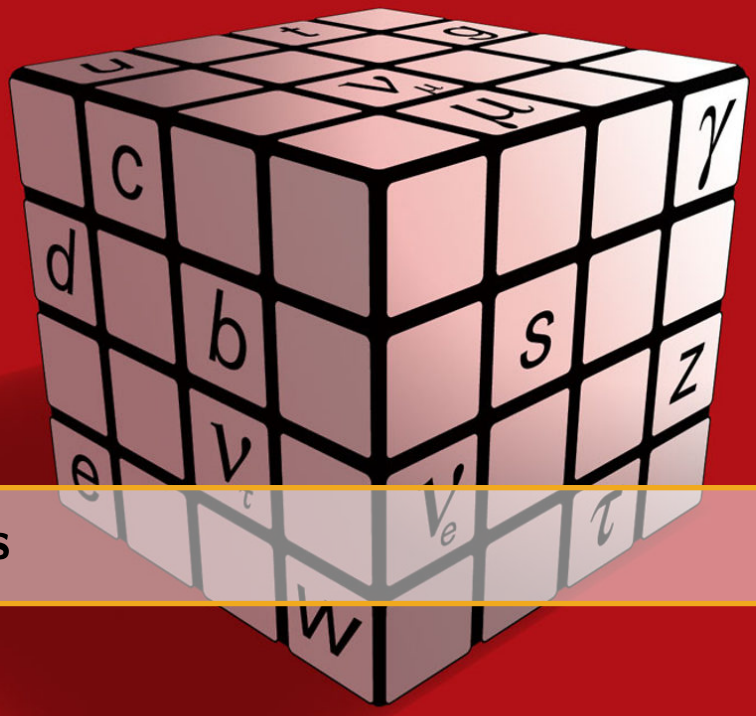


Table of Contents

Introduction and Summary

Paragraphs of Text

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo

velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

I Theory and Experiment

II Simulation, Processing and Analysis

1. Simulation: Event Generation and Propagation

Soon there will be virtual reality, and augmented reality. If you assume any rate of improvement at all, then games will become indistinguishable from reality . . ., it would seem to follow that the odds we are in base reality are one in billions. ∼ *Elon Musk*

In order to be able to search for new physics, one has to have a good handle on the detector response on known physics processes. Depending on the analysis, some processes are more interesting than others. In general, the particle interactions of interest are referred to as *signal events*. Other interactions, which mimic or obscure the signal events, are typically called *background events*. These events are simulated using Monte Carlo* (MC) simulations, where one makes use of a model that describes the interactions and their probability to occur. A typical MC simulation consists of hundreds to millions of events that are constructed using these models with the use of random number generators. To determine the detector response of a particle interaction, one first has to start with the particle generation, which sets the conditions of the initial interaction. Afterwards, the propagation of the particle in the detector (medium) is simulated as best as possible. Below, we give an overview of the important background and signal simulations that are used in this analysis. A flowchart of the simulations steps is given in Fig. 1.1.

Corollary 1.0.1 — The software framework. *IceTray* is a modular framework written and used by the IceCube collaboration and mostly written in C++ for fast computation. A python interface for most modules is provided for fast and easy implementation of the code. The framework is used in both online and offline processing and is stream-based with modules that act on events in the stream and is essentially follows the flowchart that is provided by the user.

To process the large amount of simulation that is required for the collaboration, a data processing and management framework called *IceProd* was developed. The setup is very light-weight, running as a python application. It uses (complex) workflow DAGs (see below)

*While recovering from an illness in 1946, Stanislaw Ulam figured that the actual counting of succesful attempts in playing a card game would yield him a much faster answer to the probability of succes rather than doing the actual calculus. His work, shared with John von Neumann, needed to remain secret and adopted the code word "Monte Carlo", referring to the gambling games in the Monte Carlo Casino in Monaco.

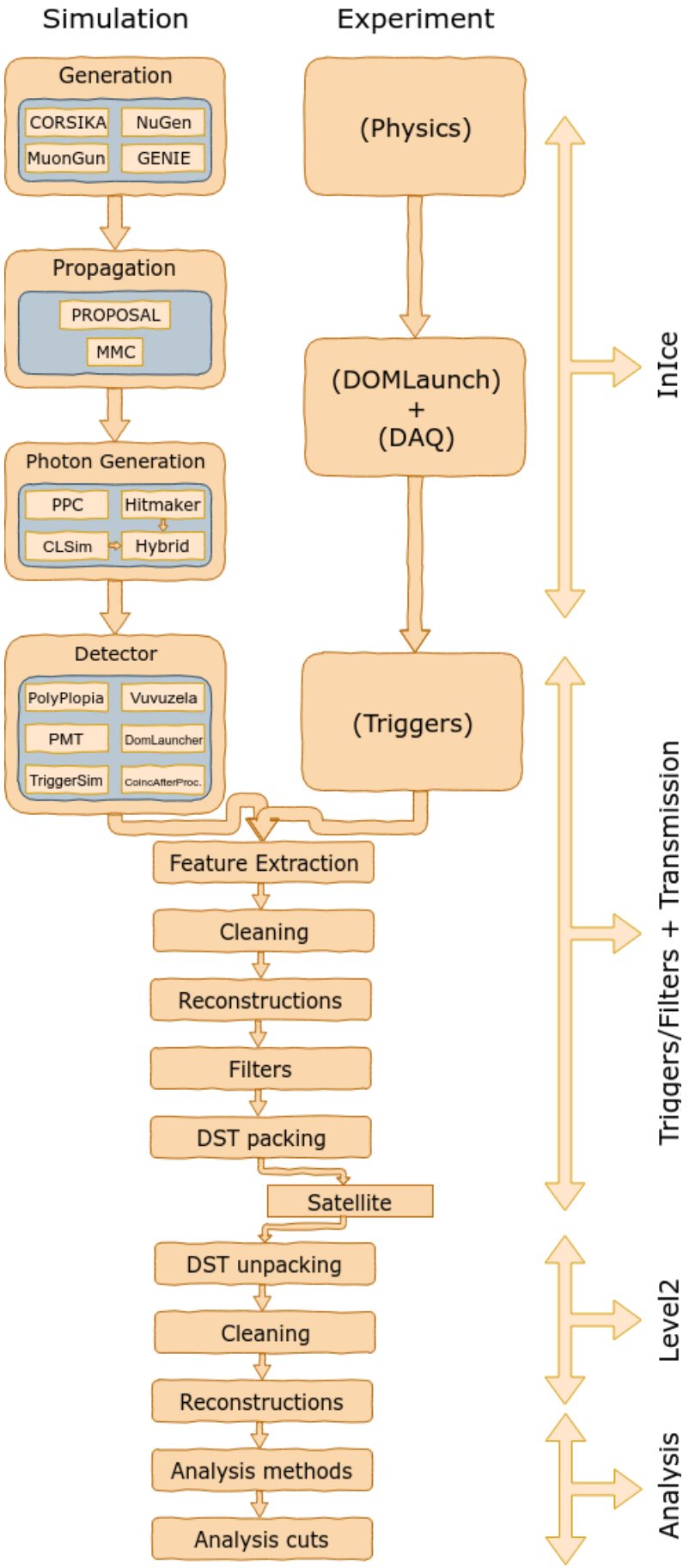


Figure 1.1: Flowchart of the simulation layout. On the left it is shown that particles are injected and their interactions are simulated to digitized waveforms. The right part shows real data processing. After triggering, data and simulation go through the same processing chain to prepare for analysis.

across distributed computing grids in order to optimize usage of resources. A *dataset* is set up by running hunders to thousands of jobs in parallel over multiple computing resources all over the world. Each dataset has specific input parameters that are fixed. Distributions in physical parameters such as the direction, energy, position, etc. of the particle(s) are provided by random number generators [1].

HTCondor is an open source computing software that provides a job queueing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel jobs to HTCondor and places them into a queue. It chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion.

DAGMans (Directed Acyclic Graph Managers) are meta-schedulers for the execution of computations. They submit the programs to HTCondor in an order that is represented by a DAG and processes the results. DAGMans are often used by analyzers for bulk computations on large amounts of data.

1.1 Generation

Simulations start with setting up the starting conditions of the physical processes one wants to simulate. For example, a shower event by itself is not well defined. The type of primary particle (H, He, Fe,...), the energy, the inclination and so on will all define the properties of the full air shower that will be produced. Multiple different generators used in the IceCube collaboration serve other purposes; some are explained in more detail below.

1.1.1 Background simulation

1.1.1.1 CORSIKA

A free, publicly available software framework that is widely used in the astrophysics community for the simulation of cosmic ray interactions is called CORSIKA (COsmic Ray SImulations for Kascade). It was originally developed for the KASCADE experiment and now used by most people and collaborations to simulate air shower events. IceCube analyses, such as this one, use CORSIKA simulations to simulate the muonic component that is able to reach the in-ice detector.

A particle of specific type, energy, direction and position is injected in the top of the atmosphere and propagated. The distribution of particles in the shower is saved and read out according to a certain altitude. Because the flux of cosmic rays is exceedingly small at the highest energies, too many resources and too much time would be required to simulate an energy distribution as measured in experiments. Therefore, one often simulates a much harder spectrum and reweights the events accordingly later on (see Section D.4). Simulation datasets are often subdivided into a low-energy and high-energy dataset. The former ranges from primary energies between 600 GeV to 100 TeV and uses a spectral index that is close to what is measured. The spectral index of the latter is smaller, resulting in a harder spectrum, and the primary energy ranges from 100 TeV to 100 EeV. The lower limit of the energy range is due to the limited penetration depth of muons through the ice. An overview is given in Table 1.2.

The spectrum used for this analysis, after reweighting, follows the following energy distribution:

$$\Phi_i\left(E_{\text{prim}}\right) = \sum_{j=1}^{3} a_{i,j} E^{-\gamma_{i,j}} \cdot \exp\left[-\frac{E}{Z_i R_{c,j}}\right]. \tag{1.1}$$

where we sum over the three populations that are mentioned in Section **??**, γ is the spectral index, Z the particle atomic number and $a_{i,j}$ the normalization constants for primary i in population j. The 5 groups that are assumed to contribute significantly to the flux are: p, He, CNO, Mg-Si

Table 1.1: Best fit for parameters in Eq. 1.1. Numbers taken from Ref. [2].

j	R_c [V]	γ					$a_{i,j}$				
		p	He	CNO	Mg-Si	Fe	p	He	CNO	Mg-Si	Fe
1	$4 \cdot 10^{15}$	1.66	1.58	1.63	1.67	1.63	7860	3550	2200	1430	2120
2	$30 \cdot 10^{15}$	1.4					20		13.4		
3	$2 \cdot 10^{18}$	1.4					1.7		1.14		

and Fe. This is the convention that is used in Ref. [2]. Table 1.1 shows the best fits for the normalization constants to describe the data.

Interactions

The atmosphere composition is always set at 78.1% N_2, 21% O_2, and 0.9% Ar, which is a good description of reality on average. However, the density of the air above the detector changes significantly during the year because of temperature differences in the Arctic Summer and Winter. Most analyses that treat the muonic component as a background and are not interested in the details of the showers and how it changes during the year use an average of the atmospheric density.

The shower propagation and composition depends on the models that are used to simulate these high-energy interactions. The lowest energies are simulated with FLUKA (FLUktuierende KAskade) [3]. This model covers the energy range that can be compared with accelerator experiments. Which model is the best for the highest energies is not known at the time of writing, if there even is one that describes nature well enough, since there are no controlled laboratory measurements that are capable of reaching these energies. Several studies seem to indicate that the composition changes drastically at the highest energies SAMCITEREN+andere. Fortunately, this is of no importance for this analysis.

CORSIKA is written in FORTRAN 77, but a C++ version is currently in the making [4].

1.1.1.2 NuGen

The neutrino-generator is a neutrino event generator program that works with the IceTray framework. With this module, one can inject a primary neutrino on the surface of the Earth by specifying a few parameters in the steering file.

The physics implemented in this program is based on the ANIS-All Neutrino Interaction Generator [5]. However, the cross sections have been updated and the structure of code has been changed significantly from ANIS to incorporate it in the IceTray framework.

The generator requires the first interaction to be near the detector and

- prepares a primary neutrino and injects it to the Earth,
- propagates the neutrino and work out interactions inside the Earth* (when they occur),
- makes a forced interaction inside the detection volume† (only if any neutrino reaches the detector site),
- stores injected neutrinos and all generated secondaries,
- stores interaction weight information.

The generator also does not distinguish between neutrino and antineutrino and assumes a ratio of (1:1).

The spectrum used for this analysis, after reweighting, follows the Honda2006 spectrum [6] for

*Possible interactions are CC, NC, Glashow resonance for $\bar{\nu}_e$ and tau decay for $\overset{(-)}{\nu}_\tau$. CC interactions produces no new neutrinos and the simulation stops at the vertex point. The other interactions create new secondary neutrinos.

† In most cases, a neutrino will not interact within the medium, but for computational reasons at least one neutrino is forced to interact and the simulation is reweighted afterwards accordingly.

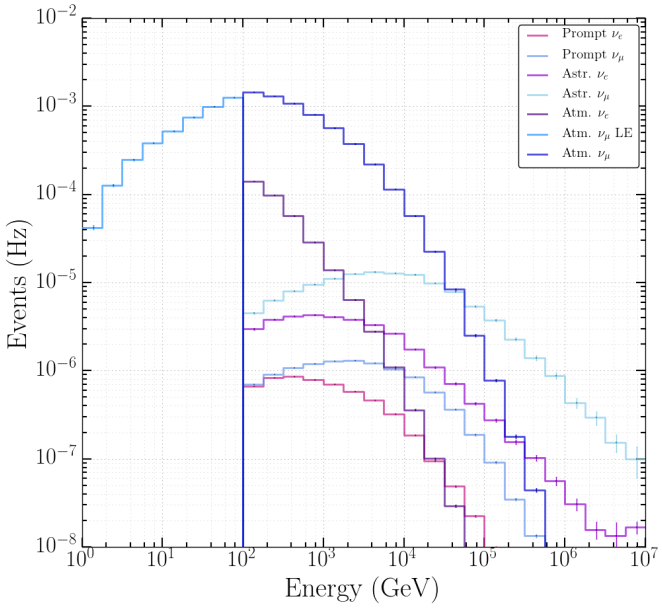


Figure 1.2: Distribution of weighted neutrino fluxes that were used for this analysis. The atmospheric ν_μ and ν_e fluxes were derived from Ref. [6], prompt from Ref. [7], and astrophysical from Ref. [8].

atmospheric neutrinos, SarcevicStd for the prompt component [7], and astrophysical from [8] (see Section **??** for more information on these fluxes). The astrophysical flux measured by the IceCube collaboration follows the following energy spectrum

$$E^2\,(\Phi) = 1.5 \cdot 10^{-8} \left(\frac{E}{100\ \text{TeV}}\right)^{-0.3} \text{GeV cm}^{-2}\text{s}^{-1}\text{sr}^{-1}. \tag{1.2}$$

The distribution for these different components can be seen in Fig. 1.2.

1.1.1.3 GENIE

To include the lowest energies, which are not accounted for by ANIS/NuGen, the GENIE (Generates Events for Neutrino Interaction Experiments) neutrino generator was implemented in IceTray. It is a well established generator, used by collaborations worldwide and written in C++ [9, 10].

The spectrum used for this analysis, after reweighting, follows the Honda2015 spectrum [11] for low-energy atmospheric neutrinos.

1.1.2 Signal simulation

As mentioned in Section **??**, the signal flux is assumed to be isotropic close to the detector. The SMP starting points are randomly placed on a disk with a direction perpendicular to it as shown in Fig. 1.3. The disk has a radius of 800 m and is located at a distance of 1000 m from the detector center. The disk itself is randomly rotated around the detector center to simulate an isotropic flux. The distribution of the azimuth, ϕ and cosine of the zenith*, $\cos(\theta)$, is shown in Fig. 1.4.

Because slow moving particles would require specialized treatment, the minimal velocity of the particles is set as $\beta > 0.95$ and simulated with am E^{-1} spectrum. The spectrum is later normalized to a flux of 10^{-14} GeV cm^{-2} s^{-1} sr^{-1} with an E^{-2} spectrum. The absolute flux is

*See Appendix D.3 why we show the cosine of the zenith.

Table 1.2: Overview of the datasets used in this analysis. GaisserH3a from Ref. [2], Honda2015 from Ref. [11], Honda2006 from Ref. [6], Sarcevic from Ref. [7], and astrophysical from Ref. [8].

Generator	Type	Range [GeV]	Simulated γ	Weighted γ	Ice	Dataset
CORS.	5-comp.	$10^5 - 10^{11}$	2	GaisserH3a	SpiceLea	11937
CORS.	5-comp.	$600 - 10^5$	2.6	GaisserH3a	SpiceLea	11499
CORS.	5-comp.	$600 - 10^5$	2.6	GaisserH3a	SpiceLea	11808
CORS.	5-comp.	$600 - 10^5$	2.6	GaisserH3a	SpiceLea	11865
CORS.	5-comp.	$600 - 10^5$	2.6	GaisserH3a	SpiceLea	11905
CORS.	5-comp.	$600 - 10^5$	2.6	GaisserH3a	SpiceLea	11926
CORS.	5-comp.	$600 - 10^5$	2.6	GaisserH3a	SpiceLea	11943
CORS.	5-comp.	$600 - 10^5$	2.6	GaisserH3a	SpiceLea	12161
CORS.	5-comp.	$600 - 10^5$	2.6	GaisserH3a	SpiceLea	12268
GENIE	ν_μ	$0.5 - 100$	1	Honda2015	SpiceMie	12475
NuGen	ν_μ	$100 - 10^8$	2	atmos.: Honda2006 prompt: Sarcevic astro.: Astro.	SpiceLea	11029
NuGen	ν_μ	$100 - 10^8$	2	atmos.: Honda2006 prompt: Sarcevic astro.: Astro.	SpiceLea	12346
NuGen	ν_μ	$100 - 10^8$	2	atmos.: Honda2006 prompt: Sarcevic astro.: Astro.	SpiceLea	11883
NuGen	ν_e	$100 - 10^8$	2	atmos.: Honda2006 prompt: Sarcevic astro.: Astro.	SpiceLea	12034
NuGen	ν_e	$100 - 10^8$	2	atmos.: Honda2006 prompt: Sarcevic astro.: Astro.	SpiceLea	12646

Table 1.3: Overview of the datasets used for systematic uncertainties. Polyg(onato) follows from Ref. [12], GaisserH4a from Ref. [13] and Bartol from Ref. [14].

Generator	Type	Range [GeV]	Sim. γ	Weighted γ	Ice	Dataset	Syst. Eff.
CORS.	Hoerandel	$600 - 10^5$	Polyg.	GaisserH3a	SpiceLea	11527	DOM eff. -10%
CORS.	Hoerandel	$600 - 10^{11}$	Polyg.	GaisserH3a	SpiceLea	11526	DOM eff. +10%
CORS.	5-comp.	$600 - 10^{11}$	2.6	GaisserH3a	SpiceLea	12388	Abs. +10%
							Scat. +10%
							Abs./Scat. -7.1%
CORS.	All datasets from Table 1.2			GaisserH4a	SpiceLea	Table 1.2	GaisserH4a
NuGen	ν_μ	$100 - 10^8$	2	Bartol (syst.)	SpiceLea	11029	Bartol flux
NuGen	ν_μ	$100 - 10^7$	2	Honda2006 + Bartol (syst.)	SpiceLea	11883	DOM eff. +10% DOM eff. -10% Abs. +10% Scat. +10% Abs./Scat. -7.1% Bartol flux
NuGen	ν_μ	$100 - 10^8$	2	Honda2006 + Bartol (syst.)	SpiceLea	12346	DOM eff. +10% DOM eff. -10% Abs./Scat. -7.1% Bartol flux

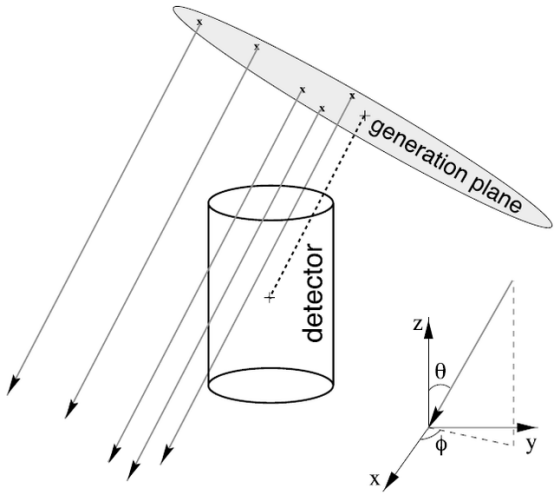


Figure 1.3: Illustration of how the particle injection works. The particle is first randomly positioned on a disk following a uniform distribution. The disk is then randomly rotated to simulate an isotropic flux.

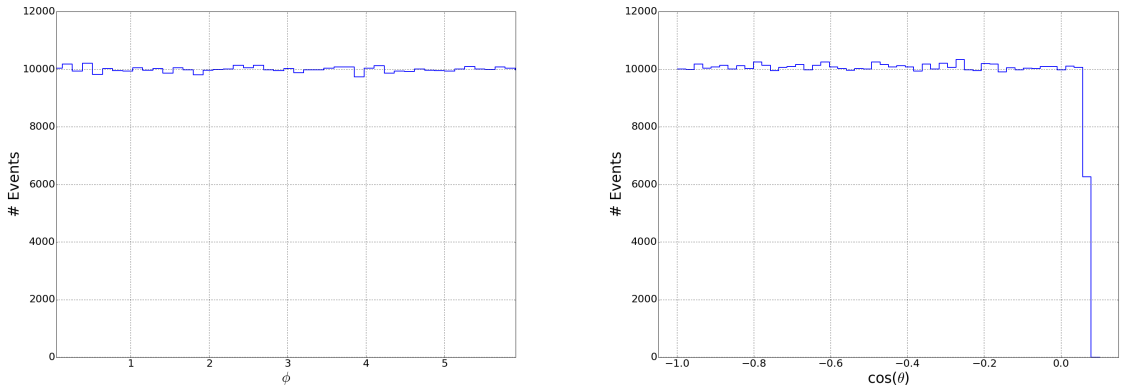


Figure 1.4: Illustration of uniform distributions of azimuth and cosine of the zenith for the particle injection in agreement with an isotropic flux (see Appendix D.1).

.

only necessary for illustrative purposes, see Section refer to this section when it's written!.

Similar to the background, SpiceLea was used as the nominal ice model.

1.2 Propagation

After generation, the particles need to be propagated through the medium. The particles will interact, lose energy, produce new particles, and generate light. The particle interactions and light production are done in two different modules as photon simulation is done with GPU. The former module is called `PROPOSAL`, the latter `ppc`.

1.2.1 `PROPOSAL`

Using the cross sections of the important interaction, together with the properties of the traversing medium and the particles (mass, charge, spin, decay time, etc.) it is possible to simulate the energy losses, secondary production and the consequent interaction of these daughter particles. This is done in the software package `PROPOSAL` (the Propagator with Optimal Precision and Optimized Speed for All Leptons), fully written in C++. It was based on the former program `MMC` (Muon Monte Carlo), which was written in Java. In 2018, a substantial improved version of

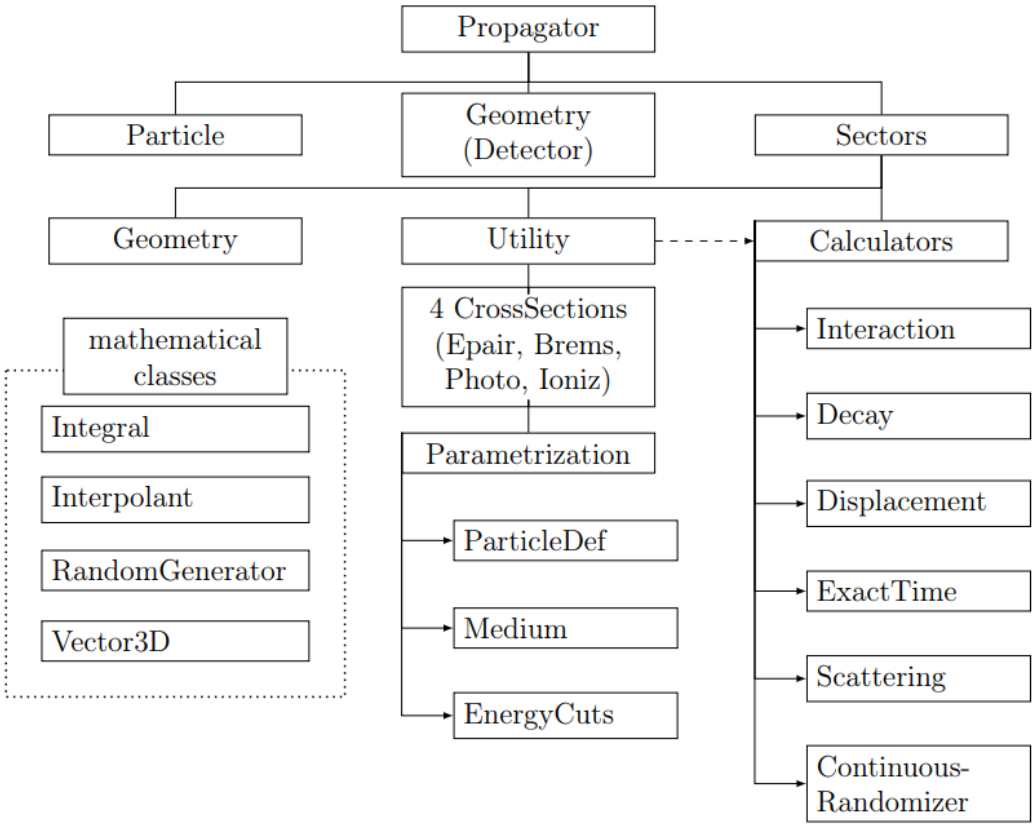


Figure 1.5: Overview of the class structure in PROPOSAL, from Ref. [15].

PROPOSAL was finalized. An illustration of the workings of the code is given in Fig. 1.5 and an in depth documentation is given in Ref. [15].

PROPOSAL for SMPs

Since we assume the SMPs to behave leptonically, it was chosen to use PROPOSAL for the signal propagation as well. The mass and charge of the particle are set in the input parameters and the cross section dependence on these parameters can be seen in Section **??**. In general, there is a small dependence on the mass and a squared dependency on the charge (Eq. ???), except for bremsstrahlung that has a quadratic charge dependency. These effects only become prominent and important for highly relativistic particles, which as will be seen in Section verwijs hiernaar als je dit geschreven hebt, do not have a dominating contribution to the total signal.

The PROPOSAL module keeps track of all the particles that are produced during propagation and the accompanying energy losses in a tree-like structure (called an *I3MCTree*). This collection of particles and their interactions are forwarded to a light production computation module.

1.2.2 Photoelectron generators

In Section **??** we already explained how the ice is simulated in the IceCube detector. The parameters $b_e(400)$ and $a_{dust}(400)$ define the photon propagation through the ice and determine if they are absorbed or hit a DOM. To optimize computing time, the DOMs were scaled (mostly with a factor of 5) to force more photon interactions. The number of photons emitted was then appropriately scaled down with the square of this scaling factor*. The DOM acceptance curves, as shown in Fig. **??**, together with the Frank-Tamm formula (Eq. **??**) allow to calculate the expected number of photons produced per unit length:

*The surface of a sphere scales with the square of the radius.

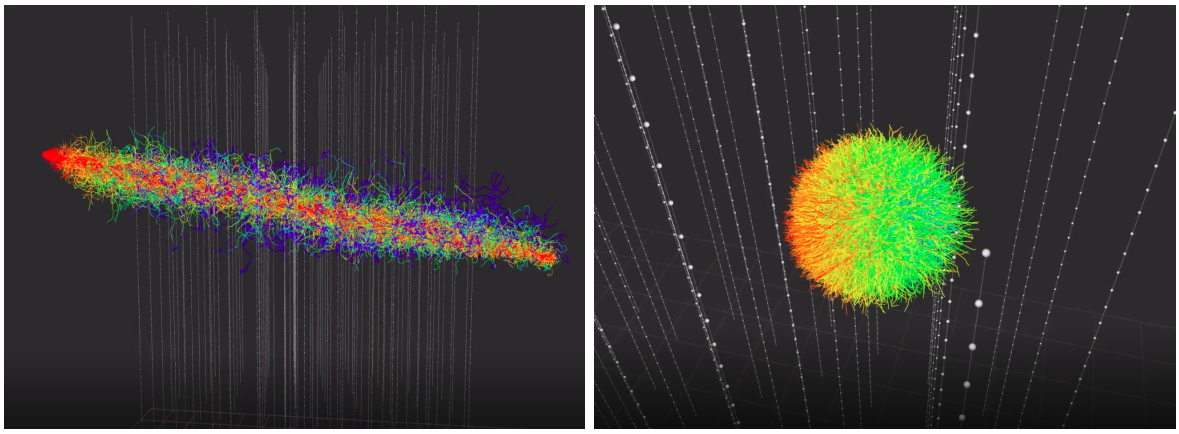


Figure 1.6: *Left:* simulation of a track event in IceCube. Each line represents a photon path and colors indicate how far they have traveled from their generation point. *Right:* simulation of a cascade event in IceCube.

$$\frac{dN}{dx} = \int_{\lambda_1}^{\lambda_2} \frac{2\pi\alpha}{\lambda^2} \sin^2(\theta_c)\, d\lambda = 2\pi\alpha \sin^2(\theta_c) \left(\frac{1}{\lambda_1} - \frac{1}{\lambda_2}\right). \tag{1.3}$$

From this formula, we find that the expected rate of a Cherenkov emission profile is equal to ≈ 350 photons/cm. Together with the DOM acceptance, which has an overall average of around 7%, the expected *seen* number of photons per meter is equal to 2450 m^{-1}.

`PPC` is a Photon Propagation Code, written in C++ and runs on graphic processing units (GPUs). This allows the code to run up to a hunderd times faster than in a CPU-only environment. `PPC` employs both CUDA (NVIDIA GPU only) and OpenCL programming interface (both NVIDIA and AMD GPUs) together with multiple CPU environments. GPU environments allow the tracking of thousands of photons simultaneously, vastly improving the computational speed. For more information, see Ref. [16].

Previous photon propagation code, such as `Photonics` [17], produced 6-dimensional photon tables (3 spatial, 2 directional and 1 temporal). This meant that at least one set of tables had to be produced per particle type and per velocity and interpolation methods had to be used, with the accompanying inaccuracies. These tables also required significant disk space and the method was therefore replaced with the GPU-codes. Direct photon simulation also allows for other non-trivial implementations such as the tilting of ice layers.

Another photon propagation code is called `CLSim`, which uses GEANT4 to propagate particles. A hybrid version called `HybridCLsim` is sometimes used. Muons are propagated using `PROPOSAL`/`MMC` and their stochastic losses (which are showers) are simulated from tables whereas the "bare muons" (with their stochastics) are simulated using direct propagation. This avoids time loss for the rare but very computational high-energy cascade events.

An illustration of photon propagation in the IceCube detector for both a cascade and track simulation is shown in Fig. 1.6.

1.3 Detector simulation

Further processing of the simulations involve:

- `Polyplopia`: a project dedicated to merge multiple events to account for coincident events (that are simulated independently). An estimated 15% of CORSIKA events result in coincident events and make up de bulk of bad reconstructions where down-going muons are simulated as up-going (example see Fig. 2.4);

- `Vuvuzela`: the PMT noise is simulated as having an exponential component from thermal and radioactive decays, and a log-normal contribution for scintillation;
- `PMT`: the time from the first photon entering the PMT to the readout after passing along multiple dynodes has an uncertainty, referred to as "PMT jitter" . The amplification of photoelectrons by the PMT is also not constant and is simulated in this module. Additionally, the module accounts for prepulses, late pulses, afterpulses and saturation of the PMT. More information can be found in Refs. [18, 19].
- `DOMLauncher`: the digitization of the PMT pulses and other behavior of the DOM mainboard (as explained in Section **??**) is done in this module. The three main features of the DOM that are simulated to generate launches are the discriminator, LC, and digitization.
- `trigger-sim`: simulation of the trigger behavior as explained in Sec. **??**.

1.4 Burn sample

Getting the intricate details of physical events in non-trivial environments just right is not an easy task. In many steps of the way, simulations use fits and estimations. Some simulation datasets are reasonable to compare to the data, depending on the phase space one is looking at, while other datasets need other specifications. For example, analyses dedicated to measuring the cosmic ray interactions need much more fine-tuning in their models for the atmosphere, composition, interaction models, etc. than an analysis dedicated to search for muon tracks that first propagated through the Earth and have atmospheric muons as a background.

It is for this reason, most analyses select a certain subset of the data they want to analyze to compare to the Monte Carlo simulations. For this analysis, 10% of the total data, called the *burn sample* , was used to compare data to Monte Carlo. As indicated in Section **??**, the data is saved in 8-hour runs and the burn sample consists of every run ending with a '0'. The burn sample also allows to estimate the robustness of certain reconstructions and variables regarding differences in data and simulation.

More info on the burnsample see Section???

1.4.1 Standard ??

Processing ergens: online L0, L1, L2, uw filters,... Beter

1.4.2 Event viewer

After a full simulation, it is possible to visualize the event in an event viewer called *Steamshovel* . Typical events in the IceCube detector are shown with this interface and are loaded from *i3files* that contain information about the detector geometry and the full event (DOM positions and calibrations, detector hits, timestamps, trigger hits, etc.). Simulated events also contain the true values of the particles and can be compared with reconstructed variables. Event viewers allow for first guesses in how background events are able to be separated from signal, although both can have wide varieties in possible outputs.

An example is given in Fig. ??? wat is goed?.

2. Reconstruction, Cleaning and Analysis Techniques

Shall I refuse my dinner because I do not fully understand the process of digestion? ~ Oliver Heaviside

Because the in-ice IceCube detecor is a sparsely distributed detector, it is not straightforward to unambiguously reconstruct the particle (interactions). The scattering and absorption of photons, tilt of ice sheets, bubble column, etc. lead to uncertainties and make reconstruction challenging. Over the years, multiple reconstruction methods have been developed in the collaboration. They range from very fast (and simple) reconstructions, necessary for online filtering, to slow (and more refined) ones. Multiple reconstruction algorithms have been used in this analysis and are explained in more detail in this chapter.

2.1 Reconstruction

2.1.1 Likelihood

Reconstruction algorithms usually have no unique solutions to describe the set of measured values of an event. The likelihood $\mathcal{L}(\vec{x}|\vec{a})$ describes the probability of a set of parameters $\vec{a}$ to be expressed in a set of experimentally measured values $\vec{x}$. The parameters, $\vec{a}$, typically define the particle's characteristics (energy, direction, position, type, etc.) while the measured values $\vec{x}$ are determined from the detector response (number of PE, timing, position of hit DOMs, etc.). This likelihood is equal to the cumulative probability

$$\mathcal{L}(\vec{x}|\vec{a}) = \prod_i p(x_i|\vec{a}), \tag{2.1}$$

where $p(x, \vec{a})$ is the probability that we measure a certain value x from a set of independent values $\vec{x}$ given an initial set of parameters $\vec{a}$. The best possible guess for the unknown parameters $\vec{a}$ is the most likely set that will result into the experimental values. This is done by maximizing the likelihood $\mathcal{L}$. The reconstruction algorithms below rely on following parameters that assume a single, long track

$$\vec{a} = (\vec{r_0}, t_0, \vec{p}, E_0), \tag{2.2}$$

where $\vec{r_0}$ is the position vector of the particle at a time t_0 with a direction $\vec{p}$ and initial energy E_0.

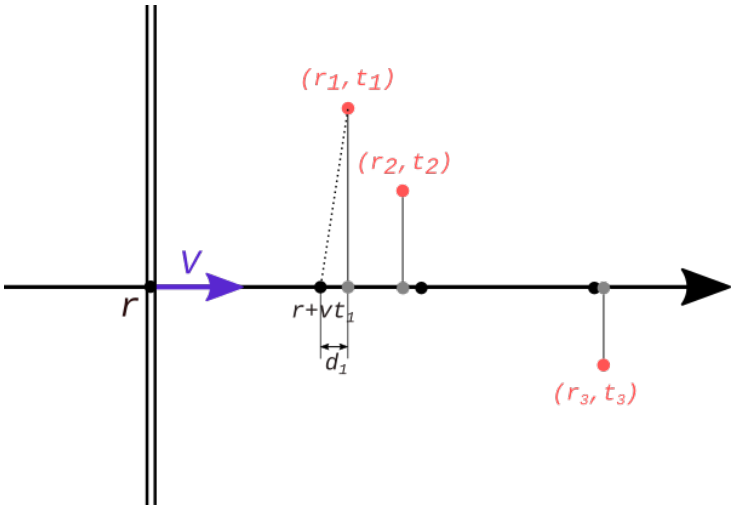


Figure 2.1: Figure illustrating how LF works. The position, $\vec{r}$, and velocity, $\vec{v}_{\text{part}}$ minimizing the distance of the DOMs to the track is calculated. The dotted line is one distance that is minimized in Eq. 2.4.

2.1.2 `Line-Fit`

One of the most simple approaches in constructing a parameter profile is by calculating the track that, overall, has the closest approach of all the hit optical modules and is called `Line-Fit` (LF) [20]. If we assume that a particle starts at a position, $\vec{r}$ at a time 0 and travels at a velocity of $\vec{v}_{\text{part}}$, then its position at any given time is

$$\vec{r'} = \vec{r} + \vec{v}_{\text{part}} t. \tag{2.3}$$

We want to calculate the best possible estimate of the velocity $\vec{v}_{\text{part}}$ and the position $\vec{r}$. Each DOM has a known location, $\vec{r}_i$, and measured time of a pulse, t_i. In this algorithm one assumes that a wavefront perpendicular to the particle's direction is traveling along with the particle. If the velocity $\vec{v}_{\text{part}}$ is fixed, then the position of the particle is known (black points in Fig. 2.1). However, because the Cherenkov wavefront should be set at an angle and scattering, PMT jitter, noise, etc. are not taken into account, this will not agree with the DOM position projected along the particle path (grey dots). The unknown velocity $\vec{v}_{\text{part}}$ and position $\vec{r}$ are the analytical solutions after minimizing the distances d_i as shown in the figure*

$$\begin{aligned} S(\vec{r}, \vec{v}_{\text{part}}) &= \sum_{i=1}^{N_{\text{hit}}} \rho(\vec{r}, \vec{v}_{\text{part}}, \vec{r}_i, t_i)^2 \\ &\equiv \sum_{i=1}^{N_{\text{hit}}} \left(\vec{r}_i - \vec{r} - \vec{v}_{\text{part}} t_i\right)^2, \end{aligned} \tag{2.4}$$

where N_{hit} are the number of hits. The analytical solution by minimizing this equation is equal to

$$\vec{r} = \langle \vec{r}_i \rangle - \vec{v}_{\text{part}} \langle t_i \rangle \quad \text{and} \quad \vec{v}_{\text{part}} = \frac{\langle \vec{r}_i t_i \rangle - \langle \vec{r}_i \rangle \langle t_i \rangle}{\langle t_i^2 \rangle - \langle t_i \rangle^2}, \tag{2.5}$$

where $\langle x \rangle$ denotes the average of a parameter x over all hits i.

Because this is an analytical equation, this algorithm is very fast and therefore often used in online processing.

*Minimizing $r_i - r'$ (dotted line in Fig. 2.1) is the same as minimizing d.

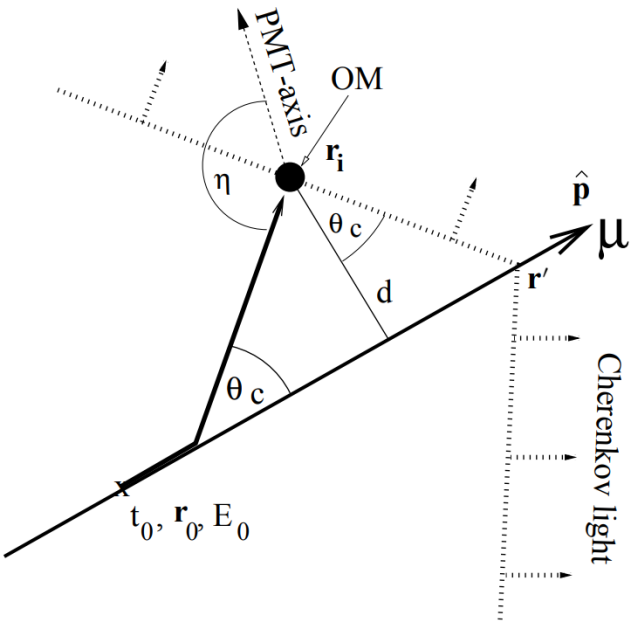


Figure 2.2: Figure illustrating a muon track passing close by an optical module and defining the parameters used in the reconstruction algorithms. Illustration from Ref. [20].

2.1.2.1 `Improved Line-Fit`

While disregarding the Cherenkov profile is inherent to the simplified LF model chosen for computational reasons, removing hits generated by photons that scattered for a significant length of time will mitigate the effect of ignoring the photon scattering in the ice. It was found that a basic filter could identify these scattered hits, and improve accuracy by almost a factor of two by removing them from the dataset. More formally, for each hit h_i, the algorithm looks at all neighboring hits within a neighborhood of μ, and if there exists a neighboring hit h_j with a time stamp that is t earlier than h_i, then h_i is considered a scattered hit, and is not used in the basic reconstruction algorithm. Optimal values of μ and t were found to be 156 m and 778 ns by tuning them on simulated muon data [21].

This "delay cleaning" is done by computing the Huberfit on the remaining data points and minimizing

$$\sum_{i=1}^{N_{\text{hit}}} \phi(\rho(\vec{r}, \vec{v}_{\text{part}}, \vec{r}_i, t_i)), \tag{2.6}$$

where ρ is defined in Eq. 2.4 and the Huber penalty function ϕ is defined as

$$\phi(\rho) \equiv \begin{cases} \rho^2 & \text{if } \rho < \mu \\ \mu\,(2\rho - \mu) & \text{if } \rho \leq \mu \end{cases} . \tag{2.7}$$

Because of the overal performance increase of this method, all LF computations were done with the improved version (although still often referred to as "Line-Fit").

2.1.3 SPE and MPE

A more intricate method of track reconstruction is done by taking the geometrical shape of the Cherenkov cone into account and relying on simulation fits where a seed track is implemented (usually from the fast `Line-Fit` algorithm).

Let us assume a particle is traveling close to a DOM with parameters defined in Eq. 2.2 as illustrated in Fig. 2.2. The minimal distance of the track to the DOM is equal to d and the PMT-axis (downwards relative to DOM) has an angle offset of η degrees of the Cherenkov wave

direction. In perfect conditions, the *time residual* (time between the observed hit time and the "expected" time) is a delta function centered around 0, where

$$t_{\text{res}} \equiv t_{\text{hit}} - t_{\text{geo}}, \tag{2.8}$$

with

$$t_{\text{geo}} = t_0 + \frac{\vec{p} \cdot (\vec{r}_i - \vec{r}_0) + d \cdot \tan(\theta_c)}{c_{\text{vac}}}, \tag{2.9}$$

which is equal to the time of the particle to travel from the position $\vec{r}_0$ to $\vec{r}'$ as illustrated in the figure. The accompanying Cherenkov wavefront that sent out photons at a time t_0 from $\vec{r}_0$ will cross the DOM when the particle is at a position $\vec{r}'$. Due to noise effects, PMT jitter, light from secondary interactions, DOM orientation, etc. the time risidual is smeared and shifted. The p.d.f. was estimated with photon simulations in ice and fitted to a Pandel function [20]. The time likelihood profile for single photons i at the locations of the hit DOMs is then

$$\mathcal{L}_{\text{time}} = \sum_{i=1}^{N_{\text{hit}}} p_1(t_{\text{res}} | \vec{a} = d_i, \eta_i, ...). \tag{2.10}$$

An initial particle position and direction are found by maximizing the likelihood and iterated a couple of times to find the global maximum instead of a local. This fitting is called the Single PhotoElectron (SPE) fit.

The description of single photons arriving at the optical modules cannot be correct since electrical and optical signal channels can only resolve multiple photons separated by a few 100 ns and $\approx$ 10 ns, respectively. In the Multi-PhotoElectron (MPE) fit, one accounts for the fact that the early photons in a DOM hit scattered less in the ice. The p.d.f. for the first photon out of a total of N to arrive with a time residual of t_{res} is

$$p_N^1(t_{\text{res}}) = N \cdot p_1(t_{\text{res}}) \cdot \left(\int_{t_{\text{res}}}^{\infty} p_1(t) dt \right)^{(N-1)} = N \cdot p_1(t_{\text{res}}) \cdot (1 - P_1(t_{\text{res}}))^{(N-1)}, \tag{2.11}$$

where P_1 is the cumulative distribution of the single photon p.d.f..

2.1.4 `Millipede`

To have a better handle on the particle energy and cascades along the track, the module `Millipede` was developed. The number of photons seen at each optical module depends on multiple factors (that were mentioned throughout this text, such as the ice characteristics, timing, etc. In this module, the expected number of photons is said to depend on the energy that was deposited and a *light yield factor* that depends on the DOM position and the location of emission along the track

$$\begin{aligned} N_{\text{exp},k} &= \rho_k + \sum_{i=1}^{n} \Lambda(\vec{r}_k, \vec{r}_i') E_i \\ &= \rho_k + \vec{\Lambda}(\vec{r}_k, \vec{r}_i') \cdot \vec{E}, \end{aligned} \tag{2.12}$$

where k refers to a certain DOM and i refers to a certain energy deposit such as illustrated in Fig. 2.3.

The likelihood is assumed to follow a Poisson distribution with λ, the expected amount of photons, equal to $N_{\text{exp},k}$

$$\mathcal{L}_k = \frac{\left(\vec{\Lambda} \cdot \vec{E} + \rho_k \right)^N_{\text{seen},k}}{N_{\text{seen},k}!} e^{-\vec{\Lambda} \cdot \vec{E} + \rho_k}. \tag{2.13}$$

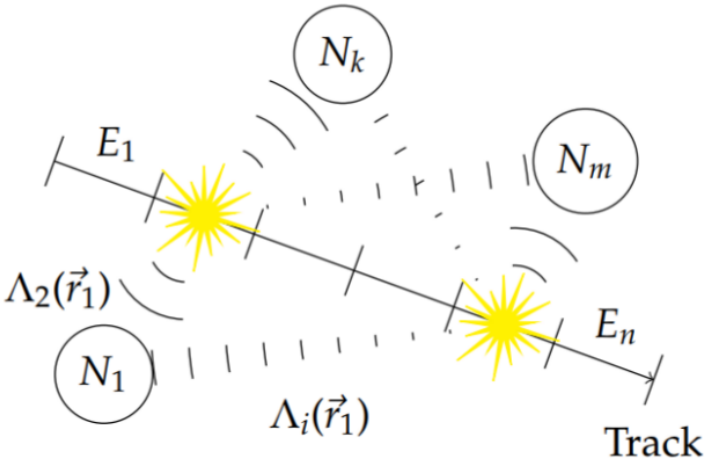


Figure 2.3: Illustration of the working principles of the `Millipede` toolkit. A track is subdivided into segments that each deposit a certain energy, E_i. Different segments can contribute to the total number photons seen per DOM, N_k.

For easier, faster and more accurate computation the logarithm of the likelihood is used

$$\begin{aligned}\ln \mathcal{L}_k &= N_{\text{seen},k} \ln \left(\rho_k + \sum_{i=1}^{n} \Lambda(\vec{r}_k, \vec{r}_i') E_i \right) - \ln\left(N_{\text{seen},k}!\right) - \sum_{i=1}^{n} \Lambda(\vec{r}_k, \vec{r}_i') E_i - \rho_k \\ &= N_{\text{seen},k} \ln \left(\rho_k + \vec{\Lambda}(\vec{r}_k)\vec{E} \right) - \vec{\Lambda}(\vec{r}_k)\vec{E} - \rho_k - \ln\left(N_{\text{seen},k}!\right)\end{aligned} \tag{2.14}$$

Maximizing the total likelihood (summing over all m DOMs) with respect to the energy gives

$$\nabla_{\vec{E}} \ln \mathcal{L} = \nabla_{\vec{E}} \sum_{k=1}^{m} \ln \mathcal{L}_k = \sum_{k=1}^{m} \left(\frac{N_{\text{seen},k} \vec{\Lambda}(\vec{r})}{\vec{\Lambda}(\vec{r}_k) \cdot \vec{E} + \rho_k} - \vec{\Lambda}(\vec{r}_k) \right) = 0. \tag{2.15}$$

This equation holds if all terms in the sum vanish, i.e. if for all DOMs holds that

$$\begin{aligned}N_{\text{seen},k} &\overset{\text{Eq.2.12}}{=} N_{\text{exp},k} \\ &= \vec{\Lambda}(\vec{r}_k) \cdot \vec{E} + \rho_k.\end{aligned} \tag{2.16}$$

This can be written in a set of linear equations

$$\vec{N} - \vec{\rho} = \mathbf{\Lambda}\vec{E}, \tag{2.17}$$

where

$$\mathbf{\Lambda} = \begin{pmatrix} \Lambda(\vec{r}_1, \vec{r}_1') & \cdots & \Lambda(\vec{r}_1, \vec{r}_n') \\ \vdots & \ddots & \cdots \\ \Lambda(\vec{r}_m, \vec{r}_1') & \cdots & \Lambda(\vec{r}_m, \vec{r}_n') \end{pmatrix}, \tag{2.18}$$

is the *response matrix* and has to be inverted to find the energies in the vector $\vec{E}$. It describes the DOM response to light output from certain segments along a track. The entries in this matrix come from simulations that produce spline tables. Simplified sources, such as minimum ionizing muons and isotropically emitting point sources are simulated in Monte Carlo simulations at certain discrete points. Interpolation is done using spline functions. More information, such as how timing information can be implemented, can be found in Refs. [22, 23].

2.1.5 `FiniteReco`

`FiniteReco` is a module that tries to reconstruct if particles are starting, stopping, contained or through-going. The hit DOMs around a seed track are checked to have a seen light and the first and last emission points along the track are used to check the possible hypotheses.

Because the edges of the detector are not well defined*, the likelihoods of individual DOMs to have seen a hit lead to a total likelihood that does not give a conclusive answer, but the starting and stopping probabilities can for example be compared to a through-going track hypothesis.

2.1.6 `Paraboloid`

In sections 2.1.2 and 2.1.3, we discussed how a particle's direction could be estimated. The `Paraboloid` module tries to provide an estimate for the error on this direction. A highly energetic muon with hunderds of hit DOMs will lead to a much better directional resolution than a dim track where only a handful of DOMs are hit. In general, the likelihood space around the estimated direction is scanned and compared to the likelihood of the initial track estimation. This method also gives a robust estimation if the initial track direction is in fact located ath the global maximum likelihood or a local one. `Paraboloid` constructs a grid of zenith and azimuth points near the minimum, for each point on the grid it does a three-parameter minimization for the vertex holding the zenith and azimuth constant. The likelihood values for each point on the grid are then fit to paraboloid using a χ^2 minimization.

Die sigmas en Paraboloid pull uitleggen, beste pagina: doxygen

More information can be found in Ref. [24]

2.2 Pulse cleaning

As explained in Section **??**, each DOM in IceCube has an intrinsic noise rate. This dark noise is observed in every triggered event and seen as random hits in the detector added to the hit pattern of tracks and cascades. These spurious hits are a large nuisance factor in event reconstructions, leading to misidentification and errors in the result. Noise cleaning should be done in early stages of event processing and analysis to reduce a large rate of bad reconstructed events that pass cut selections. One of the most conservative ways is to only look at HLC hits (*HLC cleaning*), but is too demanding for most low-energetic events that will have multiple hits from isolated DOMs.

Another, more conservative method, is the *seededRT* algorithm. This method relies on the "RT cut", which was already implemented in AMANDA. R is a designed radius and T refers to the time between multiple possible pulse times (e.g. the pulse of one DOM starts during the time window of of a second DOMs pulse, or stops during the time window). The full description can be found in Ref. [25], but can be summarized as: DOMs are required to be in a temporal and spacial coincidence that is physically possible (e.g. signal between DOMs cannot exceed the speed of light in vaccuum). This method is however computationally expensive since all DOM pairs have to be looped over†. The seededRT algorithm takes a subset of seeds that are considered to be mostly signal related hits. These seeds can be provided by, for example, using HLC information. By adding all further hits found within the seeds RT-range to the list of seed hits and iterating until a convergence, only those (SLC) hits are kept which cluster around the initial seed hits. Outlying noise hits are supposedly not added and thus removed in the cleaned output. This method does not scale as drastically as the original RT cut method.

2.3 `IceHive`

In Section **??**, it was explained how multiple triggers were combined into one global trigger. In a first step, Q-frames are simply re-split into the individual events that belong to the different subtriggers. In about 10% of cases the data read out in one of these P-frames contains more than one primary interaction. This pile-up effect is referred to as *coincident events*. This is a

*Imagine a cascade 20 m below the lowest DOMs. It is still possible for light to reach the bottom modules of the detector.

†The number of pairs for n DOMs is equal to $\frac{1}{2}n(n-1)$ and scales with n^2.

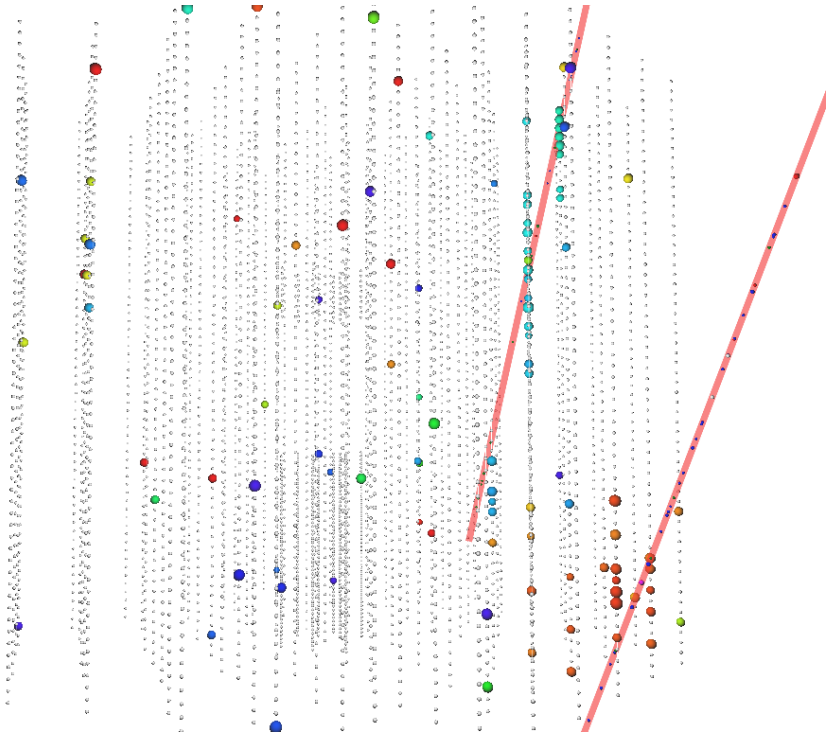


Figure 2.4: Event display of a simulated coincident event of two downgoing muons. The colors of the event range from red (early) to blue (late). The first muon hits the bottom of the detector, while the second traverses mainly the upper part. These events are often reconstructed as a single up-going event and therefore result in a large background contribution. The scattered isolated hits are due to noise effects and mostly removed by pulse cleaning.

direct result of the traversal time of a couple of microseconds in the detector*, the large flux of low-energetic events and the trigger time windows of a couple of microseconds. This can be problematic for reconstructions, as can be seen in Fig. **??** where two downgoing muons can be reconstructed as an upgoing track.

There are two modules that try to clean events more thoroughly than pulse cleaning alone. The first is *TopolocalSplitter*(TS), which starts from the Q-frames and loops over pulses and splits the event into clusters of pulses that contain at least a number of causally† connected pulses within a certain time window. Some extra cleaning, similar to seededRT cleaning, is done in addition and can split coincident events that have overlapping readout windows, but are geometrically separated.

The second module, and also used in this analysis, is called `IceHive`. A full description can be found in the doctoral thesis of M. Zoll [26]. The module consists of two main parts: one that splits events and handles coincident events, `HiveSplitter` and another that has a refined pulse cleaning, `HiveCleaning`.

2.3.1 `HiveSplitter`

The module assumes that individual particles will create *clusters* of hits in the detector. A cluster can grow within a certain time window, but is separated from another cluster if it's not spatially connected. An initial cluster is formed if the multiplicity of hits exceeds a certain threshold (usually 3 or 4). The main difference in this module versus TS is that is uses hexagons to describe the detector instead of assuming a spherical parameterization. It makes more sense to optimize the search volume, where hits are clustered together, with a shape that describes the detector

*The speed of light in vaccuum is equal to ≈ 0.3 m/ns, meaning the particle travels around 100 m in 0.3 μs, without accounting for the delayed photon propagation necessary for detection.

†The time between two DOM hits cannot be less than the time that light may have taken.

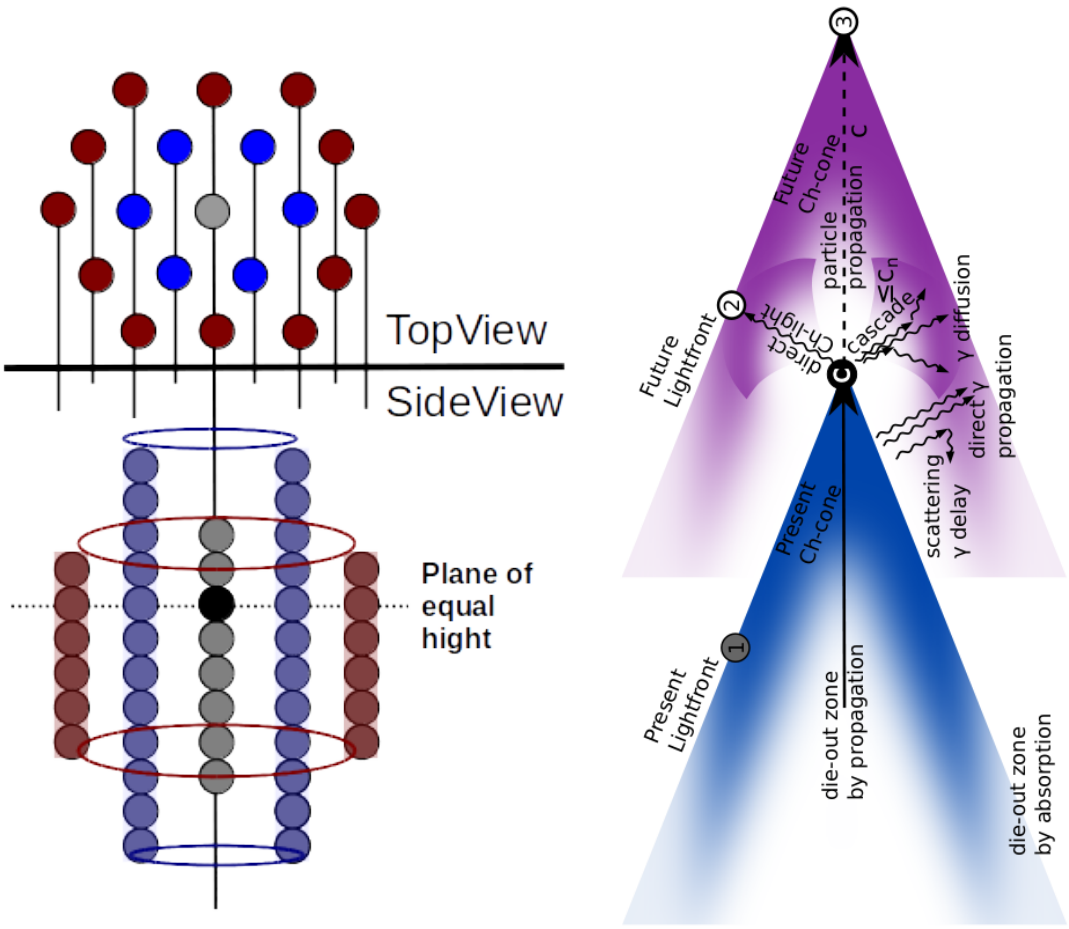


Figure 2.5: *Left:* The black circle illustrates a DOM that triggered a hit in the detector. The grey circles symbolize the DOMs along the string of the hit DOM. The number of DOMs that can be included in the active volume depends on the height defined by the module. The blue/purple DOMs belong to the neighboring strings and the red DOMs to the next-to-neighboring strings. The heights of both these sets of DOMs are also set by the module. This example shows $h_2 > h_1 > h_3$, the heights are also asymmetric in this example. *Right:* Illustration of Cherenkov emission profile of a traversing particle. Both figures from Ref. [26].

well and uses a discrete spacing between larger volumes instead of a uniformly growing sphere. The hexagonal shape is set by defining three heights. The first height is defined along the string of the hit DOM and is equal to the vertical distance along the string where another. The second height is the vertical height along the neighboring strings. The third hight is the vertical height along the next-to-neighboring strings. An example is shown in Fig. 2.5.

When the active region is set, additionally it is checked if DOMs can be "connected". `IceHive` assumes certain emission profiles (for both cascades as tracks) where light is produced. Three possible connections are checked:

1. Hits occur at the same time, but at a spatial distance in agreement with the Cherenkov emission profile (hits C&1 and 2&3 in Fig. 2.5)
2. Hits occur at a different time and a different location, but in agreement with the Cherenkov emission profile (hits C&2 in Fig. 2.5).
3. Hits on topologically identical sites of an emission pattern that has moved along with the propagation of the particle (hits C&3 and hits 1&2 on Fig. 2.5).

These clusters are finally separated into different events, P-frames.

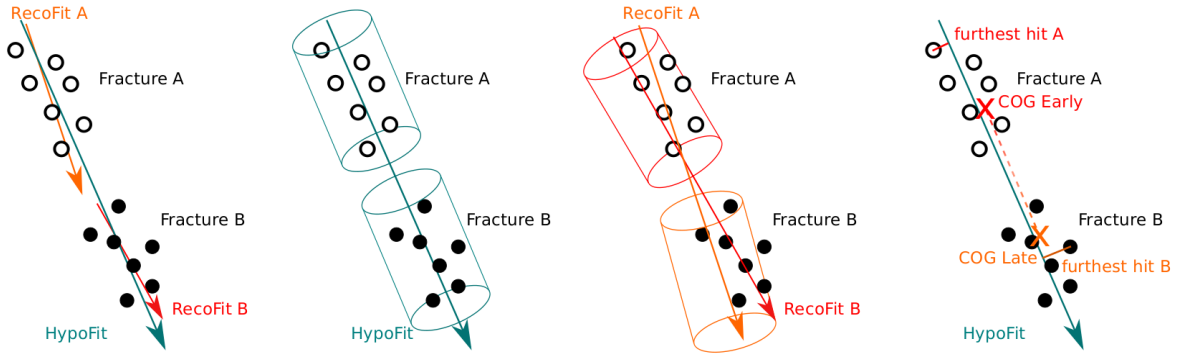


Figure 2.6: Schematic illustrations of possible recombination scenarios with the `CoincSuite` module.

2.3.2 `HiveCleaning`

Additionally, a similar cleaning as explained in Section 2.2 can be performed. Isolated hits that do not have neighbouring hits occurring within a certain distance within a certain time window, are removed. The main difference between this and seededRT cleaning is that the module again uses the hexagons as defined in the previous section.

the usage of `IceHive` has a great performance in separating coincident events, but often "overperforms" and splits clusters of hits that are originating from the same particle. This is predominantly the case for dim tracks that have large separations in between clusters (most of the triggered SMP events are of this type). It is because of this that the module *CoincSuite* was designed.

2.4 CoincSuite

Several testing algorithms allow to check if two or more split P-frames can originate from a single event. Five different scenarios were tested in this analysis, the first four are also chronologically shown in Fig. 2.6:

1. Cluster alignment: the reconstructed direction of the individual clusters is compared to the direction of a reconstruction that uses the combined hits (HypoFit). The directions should be within a certain criticle angle.
2. Cylinder cluster containment: the DOMs of the individual clusters should be able to be grouped together in a cylinder that has its center and direction along the HypoFit.
3. Cylinder cluster alignment: a cylinder around the recontruction of each cluster is draw. The cylinders should overlap within a certain fraction.
4. COG* connection: the second quarter of the COG of the first cluster and the third quarter of the COG of the second cluster are computed. These COG should lie close enough and have to be in the vicinity of the HypoFit.
5. Velocity test: tests if the velocity of the HypoFit is close to the speed of light.

The combination of `IceHive`, which does a very good job in cleaning events, but often overperformes and splits events that shouldn't be, and `CoincSuite`, which recombines events that were wrongfully split leads to a very powerful tool to clean events.

2.5 Analysis techniques

Komt hier iets dat je gebruikt in het volgende hoofdstuk maar nog niet hebt uitgelegd?

*Similar to COM (Center Of Mass), the COG is a weighted average position of the hit DOMs. DOMs that register more light get a heigher weight.

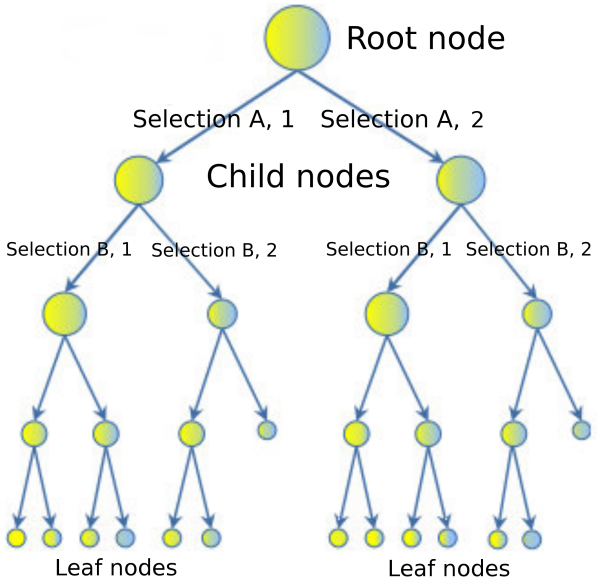


Figure 2.7: Illustration of decision tree scheme.

2.6 Boosted Decision Tree classifiers

Given a certain event with a fixed set of variables that are constructed in an analysis, the question remains if the event is in fact a *signal* or *background* event. One can rely on Monte Carlo simulations to get a handle on the variable distribution in both sets. The most general and still widely used method is to use a cut-and-count approach where a cut is placed on a certain variable that discards events that fail the requirement. A Boosted Decision Tree (BDT) inspects a set of set variables and classifies an event with a score that ranges from -1 to 1. The higher the score, the more an event is regarded as signal-like. How this is done is given in more detail below. Boosted decision trees rely on multiple individual trees. Therefore, we will first explain how a single tree classification works.

2.6.1 Stucture

The goal of a decision tree is to determine if an event is signal- or background like. It uses a tree-like structure where certain selection criteria are used at different nodes as illustrated in Fig. 2.7.

An decision tree is a binary tree that places an event into a certain node depending on the selection at a node. The depth of a decision tree can be arbitrarily long, but is but are determined by a set of criteria defined in Section 2.6.2). An event consists of a certain set of variables $X = x_1, x_2, ...x_n$ that are used in the classification. Before any selection criteria, the event is said to be represented by the *root node*. A binary selection then determines to which *child node* the event should be classified, for example:

$$\begin{aligned} \text{Selection A} &= x_1 > y_1 \quad \text{(option 1)} \\ &= x_1 \leq y_1 \quad \text{(option 2)}, \end{aligned} \tag{2.19}$$

where y_1 is the cut value for variable x_1. Similarly, the other selections deterimine where the event is eventually placed. The last nodes are referred to as *leaf nodes* and hold the probabilities of whether an event is more signal- or background-like. These probabilities are translated into a score ranging between -1 (background) and 1 (signal).

2.6.2 Training

To construct a decision tree, one first has to "train" the algorithm. Given a certain "signal set" and "background set", all variables used in the BDT are histogrammed and at each bin for each

variable the "best cut" is set at the first node selection. This best cut uses the purity of a node, p, to select the optimal cut:

$$p = \frac{\sum_s w_s}{\sum_s w_s + \sum_b w_b}, \tag{2.20}$$

where w_s and w_b refer to the weights of the signal and background events and

$$g(p) = p(1-p), \tag{2.21}$$

where g refers to the Gini index that is used as a separation variable in this work*. Using the Gini index, the separation gain determins the effectiveness of the cut

$$\Delta S = g_p \cdot \sum w_p - \left(g_l \cdot \sum w_l + g_r \cdot \sum w_r \right), \tag{2.22}$$

where g_p and w_p denote the Gini index and weights of the parent nodes and similarly for the left and right child nodes. The cut that gives the highest separation gain is subsequently selected. The algorithm stops when one of the following criteria is met:

- A node only consists of signal or background events.
- A certain predefined maximal depth is reached.
- Splitting would cause a child node to have less than a predefined minimal amount of events left;

and therefore determines the size of a tree. These selection criteria are necessary to avoid overtraining (see Section 2.6.4).

2.6.3 Boosting

As already implied in the text above, a BDT consists of a *forest* of decision trees. A user specified number of individual decision trees are trained sequentially, with a boosting process in between each training. Boosting consists of adjusting the weights of individual events according to whether the previously trained tree classifies them correctly. In this work, the AdaBoost† algorithm was used for boosting in which the score of an event is a weighted average of the scores the event receives from each tree in the forest [27].

A BDT may informally be called a "boosted decision tree", but it must be understood that there are actually many trees (typically hundreds), and that boosting is a process that occurs between the training of consecutive trees. The approach makes use of the power of numbers: many weak single decision trees combined can be more powerful than one very good decision tree. In general, boosting follows the following steps:

1. Train a weak model on training data.
2. Compute the error of the model on each training example.
3. Give higher importance to examples on which the model made mistakes.
4. Re-train the model using "importance weighted" training examples.
5. Go back to step 2.

An example is given in Appendix E.

If $I(s, y)$ is a function equal to 0 if the sample test score after tree classification, s, is equal to it's true identity, y, then the error rate for a tree is equal to

$$\epsilon = \frac{\sum_i w_i I(s, y)}{\sum_i w_i}. \tag{2.23}$$

The boosting factor for the tree is defined as

$$\alpha = \beta \cdot \ln\left(\frac{1-\epsilon}{\epsilon}\right), \tag{2.24}$$

*Other possible separation variables include the cross entropy $-p \cdot \ln(p) - (1-p) \cdot \ln(1-p)$ or the misclassification error $1 - \max(p, 1-p)$

†Short for Adaptive Boosting.

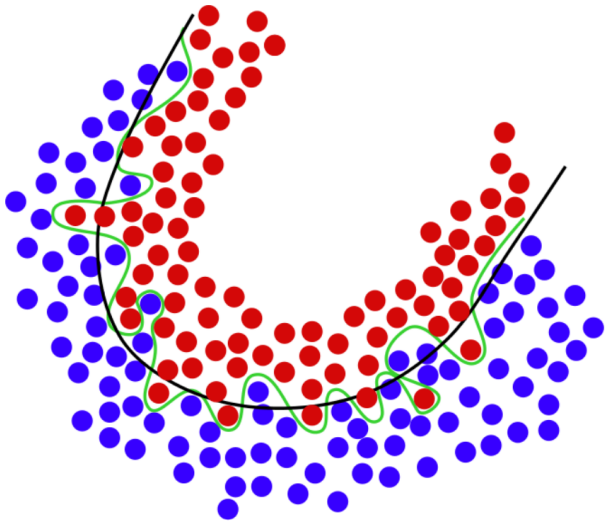


Figure 2.8: Example of overtraining. In black, the theoretical line between background (blue points) and signal (red points) is given. An overtrained BDT will have a (perfect) selection such as the green line. Illustration from [28].

and changes the weight of the tree to

$$\begin{array}{cc} w' = w \cdot \exp(\alpha), & w' = w \cdot \exp(-\alpha) \\ \text{(correct classification)} & \text{(incorrect classification)}, \end{array} \tag{2.25}$$

after which the weights are renormalized so that $\sum w = 1$. The process is repeated until the number of predefined trees is reached.

Due to its definition, the boost factor α will give good classifiers (which have low error rates, ϵ) large boost factors. Events that are misclassified are then given larger weights, making the algorithm more likely to classify them correctly in the subsequent tree classifier.

Once the entire BDT is trained, the events can be given a score based on the multiple tree classifiers. This is done by taking the weighted average of all the scores in the individual tree classifiers, using its boost factor α as the weight of the tree. The score of an event i is then given by

$$s_i = \frac{\sum_m \alpha_m \cdot s_{i,m}}{\sum \alpha_m}, \tag{2.26}$$

where we loop over the individual trees denoted with index m.

2.6.4 Overtraining

BDTs are very powerful tools, but if not used correctly could lead to problems that are not easy to spot at first sight. Assume we train our BDT with a certain signal set and background set. If the BDT is trained up to the point of classifying statistical fluctuations, there is said to be *training sample overtraining*. An example is given in Fig. 2.8. Another example is data/MC overtraining. The former can be dealt with with the use of *pruning*, while the latter should show clear data/MC agreement.

2.6.4.1 Pruning

The problem with overtraining is essentially that there are certain splits in a classifier tree that are too specific and less important. In the method of *cost complexity pruning*, for each node the complexity is calculated as

$$\rho = \frac{\Delta S}{n_{\text{leaves}} - 1}, \tag{2.27}$$

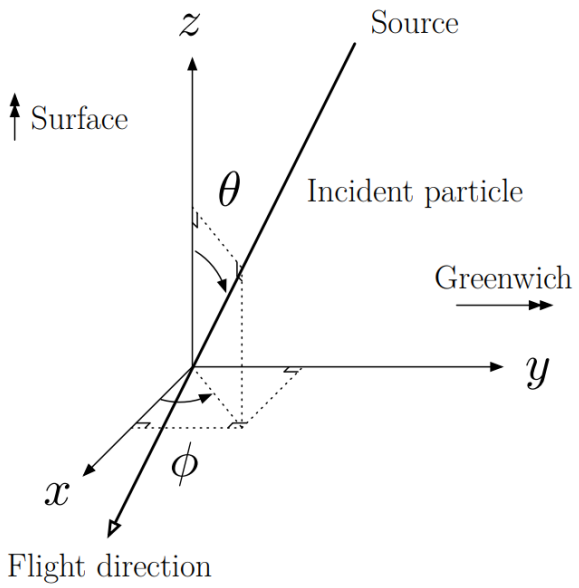


Figure 2.9: The IceCube Coordinate system.

with ΔS th separation gain as defined in Eq. 2.22. The subtree of the node with the smallest value of ρ is removed and this is done repeatedly until a desired *pruning strength** is reached.

BDTs are always trained and tested with different subsets of the same type (background or signal) of event. If the training set has significantly different scores than the testing set, the sample is most probably overtrained and one has to change the BDT input parameters. Typically, one uses Kolmogorov-Smirnov testing to indicate if the BDT score distributions from the training and testing sets could follow the same distribution pattern. As a rule of thumb, a $p_{KS} \lesssim 0.01$ indicates overtraining beyond the level of comfort.

2.6.4.2 Data-MC agreement

Nodig?

2.7 Paraboloid

2.8 IceCube coordinate system

Lastly, when referring to positions and directions one first has define a coordinate system to be able to uniquely define these variables. The system is shown in Fig. 2.9.

The center of the coordinate system is set close to the geometric center of the detector, at about 2000 m below the surface of the ice. The y-axis of the coordinate system is aligned with the Prime Meridian pointing toward Greenwich (United Kingdom). The x-axis is set perpendicular to the y-axis pointing in a, 90° clockwise direction. The z-axis is set perpendicular to the xy-plane, pointing upwards, normal to the Earth's surface.

A particle's direction is defined with zenith and azimuth angles, θ and ϕ respectively. The zenith angle is measured relative to the positive z-axis and the azimuth angle is measured counterclockwise from the positive xy-plane.

*A parameter on a scale from 0 to 100, which specifies the percentage of the pruning sequence to actually execute. 0 means no pruning is done and 100 signifies only a single root node remaining.

3. The SPACE Analysis

De uitleg van temperature modulations is goed in Sam zijn thesis `https://arxiv.org/pdf/physics/0312102v1.pdf`
`https://arxiv.org/pdf/1310.1284.pdf`

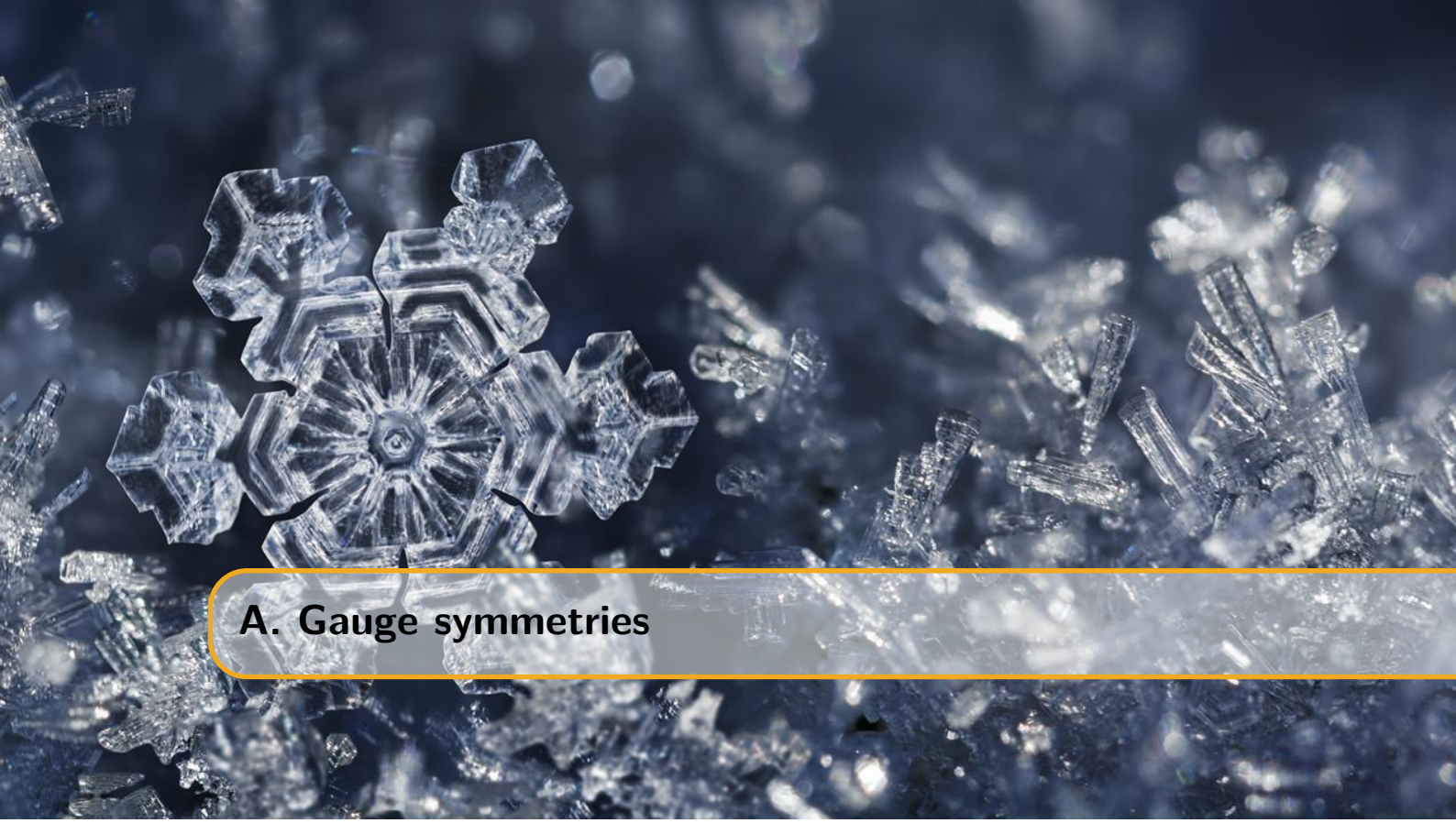
Ook ergens een tabel maken met info over je data runs. Duidelijk maken wat de livetime is bv en ook zeggen van wanneer to wanneer een bepaalde run liep (2011: mei 2011- mei 2012)

Klaus zijn paper? `https://arxiv.org/abs/1806.05696`

4. Summary and Discussion

III Additions

Appendices



A. Gauge symmetries

NOG NIET GEDAAN

The difference between global and local symmetries are not straightforward for everybody. In this appendix I try to give a better view of the matter.

Imagine that at each point in space and time there is a circle attached to it. If one shifts all circles of all points with a fixed angle the underlying physics hasn't changed. If we look at the whole in a different angle, nothing seems to be changed as everything holds the same relative orientation. This is a global symmetry. For local symmeries we instead shift each circle through a different angle, but an angle that changes smoothly from point to point and in a way that we can say how that angle is varying between different nearby regions. Then it will turn out that we can describe that rotation angle by means of a so-called gauge field, which just lets us transport the charged scalar field from one point in space time to another, taking account of how the rotation angle of the circle is changing. A gauge is a kind of coordinate system that is varying depending on the location with respect to some underlying space. In physics we are almost always concerned with space-time as the underlying space, and we are typically interested in theories that are invariant with respect to the choice of gauge or coordinate system.

Dan wat uitleg vanuit je QFT boek en de dingen hieronder: Je wilt je derivative anders doen werken in je theory onder een transformatie, maar daarvoor heb je een veld nodig. M.a.w.: dankzij een veld heb je lokale ijktransformatie mogelijk!

B. Planck's law

bron: http://hyperphysics.phy-astr.gsu.edu/hbase/quantum/rayj.html

B.1 Electromagnetic waves in a cubical cavity

Suppose we have EM waves in a cavity at equilibrium with its surroundings. These waves must satisfy the wave equation in three dimensions:

$$\frac{\partial^2\Psi}{\partial x^2}+\frac{\partial^2\Psi}{\partial y^2}+\frac{\partial^2\Psi}{\partial z^2}=\frac{1}{c^2}\frac{\partial^2\Psi}{\partial t^2}. \tag{B.1}$$

The solution must give zero amplitude at the walls. A non-zero value would mean energy is dissipated through the walls which is in contradiction to our equilibrium assumption. A general solution takes the form of

$$\Psi(x,y,z,t)=\Psi_0\sin k_1x\sin k_2y\sin k_3z\sin k_4t, \tag{B.2}$$

which, after requiring $k_nL=n\pi$ with $n=0,1,2...$ and $k_4\frac{\lambda}{2c}=\pi$, leads to

$$\Psi(x,y,z,t)=\Psi_0\sin\left(\frac{n_1\pi x}{L}\right)\sin\left(\frac{n_2\pi y}{L}\right)\sin\left(\frac{n_3\pi z}{L}\right)\sin\left(\frac{2\pi ct}{\lambda}\right). \tag{B.3}$$

From the wave equation it is easy to find that

$$n^2=n_1^2+n_2^2+n_3^2=\frac{4L^2}{\lambda^2}, \tag{B.4}$$

which span up a sphere in "n-space" with a volume of $\frac{1}{8}\frac{4}{3}\pi n^{3/2}$, where the first term originates from the positive nature of $n_{1,2,3}$. Because there are two possible polarizations of the waves one has to multiply with an additional factor 2. The number of modes per unit wavelength is equal to

$$\frac{dN}{d\lambda}\times\frac{1}{L^3}=\frac{d}{d\lambda}\left[\frac{8\pi L^3}{3\lambda^3}\right]\times\frac{1}{L^3}=-\left[\frac{8\pi}{\lambda^4}\right]. \tag{B.5}$$

B.1.1 Classical approach

Following the principle of equipartition of energy, each standing wave mode will have an average enkergy kT with k the Boltzmann constant and T the temperature in Kelvin. The energy density is then:

$$\frac{du}{d\lambda} = -kT\frac{8\pi}{\lambda^4}. \tag{B.6}$$

In function of frequency $\nu = \frac{c}{\lambda}$:

$$\frac{du}{d\nu} = -\frac{c}{\lambda^2}\frac{du}{d\lambda} = \frac{8\pi kT\nu^2}{c^3}, \tag{B.7}$$

also known as the Rayleigh-Jeans law*. Problem: divergence

B.1.2 Quantum approach

The energy levels from a quantized harmonic oscillator are equal to

$$E_r = h\nu\left(r + \frac{1}{2}\right) = \frac{hc}{\lambda}\left(r + \frac{1}{2}\right) \quad \text{with} r = 0, 1, 2, ... \tag{B.8}$$

Implementing eq. B.4

$$E = \left(r + \frac{1}{2}\right)\frac{hc}{2L}\sqrt{n_1^2 + n_2^2 + n_3^2} \tag{B.9}$$

According to statistical physics the average energy is now not equal to kT but follows a probability distribution

$$p(\nu, r) = \frac{e^{\frac{-rh\nu}{kT}}}{\sum_{r=0}^{\infty} e^{\frac{-rh\nu}{kT}}}, \tag{B.10}$$

where we reference to the ground state of the oscillator: $E'_r = E_r - E_0$.

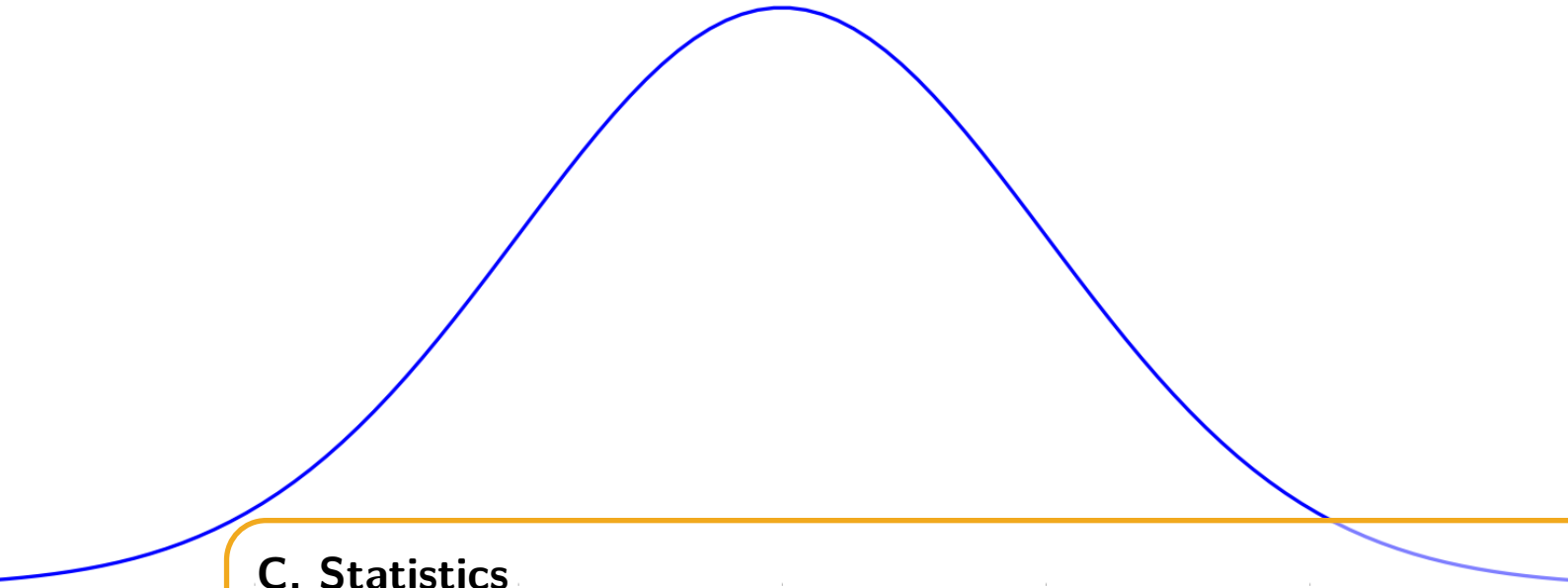
The average energy is now:

$$\begin{aligned}\langle E(\nu)\rangle &= \sum_{r=0}^{\infty} E(\nu, r)\cdot p(\nu, r) = \frac{\sum_{r=0}^{\infty} rh\nu\ e^{\frac{-rh\nu}{kT}}}{\sum_{r=0}^{\infty} e^{\frac{-rh\nu}{kT}}} \\ &= \frac{h\nu}{e^{h\nu/kT} - 1}\end{aligned} \tag{B.11}$$

Substituting this for kT in eq. B.7 we find Planck's equation:

$$\frac{du}{d\nu} = \frac{8\pi h\nu^3}{c^3}\frac{h\nu}{e^{h\nu/kT} - 1} \tag{B.12}$$

*This is often quoted per unit of staradian, which results in $\frac{2kT\nu^2}{c^3}$



C. Statistics

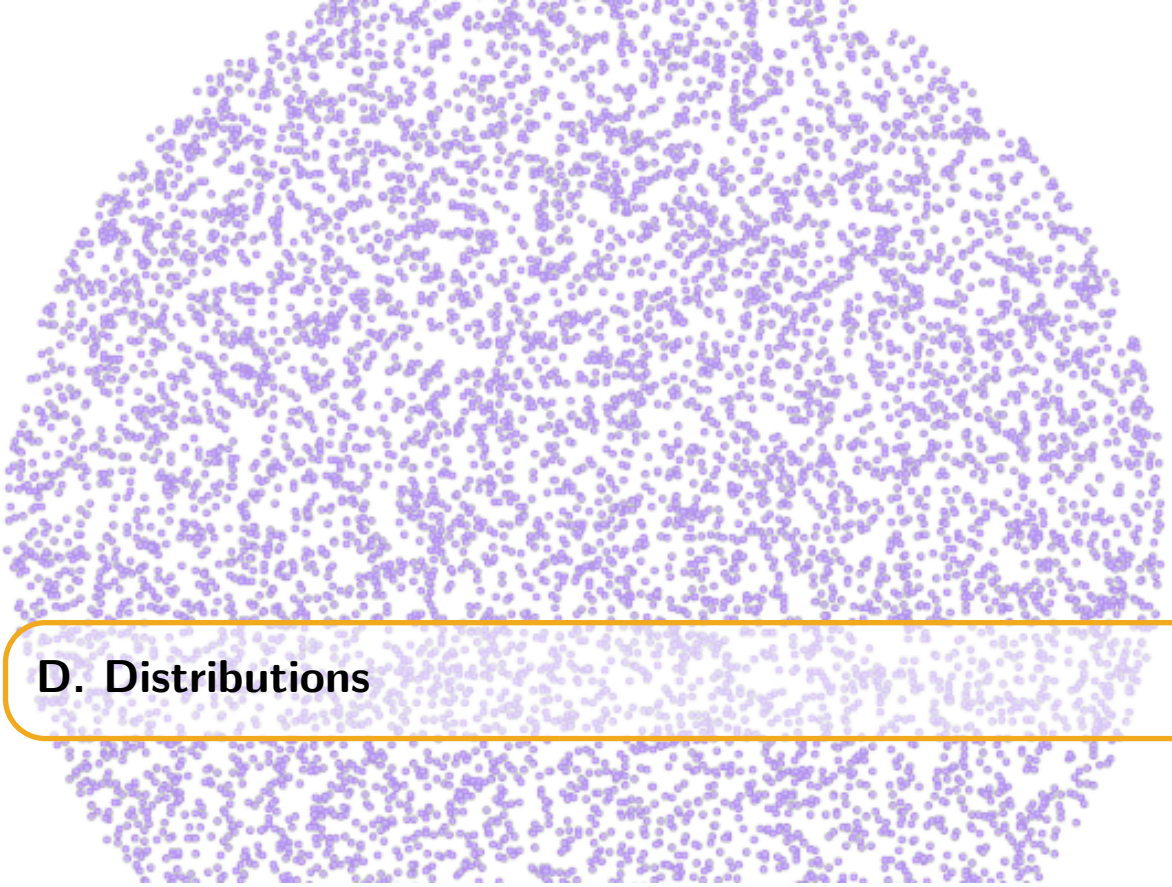
A word that is often mentioned in this work is “statistics”. It refers to the statistical error of a counting experiment, i.e. the Poissonian error. The Poisson distribution is a discrete probability of a certain number of $n$events occurring in a fixed time interval. The Poisson probability function is given by

$$P(n) = \frac{\lambda^n e^{-\lambda}}{n!}, \tag{C.1}$$

where λ is the expected number of events and also equal to the variance. An experiment that counted N events therefore has a statistical error of

$$\sigma = \sqrt{N} \tag{C.2}$$

In other words: higher statistics denotes a lower statistical error.



D. Distributions

D.1 Spherical random numbers

Most random number generators provide uniform distributions from between the range $[0, 1]$. Assume we want to make a uniform distribution along a sphere with angles ϕ and θ and radius r, in spherical coordinates. Random numbers between $[0, \pi]$, $[0, 2\pi]$ and $[0, R]$ (the ranges of the coordinates) would not give a uniform distribution as illustrated in Fig. D.1 (left).

The differential surface area, dA, is equal to $dA(d\phi, d\theta) = r^2 \sin(\phi) d\phi d\theta$. If we want the distribution $f(v)$ to be constant for a uniform distribution, then $f(v) = \frac{1}{4\pi}$ since $\int \int_S f(v) dA = 1$ and $\int \int_S dA = 4\pi$. We want the distribution in function of the angles, so

$$f(v)dA = \frac{1}{4\pi} dA = f(r) f(\phi, \theta) d\phi d\theta. \tag{D.1}$$

Since we know the expression for dA, we find that

$$f(\phi, \theta) = \frac{1}{4\pi} \sin(\phi), \tag{D.2}$$

and separating the angles:

$$f(\theta) = \int_0^{\pi} f(\phi, \theta) d\phi = \frac{1}{2\pi}, \tag{D.3}$$

$$f(\phi) = \int_0^{2\pi} f(\phi, \theta) d\theta = \frac{\sin(\phi)}{2}, \tag{D.4}$$

where it is clear that $f(\phi)$ scales with $\sin(\phi)$; there are more points needed at the equator (this makes sense, as the surface at the equator is much larger!).

The question is now how one can get a sample to follow the distribution of $f(\phi)$. For this, we use the *inverse transform sampling* method where one makes use of the cumulative distribution function, $F(\phi)$, which increases monotonically

$$F(\phi) = \int_0^{\phi} f(\phi') d\phi' = \frac{1}{2} \left(1 - \cos(\phi)\right). \tag{D.5}$$

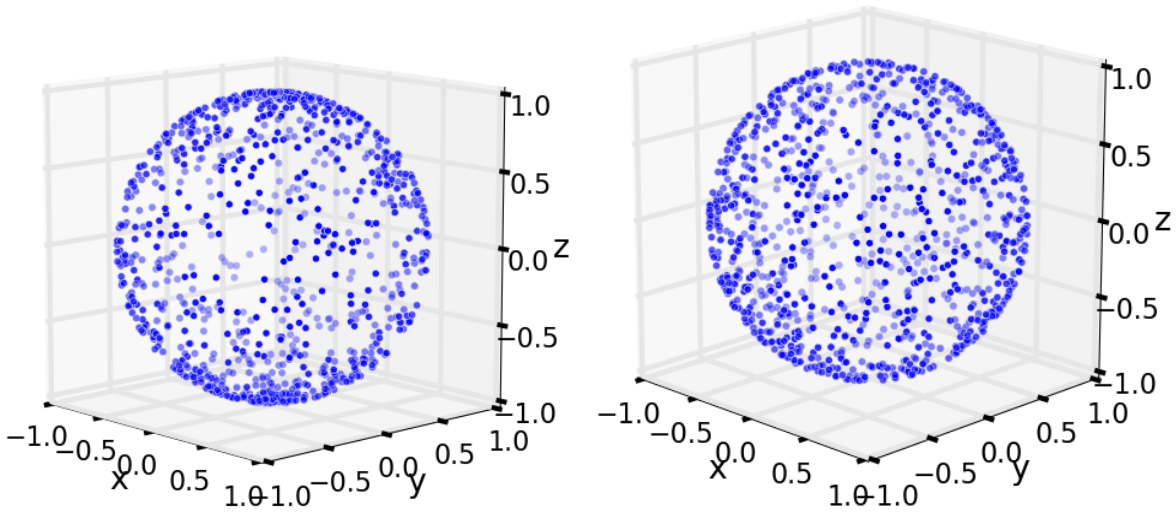


Figure D.1: *Left:* Illustration of a uniform sampling in angles ϕ and θ that doesn't give a uniform spherical distribution. *Right:* Illustration of a good spherical distribution.

The method shows that if u is a random variable drawn from a uniform distribution, we have to find the inverse function of F,

$$F(F^{-1}(u)) = u \tag{D.6}$$

$$\frac{1}{2}\left(1 - \cos(F^{-1}(u))\right) = u \tag{D.7}$$

$$F^{-1}(u) = \arccos(1-2u). \tag{D.8}$$

In other words: if u is a random variable drawn from a uniform distribution, then $\phi = \arccos(1-2u)$ follows a distribution necessary for a uniform spherical distribution. Similarly, $\theta = \frac{1}{2\pi}u$.

D.2 Power law distributions

Analogous to what was written in the previous section, one can produce a power law distribution from random numbers using the inverse transform sampling method:

$$\begin{aligned} f(E) &= A \cdot E^{-\gamma} \quad \text{(powerlaw)} \\ F(E) &= \int_{E_{min}}^{E} A \cdot E^{-\gamma} dE = u \quad \text{(inverse sampling, u random number [0,1])} \\ &= A \left[\frac{E^{-\gamma+1}}{-\gamma+1}\right]_{E_{min}}^{E} \\ &= \frac{A}{-\gamma+1}\left(E^{-\gamma+1} - E_{min}^{-\gamma+1}\right) \end{aligned} \tag{D.9}$$

Because we know that $F\left(F^{-1}(u)\right) = u$, we can find an expression for $F^{-1}(u)$:

$$\begin{aligned} u &= \frac{A}{-\gamma+1}\left(\left(F^{-1}(u)\right)^{-\gamma+1} - E_{min}^{-\gamma+1}\right) \\ &\Rightarrow \\ F^{-1}(u) &= \left(\left(\frac{-\gamma+1}{A} \cdot u\right) + E_{min}^{-\gamma+1}\right)^{1/(-\gamma+1)} \end{aligned} \tag{D.10}$$

To find A, we use the property of a CDF:

$$F(E_{max}) = 1 \Rightarrow A = \frac{-\gamma + 1}{E_{max}^{-\gamma+1} - E_{min}^{-\gamma+1}}, \tag{D.11}$$

leading to

$$F^{-1}(u) = \left((1-u) \cdot E_{min}^{-\gamma+1} + u \cdot E_{max}^{-\gamma+1}\right)^{1/(-\gamma+1)}, \tag{D.12}$$

which shows how one can draw a distribution in function of E following $f(E)$ with a uniform random number u.

For γ = -1, the computations are analogous and one can see that this will produce a uniform distribution in log space. This is shown in Fig. D.2.

$$\begin{aligned} E &= E_{min} \cdot 10^{u \cdot \log \frac{E_{max}}{E_{min}}} \\ &= 10^{u[\log E_{min}, \log E_{max}]} \end{aligned} \tag{D.13}$$

D.3 Angular distributions

As seen in Section D.1, the differential space angle $d\Omega$ is equal to

$$d\Omega = \sin(\theta) d\theta d\phi. \tag{D.14}$$

If one shows the distribution of ϕ and/or θ, then this is the same as showing partial integrations per bin. We find that

$$\Omega \propto \cos(\theta), \tag{D.15}$$

or in other words: the space angle is proportional to the azimuth and the cosine of the zenith. An example is shown in Fig. D.3.

D.4 Weighting

A method that is often used in simulations is *weighting*. The simulated and expected differential flux of particles is often not the same, mainly due to two reasons:

- The flux has no uniform power law behavior. As can bee seen in Fig. **??**, there are multiple "kinks" and changes in the spectrum. Instead of simulating the flux according to one model, a general uniform flux is used and later reweighted to be able to fit to other models more easily.
- A steep power law indicates very few events at the highest energy bins. This means large CPU time would be necessary to simulate these events. As an example, let us assume two different fluxes

$$f_1 = A \cdot x^{-1} \tag{D.16}$$

$$f_2 = B \cdot x^{-2}, \tag{D.17}$$

where $A = 10^3$ and $B = 10^4$, so the fluxes cross at a value of $x^{-1+2} = x = \frac{10^4}{10^3} = 10$. In the interval $x \in [10^3, 10^4]$, the number of events for f_1 is equal to 10^3, whereas for f_2 this is equal to 9.
Simulating with harder spectra* leads to more statistics in high-energy bins.

*Harder spectra equals to a lower gamma, since there will be more high-energy events.

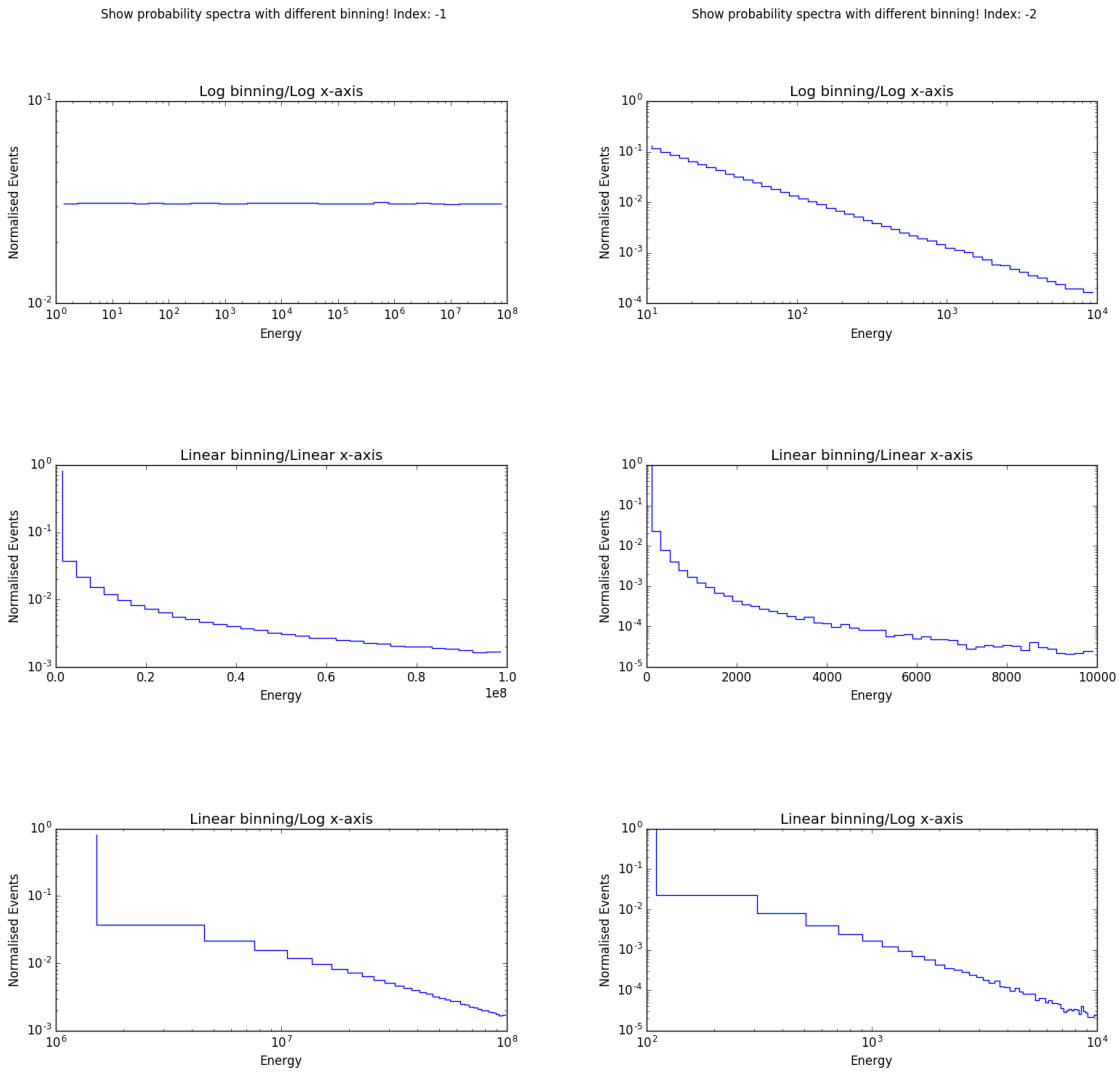


Figure D.2: *Left:* Histograms with different binnings showing the behavior of an energy spectrum with spectral index -1. *Right:* Histograms with different binnings showing the behavior of an energy spectrum with spectral index -2.

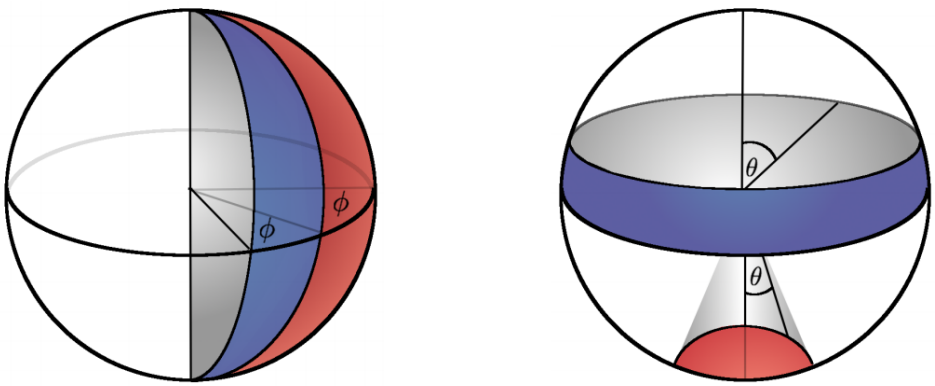


Figure D.3: Illustration of angle distributions in spherical coordinates. The blue and red surfaces are equal in size. The left figure clearly shows the surface to be proportional to the azimuth. The right figure shows how there is a non-trivial dependence on the zenith angle for equal partitions on the surface of a sphere.

The weights can be generally written down as

$$w = \frac{dN_{exp}}{dAd\Omega dEdt} \times \frac{dAd\Omega dE}{dN_{sim}}. \tag{D.18}$$

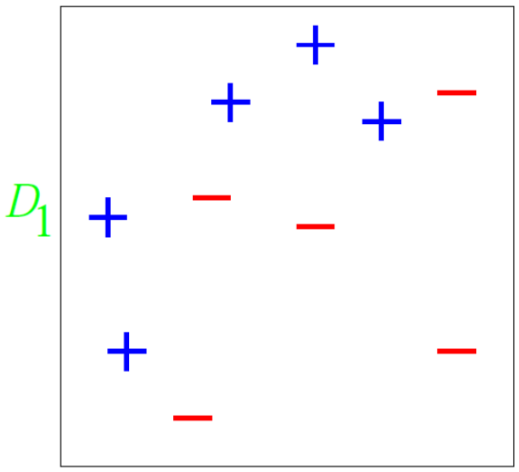
A disadvantage of using weights is that certain events with a high weight are rare but can dominate or obscure the sample in the tails of certain distributions.

E. AdaBoost: simple example

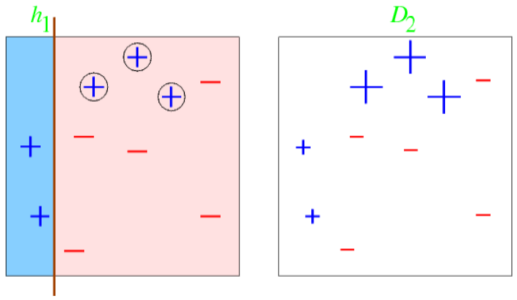
Consider a binary decision tree classification with 10 training examples. The illustrations below are 2D variable distributions.

We give each event an equal weight, making the weight distribution D_1 uniform. For this simple example, each of our classifiers will be an axis-parallel linear classifier (simple cut in one of the two variables).

Initial distribution



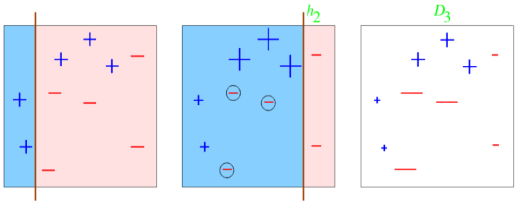
Round 1



- Error rate of h_1: $\epsilon_1 = 0.3$; weight of h_1 (see Eq. 2.24): $\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = 0.42$

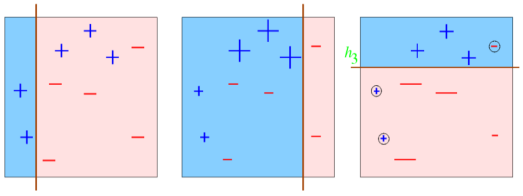
- An event that is misclassified gets a higher weight: weight multiplied with $\exp(\alpha_1)$
- An event that is correcly classified gets a lower weight: weight multiplied with $\exp(-\alpha_1)$

Round 2



- Error rate of h_1: $\epsilon_1 = 0.21$; weight of h_2 (see Eq. 2.24): $\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\epsilon_2}{\epsilon_2}\right) = 0.65$
- An event that is misclassified gets a higher weight: weight multiplied with $\exp(\alpha_2)$
- An event that is correcly classified gets a lower weight: weight multiplied with $\exp(-\alpha_2)$

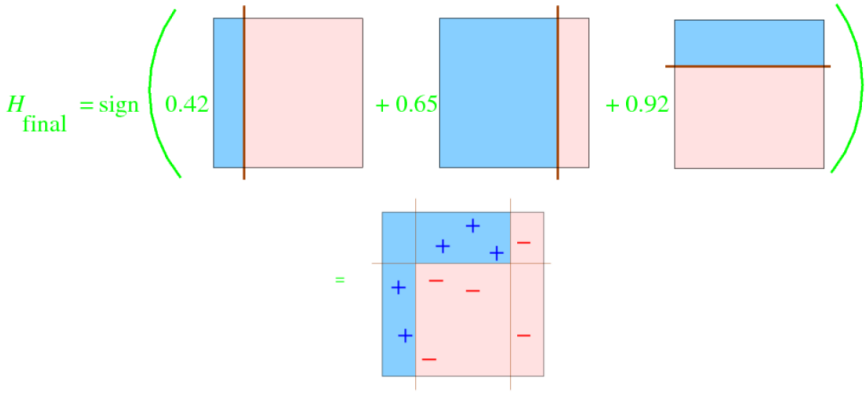
Round 3

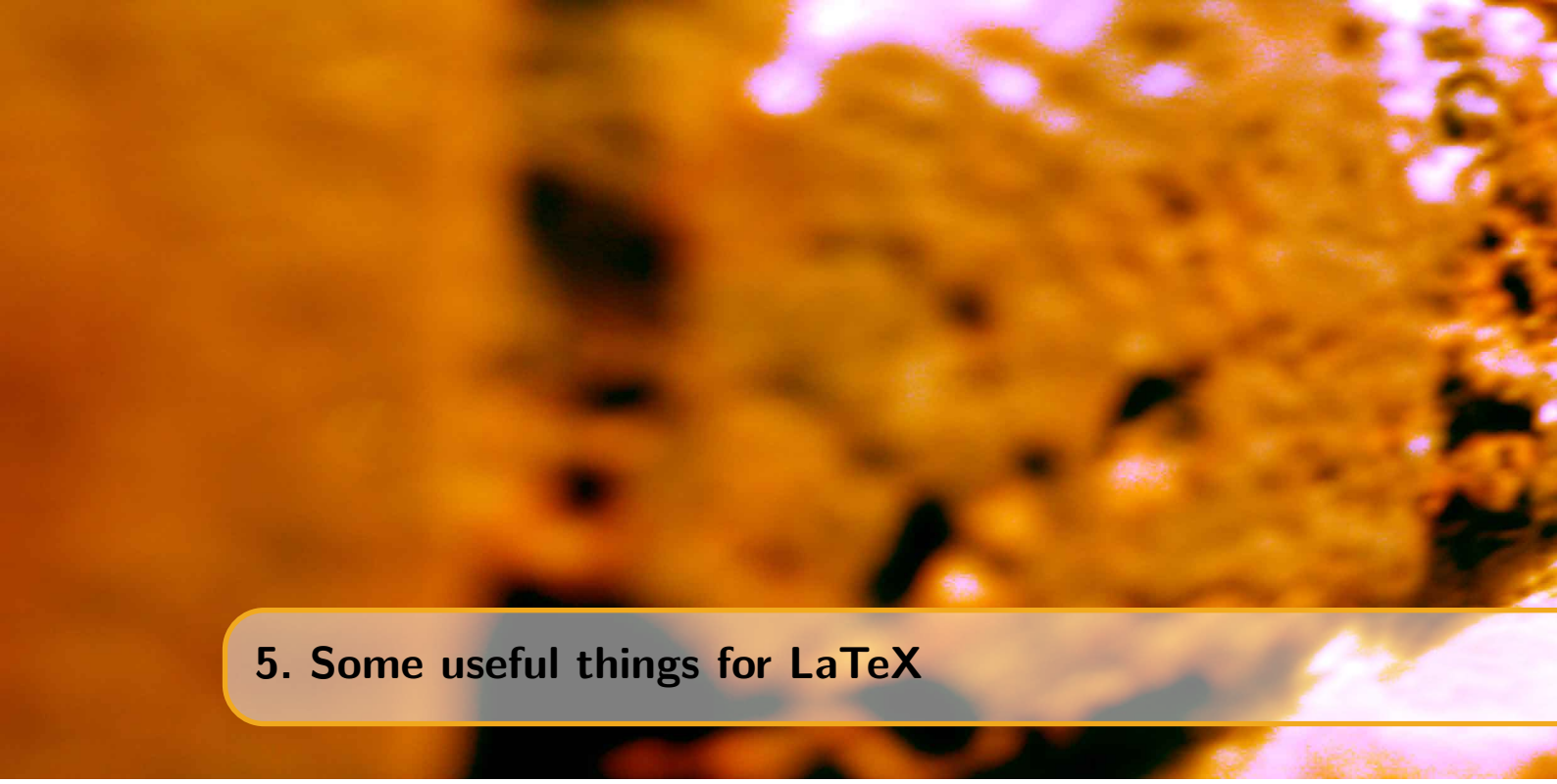


The error rate of h_1: $\epsilon_1 = 0.21$; weight of h_2 (see Eq. 2.24): $\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\epsilon_2}{\epsilon_2}\right) = 0.65$
Let us suppose to stop after this round, we now have a forest of 3 decision classifiers: h_1, h_2, h_3.

Final step

The final classifier is a weighted linear combination of all the classifiers:





5. Some useful things for LaTeX

5.1 Definitions

This is an example of a definition. A definition could be mathematical or it could define a concept.

Definition 5.1.1 — Definition name. Given a vector space E, a norm on E is an application, denoted $||\cdot||$, E in $\mathbb{R}^+ = [0, +\infty[$ such that:

$$||\mathbf{x}|| = 0 \;\Rightarrow\; \mathbf{x} = \mathbf{0} \tag{5.1}$$

$$||\lambda\mathbf{x}|| = |\lambda| \cdot ||\mathbf{x}|| \tag{5.2}$$

$$||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}|| \tag{5.3}$$

5.2 Remarks

This is an example of a remark.

R The concepts presented here are now in conventional employment in mathematics. Vector spaces are taken over the field $\mathbb{K} = \mathbb{R}$, however, established properties are easily extended to $\mathbb{K} = \mathbb{C}$.

5.3 Corollaries

This is an example of a corollary.

Corollary 5.3.1 — Corollary name. The concepts presented here are now in conventional employment in mathematics. Vector spaces are taken over the field $\mathbb{K} = \mathbb{R}$, however, established properties are easily extended to $\mathbb{K} = \mathbb{C}$.

5.4 Propositions

This is an example of propositions.

5.4.1 Several equations

Proposition 5.4.1 — Proposition name. It has the properties:

$$|||\mathbf{x}|| - ||\mathbf{y}||| \leq ||\mathbf{x} - \mathbf{y}|| \tag{5.4}$$

$$||\sum_{i=1}^{n} \mathbf{x}_i|| \leq \sum_{i=1}^{n} ||\mathbf{x}_i|| \quad \text{where } n \text{ is a finite integer} \tag{5.5}$$

5.4.2 Single Line

Proposition 5.4.2 Let $f, g \in L^2(G)$; if $\forall \varphi \in \mathcal{D}(G)$, $(f, \varphi)_0 = (g, \varphi)_0$ then $f = g$.

5.5 Examples

This is an example of examples.

5.5.1 Equation and Text

■ **Example 5.1** Let $G = \{x \in \mathbb{R}^2 : |x| < 3\}$ and denoted by: $x^0 = (1, 1)$; consider the function:

$$f(x) = \begin{cases} \mathrm{e}^{|x|} & \text{si } |x - x^0| \leq 1/2 \\ 0 & \text{si } |x - x^0| > 1/2 \end{cases} \tag{5.6}$$

The function f has bounded support, we can take $A = \{x \in \mathbb{R}^2 : |x - x^0| \leq 1/2 + \epsilon\}$ for all $\epsilon \in]0\,;5/2 - \sqrt{2}[$. ■

5.5.2 Paragraph of Text

■ **Example 5.2 — Example name.** Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

■

5.6 Exercises

This is an example of an exercise.

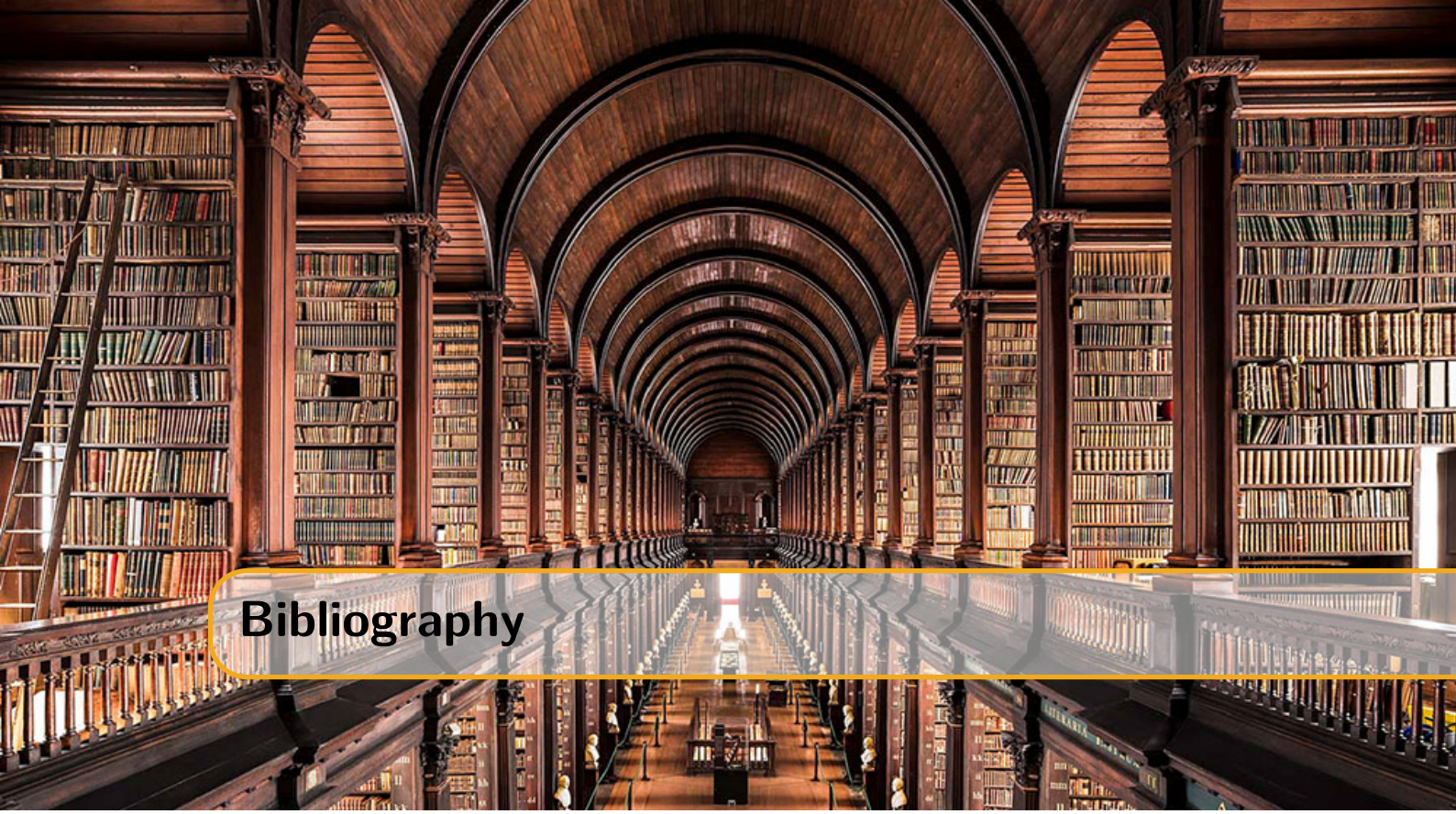
Exercise 5.1 This is a good place to ask a question to test learning progress or further cement ideas into students' minds. ■

5.7 Problems

Problem 5.1 What is the average airspeed velocity of an unladen swallow?

5.8 Vocabulary

Define a word to improve a students' vocabulary.
Vocabulary 5.1 — Word. Definition of word.



Bibliography

[1] D Schultz. "IceProd 2: A Next Generation Data Analysis Framework for the IceCube Neutrino Observatory". In: *Journal of Physics: Conference Series* 664.6 (2015), page 062056. URL: http://stacks.iop.org/1742-6596/664/i=6/a=062056 (cited on page 13).

[2] Thomas K. Gaisser, Todor Stanev, and Serap Tilav. "Cosmic Ray Energy Spectrum from Measurements of Air Showers". In: *Front. Phys.(Beijing)* 8 (2013), pages 748–758. DOI: 10.1007/s11467-013-0319-7. arXiv: 1303.3565 [astro-ph.HE] (cited on pages 14, 16).

[3] Giuseppe Battistoni et al. "Overview of the FLUKA code". In: *Annals Nucl. Energy* 82 (2015), pages 10–18. DOI: 10.1016/j.anucene.2014.11.007 (cited on page 14).

[4] Ralph Engel et al. "Towards the next generation of CORSIKA: A framework for the simulation of particle cascades in astroparticle physics". In: (2018). arXiv: 1808.08226 [astro-ph.IM] (cited on page 14).

[5] Askhat Gazizov and Marek P. Kowalski. "ANIS: High energy neutrino generator for neutrino telescopes". In: *Comput. Phys. Commun.* 172 (2005), pages 203–213. DOI: 10.1016/j.cpc.2005.03.113. arXiv: astro-ph/0406439 [astro-ph] (cited on page 14).

[6] Morihiro Honda et al. "Calculation of atmospheric neutrino flux using the interaction model calibrated with atmospheric muon data". In: *Phys. Rev.* D75 (2007), page 043006. DOI: 10.1103/PhysRevD.75.043006. arXiv: astro-ph/0611418 [astro-ph] (cited on pages 14–16).

[7] Rikard Enberg, Mary Hall Reno, and Ina Sarcevic. "Prompt neutrino fluxes from atmospheric charm". In: *Phys. Rev.* D78 (2008), page 043005. DOI: 10.1103/PhysRevD.78.043005. arXiv: 0806.0418 [hep-ph] (cited on pages 15, 16).

[8] M. G. Aartsen et al. "Observation of High-Energy Astrophysical Neutrinos in Three Years of IceCube Data". In: *Phys. Rev. Lett.* 113 (2014), page 101101. DOI: 10.1103/PhysRevLett.113.101101. arXiv: 1405.5303 [astro-ph.HE] (cited on pages 15, 16).

[9] C. Andreopoulos et al. "The GENIE Neutrino Monte Carlo Generator". In: *Nucl. Instrum. Meth.* A614 (2010), pages 87–104. DOI: 10.1016/j.nima.2009.12.009. arXiv: 0905.2517 [hep-ph] (cited on page 15).

[10] Costas Andreopoulos et al. "The GENIE Neutrino Monte Carlo Generator: Physics and User Manual". In: (2015). arXiv: 1510.05494 [hep-ph] (cited on page 15).

[11] M. Honda et al. “Atmospheric neutrino flux calculation using the NRLMSISE-00 atmospheric model”. In: *Phys. Rev.* D92.2 (2015), page 023004. DOI: 10.1103/PhysRevD.92.023004. arXiv: 1502.03916 [astro-ph.HE] (cited on pages 15, 16).

[12] Joerg R. Hoerandel. “On the knee in the energy spectrum of cosmic rays”. In: *Astropart. Phys.* 19 (2003), pages 193–220. DOI: 10.1016/S0927-6505(02)00198-6. arXiv: astro-ph/0210453 [astro-ph] (cited on page 16).

[13] Thomas K. Gaisser. “Spectrum of cosmic-ray nucleons, kaon production, and the atmospheric muon charge ratio”. In: *Astropart. Phys.* 35 (2012), pages 801–806. DOI: 10.1016/j.astropartphys.2012.02.010. arXiv: 1111.6675 [astro-ph.HE] (cited on page 16).

[14] G. D. Barr et al. “Three-dimensional calculation of atmospheric neutrinos”. In: *Phys. Rev. D* 70 (2 July 2004), page 023006. DOI: 10.1103/PhysRevD.70.023006. URL: https://link.aps.org/doi/10.1103/PhysRevD.70.023006 (cited on page 16).

[15] Mario Dunsch et al. “Recent Improvements for the Lepton Propagator PROPOSAL”. In: (2018). arXiv: 1809.07740 [hep-ph] (cited on page 18).

[16] Dmitry Chirkin. “Study of South Pole ice transparency with IceCube flashers”. In: *Proceedings, 32nd International Cosmic Ray Conference (ICRC 2011): Beijing, China, August 11-18, 2011*. Volume 4, page 161. DOI: 10.7529/ICRC2011/V04/0333 (cited on page 19).

[17] Johan Lundberg et al. “Light tracking for glaciers and oceans: Scattering and absorption in heterogeneous media with Photonics”. In: *Nucl. Instrum. Meth.* A581 (2007), pages 619–631. DOI: 10.1016/j.nima.2007.07.143. arXiv: astro-ph/0702108 [ASTRO-PH] (cited on page 19).

[18] R. Abbasi et al. “Calibration and Characterization of the IceCube Photomultiplier Tube”. In: *Nucl. Instrum. Meth.* A618 (2010), pages 139–152. DOI: 10.1016/j.nima.2010.03.102. arXiv: 1002.2442 [astro-ph.IM] (cited on page 20).

[19] K. J. Ma et al. “Time and Amplitude of Afterpulse Measured with a Large Size Photomultiplier Tube”. In: *Nucl. Instrum. Meth.* A629 (2011), pages 93–100. DOI: 10.1016/j.nima.2010.11.095. arXiv: 0911.5336 [physics.ins-det] (cited on page 20).

[20] J. Ahrens et al. “Muon track reconstruction and data selection techniques in AMANDA”. In: *Nucl. Instrum. Meth.* A524 (2004), pages 169–194. DOI: 10.1016/j.nima.2004.01.065. arXiv: astro-ph/0407044 [astro-ph] (cited on pages 22–24).

[21] M. G. Aartsen et al. “Improvement in Fast Particle Track Reconstruction with Robust Statistics”. In: *Nucl. Instrum. Meth.* A736 (2014), pages 143–149. DOI: 10.1016/j.nima.2013.10.074. arXiv: 1308.5501 [astro-ph.IM] (cited on page 23).

[22] IceCube collaboration. URL: https://internal-apps.icecube.wisc.edu/reports/data/icecube/2014/04/001/icecube_201404001_v1.pdf (cited on page 25).

[23] Stef Verpoest. URL: https://lib.ugent.be/fulltxt/RUG01/002/479/620/RUG01-002479620_2018_0001_AC.pdf (cited on page 25).

[24] Till Neunhoffer. “Estimating the angular resolution of tracks in neutrino telescopes based on a likelihood analysis”. In: *Astropart. Phys.* 25 (2006), pages 220–225. DOI: 10.1016/j.astropartphys.2006.01.002. arXiv: astro-ph/0403367 [astro-ph] (cited on page 26).

[25] URL: https://wiki.icecube.wisc.edu/index.php/SLC_hit_cleaning (cited on page 26).

[26] URL: https://www.dissertations.se/dissertation/fef1f8e4ba/ (cited on pages 27, 28).

[27] Yoav Freund and Robert E Schapire. “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting”. In: *Journal of Computer and System Sciences* 55.1 (1997), pages 119–139. ISSN: 0022-0000. DOI: https://doi.org/10.1006/jcss.1997.1504. URL: http://www.sciencedirect.com/science/article/pii/S002200009791504X (cited on page 31).

[28] S. Fink C. Böser and S. Röcker. *Introduction to boosted decision trees: A multivariate approach to clasification problems*. Presented at the KSETA Doctoral Workshop, Berlin, Germany. July 2014 (cited on page 32).

Index

V