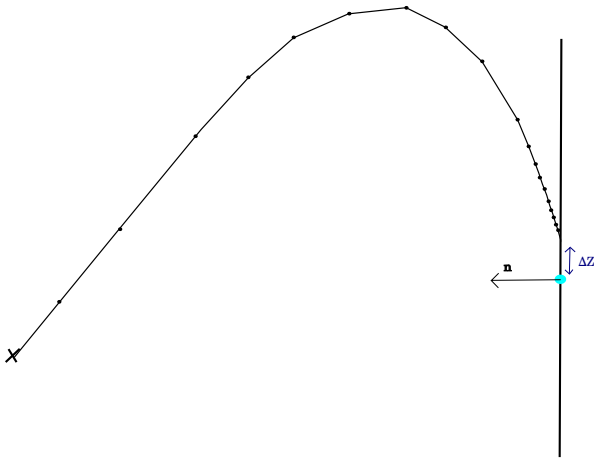


Figure 5.1: how normal vector size influences the stepsize

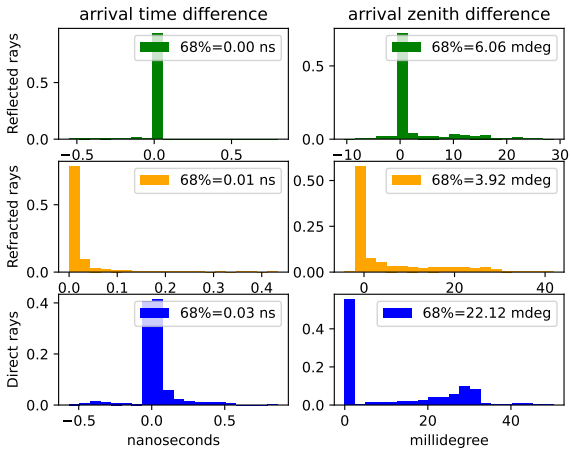


Figure 5.2: Hybrid

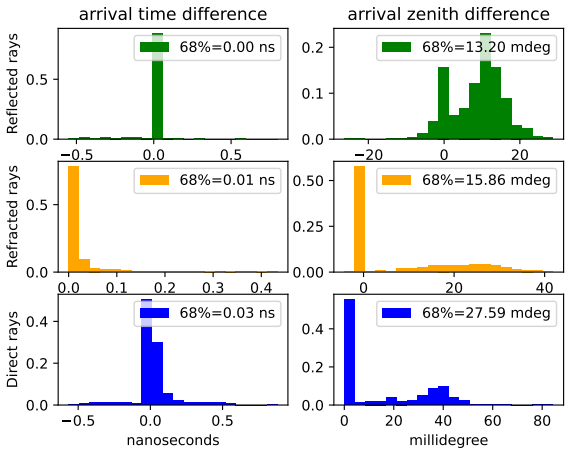


Figure 5.3: Iterative

5.3.3 Sphere Size & Step Size

The initial rays are sent out in steps of a certain angle and with a sphere around the detector of a certain size. As this initial search for launch angle regions is the slowest step in the hybrid ray tracer it's imperative to optimize this. The procedure was: change the sphere size and loop over various step sizes, recording the speed. The results are shown in figure 5.8 the lower on the chart the better, zooming in onto the lowest point as is shown in a combined plot on figure 5.9, we see that an optimum seems to be around a spheresize of 45m and a stepsize of 0.7°.

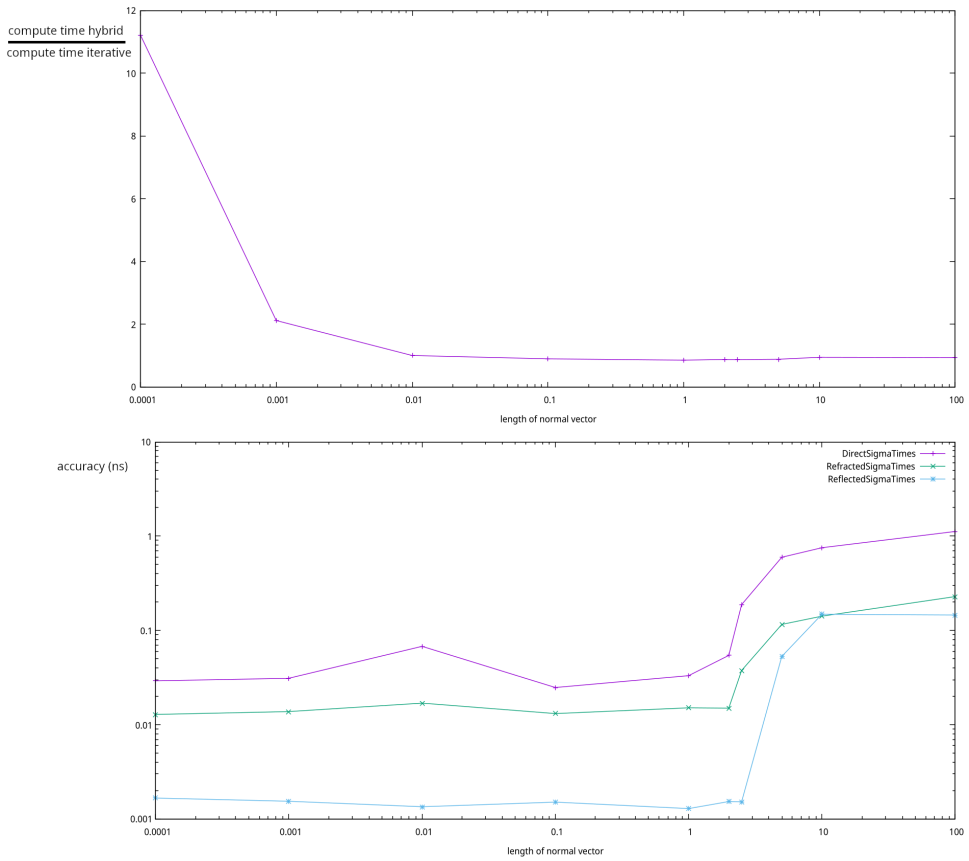


Figure 5.4: influence of the length of the normal vector

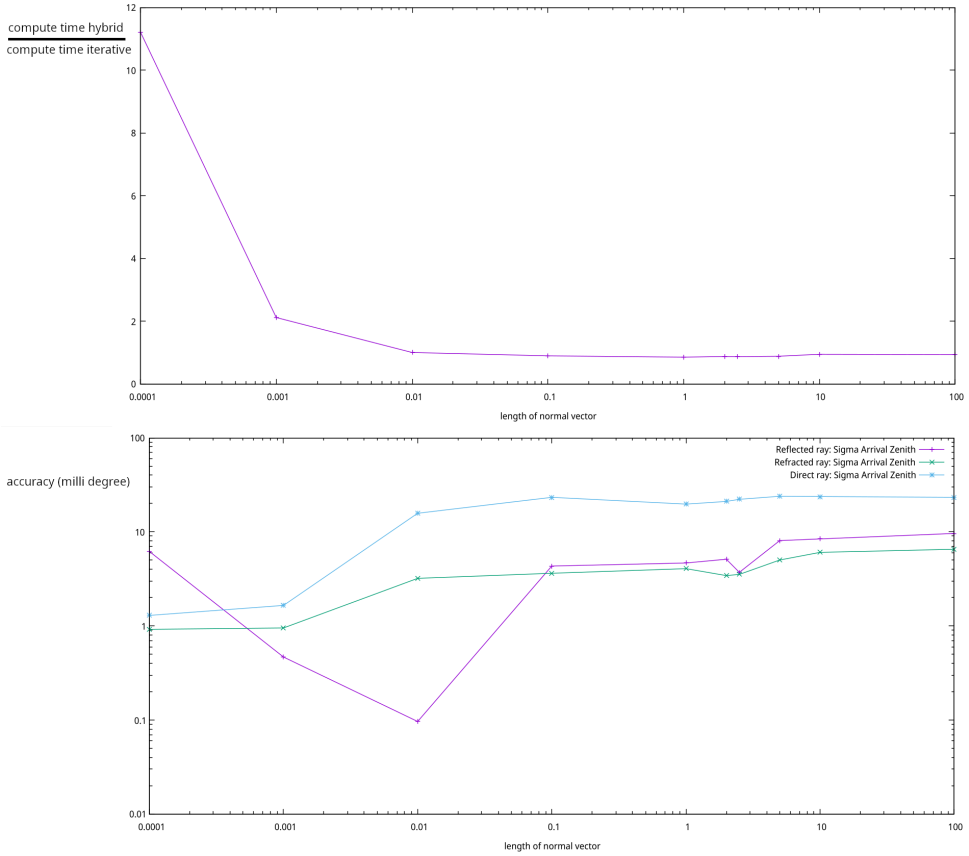


Figure 5.5: influence of the length of the normal vector

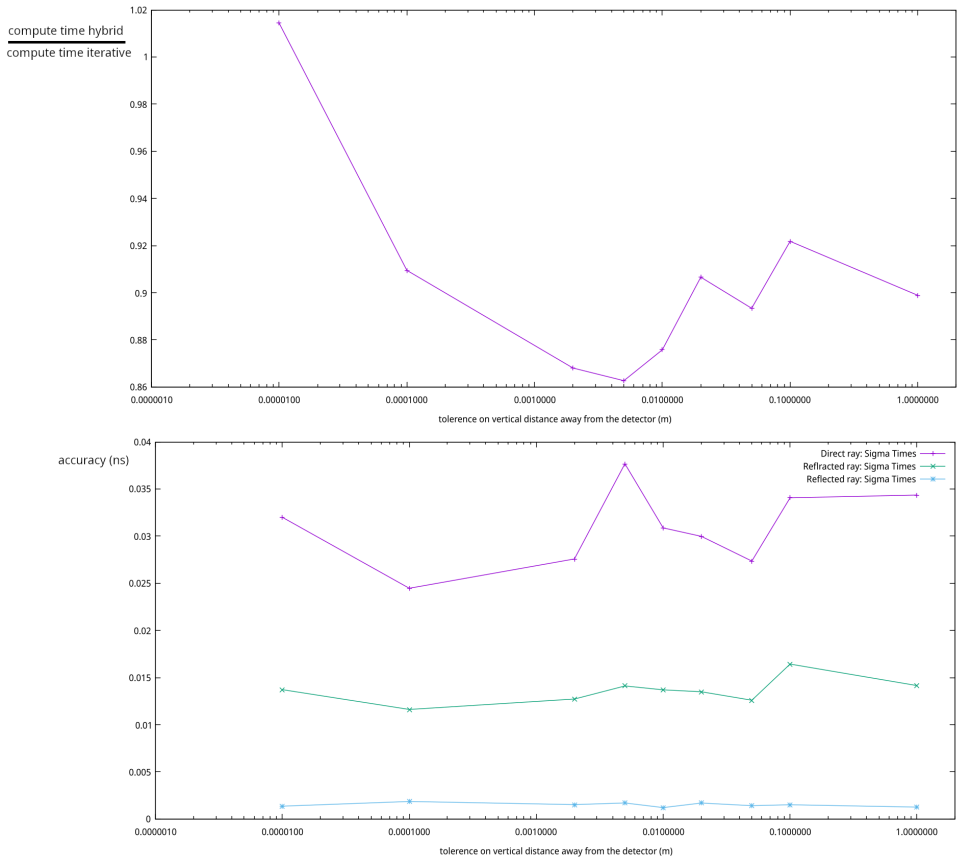


Figure 5.6: influence of the tolerence on vertical distance

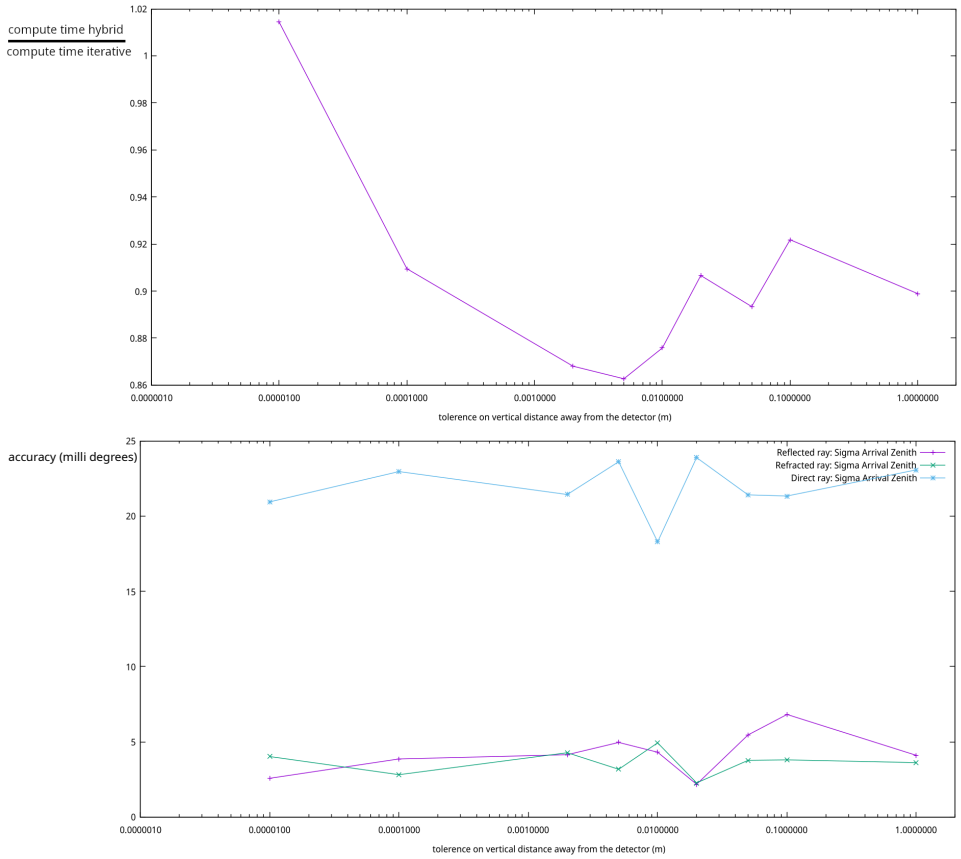


Figure 5.7: influence of the tolerence on vertical distance

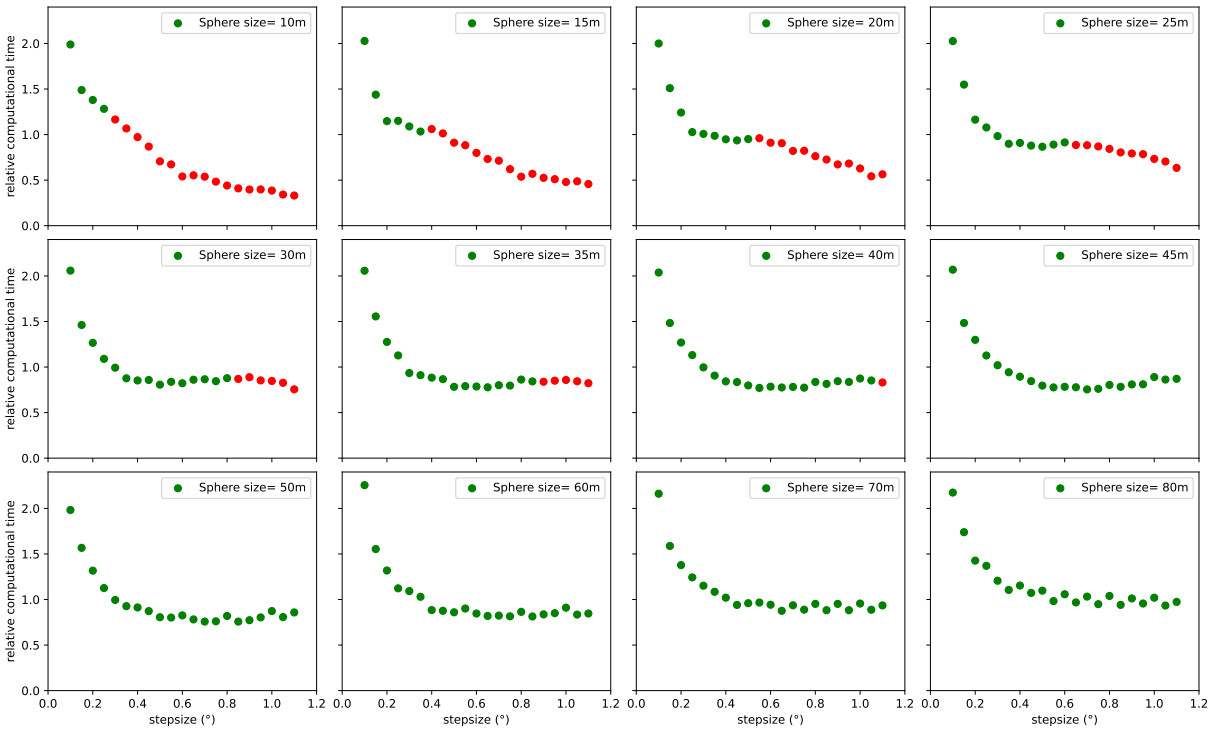


Figure 5.8: Variation in Sphere and angle step size with report on relative time.

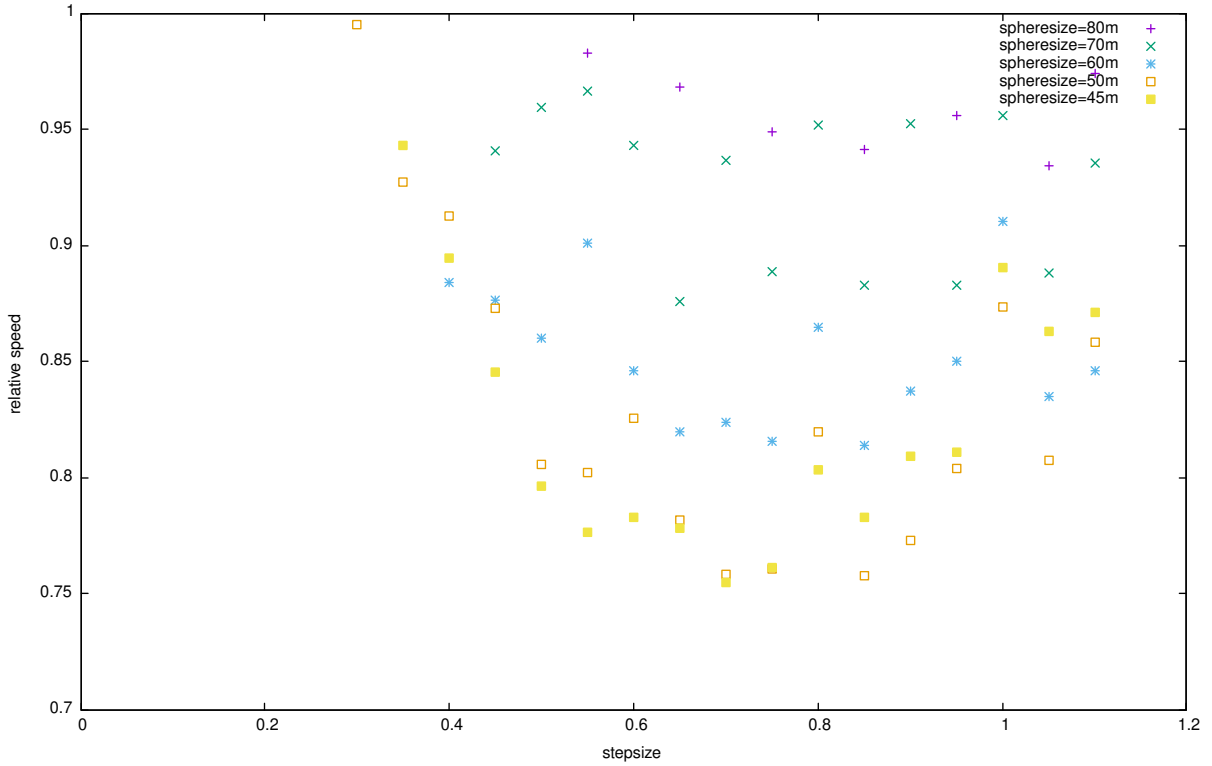


Figure 5.9: Green values in variation in sphere and angle step size with report on relative time.

CHAPTER

6

WEATHER BALLOON

In this chapter we'll simulate radio signals coming from a weather balloon flying over the stations. Our goal is to use the plane waves method to reconstruct the position of the weather balloon with the timing information inferred from the detected radio signals in the detectors. If this is somewhat succesfull it can be used (as we'll get to shortly) to find out the local index of refraction in the ice.

There are 2 changes that need to happen first however to our algorithm for us to be able to simulate this: The air observers needs to be hard removed as to make a ray tracing possible from within the air to the detector and second off, we'll need to implement secondaries. The problem is essentially that a ray coming from the balloon propagates to the ice, refracting and reflecting and then one of the refracted rays hits the detector. The "primary" ray is concidered the reflected one so we adjusted the algorithm to look at the secondary that ends closest to the detector and return that ray. What this has as a consequence is that the "path" you get back is only the path from when it "became" a secondary (so only the part below the ice) but this can be easily fixed as the radio wave just propagates straight from the balloon to the beginning of that ray, making the full ray reconstructable by just assuming a line from the balloon to the "start" of the ray.

From this information the propagation time from the balloon to the ice t can later be added to the time of the path in the ice by measuring the length d of the drawn line from the balloon to the beginning of the recorded path, setting the speed of radio waves in air to c and then just $t = d/c$.

6.1 Plane Wave Reconstruction

Now having modified our ray tracer, the first problem we'll consider is plane wave reconstruction of the original position, an example path to some of the deep sensors is given in figure 6.1 The plane wave reconstruction can easily be understood using figure 6.2, the waves coming in are drawn in blue and make a certain angle with the detectors. the top detector (top box) detects the wave at a certain time t_1, the bottom detector detects it at a time t_2. In our database, after

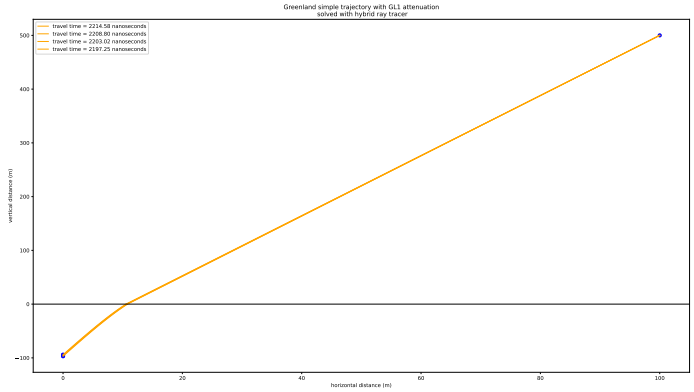


Figure 6.1: Example trajectory of rays coming from a weather balloon (blue dot top right) and going through the ice to the various detectors (blue dots bottom left)

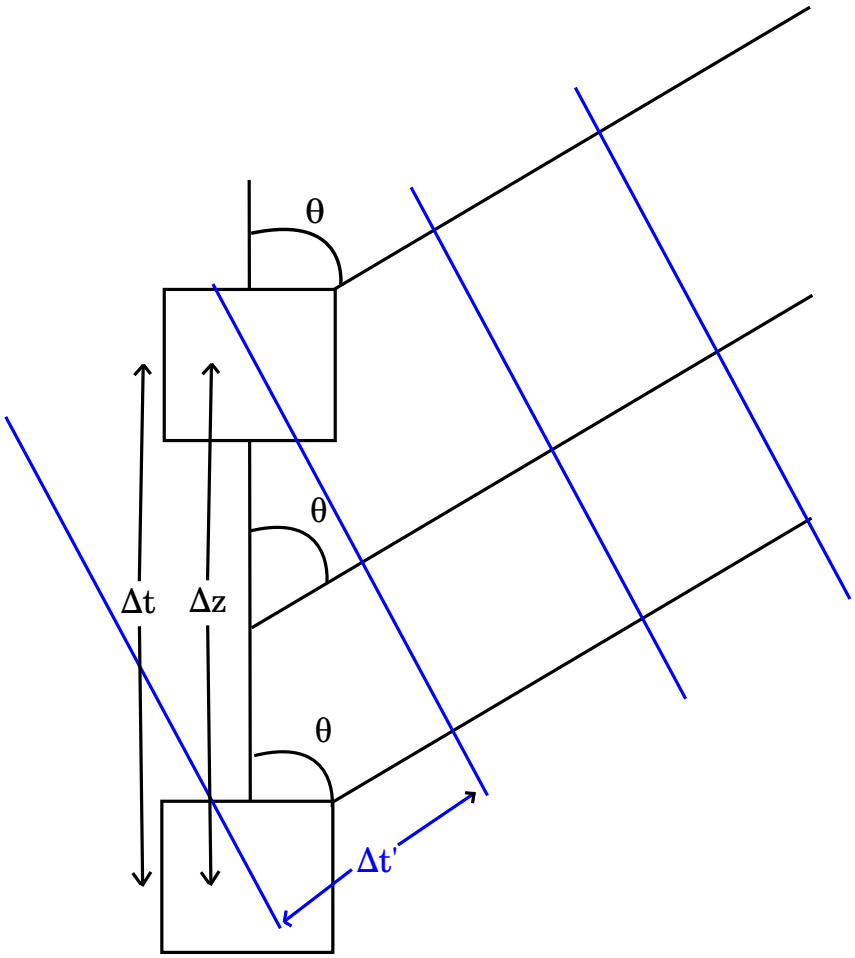


Figure 6.2: Illustration of Plane waves

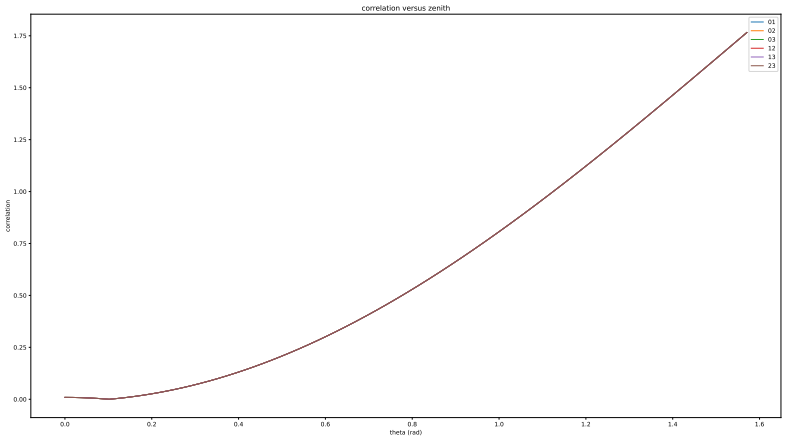


Figure 6.3: Example normed correlation functions

decoding the signal we'd thus see that these two detectors got a signal $\Delta t = t_1 - t_2$ seconds apart from eachother. Now ideally this is equal to the time $\Delta t'$ which is the time it took the wave to propagate that distance which we can calculate from basic trigonometry and dimensional analysis:

$$\Delta t' = \frac{m}{(m/s)} = (s/m) * m = v^{-1} * m = v^{-1} \cos\theta \Delta z \tag{6.1}$$

With $v = c/n$ the local speed of light. As previously discussed this n is depth-dependent and we'll be using the local index of refraction. Say we have 4 detectors at depths -94,-95,-96,-97, this then would mean that we'll ask for n at a depth of -95.5, exactly inbetween:

```
ice = medium.greenland_simple()
n = ice.get_index_of_refraction(np.array[0,0,-95.5])
```

Now in reality we don't know the angle a priori, we'll only have the timing information, so we'll perform a scan by minimizing something we define here as *the correlation function*:

$$Correlation(\theta) := \Delta t - \Delta t' = \Delta t - \frac{\cos\theta \Delta z}{v} \tag{6.2}$$

If we have more than 1 detector however (which of course will be the case in RNO-G), we'll need to specifiy various correlation functions. E.g if we have four detectors labeled 0 to 3 we'll have to construct correlation functions between detectors 0&1, 0&2, 0&3, 1&2, 1&3 and 2&3 . As all of these correlation functions will have different sizes we'll norm them as follows:

$$Correlation_{Normed}(\theta) = \frac{Correlation(\theta)}{\int Correlation(\theta) \Delta\theta} \tag{6.3}$$

An example of these correlation functions is shown in figure 6.3, notice how you can't differentiate between the correlation functions, this is only possible because of the hybrid ray tracer having that high of a precision. After this we can sum them, as shown in figure 6.4, and look where it reaches it's minimum. Using this angle we can then reconstruct a ray and guess where the weather balloon is approximately, this is illustrated in figure 6.11.

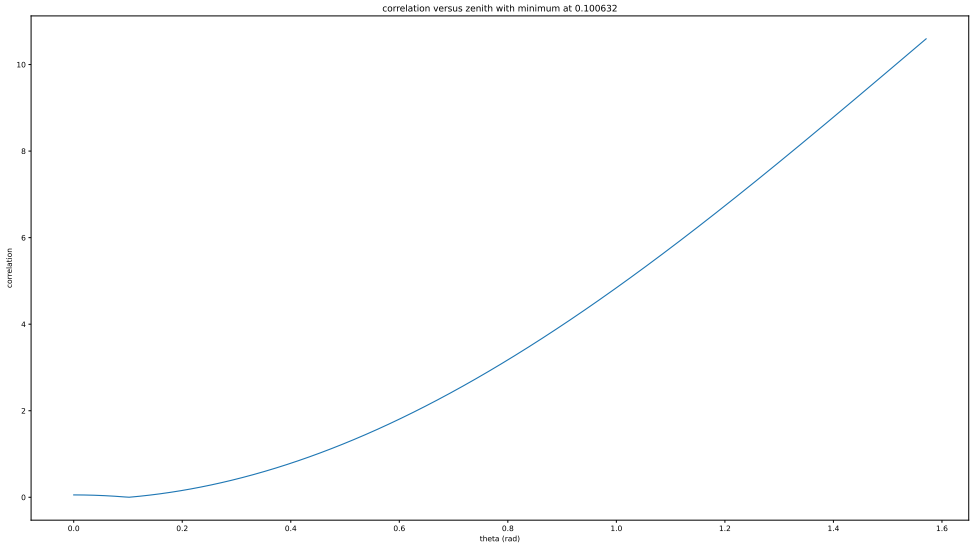


Figure 6.4: Example sum of the normed correlation functions

6.2 Is the goal feasible?

In the example reconstruction illustrated in figure 6.11 the difference in angle between direct to balloon and plane wave reconstruction is already quite small (0.65329617%) but as the balloon gets closer to the detector this reduces significantly as is shown in figure 6.12 where the difference in angle between direct to balloon and plane wave reconstruction is only 0.06788141%. Our goal is to find the local index of refraction n by using the plane wave reconstruction with the recorded timing and the positional data from the weather balloon as the plane wave reconstruction is heavily dependent on the index of refraction (as can be seen in equation 6.2). As was already established we'll consider the index of refraction in the middle of all the detectors (instead of a different one for every pair of detectors), after experimenting this doesn't seem to have an impact on the accuracy of the plane wave reconstruction.

Now we need to ask ourselves the question, within which angles should the weather balloon fly for the data to be useful? As was previously stated, the further the weather balloon is away (in the x direction) the bigger the zenith angle with the detector the less accurate the plane wave reconstruction. So which angles are acceptable? To determine this our method works as follows:

we vary the position of the weather balloon in the x direction (keeping the height constant), reconstruct the ray path from channels 0 to 3 and then fit n such that the difference between the reconstructed angle and the direct angle is the smallest possible. Then we compare the n we have fit to the one we know from the model at that position. We quantise the discrepency between these two indices of refraction using what we here define as the *relative accuracy*:

$$\varepsilon\ (\%) = \frac{n_{\text{fit}} - n_{\text{actual}}}{n_{\text{actual}}} \times 100 \tag{6.4}$$

Carrying this out we get figure 6.5, i.e it gets exponentially less accurate as the balloon moves further away. As we wish our accuracy to be within 1%, the angle the balloon makes with the detector needs to be less than 10° for the deep channels 0 to 3. An example path of

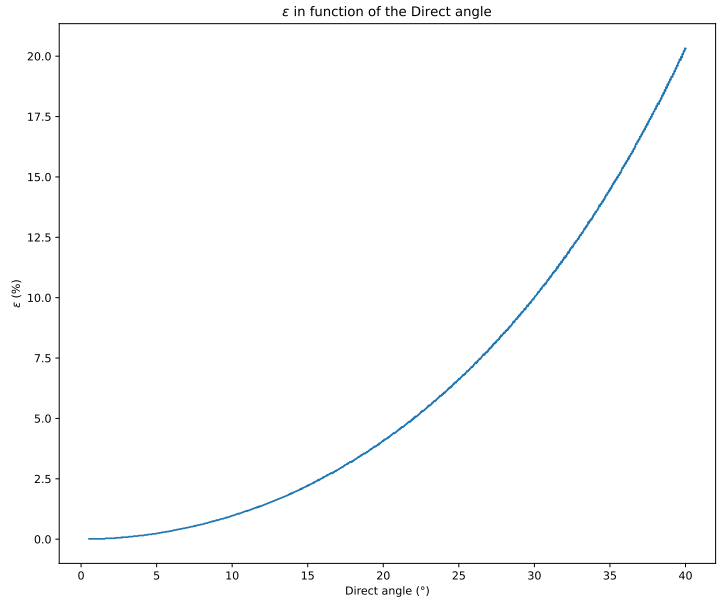


Figure 6.5: Epsilon in function of the direct angle

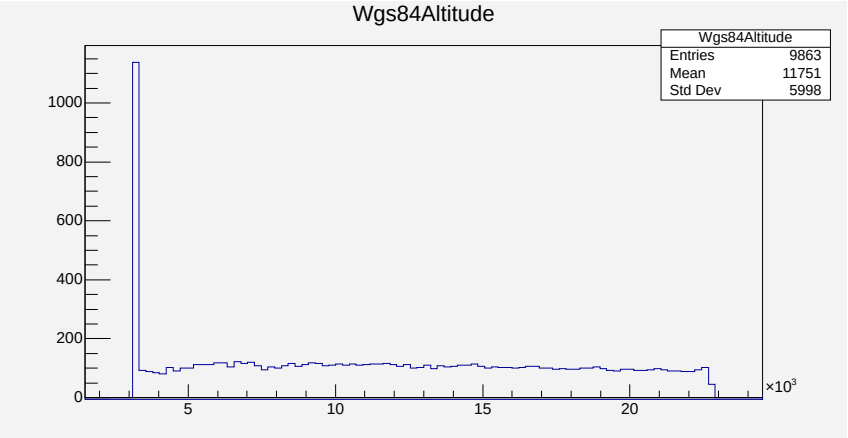


Figure 6.6: Height data viewed in ROOT

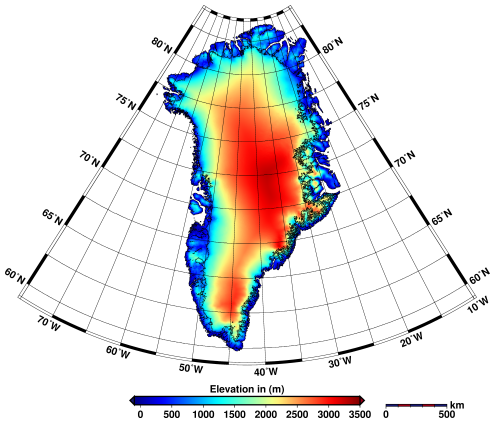


Figure 6.7: Height map of greenland

a weather balloon is shown in figure 6.10, looking specifically at the height using ROOT[1] we get what is shown in figure 6.6. It can be seen that the elevation varies between 3228m (read the graph as: Over a 1000 data entries at that height) and 22755m (entries go to zero after that height). This is relative to sea-level, looking at the height map of greenland as shown in figure 6.7 this is obvious. It's quite difficult to work directly with the global geographic coordinate system (longitudinal, latitude and elevation coordinates), that's why we convert them to local ENU coordinates (north, east, up) this is all relative to the 'DISC' which is located at 72.58279265212887 latitude, -38.45581495328228 longitude and 3251.9489147560234 elevation.

Now what would our <10° policy entail? And is it even possible? Say we take a look at station Terianniaq (station 12) it's located at 72.6000868058195° N -38.4962265332872° W. converting this into ENU coordinates we get -137.67250176003688N 1727.5184983294744E now our "up" is then -95.5m as this is the approximate location of the middle of channels 0-3. Now looking at e.g the Balloon path recorded on the 20th of august 2022 (figure 6.9) we see that the balloon crosses paths with detector 12, how close was this encounter? We can just look at every data entry (there are >12000 entries) individually and compute the angle the balloon makes with the detector by first converting the location of the balloon in ENU coordinates, then calculating the horizontal distance ($\sqrt{(x-x')^2+(y-y')^2}$) then the relative vertical distance $|z-z'|$ and then from those compute the angle ($tan(\theta) = \frac{Hor.}{Vert.}$) doing this and recording when the balloon gets close enought to get below the 10° mark we get figure 6.8. We thus see that in this example the $< 10°$ policy is viable.

6.3 Fitting the index

Now that we know our goal to be feasible for angles smaller than 10 degrees, let's analyse the data. The positional data of the weather balloons was obtained from the `ftp1.esrl.noaa.gov` website using the rno-g-sonde script Of the official RNO-G github page. We'll be looking at the data recorded over the summer of 2022, more particularly 15/06/2022 - 30/09/2022. The detected events can be obtained from the DESY database[2].

Let's do an example rundown of how this analysis is going to go: let's start by looking at the weather balloon file SMT_20220820_111609.root, it was recorded the 20th of august 2022 as can be derived from it's name, we have a complementary file SMT_20220820_111609.gpx this can

[1] https://root.cern/
[2] https://rnog-monitor.zeuthen.desy.de/

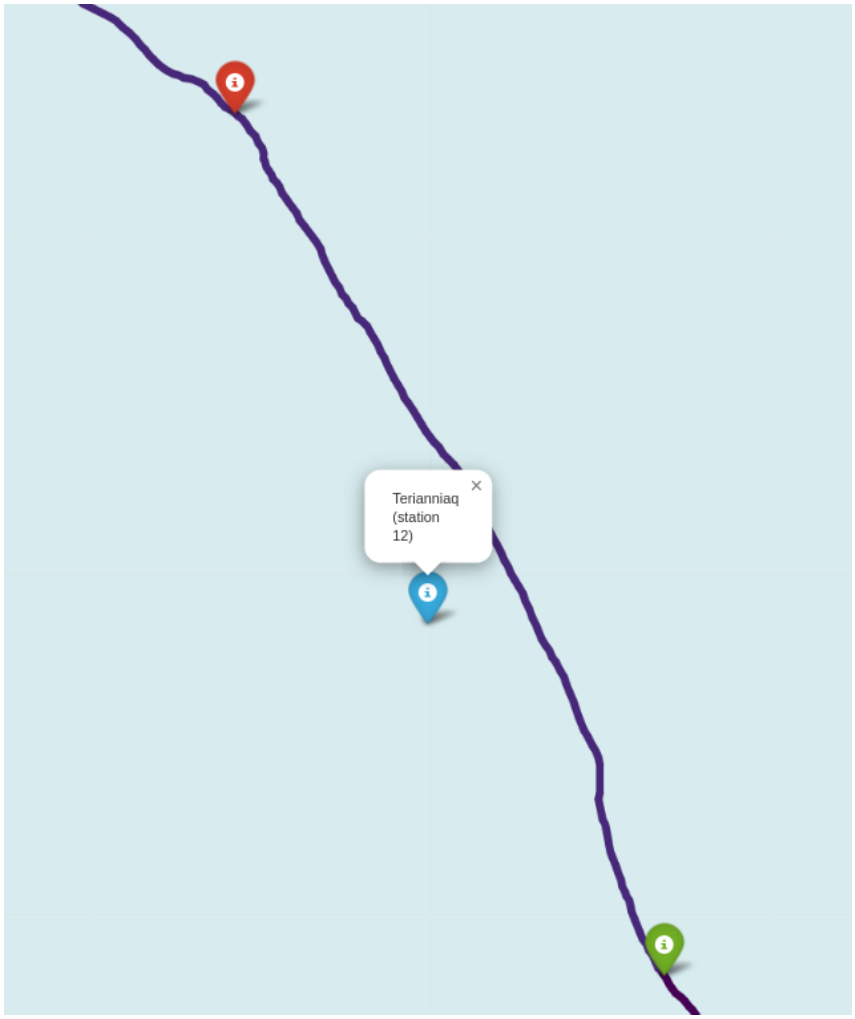


Figure 6.8: Illustration of when the angle with the deep array (channels 0-3) with the weather balloon is less than 10 degrees, the green mark indicates when it starts being less than 10 degrees and the red mark when it stops being less than 10 degrees.

Figure 6.9: Path traced out by a weather balloon released at 20/08/2022

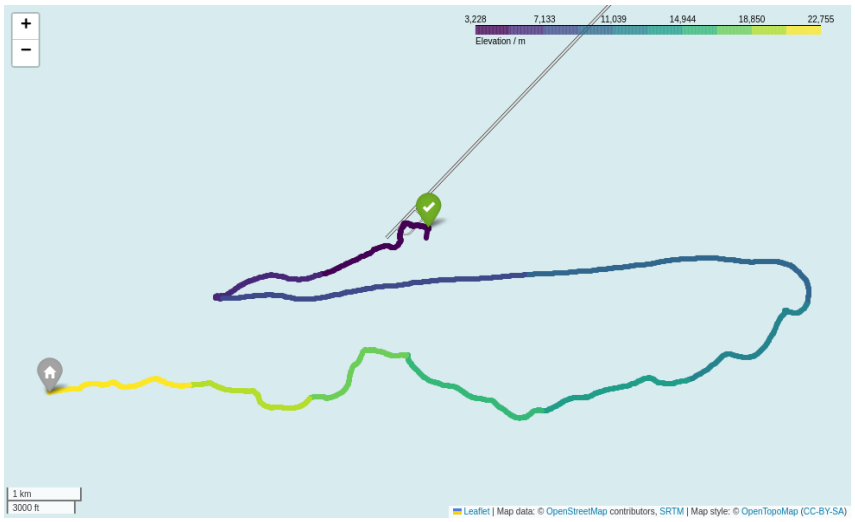


Figure 6.10: Path traced out by a weather balloon released by Bob Oeyen, the checkmark indicating the start and the house mark the end

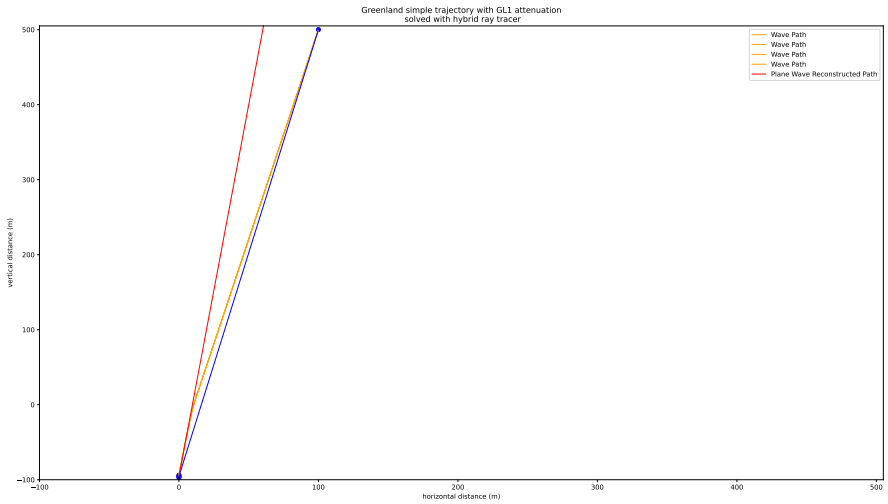


Figure 6.11: Example plane wave reconstruction of weather balloon position

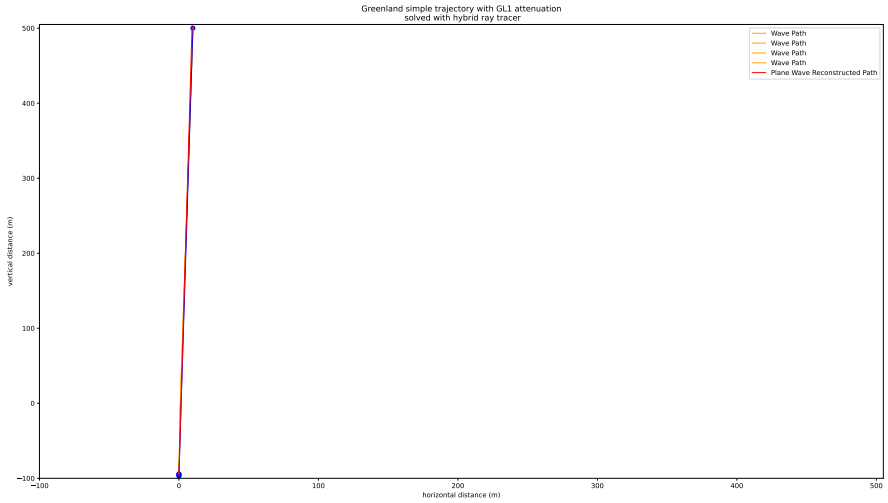


Figure 6.12: Example plane wave reconstruction of near-flying weather balloon position

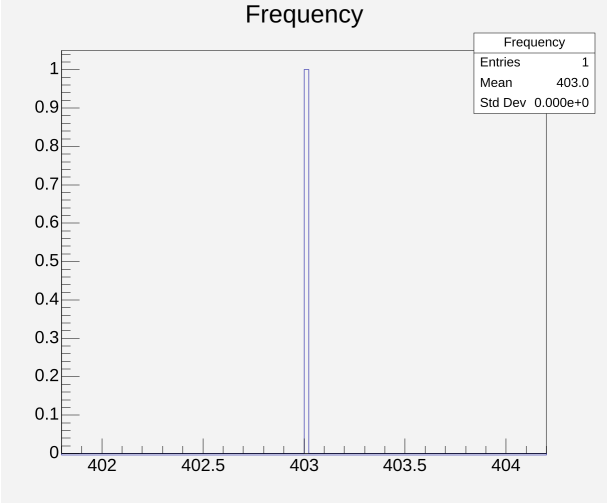


Figure 6.13: Frequencies sent out by the weather balloon

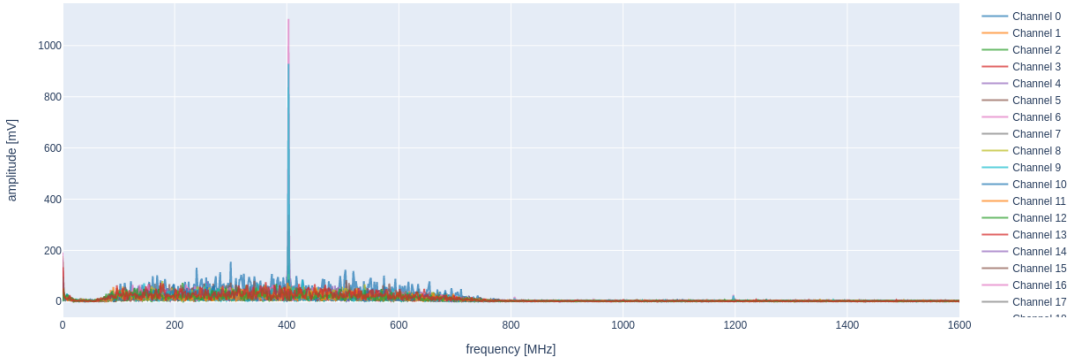


Figure 6.14: Frequencies received by the various detectors

be graphically displayed using the python module gpxplotter resulting in the trajectory shown in figure 6.15 We can see it clearly first coming close to the station Amaroq (station 21) and subsequently very close to Terianniaq (station 12), making this event usable for our analysis. If we take a look at this day in the event browser we see quite a lot of activity as is shown in figure 6.16. We, however, are only interested in the brief timeframe where the weather balloon passes these detectors closely and only in signals coming from these particular detectors. Let's do the analysis previously described to find when the angle the detector makes (at -95.5m depth) with the Balloon is less than 10° and record the timestamps, if we do this we'll find that we'll need to look inbetween 11:24:02 and 11:26:23. This seems to be within run 528 as it runs from 10:45 till 12:45. If we now take a look at the file SMT_20220820_111609.root with ROOT we can see that the balloon sends out signals at a frequency of 403 as shown in figure 6.13, if we now take a look at event 1981 in run 528 which was recorded at 20/08/2022 11:25:51 we see the spectrum shown in figure 6.14 which shows a spike at 403MHz for the surface detectors. Let's now go through all the different files and see in which ones the Balloon gets close "enough"

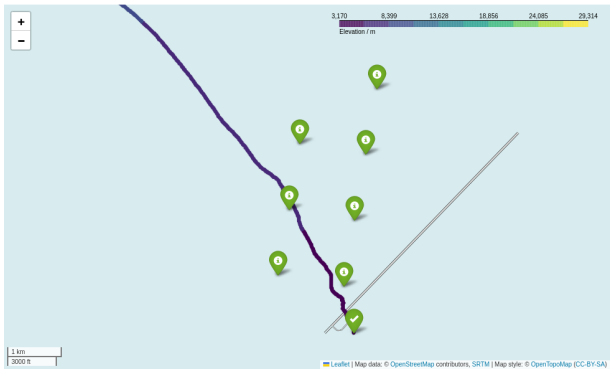


Figure 6.15: Part of a weather balloon path on the 20th of august, the detectors placed as of 2023 are illustrated with a green info mark

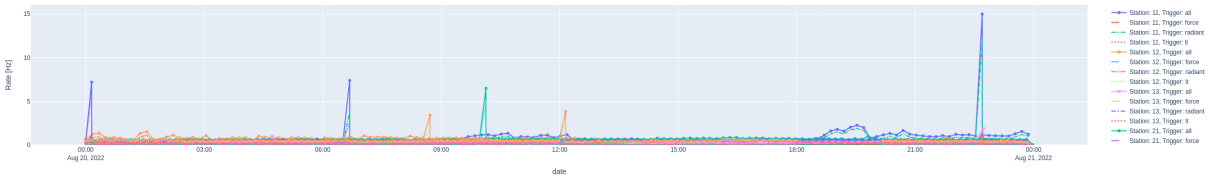


Figure 6.16: Event rate recorded on the 20th of august

APPENDIX

A

LIST OF ABBREVIATIONS

APPENDIX

B

HYBRID MINIMIZER CODE

```
    def raytracer_hybrid_minimizer(self,n_reflections=0):
        """
        Uses RadioPropa to first find all the numerical ray tracing solutions between sphere x1
        and sphere x2 for a big sphere. After which it uses the scipy optimize.minimize module
        to find the best path in these angle intervals.
        Tracer does not work for reflective bottoms or secondary creation at the moment
        """

        try:
            X1 = self._X1 * (radiopropa.meter/units.meter)
            X2 = self._X2 * (radiopropa.meter/units.meter)
        except TypeError:
            self.__logger.error('NoneType: start or endpoint not initialized')
            raise TypeError('NoneType: start or endpoint not initialized')

        def get_ray(theta,phi):
            ray_dir = hp.spherical_to_cartesian(theta,phi)
            if ray_dir.shape==(3,1): ray_dir = ray_dir.T[0] #doesn't always give the right shape
            source = radiopropa.Source()
            source.add(radiopropa.SourcePosition(radiopropa.Vector3d(*X1)))
            source.add(radiopropa.SourceDirection(radiopropa.Vector3d(*ray_dir)))
            ray = source.getCandidate()
            return ray

        def shoot_ray(theta):
            ray = get_ray(theta,phi_direct)
            sim.run(ray, True)
            return ray

```

```
30          def cot(x):
31              return 1/np.tan(x)
32
33          def arccot(x):
34              return np.arctan(-x) + np.pi/2
35
36          def delta_z(cot_theta):
37              theta = arccot(cot_theta)
38              ray = shoot_ray(theta)
39              ray_endpoint = self.get_path_candidate(ray)[-1]
40              return (ray_endpoint-self._X2)[2]
41
42          def delta_z_squared(cot_theta):
43              return delta_z(cot_theta)**2
44
45          def MinimizeAble(lower,upper):
46              #This checks if there are 2 distinct regions, takes about 10^-6% of
                ↪  the total time
47              if ((len(lower) == 2) and (len(upper)==2)):
48                  return ((lower[1] > upper[0]) and (lower[0] < upper[0]) and
                    ↪  (lower[1] < upper[1]))
49              else:
50                  return False
51
52          LetsMinimize = False
53
54          v = (self._X2 - self._X1)
55          u = copy.deepcopy(v)
56          u[2] = 0
57          theta_direct, phi_direct = hp.cartesian_to_spherical(*v) # zenith and
            ↪  azimuth for the direct linear ray solution (radians)
58          cherenkov_angle = np.arccos(1. /
            ↪  self._medium.get_index_of_refraction(self._X1))
59
60          if np.linalg.norm(u) != 0:
61              delta_theta =
                ↪  2*abs(self.delta_theta_direct(dz=self._sphere_sizes[0]))
62          else:
63              delta_theta = self._step_sizes[0]/units.radian
64
65          ## regions of theta with posible solutions (radians)
66          launch_lower = [0]
67          launch_upper = [theta_direct + delta_theta] # below theta_direct no
            ↪  solutions are possible without upward reflections
68
69          if n_reflections > 0:
70              if self.medium.reflection is None:
71                  self.__logger.error("a solution for {:d} reflection(s) off the
                    ↪  bottom reflective layer is requested,"
72                                      +"but ice model does not specify a reflective
                                        ↪  layer".format(n_reflections))
73                  raise AttributeError("a solution for {:d} reflection(s) off the
                    ↪  bottom reflective layer is requested,"
```

```
                                    +"but ice model does not specify a reflective
                                    ↪  layer".format(n_reflections))
            else:
                z_refl = self._medium.reflection
                rho_channel = np.linalg.norm(u)
                if self._X2[2] > self._X1[2]:
                    z_up = self._X2[2]
                    z_down = self._X1[2]
                else:
                    z_up = self._X1[2]
                    z_down = self._X2[2]
                rho_bottom = (rho_channel * (z_refl - z_down)) / (2*z_refl - z_up
                ↪  - z_down)
                alpha = np.arctan((z_down - z_refl)/rho_bottom)
                ## when reflection on the bottom are allowed, a initial region
                ↪  for theta from 180-alpha to 180 degrees is added
                launch_lower.append(((np.pi/2 + alpha) -
                ↪  2*abs(self.delta_theta_bottom(dz=self._sphere_sizes[0],
                ↪  z_refl=z_refl) / units.radian)))
                launch_upper.append(np.pi)
        #we first do the same as in the iterative solver
        for s,sphere_size in enumerate(self._sphere_sizes):

            sphere_size = sphere_size * (radiopropa.meter/units.meter)
            detected_rays = []
            results = []

            ##define module list for simulation
            sim = radiopropa.ModuleList()
            sim.add(radiopropa.PropagationCK(self._ice_model.get_scalar_field(),
            ↪  1E-8, .001, 1.)) ## add propagation to module list
            for module in self._ice_model.get_modules().values():
                sim.add(module)
            sim.add(radiopropa.MaximumTrajectoryLength(self._max_traj_length *
            ↪  (radiopropa.meter/units.meter)))

            ## define observer for detection (channel)
            obs = radiopropa.Observer()
            obs.setDeactivateOnDetection(True)
            channel =
            ↪  radiopropa.ObserverSurface(radiopropa.Sphere(radiopropa.Vector3d(*X2),
            ↪  sphere_size)) ## when making the radius larger than 2 meters,
            ↪  sometimes three solution times are found

            obs.add(channel)
            sim.add(obs)

            ## define observer for stopping simulation (boundaries)
            obs2 = radiopropa.Observer()
            obs2.setDeactivateOnDetection(True)
            w = (u / np.linalg.norm(u)) * 2*sphere_size
            boundary_behind_channel =
            ↪  radiopropa.ObserverSurface(radiopropa.Plane(radiopropa.Vector3d(*(X2
            ↪  + w)), radiopropa.Vector3d(*w)))
```

```
116             obs2.add(boundary_behind_channel)
117             boundary_above_surface =
                ↪  radiopropa.ObserverSurface(radiopropa.Plane(radiopropa.Vector3d(0,
                ↪  0, 1*radiopropa.meter), radiopropa.Vector3d(0, 0, 1)))
118             obs2.add(boundary_above_surface)
119             sim.add(obs2)
120
121             #create total scanning range from the upper and lower thetas of the
                ↪  bundles
122             step = self._step_zeniths[s] / units.radian
123             theta_scanning_range = np.array([])
124             for iL in range(len(launch_lower)):
125                 new_scanning_range = np.arange(launch_lower[iL],
                    ↪  launch_upper[iL]+step, step)
126                 theta_scanning_range = np.concatenate((theta_scanning_range,
                    ↪  new_scanning_range))
127
128             for theta in theta_scanning_range:
129                 ray_dir = hp.spherical_to_cartesian(theta, phi_direct)
130
131                 def delta(ray_dir,shower_dir):
132                     viewing = np.arccos(np.dot(shower_dir, ray_dir)) *
                        ↪  units.radian
133                     return viewing - cherenkov_angle
134
135                 if (self._shower_axis is None) or
                    ↪  (abs(delta(ray_dir,self._shower_axis)) <
                    ↪  self._cut_viewing_angle):
136                     source = radiopropa.Source()
137
                        ↪  source.add(radiopropa.SourcePosition(radiopropa.Vector3d(*X1)))
138
                        ↪  source.add(radiopropa.SourceDirection(radiopropa.Vector3d(*ray_dir)))
139                     sim.setShowProgress(True)
140                     ray = source.getCandidate()
141                     sim.run(ray, True)
142
143                     current_rays = [ray]
144                     while len(current_rays) > 0:
145                         next_rays = []
146                         for ray in current_rays:
147                             if channel.checkDetection(ray.get()) ==
                                ↪  radiopropa.DETECTED:
148                                 detected_rays.append(ray)
149                                 result = {}
150                                 if n_reflections == 0:
151                                     result['reflection']=0
152                                     result['reflection_case']=1
153                                 elif self._ice_model.get_modules()["bottom
                                    ↪  reflection"].get_times_reflectedoff(ray.get())
                                    ↪  <= n_reflections:
154
                                        ↪  result['reflection']=self._ice_model.get_modules()["b
                                        ↪  reflection"].get_times_reflectedoff(ray.get())
```

```
155
                                        ↪ result['reflection_case']=int(np.ceil(theta/np.deg2ra
156                             results.append(result)
157                         for secondary in ray.secondaries:
158                             next_rays.append(secondary)
159                     current_rays = next_rays
160             #loop over previous rays to find the upper and lower theta of each
                ↪ bundle of rays
161             #uses step, but because step is initialized after this loop this ios
                ↪ the previous step size as intented
162             if len(detected_rays) > 0:
163                 launch_lower.clear()
164                 launch_upper.clear()
165                 launch_theta_prev = None
166                 for iDC,DC in enumerate(detected_rays):
167                     launch_theta = DC.getLaunchVector().getTheta()/radiopropa.rad
168                     if iDC == (len(detected_rays)-1) or iDC == 0:
169                         if iDC == 0:
170                             launch_lower.append(launch_theta-step)
171                         if iDC == (len(detected_rays)-1):
172                             launch_upper.append(launch_theta+step)
173                     elif abs(launch_theta - launch_theta_prev) > 1.1*step: ##take
                        ↪ 1.1 times the step to be sure the next ray is not in the
                        ↪ bundle of the previous one
174                         launch_upper.append(launch_theta_prev+step)
175                         launch_lower.append(launch_theta-step)
176                     else:
177                         pass
178                     launch_theta_prev = launch_theta
179                 isMinimizeAble = MinimizeAble(launch_lower,launch_upper)
180                 #check if we have two distinct launch regions so we can minimize
181                 #accuracy is chosen above time duration, so this will take a
                    ↪ while
182                 if isMinimizeAble and (s == 0): # speed hack: Only check on first
                    ↪ iteration
183                     LetsMinimize = True
184                     break
185
186             else:
187                 #if detected_rays is empty, no solutions where found and the
                    ↪ tracer is terminated
188                 break
189
190         if LetsMinimize:
191             iterative = False
192             ##define module list for simulation, this needs to be redone to get
                ↪ rid of the spherical observer
193             sim = radiopropa.ModuleList()
194             sim.add(radiopropa.PropagationCK(self._ice_model.get_scalar_field(),
                ↪ 1E-8, .001, 1.)) ## add propagation to module list
195             for module in self._ice_model.get_modules().values():
196                 sim.add(module)
197
                ↪ sim.add(radiopropa.MaximumTrajectoryLength(self._max_traj_length*(radiopropa.
```

```
198
199             ## define observer for detection (channel)
200             obs = radiopropa.Observer()
201             obs.setDeactivateOnDetection(True)
202             #a bigger normal value makes the calculation faster, a smaller one
                ↪  more precise
203             #NormalScale = 2.5
204             w = (u / np.linalg.norm(u)) #* NormalScale
205             plane_channel =
                ↪  radiopropa.ObserverSurface(radiopropa.Plane(radiopropa.Vector3d(*X2),
                ↪  radiopropa.Vector3d(*w)))
206             obs.add(plane_channel)
207             sim.add(obs)
208
209             # handy?
210             boundary_above_surface =
                ↪  radiopropa.ObserverSurface(radiopropa.Plane(radiopropa.Vector3d(0,
                ↪  0, 1*radiopropa.meter), radiopropa.Vector3d(0, 0, 1)))
211             obs2.add(boundary_above_surface)
212             sim.add(obs2)
213
214             detected_rays = []
215             detected_theta = []
216             #we minimize the cotangens of the zenith to reflect the same
                ↪  resolution in z to the different angles (vertical vs horizontal)
217
218             for i in range(len(launch_lower)):
219                 ll = launch_lower[i]
220                 lu = launch_upper[i]
221                 cotll = cot(ll)
222                 cotlu = cot(lu)
223                 InitGuess = cot((ll+lu)/2)
224                 if cotlu > cotll:
225                     bounds = scipy.optimize.Bounds(lb=cotll, ub=cotlu,
                        ↪  keep_feasible=False)
226                 else:
227                     bounds = scipy.optimize.Bounds(lb=cotlu, ub=cotll,
                        ↪  keep_feasible=False)
228
229                 root =
                    ↪  optimize.minimize(delta_z_squared,x0=InitGuess,bounds=bounds,options={'xa
230                 theta = arccot(root.x)
231                 detected_theta.append(theta)
232                 detected_rays.append(shoot_ray(theta))
233
234             self._rays = detected_rays
235             self._results =  [{'reflection':0,'reflection_case':1} for ray in
                ↪  detected_rays]
236             launch_bundles = None
237             self.__used_method = 'hybrid minimizer'
238             return launch_bundles,iterative
239
240         else:
241             iterative = True
```

```
242             self._rays = detected_rays
243             self._results = results
244             self.__used_method = 'iterator'
245             launch_bundles = np.transpose([launch_lower, launch_upper])
246             return launch_bundles,iterative
```

BIBLIOGRAPHY

[1] Measurement of the multi-TeV neutrino interaction cross-section with IceCube using earth absorption. Nature, 551(7682):596–600, nov 2017.

[2] M. Aker, A. Beglarian, J. Behrens, A. Berlev, U. Besserer, B. Bieringer, F. Block, S. Bobien, et al. Direct neutrino-mass measurement with sub-electronvolt sensitivity. Nature Physics, 18(2):160–166, 2022.

[3] G A Askaryan. Excess negative charge of an electron-photon shower and the coherent radio emission from it. Zhur. Eksptl'. i Teoret. Fiz., 41, 8 1961.

[4] Jr. Cowan, C. L., F. Reines, F. B. Harrison, H. W. Kruse, and A. D. McGuire. Detection of the Free Neutrino: A Confirmation. Science, 124(3212):103–104, July 1956.

[5] Scott Dodelson. Modern Cosmology. Academic Press, Amsterdam, 2003.

[6] K. Eguchi, S. Enomoto, K. Furuno, J. Goldman, H. Hanada, H. Ikeda, K. Ikeda, K. Inoue, et al. First results from kamland: Evidence for reactor antineutrino disappearance. Physical Review Letters, 90(2), jan 2003.

[7] C. Glaser, D. García-Fernández, A. Nelles, J. Alvarez-Muñiz, S. W. Barwick, D. Z. Besson, B. A. Clark, A. Connolly, C. Deaconu, K. D. de Vries, J. C. Hanson, B. Hokanson-Fasig, R. Lahmann, U. Latif, S. A. Kleinfelder, C. Persichilli, Y. Pan, C. Pfendner, I. Plaisier, D. Seckel, J. Torres, S. Toscano, N. van Eijndhoven, A. Vieregg, C. Welling, T. Winchen, and S. A. Wissel. NuRadioMC: simulating the radio emission of neutrinos from interaction to detector. European Physical Journal C, 80(2):77, January 2020.

[8] C. Glaser, D. García-Fernández, A. Nelles, J. Alvarez-Muñiz, S. W. Barwick, D. Z. Besson, B. A. Clark, A. Connolly, C. Deaconu, K. D. de Vries, J. C. Hanson, B. Hokanson-Fasig, R. Lahmann, U. Latif, S. A. Kleinfelder, C. Persichilli, Y. Pan, C. Pfendner, I. Plaisier, D. Seckel, J. Torres, S. Toscano, N. van Eijndhoven, A. Vieregg, C. Welling, T. Winchen, and S. A. Wissel. NuRadioMC: simulating the radio emission of neutrinos from interaction to detector. The European Physical Journal C, 80(2), jan 2020.

[9] Christian Glaser, Anna Nelles, Ilse Plaisier, Christoph Welling, Steven W. Barwick, Daniel García-Fernández, Geoffrey Gaswint, Robert Lahmann, and Christopher Persichilli. NuRa-

dioReco: a reconstruction framework for radio neutrino detectors. The European Physical Journal C, 79(6), jun 2019.

[10] R. Hiller, P. A. Bezyazeekov, N. M. Budnev, et al. Tunka-rex: energy reconstruction with a single antenna station. EPJ Web of Conferences, 135:01004, 2017.

[11] Tim Huege and Dave Besson. Radio-wave detection of ultra-high-energy neutrinos and cosmic rays. Progress of Theoretical and Experimental Physics, 2017(12), nov 2017.

[12] Tobias Melson, Hans-Thomas Janka, Robert Bollig, Florian Hanke, Andreas Marek, and Bernhard Müller. Neutrino-driven explosion of a 20 solar-mass star in three dimensions enabled by strange-quark contributions to neutrino–nucleon scattering. The Astrophysical Journal Letters, 808(2):L42, jul 2015.

[13] B. Oeyen, I. Plaisier, A. Nelles, C. Glaser, and T. Winchen. Effects of firn ice models on radio neutrino simulations using a RadioPropa ray tracer. In 37th International Cosmic Ray Conference. 12-23 July 2021. Berlin, page 1027, March 2022.

[14] C. Welling, P. Frank, T. Enßlin, and A. Nelles. Reconstructing non-repeating radio pulses with information field theory. Journal of Cosmology and Astroparticle Physics, 2021(04):071, apr 2021.

[15] C. Welling, C. Glaser, and A. Nelles. Reconstructing the cosmic-ray energy from the radio signal measured in one single station. Journal of Cosmology and Astroparticle Physics, 10:075–075, oct 2019.

[16] Tobias Winchen. RadioPropa — a modular raytracer for in-matter radio propagation. EPJ Web of Conferences, 216:03002, 2019.