Westfälische Wilhelms-Universität Münster
Institut für Kernphysik

Simulation kosmischer Myonen und die Slow-Control-Datenbank für den High-Precision-Tracker des OPERA-Experiments

Diplomarbeit
vorgelegt von
Martin Hierholzer

02.01.2007

Abstract

The OPERA experiment searches for the appearance of ν_τ in the ν_μ neutrino beam produced at CERN. It presents the first long baseline appearance experiment to look for the $\nu_\mu \leftrightarrow \nu_\tau$ neutrino oscillation. The experiment is located in hall C of the LNGS underground laboratory. This diploma thesis describes a simulation of cosmic muons for the calibration of the precision tracker of the OPERA experiment and for the trigger of the precision tracker, as well as the slow control database of the precision tracker.

Cosmic muons are a convenient source for the calibration of the precision tracker. For this purpose muons, which cross a full double plane of the precision tracker while causing a trigger signal, are needed. As a result of the simulation, both the space drifttime relation and the alignment of the precision tracker modules can be calibrated within a reasonable time, as the rate of cosmic muons fulfilling the above mentioned conditions is high enough with about 650 events per day.

Furthermore, it has to be checked, if cosmic muons present a background for the measurement of beam neutrinos. Due to the shape of the shielding, given by the Gran Sasso rock, there are no muons hitting the detector in the same direction as the beam neutrino related muons. Therefore, beam events and cosmic muon events can be distinguished by a simple angular cut. A comparison of the simulated angular distributions with real data is not yet possible, as the alignment of the precision tracker has not been measured so far.

The slow control database records all necessary environmetal parameters, error messages and set points of the precision tracker. The necessary software has been written and tested as part of this diploma thesis. It will operate during the entire measuring period, and is presently already operational for testing purposes. The maximum size of the database after five years of measurement is estimated to be about 70 GByte and does not present a technical problem therefore. Further software has been developed, e.g. for displaying the measured data graphically. All software has been designed to allow future changes if required.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

1.1 Einleitung

Das OPERA-Experiment[1] ist ein Experiment zur Messung von Neutrino-Oszillationen, insbesondere der Oszillation $\nu_\mu \leftrightarrow \nu_\tau$. Es ist das erste *long baseline appearance experiment* [Crn00]. In den 1990er Jahren gab es bereits ein ähnliches *short baseline appearance experiment*, das CHORUS-Experiment. Dieses wurde ebenfalls zur Messung der Oszillation $\nu_\mu \leftrightarrow \nu_\tau$ gebaut [Esk98]. Bei CHORUS betrug die Oszillationsstrecke (*baseline*) 625 m, während die Oszillationsstrecke bei OPERA mit 732 km um gut drei Größenordnungen länger ist. CHORUS hat keine Oszillation nachgewiesen. Sein Ergebnis ist konsistent mit den kurze Zeit später gemessenen Oszillationsparametern (siehe Kapitel 2.1), welche für CHORUS ein positives Signal ausschlossen.

Das OPERA-Experiment wurde bereits 1997 vorgeschlagen und befindet sich derzeit im Aufbau. Die Datennahme hat Ende des Jahres 2006 bereits begonnen. In einer ersten Phase werden nötige Kalibrationen durchgeführt. Das eigentliche Target wird voraussichtlich Anfang 2007 installiert werden.

In dieser Einführung werden einige wichtige physikalische Grundlagen für das OPERA-Experiment und in Kapitel 2 das eigentliche Experiment erklärt. In Kapitel 3 wird eine Simulation der kosmischen Myonen beschrieben, die im Rahmen dieser Diplomarbeit für das OPERA-Experiment durchgeführt wurde. Die Myonen sind für die Messung der exakten Position der Detektorkomponenten (*alignment*) wichtig. Gleichzeitig stellen die Myonen aber auch einen Untergrund für die gesamte Messung dar und können insbesondere die Neutrinostrahlausrichtung auf den Detektor (*beam finding*) stören. Daher ist eine genaue Kenntnis der Winkel- und Energieverteilungen der Myonen und der Rate des durch die Myonen

[1] OPERA: **O**scillating **P**roject with **E**mulsion t**R**acking **A**pparatus

ausgelösten *Triggers*[2] im Voraus nötig. Mit dem Ergebnis der Simulation kann die für den Detektorbetrieb und insbesondere die für die Inbetriebnahme des Detektors nötige Software getestet und angepasst werden.

Weiterhin wird die *slow control database* des Detektors zur Spurrekonstruktion der Myonen im Myonspektrometer, dem sogenannten *precision tracker*, beschrieben. Die Datenbank wurde im Rahmen dieser Diplomarbeit entwickelt. Während der gesamten Messzeit werden alle benötigten Umgebungsparameter, wie zum Beispiel Temperatur und Gasdruck, und Fehlermeldungen aufgezeichnet und in die Datenbank geschrieben. Diese Daten sind für eine Auswertung der eigentlichen Messung wichtig, da sie Informationen über die Nachweiswahrscheinlichkeit des Detektors geben, auf einen teilweisen oder vollständigen Ausfall des *precision tracker*s hindeuten und bei der Fehlersuche helfen können. Der Aufbau und die Funktionsweise dieser Datenbank wird in Kapitel 4 beschrieben.

1.2 Neutrinos im Standardmodell

Das Neutrino (im Folgenden meist mit ν abgekürzt) ist im vereinfachten Standardmodell ein masseloses, ungeladenes Lepton. Es gibt drei Familien, die sogenannten *Flavours*[3], genau wie im Bereich der geladenen Leptonen:

$$\begin{pmatrix} \nu_e \\ e \end{pmatrix} \begin{pmatrix} \nu_\mu \\ \mu \end{pmatrix} \begin{pmatrix} \nu_\tau \\ \tau \end{pmatrix}$$

Jedem Leptonenflavour wird eine additive Quantenzahl n_e, n_μ bzw. n_τ zugeordnet, die in allen Wechselwirkungen erhalten ist. Bei der Zerfallsreaktion des μ^+ beispielsweise ($\mu^+ \longmapsto e^+ + \bar{\nu}_\mu + \nu_e$) ergibt sich sowohl für den Eingangs- als auch für den Ausgangszustand: $n_\mu = -1$ und $n_e = 0$.

Da die Leptonen an der starken Wechselwirkung nicht teilnehmen und die Neutrinos ungeladen sind, können Neutrinos nur schwach wechselwirken. Die Gravitation, der die Neutrinos natürlich auch unterliegen, spielt in der Regel in der Teilchenphysik keine Rolle.

[2] Trigger, engl. Auslöser. Der Begriff wird in der Hochenergiephysik sowohl für die Schaltung als auch für die zugehörigen Detektoren verwendet, die den Zeitpunkt eines Ereignisses feststellen und weitere Messungen initiieren.

[3] Flavour, engl. Geschmack, in der Teilchenphysik meist eher im Sinne von Leptonengeneration gebraucht, insbesondere zur Unterscheidung von Quarkgeneration

1.3 Historischer Überblick

1.3.1 Entdeckung des Neutrinos

Neutrinos wurden 1930 von Wolfgang Pauli zur Erklärung des kontinuierlichen Spektrums des β-Zerfalls in einem öffentlichen Brief postuliert [Pau61]. Bei einem Zwei-Körper-Zerfall müsste aufgrund der konstanten Massendifferenz zwischen dem Eingangs- und dem Ausgangszustand das Spektrum monoenergetisch sein. Ein zusätzliches, ungeladenes Zerfallsprodukt kann einen Teil der Energie übernehmen und somit ein kontinuierliches Energiespektrum des Elektrons hervorrufen. Der experimentelle Nachweis des Neutrinos gelang C. L. Cowan und F. Reines erst 1956 durch den inversen Beta-Zerfall ($\overline{\nu}_e + p \longmapsto e^+ + n$) in einem Reaktorexperiment [Cow56]. Zum damaligen Zeitpunkt war noch nicht bekannt, dass es verschiedene Neutrinosorten gibt.

1.3.2 Nachweis der Neutrinoflavours

Der erste Nachweis eines anderen Neutrinoflavours gelang 1963 J. Steinberger, M. Schwartz und L. Ledermann im gleichzeitig ersten Experiment mit einem am Beschleuniger hergestellten Neutrinostrahl [Led63]. In diesem Experiment wurde nachgewiesen, dass die aus dem Pion-Zerfall erzeugten Neutrinos ($\pi^{\pm} \longmapsto \mu^{\pm} + \overset{(-)}{\nu}_{\mu}$) bei einer Reaktion mit Materie überwiegend Myonen produzieren. Aufgrund dieses Experiments führte man zwei neue Quantenzahlen, die flavourunterscheidenden Leptonenzahlen, und deren Erhaltung unter allen Wechselwirkungen ein. Nach der Entdeckung des τ-Leptons (1978 durch M. L. Perl am SLAC) [Per78] und des zugehörigen τ-Neutrinos (2001 durch K. Kodama am DONUT-Experiment) [Kod01] wurde dieses Gesetz um das dritte Flavour erweitert.

1.3.3 Das Neutrinodefizit

In den 1970er Jahren wurde mit dem Homestake-Experiment der solare Neutrinofluss erstmals gemessen [Bah76]. Durch den pp-Zyklus, welcher der in der Sonne dominante Kernfusionsprozess ist, und die Nebenzyklen entsteht das in Abbildung 1.1 dargestellte Neutrinospektrum. In einem mit Tetrachlorethen gefüllten Tank wurde ^{37}Ar durch den neutrinoinduzierten β-Zerfall des ^{37}Cl-Isotops erzeugt: $\nu_e + {}^{37}Cl \longmapsto e^- + {}^{37}Ar$. Der Nachweis erfolgte dann später (*offline*) durch den Zerfall des ^{37}Ar. Aus Sonnenmodellen (und zum Beispiel Messungen der Photonenluminosität der Sonne) war ein theoretischer Wert für die Intensität

der solaren Neutrinos mit hinreichender Genauigkeit berechnet worden. Durch den Q-Wert der obigen Chlor-Reaktion ergibt sich eine Nachweisschwelle von $E_{\text{th}} = 0.814\,\text{MeV}$. Daher konnte das Experiment im Wesentlichen nur die Neutrinos des ^8B-Nebenzyklusses nachweisen. Trotz Berücksichtigung dieses Effekts konnten nur zirka 33% der erwarteten Neutrinos nachgewiesen werden.

Neben solaren Neutrinos wurden auch atmosphärische Neutrinos untersucht. Atmosphärische Neutrinos werden durch den Zerfall von Pionen und Kaonen produziert, die durch kosmische Strahlung erzeugt werden (Abbildung 1.2). Auch bei diesen Neutrinos wurde ein Defizit durch verschiedene Experimente (wie zum Beispiel Kamiokande in Japan [Hir88]) seit 1980 festgestellt. Hier wurde das Ratenverhältnis der Myonneutrinos zu den Elektronneutrinos gemessen, das um einen Faktor 0.6 unter dem aus Monte-Carlo-Simulationen vorhergesagten Wert lag (z.B. [Fuk98]).

Zur Erklärung des Defizits bei solaren und atmosphärischen Neutrinos wurde die bereits 1958 von Pontecorvo diskutierte Mischung der Neutrinozustände herangezogen, analog der damals schon bekannten Mischung der $K^0/\overline{K}^0$-Zustände [Pon58]. Anders als Pontecorvo, der die mögliche Mischung von Neutrino und Antineutrino beschrieben hatte, ging man nun von einer Mischung der Flavourzustände aus. Unter bestimmten Voraussetzungen (siehe Abschnitt 2.1) kann eine Mischung der Flavourzustände dazu führen, dass sich die Neutrinoflavours ineinander umwandeln.

Alle Experimente (z.B. Kamiokande [Hir88], IMB [Bec95], Soudan 2 [Goo95] und SuperKamiokande [Fuk01]), die den Fluss atmosphärischer oder solarer Neutrinos gemessen haben, sind kompatibel mit der Theorie der Neutrinooszillation. Durch eine Analyse der Ereignisse des SuperKamiokande-Experiments mit besonders guter $\frac{L}{E}$-Auflösung (L: Neutrinoflugstrecke, E: Neutrinoenergie) gelang es im April 2004, andere, bis dahin noch mögliche Modelle, wie den Zerfall oder die Dekohärenz der Neutrinos, auszuschließen [Ash04]. Damit wurde die Oszillation als Erklärung für das Neutrinodefizit etabliert.

Der Nachweis der Neutrinooszillation erfolgte bislang nur mit *disappearance* Experimenten. Deshalb besteht der Bedarf nach einem *appearance* Experiment, das das Neutrinoflavour identifizieren kann, zu dem die Neutrinos oszillieren. Im Falle atmosphärischer und Beschleunigerneutrinos sind dies in erster Linie τ-Neutrinos aufgrund der Oszillationsparameter und der Flugstrecken (Abschnitt 2.1). Insbesondere konnten alle bisherigen Experimente τ-Neutrinos nicht nachweisen, da die Experimente keine dafür ausreichende Ortsauflösung aufwiesen oder die Energie für die Erzeugung des τ-Leptons zu gering war.

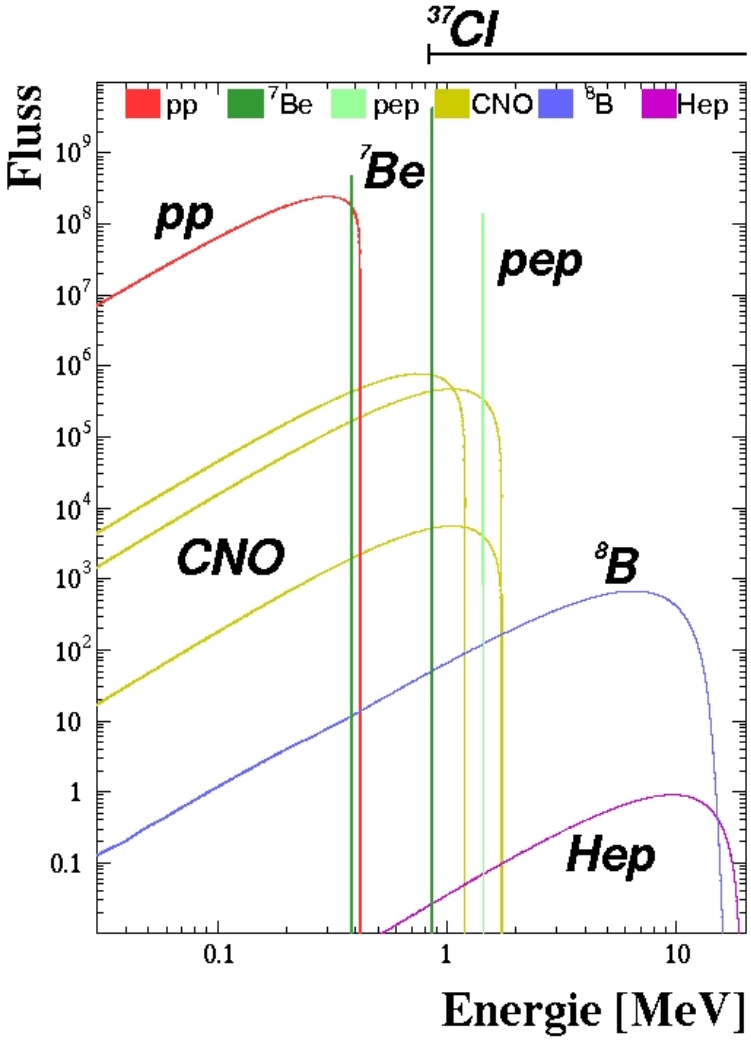


Abbildung 1.1: Spektrum der solaren Neutrinos. Im oberen Bereich ist der Sensitivitätsbereich für die Nachweisreaktion des Homestake-Experiments eingezeichnet (beschriftet mit ^{37}Cl). Durch diese Nachweisschwelle ist das Experiment in erster Linie auf die ^{8}B-Neutrinos sensitiv. Neutrinos aus dem CNO-Zyklus, dem pep-Zyklus ($p + e^- + p \longmapsto d + \nu_e$) und Hep-Zyklus (Fortsetzung des pp-Zyklus, $^3He + p \longmapsto ^4He + e^+ + \nu_e$) spielen wegen der geringen Nachweiswahrscheinlichkeit knapp oberhalb der Reaktionsschwelle oder wegen ihrer geringen Rate eine untergeordnete Rolle [Rol88]. [Www02]

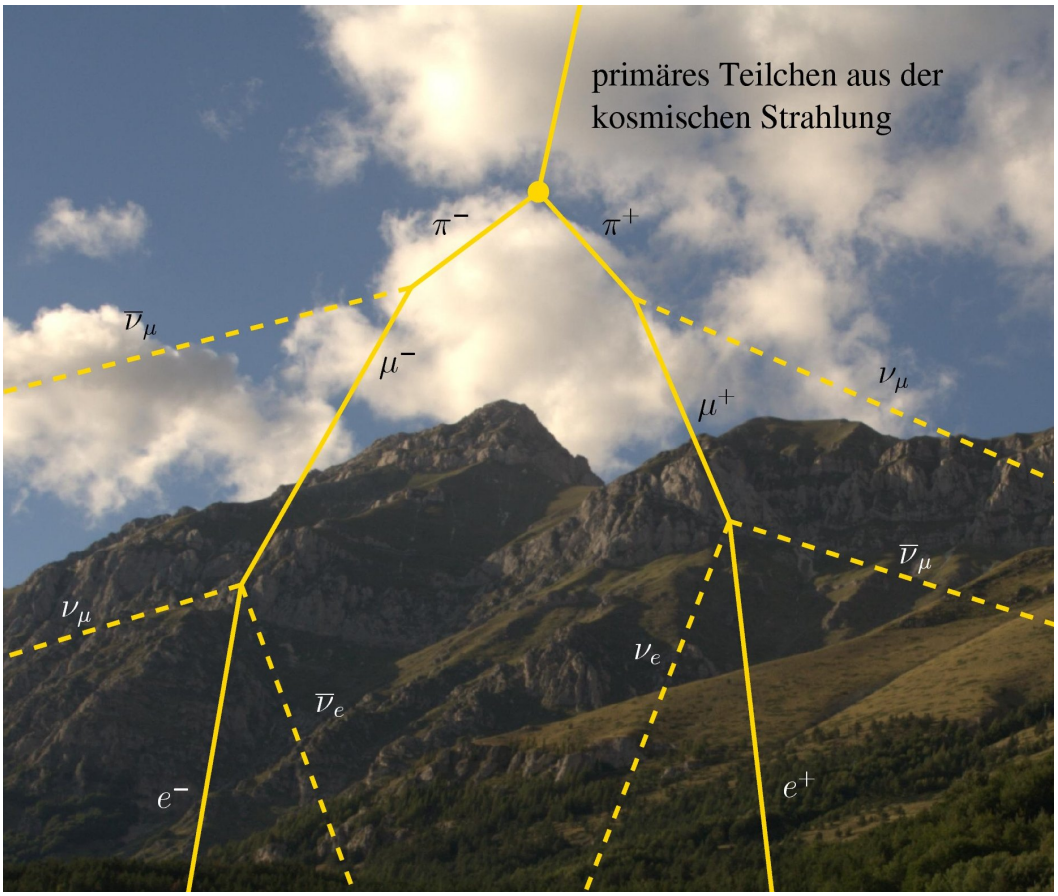


Abbildung 1.2: Entstehung atmosphärischer Neutrinos. Beim Aufprall eines hochenergetischen Teilchens aus der kosmischen Strahlung werden Teilchen erzeugt, die über verschiedene Zerfallsketten in Pionen oder Kaonen zerfallen. Diese wiederum zerfallen in der oberen Atmosphäre in je ein μ^- (im Falle eines π^- bzw. K^-) und ein $\overline{\nu}_\mu$. Die μ^- zerfallen wiederum in je ein e^-, ein ν_μ und ein $\overline{\nu}_e$. Analog ist der Prozess natürlich mit Teilchen und Antiteilchen vertauscht möglich, wenn statt eines π^-/K^- ein π^+/K^+ produziert wird. Somit ergibt sich ein Entstehungsverhältnis $\overset{(-)}{\nu}_\mu$ zu $\overset{(-)}{\nu}_e$ von 2 : 1. Eine genauere Abschätzung, die den Einfluss des erdmagnetischen Feldes und das Auftreffen einiger Myonen auf der Erdoberfläche vor ihrem Zerfall berücksichtigt, ist mittels einer Monte-Carlo-Simulation möglich.

Kapitel 2

Das OPERA-Experiment

2.1 Neutrinooszillation

Für die Flavouroszillation der Neutrinos gibt es zwei Voraussetzungen: die Flavoureigenzustände der Neutrinos $|\nu_\alpha\rangle$ ($\alpha = e, \mu, \tau$) sind nicht identisch mit den Masseneigenzuständen $|\nu_i\rangle$ ($i = 1,2,3$):

$$|\nu_\alpha\rangle = \sum_{i=1}^{3} U_{\alpha i} |\nu_i\rangle \,, \tag{2.1}$$

und die Neutrinosorten haben unterschiedliche Massen. Insbesondere können nicht alle Neutrinos masselos sein.

Ähnlich der Cabibbo-Kobayashi-Maskawa-Matrix im Quarksektor wird auch hier der Zusammenhang zwischen den Eigensystemen durch eine 3×3-Mischungsmatrix U beschrieben, die üblicherweise drei Mischungswinkel Θ_1, Θ_2 und Θ_3 und eine CP-verletzende Dirac-Phase[1] δ als freie Parameter aufweist. Die CP-verletzende Phase wäre zum Beispiel für einen Unterschied zwischen den Wahrscheinlichkeiten der Übergänge $\nu_\mu \to \nu_\tau$ und $\nu_\tau \to \nu_\mu$ verantwortlich. Da OPERA nur den Übergang $\nu_\mu \to \nu_\tau$ misst, ist das Experiment auf diese Phase nicht sensitiv.

Die Mischungsmatrix kann aus drei Drehmatrizen mit jeweils einer Drehung um nur einen Winkel (vgl. Eulerwinkel) und einer Phasenmatrix zusammengesetzt werden:

$$U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_2 & s_2 \\ 0 & -s_2 & c_2 \end{pmatrix} \cdot \begin{pmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & e^{i\delta} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_3 & s_3 \\ 0 & -s_3 & c_3 \end{pmatrix}$$

[1] CP steht für die hintereinander ausgeführten Transformationen der Ladungskonjugation (**C**, engl. *charge conjugation*) und der Parität (**P**, engl. *parity*).

$$= \begin{pmatrix} c_1 & s_1c_3 & s_1s_3 \\ -s_1c_2 & c_1c_2c_3 - s_2s_3e^{i\delta} & c_1c_2s_3 + s_2c_3e^{i\delta} \\ s_1s_2 & -c_1s_2c_3 - c_2s_3e^{i\delta} & -c_1s_2s_3 + c_2c_3e^{i\delta} \end{pmatrix}$$

mit $s_i = \sin\Theta_i$ und $c_i = \cos\Theta_i$ [Sch97]. Die Aufteilung in drei Drehmatrizen und eine Phasenmatrix ist sinnvoll, da so näherungsweise die Probleme getrennt betrachtet werden können, zum Beispiel als 2-Flavour-Oszillation.

Um die Wahrscheinlichkeit für den Übergang $\nu_\mu \longmapsto \nu_\tau$

$$P_{\nu_\mu \longmapsto \nu_\tau} = \left|\langle \nu_\tau | \hat{p}_f | \nu_\mu \rangle\right|^2$$

zu berechnen, benötigt man den Propagationsoperator im Flavourbild

$$\hat{p}_\mathrm{f} = U^{-1} \cdot \hat{p} \cdot U,$$

wobei der Propagationsoperator $\hat{p}$ definiert ist als:

$$\hat{p} = \begin{pmatrix} e^{-iE_1t} & 0 & 0 \\ 0 & e^{-iE_2t} & 0 \\ 0 & 0 & e^{-iE_3t} \end{pmatrix}.$$

E_i sind hierbei die Energien der Eigenzustände $|\nu_i\rangle$. Mit t ist die Propagationszeit und (in den im Folgenden verwendeten natürlichen Einheiten) gleichzeitig die Oszillationsstrecke bezeichnet.

Da es sich bei den Neutrinos um hochrelativistische Teilchen handelt, ist folgende Näherung für E_i möglich:

$$E_i = \sqrt{p^2 + m_i^2} \approx p + \frac{m_i^2}{2p} \approx E + \frac{m_i^2}{2E},$$

wobei p der Impuls (nicht zu verwechseln mit dem Propagationsoperator $\hat{p}$) und E die kinetische Energie des Neutrinos ist, und m_i die Massen der Eigenzustände $|\nu_i\rangle$ bezeichnen.

Die Ortsabhängigkeit der Neutrinozustände

$$|\nu_\alpha(x)\rangle = |\nu_\alpha\rangle \cdot e^{ipx}$$

braucht nicht berücksichtigt zu werden, da der Propagationsoperator ortsunabhängig ist:

$$\begin{aligned} \langle \nu_\alpha(x) | \hat{p}_f | \nu_\alpha(x) \rangle &= \langle \nu_\alpha | e^{-ipx} \cdot \hat{p}_f \cdot e^{ipx} | \nu_\alpha \rangle \\ &= \langle \nu_\alpha | \hat{p}_f \cdot e^{-ipx} \cdot e^{ipx} | \nu_\alpha \rangle \\ &= \langle \nu_\alpha | \hat{p}_f | \nu_\alpha \rangle . \end{aligned}$$

Damit ergibt sich für die Übergangswahrscheinlichkeit:

$$\begin{aligned} P_{\nu_\mu \longmapsto \nu_\tau} &= \left| \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \hat{p}_\mathrm{f} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right|^2 \\ &= \left| \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} U \cdot \hat{p} \cdot U^{-1} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right|^2 . \end{aligned} \tag{2.2}$$

Aufgrund der von anderen Experimenten (Abschnitt 1.3.3) gemessenen Differenzen der Massenquadrate $\Delta m_{ij}^2 = m_j^2 - m_i^2$ ist die Oszillationslänge $\lambda_{12} = \frac{4\pi E}{\Delta m_{12}^2}$ (hier in natürlichen Einheiten) für die Oszillation $\nu_\mu \leftrightarrow \nu_e$ groß gegen die Oszillationsstrecke des OPERA-Experiments. Deswegen kann hier zur leichteren Berechenbarkeit der Übergangswahrscheinlichkeit näherungsweise $\Delta m_{12}^2 = 0$ gesetzt werden. Des Weiteren wird die Dirac-Phase ebenfalls zu $\delta = 0$ angenommen. Damit kann die Übergangswahrscheinlichkeit berechnet werden:

$$\begin{aligned} P_{\nu_\mu \longmapsto \nu_\tau} &= \frac{1}{64} \sin^2\left(t \frac{\Delta m_{23}^2}{4E}\right) \\ &\quad \{\sin(2\Theta_2)\,[2 - 2\cos(2\Theta_1) + 6\cos(2\Theta_3) + \cos(2\Theta_1 + 2\Theta_3) + \\ &\quad + \cos(2\Theta_1 - 2\Theta_3)] + 8\cos(\Theta_1)\cos(2\Theta_2)\sin(2\Theta_3)\}^2 . \end{aligned}$$

Alle bisherigen Messungen des Winkels Θ_3 sind kompatibel mit $\Theta_3 = 0$. Die Obergrenze für diesen Winkel liegt bei 13°. Für grobe Abschätzungen kann man deshalb noch als zusätzliche Vereinfachung den Mischungswinkel $\Theta_3 = 0$ setzen. Damit erhält man:

$$\begin{aligned} P_{\nu_\mu \longmapsto \nu_\tau} &= \sin^2\left(t \frac{\Delta m_{23}^2}{4E}\right) \sin^2(2\Theta_2) \\ &= \sin^2\left(\frac{\Delta m_{23}^2}{4} \frac{L}{E}\right) \sin^2(2\Theta_2) \\ &= \sin^2\left(\pi \frac{L}{\lambda_{23}}\right) \sin^2(2\Theta_2), \end{aligned} \tag{2.3}$$

wobei mit L die Oszillationsstrecke und mit $\lambda_{23} = \frac{4\pi E}{\Delta m_{23}^2}$ die Oszillationslänge bezeichnet wurde.

2.2 Ratenverhältnis ν_τ zu ν_μ bei OPERA

Gleichung 2.3 kann genutzt werden, um das Verhältnis der Rate der τ-Neutrinos zur Rate der μ-Neutrinos im Falle des OPERA-Experiments abzuschätzen. Die mittlere Energie des Neutrinostrahls beträgt $E = 17\,\mathrm{GeV}$ und die zurückgelegte Strecke ist mit $L = 732\,\mathrm{km}$ konstant. Aktuelle Werte[2] für die Massen- und Mischungsparameter sind $\Delta m^2_{23} = 2.4 \cdot 10^{-3}\,\mathrm{eV}^2$ und $\Theta_2 = 42°$. Damit folgt für die Oszillationslänge $\lambda_{osz} = 17.7 \cdot 10^6\,\mathrm{m}$. Für die Oszillationswahrscheinlichkeit ergibt sich dann:

$$P_{\nu_\mu \to \nu_\tau} = 0.99 \cdot 0.0168 = 0.0166 = 1.66\%.$$

Das Verhältnis der Raten kann dann folgendermaßen berechnet werden:

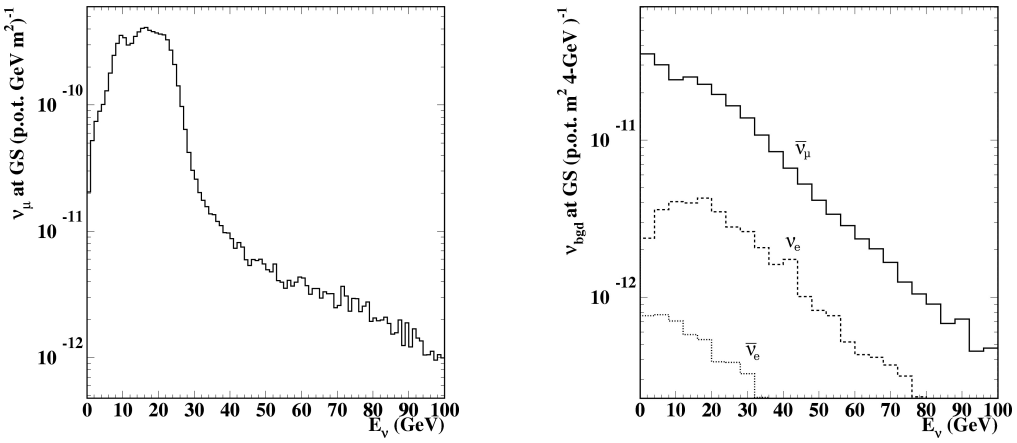
$$\frac{R_{\nu_\tau}}{R_{\nu_\mu}} = \frac{P_{\nu_\mu \to \nu_\tau}}{1 - P_{\nu_\mu \to \nu_\tau}} = 1.69\%.$$

2.3 Experimenteller Aufbau

Das OPERA-Experiment verwendet den CNGS[3] Neutrinostrahl als Myonneutrinoquelle und den OPERA-Detektor zum Nachweis von Tauneutrinos. Der Strahl wird am Protonenbeschleuniger SPS am CERN produziert. Hierbei werden Protonen auf $400\,\mathrm{GeV}$ beschleunigt und auf ein Graphit-Target gelenkt, wo Hadronen, haptsächlich $\pi^\pm$ und $K^\pm$ als metastabile Endzustände, entstehen. Durch ein magnetisches Horn können nahezu ausschließlich negativ geladene Teilchen passieren. Die Hadronen werden fokussiert und zerfallen unter anderem in μ^+ und ν_μ. Durch die nicht vollständige Ladungsselektion und untergeordnete Zerfälle ergibt sich eine Kontamination mit $\overline{\nu}_\mu$ (2.0%), ν_e (0.8%) und $\overline{\nu}_e$ (0.05%) [Crn00]. Eine Kontamination mit $\overset{(-)}{\nu}_\tau$ ist vernachlässigbar gering. Beim CHORUS-Experiment, das den Vorgänger des bei OPERA verwendeten LNGS-Neutrinostrahls verwendete, betrug sie $3\text{-}4 \cdot 10^{-6}$ [Esk97]. Die Spektren der Neutrinoenergien sind in Abbildung 2.1 dargestellt. Die Kontaminationen ergeben einen geringen Untergrund zur Messung der Oszillation $\nu_\mu \longmapsto \nu_\tau$, können aber teilweise gleichzeitig für andere Messungen (siehe Abschnitt 2.3.4) verwendet werden.

[2] Beide Werte erhältlich aus [Fog05]

[3] CNGS: *CERN neutrinos to Gran Sasso*, am CERN bei Genf produzierter Myonneutrinostrahl, auf Gran Sasso in Italien gerichtet



(a) Spektrum der ν_μ (b) Spektrum der Kontaminationen

Abbildung 2.1: Energiespektrum der ν_μ (a) und der Kontaminationen (b) des CNGS-Strahls. In (b) ist das Spektrum aufgeteilt nach $\overline{\nu}_\mu$ (durchgezogene Linie), ν_e (gestrichte Linie) und $\overline{\nu}_e$ (gepunktete Linie). Für die Messung der Oszillation $\nu_\mu \rightarrow \nu_\tau$ werden die Neutrinos im unteren Energiebereich ($\lesssim 40\,\mathrm{GeV}$) verwendet. Bei höheren Energien können möglicherweise Messungen anderer Oszillationen mit Hilfe der Strahlkontaminationen durchgeführt werden. [Crn00]

Der Detektor steht 732 km entfernt vom CERN im Untergrundlabor des LNGS[4]. Als Abschirmung gegen kosmische Myonen dienen richtungsabhängig durchschnittlich 1400 m Fels (zirka 3500 mwe[5]) des Gran Sasso Massivs in Italien.

2.3.1 Übersicht über den Detektor

Eine Übersicht über den Detektor befindet sich in Abbildung 2.2. Der Detektor besteht aus zwei identischen Baugruppen, den sogenannten Supermodulen. Jedes dieser Supermodule besteht aus einem *Neutrino-Target* mit integriertem *Target-Tracker* sowie einem Myonspektrometer. Die Teilung in zwei Supermodule verkürzt den Weg der Myonen im Target-Material zum Spektrometer und verringert damit ihren Energieverlust durch Vielfachstreuung. Dadurch kann eine höhere Genauigkeit und eine höhere Effizienz bei der Messung der Myonenergie erreicht werden. Im Folgenden wird eine Übersicht über die Detektorkomponenten gegeben. Eine detaillierte Beschreibung erfolgt in den Abschnitten 2.3.2 und 2.3.3.

[4] *Laboratori Nazionali del Gran Sasso*, Untergrundlabor des INFN im Gran Sasso
[5] Die Einheit mwe steht für *meter water equivalent*

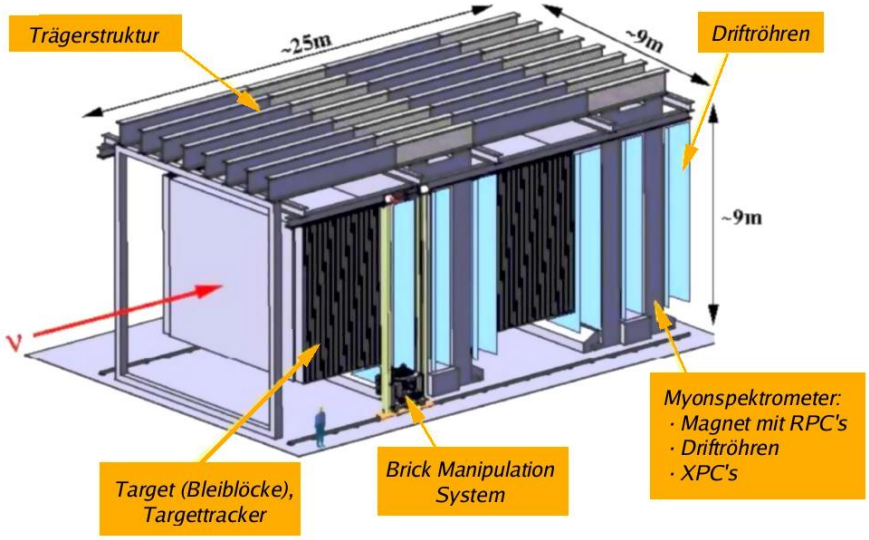


Abbildung 2.2: Schematische Darstellung des OPERA-Detektors [Www01]

Zum Nachweis der Tauneutrinos wird die *charged current* Reaktion[6] $\nu_\tau + N \longmapsto \tau^- + X$ verwendet. N bezeichnet dabei einen Kern aus dem Target-Material und X einen hadronischen Endzustand. Die Lebensdauer des τ-Leptons beträgt $t = (290.6 \pm 1.0) \cdot 10^{-15}\,\mathrm{s}$, und das τ-Lepton hat eine Masse von $1776.99^{+0.29}_{-0.26}\,\mathrm{GeV}$ [Pdg06]. Da es von einem Neutrinostrahl mit 17 GeV Energie erzeugt wird, hat es eine relativistische Geschwindigkeit und zerfällt somit im Mittel erst nach 0.7 mm ($ct = 87\,\mu\mathrm{m}$). Für das Experiment gibt es dann folgende relevante Zerfallskanäle:

$$\begin{aligned} \tau^- &\longmapsto e^- + \nu_\tau + \overline{\nu}_e \\ \tau^- &\longmapsto \mu^- + \nu_\tau + \overline{\nu}_\mu \\ \tau^- &\longmapsto h^- + \nu_\tau + (\geq 0\,\text{neutrale Teilchen}) \end{aligned}$$

Hierbei bezeichnet h^- ein beliebiges negativ geladenes Hadron. Weitere Zerfallskanäle, bei denen mehrere geladene Teilchen beteiligt sind, werden beim OPERA-Experiment nicht berücksichtigt, da diese Kanäle keine ausreichende Unterscheidung zu Untergrundereignissen ermöglichen.

Die Signatur des τ^--Zerfalls in unter anderem ein μ^- ist der Knick zwischen der τ^-- und der μ^--Bahn. Bei den beiden anderen Zerfallskanälen löst das e^- bzw. h^- Teilchenschauer aus. In diesen beiden Fällen muss der Ursprungspunkt eines

[6] Reaktion, die über den geladenen Strom stattfindet, d.h. durch Austausch eines $W^\pm$-Bosons.

Schauers mit dem Ende der τ-Bahn zusammenfallen. Da das Ende der τ-Bahn oft nicht exakt bestimmt werden kann, werden in diesen Fällen nur solche Ereignisse verwendet, bei denen eine hinreichend sichere Identifizierung möglich ist.

Durch die Beschränkung auf Ereignisse mit einer klaren Signatur ist eine gute Unterscheidung von Untergrundereignissen zum Beispiel aus Myonneutrinos möglich. Ein möglicher Untergrund ist zum Beispiel der Zerfall von Charm-Mesonen, die durch Myonneutrinos aus dem Strahl erzeugt werden und in machen Fällen als τ-Leptonen identifiziert werden könnten. Alle Untergrundquellen zusammen ergeben jedoch über die gesamte Messzeit weniger als ein Untergrundereignis [Dra04]. Das Verfahren zum Nachweis der Tauneutrinos wurde bereits beim DONUT-Experiment erfolgreich eingesetzt (Abschnitt 1.3.2, [Kod01]).

2.3.2 Target und Target-Tracker

Das Target des OPERA-Experiments besteht aus insgesamt zirka 13 Millionen $12.2\,\mathrm{cm} \times 12.7\,\mathrm{cm}$ großen und 1 mm dicken Bleiplatten. Jeweils 56 Bleiplatten sind in einer „Sandwich-Struktur" abwechselnd mit 56 Fotoemulsionsfilmen in einem *emulsion cloud chamber*[7] *brick*[8] (kurz: ECC-Brick) angeordnet. Eine zusätzliche 57. Fotoemulsion in jedem Brick ist vom 56. Film durch eine 2 mm dicke Kunststoffplatte getrennt. Mit Hilfe dieses Films wird eine Identifizierung einer τ-Interaktion in der 56. Bleiplatte ermöglicht.

Die ECC-Technik wurde bereits beim CHORUS-Experiment verwendet [Aok00]. Beim Durchgang eines ionisierenden Teilchens durch eine Emulsionsschicht wird diese entlang der Trajektorie lokal geschwärzt. Da die Filme auf beiden Seiten ihres Trägermaterials eine Fotoemulsionsschicht haben, kann eine Teilchenbahn (zum Beispiel von einem zerfallenden τ-Lepton) in drei Dimensionen rekonstruiert werden. Dazu werden die Spuren innerhalb einer Emulsionsschicht, die sogenannten *micro tracks*, zu einer vollständigen Spur verbunden (Abbildung 2.3). Da die Schicht eine gewisse Dicke hat, kann innerhalb der Schicht die Bahn des Teilchens bestimmt werden. Durch Verbinden mehrerer *micro tracks* wird die Winkelauflösung erhöht, wodurch die Signatur des τ-Zerfalls messbar wird.

Die Bricks liegen in Wänden angeordnet auf einer Trägerstruktur. Pro Wand gibt es 3328 Bricks. Zwischen den Wänden befinden sich die *Target-Tracker* (im Folgenden mit TT abgekürzt). Die TT dienen dazu, Ereignisse zu detektieren und zu lokalisieren, und so den vom Ereignis betroffenen Brick zu identifizieren. Die Detektion und Lokalisation der Ereignisse erfolgt mit Hilfe des

[7] *emulsion cloud chamber*, engl. Emulsions-Nebelkammer
[8] *brick*, engl. Backstein, Klotz

hadronischen Schauers, der durch das Ereignis ausgelöst wird. Zusätzlich können die TT benutzt werden, um elektro-magnetische Schauer, wie sie zum Beispiel beim elektronischen Zerfallskanal des τ-Leptons oder bei der Interaktion eines Elektronneutrinos entstehen, oder niederenergetische Myonen aus quasielastischer Streuung, die das Myonspektrometer nicht erreichen, zu erkennen.

Jeder TT besteht aus zirka 6.7 m langen und 2.6 cm breiten Plastikszintillatorstreifen. Um eine 2-dimensionale Information zu erhalten, werden eine Lage mit senkrechten und eine mit waagerechten Streifen hintereinander angeordnet. Die Streifen werden einzeln über Glasfasern mit PMT-Arrays[9] zur Auslese verbunden.

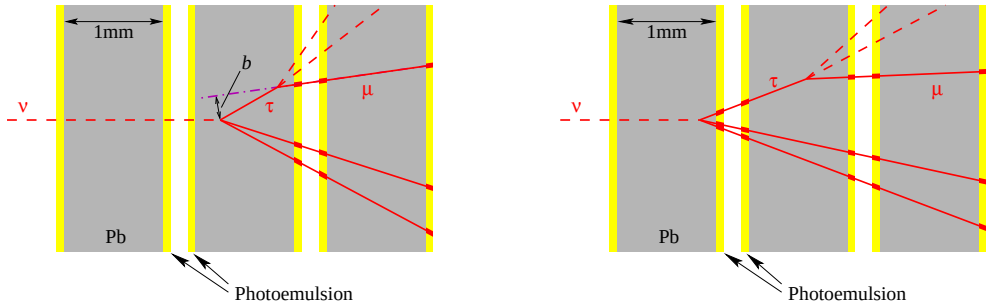
Damit die Bricks zur Analyse extrahiert werden können, sind sie in horizontalen Reihen angeordnet, in denen sie verschoben werden können. So kann das *Brick Manipulation System* (im Folgenden als BMS abgekürzt) einen einzelnen Brick aus dem Target extrahieren. Das BMS besitzt einen Vakuumsauger, der einen Brick auf eine Ablage außerhalb des Targets ziehen kann. Da so immer nur der äußerste Brick einer Reihe extrahiert werden kann, muss das BMS bei einem Ereignis im Inneren des Targets zuerst alle Bricks einer Reihe bis zum betroffenen Brick extrahieren. Nach der Extraktion des betroffenen Bricks werden die anderen Bricks zurück in das Target geschoben. Da für den extrahierten Brick kein neuer eingesetzt wird, nimmt die Masse des Targets von anfänglich 1.8 kt im Laufe der Messzeit ab. Die mittlere Target-Masse wird bei fünf Jahren Laufzeit und der erwarteten Ereignisrate zirka 1.6 kt betragen.

Damit das Target immer kompakt bleibt, müssen die Positionen einiger Bricks bei jeder Extraktion eines Bricks aus dem Inneren des Targets geändert werden. Jeder Brick ist deshalb mit einer eindeutigen Markierung versehen, die das BMS auslesen kann. Dadurch ist zu jedem Zeitpunkt bekannt, welcher Brick sich wo im Target befindet oder befunden hat.

Bevor ein Brick nach seiner Extraktion tatsächlich analysiert wird, wird zunächst ein in Strahlrichtung auf der Rückseite des Bricks angebrachter auswechselbarer Emulsionsfilm, das sogenannte *changeable sheet*, untersucht. Hierdurch können Fehlidentifizierungen des zu extrahierenden Bricks erkannt werden. Falls es zu einer Fehlidentifizierung kommt, wird der Brick wieder im Target zur Verfügung gestellt und der nächste, wahrscheinlich getroffene Brick zur Analyse extrahiert. Nach der erfolgreichen Identifikation und Extraktion des betroffenen Bricks werden diesem die Fotoemulsionen entnommen, entwickelt und die Spuren digitalisiert.

Die Genauigkeit der Rekonstruktion in Strahlrichtung beträgt ~ 1 mm und reicht zusammen mit der transversalen Genauigkeit im μm-Bereich aus, um

[9] PMT: *photo multiplying tube*, elektronischer Detektor zum Nachweis einzelner Photonen durch den Photoeffekt und Vervielfachung der Sekundärelektronen.



(a) Kurzer Zerfall (b) Langer Zerfall

Abbildung 2.3: Schematische Darstellung von ν_τ-Ereignissen beim OPERA-Experiment. Ein ν_τ wechselwirkt mit dem Target und produziert ein τ-Lepton. Dieses zerfällt in (a) noch innerhalb der Bleiplatte, während es in (b) vorher eine Emulsionsschicht durchquert (hier beispielhaft der myonische Zerfallskanal gezeigt). Detektiert werden können nur die Spuren geladener Teilchen innerhalb der Fotoemulsionsschichten. Der Fall (b) ist einfacher zu detektieren, da hier sehr leicht der Knick im einzigen Track, der bis ins Myonspektrometer gelangt, erkannt werden kann. Im Fall (a) hingegen muss die Erkennung über den Parameter *b* (*impact parameter*) erfolgen.

τ-Zerfälle identifizieren zu können (die mittlere Flugstrecke eines τ-Leptons beträgt zirka 0.7 mm, siehe Kapitel 2.3.1). In Abbildung 2.3 ist nochmals das Nachweisprinzip für τ-Ereignisse dargestellt. Neben solchen Ereignissen wird das Experiment natürlich erheblich mehr Ereignisse ohne den charakteristischen Zerfallsknick des τ-Leptons, in der Regel also von einem Myonneutrino, detektieren.

2.3.3 Myonspektrometer

Ein weiterer, wichtiger Bestandteil des Experiments ist das Myonspektrometer zur Messung von Ladung und Impuls der Myonen. Das Myonspektrometer besteht aus einem 10 m hohen und 10 m breiten Dipolmagneten mit einer Feldstärke von 1.55 T innerhalb des Eisenkerns (Abbildung 2.4). Der Kern besteht pro Magnethälfte aus zwölf 50 mm dicken Eisenplatten im Abstand von 20 mm (Abbildung 2.5). Zwischen den Platten befinden sich pro Hälfte elf RPCs[10], die bereits eine grobe Rekonstruktion der Myonspuren ermöglichen.

Zur exakten Rekonstruktion der Myonspuren dient ein an der Universität Hamburg entwickelter und aus Driftröhren bestehender *precision tracker* (im

[10] *resistive plate chambers*, ein elektronischer Detektor, unterteilt in Kammern zur 2-dimensionalen Ortsauflösung

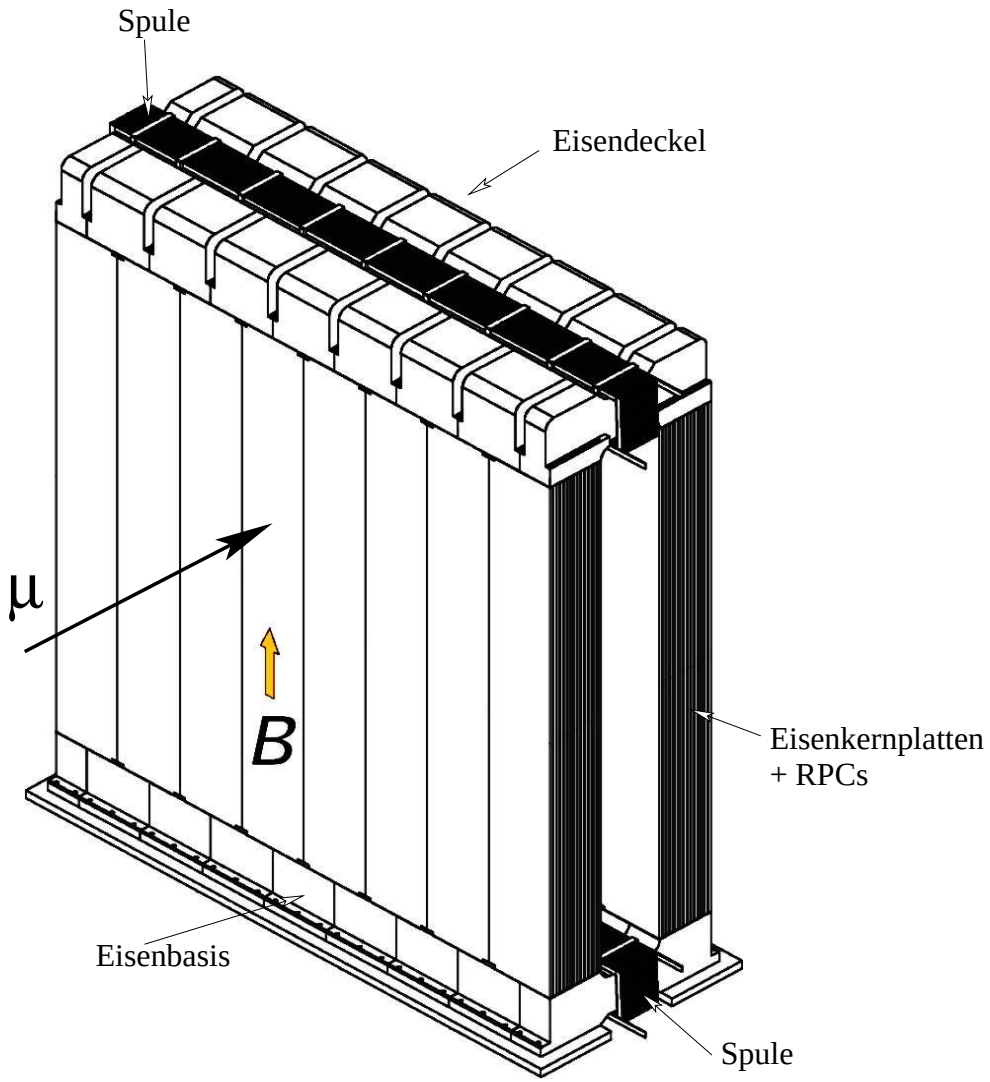


Abbildung 2.4: Der Magnet des Myonspektrometers. Zwei Spulen erzeugen ein 1.55 T starkes Magnetfeld innerhalb der Eisenkernplatten des Magneten. Außerhalb darf jedoch kein nennenswertes Feld vorhanden sein, um die Funktion der Drifträhren nicht zu beeinträchtigen. Die senkrechten Eisenkernplatten sind abwechselnd mit RPCs angeordnet, um die Myonspur innerhalb des Kerns verfolgen zu können.

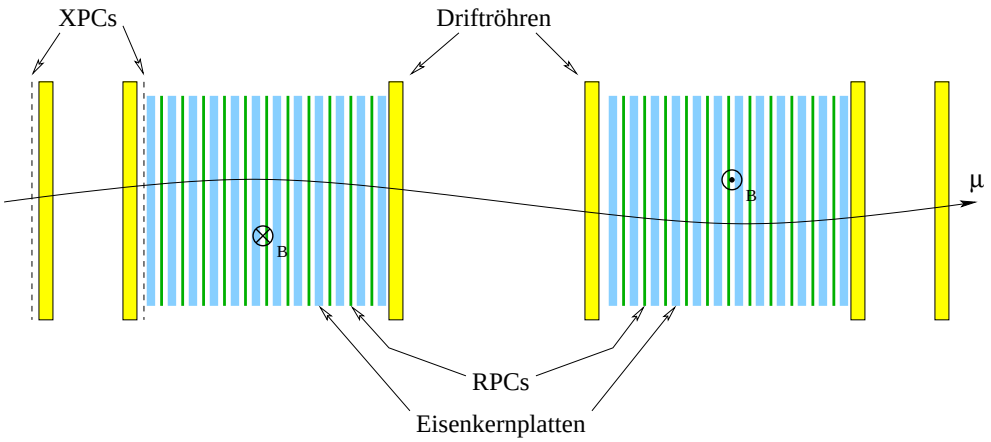


Abbildung 2.5: Horizontaler Schnitt durch das Myonspektrometer. Die Abbildung zeigt die Anordnung der Driftröhren, der Eisenkernplatten des Magneten und der RPCs/XPCs im Spektrometer. Die Eisenkernplatten bilden abwechselnd mit den RPCs/XPCs die beiden Seitenteile des Magneten. Jedes der beiden Supermodule besitzt ein eigenes Spektrometer.

Folgenden als PT abgekürzt). Als Trigger für die Driftzeitmessung werden die im Magneten installierten RPCs sowie XPCs[11] verwendet.

Um die Ablenkung der Myonen exakt messen zu können, befinden sich vor, zwischen und hinter den Magnethälften je zwei Wände des PT. Eine Wand des PT besteht aus 720 (Wände 1, 6 und 7) bzw. 816 (alle anderen Wände) Driftröhren, die in vier Lagen angeordnet sind. Durch die spezielle Anordnung (Abbildung 2.6) kann kein Myon die Wand passieren, ohne mindestens drei Röhren zu durchqueren. Die Driftröhren haben einen Außendurchmesser von 38 mm, sind 7.9 m lang, mit einem Gasgemisch aus 80% Ar und 20% CO_2 gefüllt und besitzen einen 45 μm dicken Draht in der Mitte. Zwischen Draht und Röhre liegt eine Hochspannung von 2.35 kV an. Die Driftröhren werden bei einer Spannung im Proportionalbereich betrieben, so dass die Driftzeit der Elektronen ein Maß für die Entfernung zum Draht ist. Um damit den Abstand zum Draht messen zu können, wird ein Triggersignal benötigt, welches die Driftzeitmessung aller Driftröhren für ein Ereignis stoppt (*common stop*). Dieses wird von den RPCs und XPCs geliefert. Da jede RPC/XPC-Lage jedoch mit zirka 1 kHz rauscht, werden jeweils drei Lagen in einer *2-of-3-Majority*[12] zusammengeschaltet (siehe Kapitel 3.3.1). Wie in Abbildung 2.7 dargestellt, gibt es in jedem Supermodul drei Triggerstationen, die jeweils für zwei benachbarte PT-Wände zuständig sind.

[11] RPCs mit diagonal angeordneten Kammern

[12] *Majority*, engl. Mehrheit. In diesem Fall bezeichnet der Ausdruck eine Koinzidenzschaltung, bei der nur die Mehrheit der Eingänge ein Signal liefern muss.

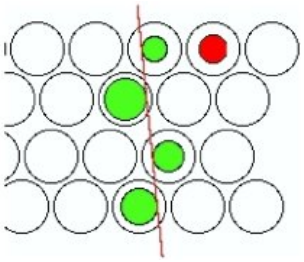


Abbildung 2.6: Schematische Darstellung eines horizontalen Schnitts durch eine Wand des PT. Die spezielle Anordnung der Driftröhren erlaubt es einem Myon nicht, den PT zu durchqueren, ohne nicht mindestens drei Röhren zu treffen. Die grünen Kreise stellen die in Abstände zum Draht umgerechneten Driftzeiten dar, die rote Linie ist eine daran angepasste Myonenspur. Der rote Kreis abseits der Myonenspur ist ein von der Mustererkennung verworfener Treffer, der durch Untergrund verursacht wurde.

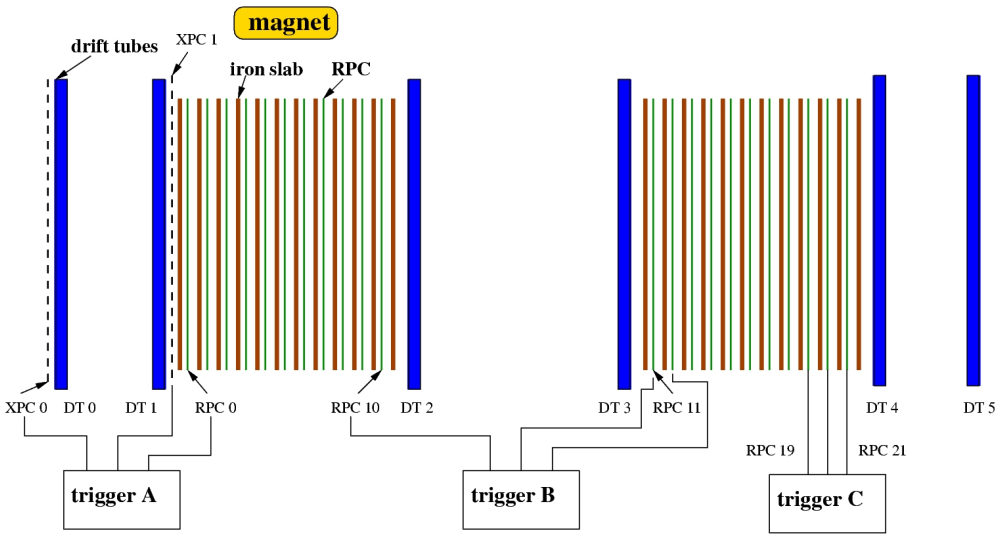


Abbildung 2.7: Stationen des Triggers für den PT. Pro Supermodul gibt es drei Stationen, die jeweils für ein Paar PT-Wände zuständig sind. [Fel05]

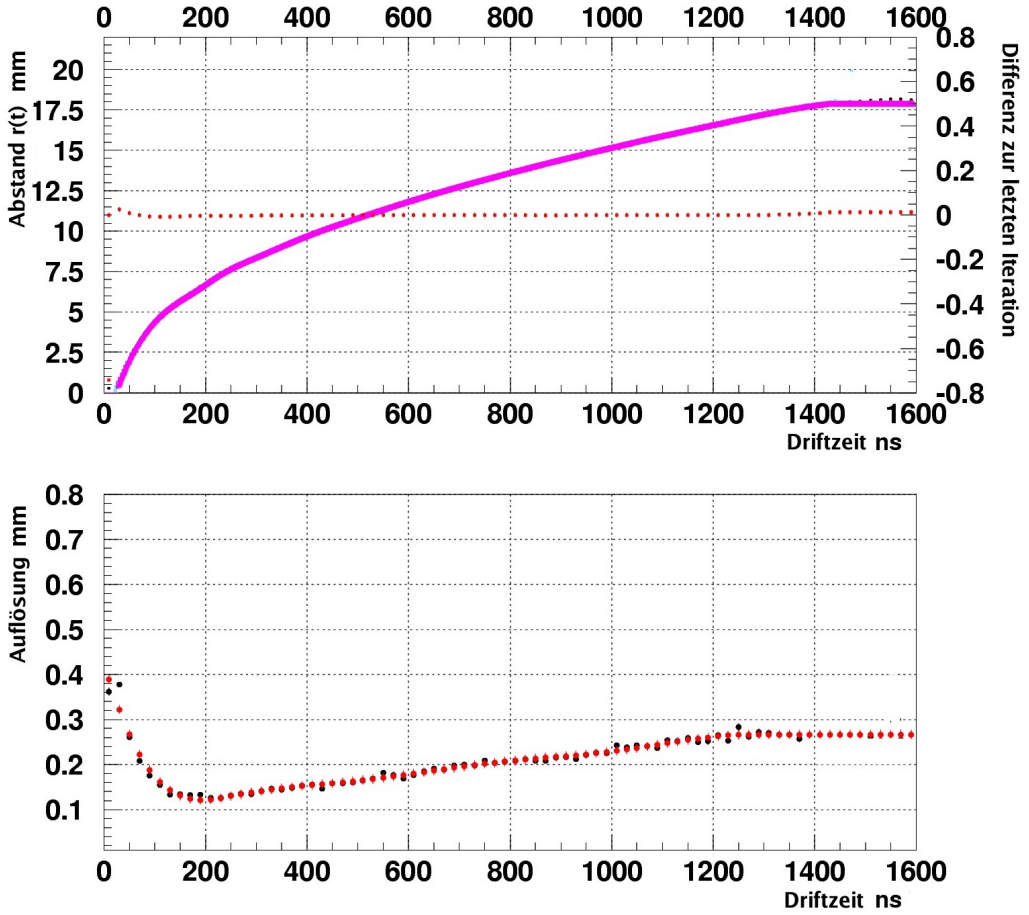


Abbildung 2.8: Driftzeit-Ortsbeziehung $r(t)$ (oberes Bild) sowie Ortsauflösung der Einzelröhren in Abhängigkeit der Driftzeit (unteres Bild). Der gepunktete Graph im oberen Bild gibt die Differenz des $r(t)$ zur vorhergehenden Iteration an (rechte Skala). [Sew06]

Die Driftzeit-Orts-Beziehung kann mittels Selbstkalibration bestimmt werden. Dazu benötigt man eine Messung mit dem PT von möglichst vielen geraden Teilchenspuren. Mit Hilfe einer anfänglichen Driftzeit-Orts-Beziehung (zum Beispiel ein linearer Zusammenhang) wird eine Gerade an jede Trajektorie angepasst. Durch einen iterativen Prozess wird dann die Driftzeit-Orts-Beziehung verbessert. Für diese Kalibrationsmessung können Myonen aus der kosmischen Strahlung verwendet werden, wie sie im Rahmen dieser Arbeit simuliert wurden (siehe Kapitel 3). Eine an einem Testaufbau des PT gemessene Driftzeit-Orts-Beziehung ist in Abbildung 2.8 dargestellt. Abweichungen von einer linearen Beziehung sind vor allem in der Nähe des Drahtes ($r \lesssim 5\,\mathrm{mm}$) und am Rand der Röhre vorhanden. In der Nähe des Drahtes ist das elektrische Feld besonders hoch,

$\Delta m^2_{23}/10^{-3}\,\mathrm{eV}^2$	Zahl der Ereignisse
Signal	
1.3	4.7
2.0	11.7
3.0	14.6
Untergrund	
unabhängig von Δm^2_{23}	1.06

Tabelle 2.1: Anzahl der erwarteten und zugeordneten ν_τ-Ereignisse im OPERA-Experiment in Abhängigkeit des Parameters Δm^2_{23} bei maximaler Mischung ($\Theta_{23} = 45°$) und fünf Jahren Messzeit. [Dra04]

wodurch eine stärkere Lawinenbildung zu einem nichtlinearen Verhalten führt. Am Röhrenrand verlangsamen Rekombinationsprozesse die Drift.

2.3.4 Erwartungen an das OPERA-Experiment

Das Experiment dient in erster Linie dem direkten Nachweis der $\nu_\mu \leftrightarrow \nu_\tau$-Oszillation. Ferner kann das Experiment bei bekanntem Mischungswinkel Θ_{23} (zum Beispiel maximale Mischung, $\Theta_{23} = 45°$) die Massendifferenz Δm^2_{23} messen. Die Zahl der nachgewiesenen ν_τ hängt gemäß Gleichung 2.3 direkt von Δm^2_{23} ab (Tabelle 2.1). Diese Messung wird allerdings wegen der geringen Statistik nicht wesentlich die Fehlergrenzen des Δm^2_{23} verkleinern können. Sollte der Wert aber signifikant von dem bisher gemessenen (siehe Abschnitt 1.3.3) abweichen, wären weitere Messungen mit *appearance*-Experimenten nötig.

Des Weiteren kann das Experiment aber auch Elektronneutrinos über elektromagnetische Schauer im Target-Tracker nachweisen. Die Kontamination mit $\overset{(-)}{\nu}_e$ ist im hochenergetischen Bereich größer als bei der mittleren Energie des Strahls (siehe Abbildung 2.1). Dadurch kann statistisch zwischen direkten und aus der Oszillation entstandenen Myonneutrinos unterschieden werden. Durch die Messung der Rate hochenergetischer Myonneutrinos kann möglicherweise die zulässige Parameterregion in der $\sin^2\Theta_{12}$-Δm^2_{12}-Ebene verkleinert werden (Abbildung 2.9).

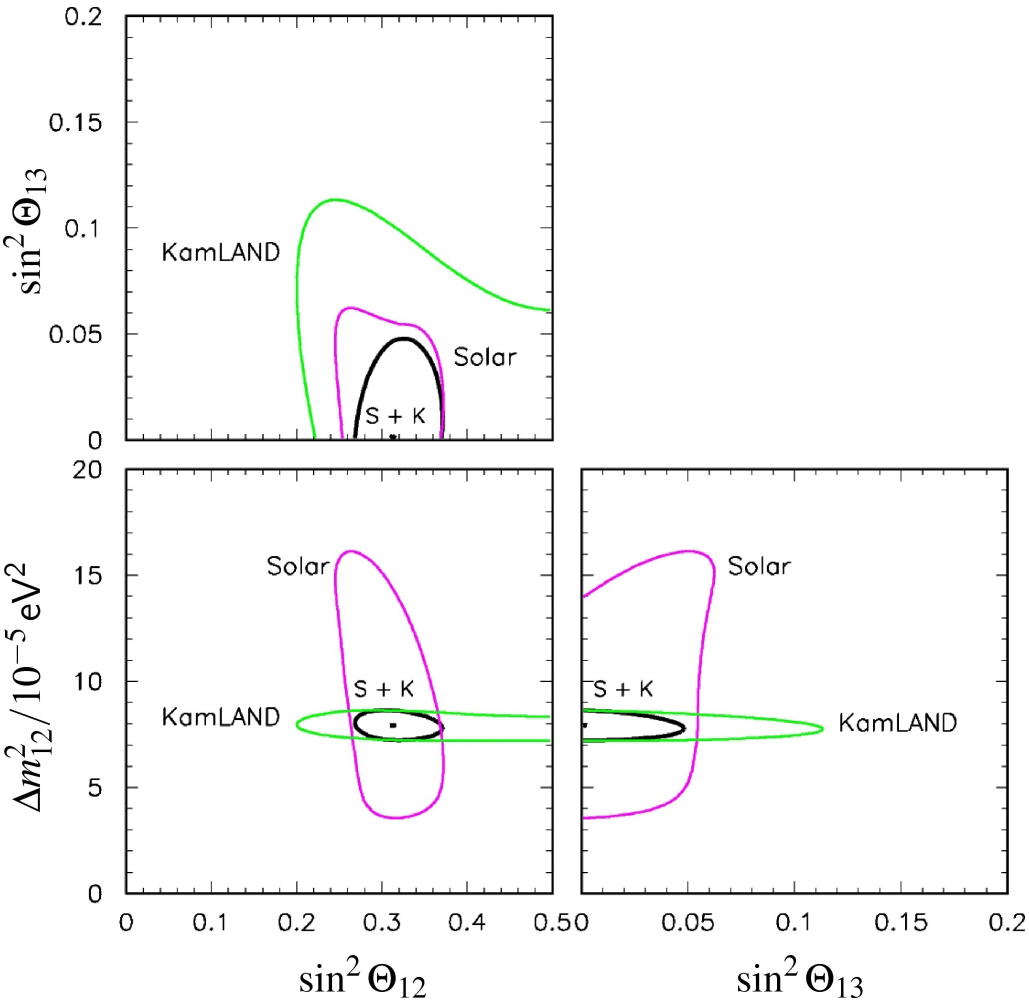


Abbildung 2.9: Darstellung der Mischungsparameter von solaren und Reaktorneutrinos (KamLAND). Die Konturen stellen die 2σ-Vertrauensregionen dar, in denen die Parameter nach den Messungen liegen. Der mit „S+K" beschriftete Bereich ist die Kombination der Ergebnisse von solaren und Reaktorneutrinos. Möglicherweise kann OPERA durch eine Analyse der Ereignisse aus der $\overset{(-)}{\nu}_e$-Kontamination des Strahls die Parameterregion verkleinern. [Fog05]

Kapitel 3

Simulation kosmischer Myonen

Myonen aus der kosmischen Strahlung stellen einen möglichen Untergrund für den *precision tracker* und seinen Trigger dar. Da die zu erwartende Rate von Neutrinoereignissen klein gegen die Rate der kosmischen Myonen ist, werden über die Auswirkungen der Myonen auf den Detektor genaue Informationen benötigt. Gleichzeitig können aber die Myonen dazu genutzt werden, um die Driftzeit-Orts-Beziehung der Driftröhren zu messen, da hierfür die Rate der neutrinoinduzierten Ereignisse nicht ausreichend ist.

Ziel dieser Diplomarbeit ist es, das bereits existierende OPERA-Softwarepaket um die Simulation hadronischer Schauer, die im Fels des umgebenden Gran-Sasso-Massivs erzeugt werden und den Detektor treffen, zu erweitern. Hadronische Schauer können wegen der hohen Teilchendichte die Rekonstruktion eines Ereignisses in den Driftröhren erheblich beeinträchtigen. Deswegen ist es auch wichtig zu wissen, mit welcher Rate solche Ereignisse den Trigger auslösen.

3.1 OPERA-Software als Grundlage

Für die Simulation der kosmischen Myonen wurde die OPERA-Software *OpRelease* verwendet. Die Software basiert unter anderem auf dem *ROOT-Framework*[1] und besteht aus mehreren Paketen. Für die Simulation wichtig sind das Simulationspaket *OpSim*, das Digitalisierungspaket *OpDigit*, das Rekonstruktionspaket *OpRec* sowie als Ereignisgenerator wahlweise *OpNegn* oder *OpCosmics*.

[1] ROOT ist eine am CERN entwickelte Software zur Datenanalyse und wird insbesondere in der Hochenergiephysik verwendet. Eine Analyse kann auf Basis von *ROOT* in der Programmiersprache C++ geschrieben werden.

OpNegn generiert die Orts- und Impulsinformationen primärer Neutrinos aus dem CNGS-Strahl und wurde deshalb im Rahmen dieser Diplomarbeit nicht verwendet. *OpCosmics* hingegen erzeugt die Orts- und Impulsinformationen von Myonen aus der kosmischen Strahlung, wie sie im LNGS-Labor untertage gemessen werden können. Dieser Ereignisgenerator verwendet eine Parametrisierung, die auf Messungen des MACRO-Experiments basiert, das unter anderem Energie- und Winkelverteilungen kosmischer Myonen im LNGS-Labor gemessen hat [Amb95]. In der ursprünglichen Version von *OpCosmics* wurden ausschließlich Myonen simuliert, die direkt den Detektor treffen. Hadronische Schauer, die von Myonen im Fels ausgelöst werden und den Detektor treffen, wurden nicht berücksichtigt. Als Teil dieser Diplomarbeit wurde diese Einschränkung behoben.

OpSim führt die eigentliche Monte-Carlo-Simulation durch und verwendet als Eingangsdaten einen ROOT-Datensatz aus einem der beiden Ereignisgeneratoren. *OpSim* basiert auf der Simulationssoftware GEANT 3.

In einer Monte-Carlo-Simulation wird ein komplexes Problem nummerisch auf Basis von Zufallszahlen gelöst. Dadurch muss es nicht durch einen speziellen Algorithmus gelöst werden. Als Ausgangsdaten stehen für jedes Ereignis Anfangsparameter wie Ort und Impuls eines Teilchens aus einem Ereignisgenerator zur Verfügung. Auf statistischer Basis werden dann physikalische Prozesse wie zum Beispiel Streuung oder Zerfall von Teilchen behandelt. Dazu müssen nur die Wirkungsquerschnitte dieser Prozesse in den verwendeten Materialien bekannt sein. Ausgegeben werden dann Trefferinformationen der aktiven Detektorelemente für jeden Prozess, also zum Beispiel die Koordinaten eines Streuprozesses und die Energiedeposition.

Die von *OpSim* generierten Daten werden von *OpDigit* weiterverarbeitet. *OpDigit* simuliert dabei die Messung und Digitalisierung dieser Daten einschließlich aller damit verbundenen Unsicherheiten, wie zum Beispiel eine begrenzte Orts- oder Zeitauflösung. Der Ausgabedatensatz von *OpDigit* ist von der Struktur und den zur Verfügung stehenden Daten her identisch mit einem aus einer realen Messung gewonnenen Datensatz. *OpRec* kann dann sowohl simulierte als auch reale Daten in identischer Weise verarbeiten.

Die gesamte OPERA-Software ist so konzipiert, dass alle Daten von allen vorangegangenen Schritten stets erhalten bleiben. So kann auf die Ausgabedaten von *OpCosmics* auch im Ausgabedatensatz von zum Beispiel *OpRec* noch zugegriffen werden. Abbildung 3.1 zeigt den Verlauf der Simulation von kosmischen Myonen.

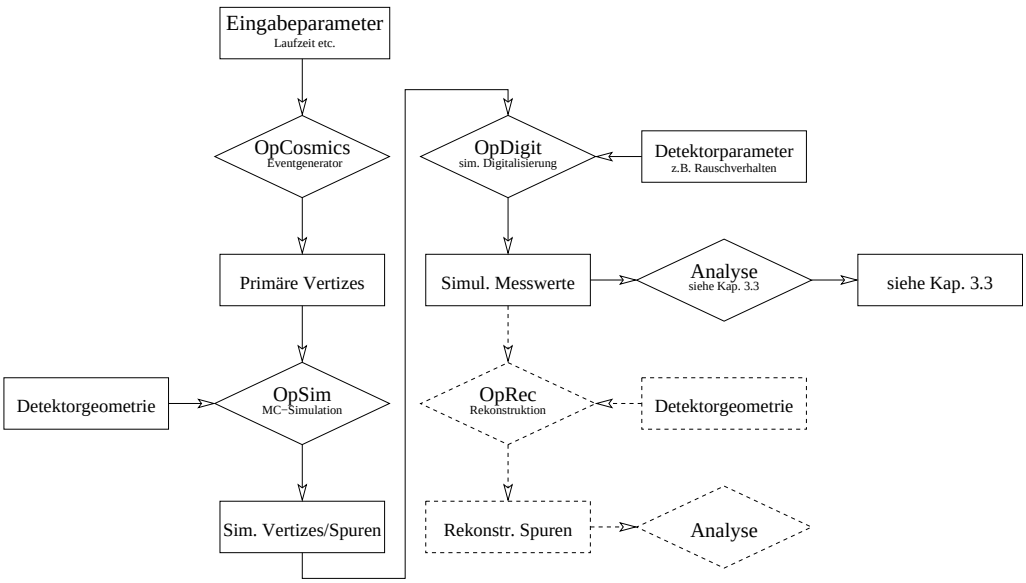


Abbildung 3.1: Flussdiagramm der Simulation kosmischer Myonen im OPERA-Detektor. Jeder Ausgabedatensatz ist ereignisbasiert, er enthält also für jedes Ereignis einen unabhängigen Eintrag. Die Rekonstruktion mit *OpRec* wurde im Rahmen dieser Diplomarbeit nicht durchgeführt.

3.2 Erweiterungen der OPERA-Software

Die wichtigste Anpassung der Software musste zunächst in *OpCosmics* geschehen. Die Originalversion generierte primäre Vertizes, also die Startpositionen der Myonen, auf der Oberfläche eines Quaders unmittelbar um den Detektor. Dadurch wurde der Detektor direkt nur von primären Myonen getroffen. Schauer, die im umgebenden Fels entstehen, konnten nicht berücksichtigt werden.

Um durch Myonen ausgelöste Schauer zu generieren, wurde der Quader so weit ausgedehnt, dass die gesamte Halle C des LNGS, in der der OPERA-Detektor steht, inklusive 13 m umgebenden Fels im Quader enthalten ist. Die Dicke von 13 m des umgebenden Felses entspricht 35 hadronische Strahlungslängen[2,3]. Die Halle ist 100 m lang und zirka 20 m breit und hoch. Die Größe des Quaders in der Originalversion der Simulation betrug zirka $18 \times 10 \times 10\,\mathrm{m}^3$.

Die Zahl der simulierten primären Myonen entspricht einer Messzeit von einem Tag. Es wurden 4418 Treffer im Detektor registriert. Aufgrund der Größe des

[2]Eine Strahlungslänge ist definiert als die durchschnittliche Wegstrecke, nach der ein geladenes Teilchen beim Durchqueren von Materie seine Energie bis auf $\frac{1}{e}$ verloren hat.

[3]nach Berechnung und Empfehlung von M. Sioli mittels der Simulations-Software FLUKA

simulierten Volumens im Vergleich zum Detektor und der damit verbundenen geringen Wahrscheinlichkeit, dass ein Ereignis überhaupt den Detektor trifft (knapp 3%), mussten allerdings 158532 primäre Myonen generiert werden. Die Rechenzeit auf dem Cluster des ZIV[4] und dem GridIKP[5] zusammen betrug zirka 3 Tage. Da diese Simulation lediglich einmal durchgeführt werden musste, wurde sie nicht optimiert.

Des Weiteren wurden an *OpSim* Änderungen zur einfacheren Automatisierung des Ablaufes vorgenommen. Für eine vollständige Übersicht über die Änderungen an beiden Paketen siehe Anhang A.

3.3 Analyse der Simulation

Zur Analyse der simulierten Ereignisse wurden diese zunächst klassifiziert. Dazu wurden eine Bedingung zur Erkennung von Schauern (Abschnitt 3.3.2), eine zur Rekonstruierbarkeit der Ereignisse durch den PT (Abschnitt 3.3.3) sowie verschiedene Triggerbedingungen (Abschnitt 3.3.1) auf die Ereignisse angewendet. Weiterhin wurden aus den Impulsen der primären Myonen die Einfallswinkel Θ und φ (siehe Abbildung 3.5) und die Energie berechnet. Diese Daten wurden dann in den entsprechenden Kategorien gegeneinander aufgetragen. Der vollständige Quellcode der Analyse ist in Anhang B wiedergegeben.

3.3.1 Triggerbedingungen

Der Trigger für den PT ist durch eine *2-of-3-Majority*[6] von drei RPC/XPC-Lagen realisiert. Die Logikschaltung für den Trigger ist in Abbildung 3.2 dargestellt. Jede der RPC-/XPC-Lagen „rauscht" mit zirka 1 kHz aufgrund von spontanen Gasentladungen. Damit ergibt sich bei einer Länge des *Gates* (siehe Abbildung 3.2) von 200 ns eine zufällige Triggerrate von

$$R = 1\,\mathrm{kHz} \cdot 1\,\mathrm{kHz} \cdot (200\,\mathrm{ns} + 200\,\mathrm{ns}) \cdot 3 = 1.2\,\mathrm{Hz} \tag{3.1}$$

Dieses Ergebnis wurde im Rahmen dieser Diplomarbeit mit einer Simulation verifiziert, deren Ergebnis mit $R = 1.19 \pm 0.01\,\mathrm{Hz}$ mit der analytischen Berechnung

[4] Zentrum für Informations-Verarbeitung der Universität Münster. Der Cluster besitzt über 100 Rechenknoten mit 3.2 GHz-Prozessoren und 4 GByte Arbeitsspeicher, von denen jeweils max. vier von einem Nutzer gleichzeitig für serielle Programme wie diese Simulation genutzt werden können.

[5] Cluster des Instituts für Kernphysik Münster, der zirka 100 Rechenknoten mit jeweils 1 GByte Arbeitsspeicher besitzt.

[6] *Majority*, engl. Mehrheit. In diesem Fall bezeichnet der Ausdruck eine Koinzidenzschaltung, bei der nur die Mehrheit der Eingänge ein Signal liefern muss.

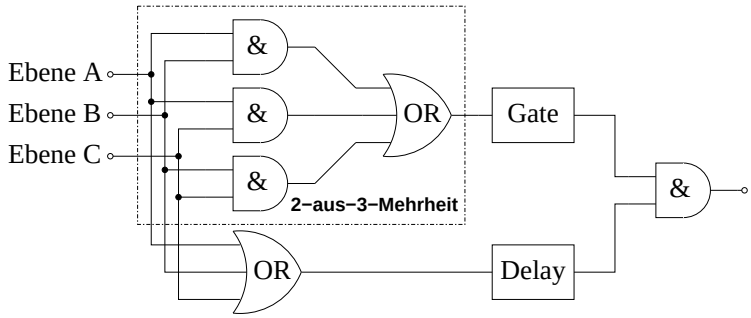


Abbildung 3.2: Blockschaltbild der Triggerlogik für den PT. Bei den drei Triggerebenen A, B und C handelt es sich um RPC- oder XPC-Ebenen. Um ein exaktes Zeitsignal zu erhalten, werden alle drei Signale über eine ODER-Schaltung (*OR*) und eine bekannte Verzögerung (*Delay*) geschaltet. Dieses Signal wird jedoch nur dann als Trigger verwendet, wenn gleichzeitig das *Gate* durch die 2-of-3-Majority für einen definierten Zeitraum geöffnet ist.

(Gleichung 3.1) übereinstimmt. Der angegebene Fehler ist nur der statistische Anteil; der systematische Fehler ist unbekannt, aber identisch mit dem der Berechnung. Der Quellcode dieser Simulation befindet sich in Anhang C.

Die Triggerlogik wurde bei der Analyse der Simulation direkt auf die Trefferinformation der RPCs und XPCs unter Berücksichtigung ihrer Effizienz von 96% [Rep05] angewendet. Dazu wurden für jedes Ereignis die getroffenen RPC/XPC-Lagen in ein „Musterarray" eingetragen, das für jede Lage ein Element mit einer Boole'schen Trefferinformation besitzt. Auf dieses Array wurde anschließend die Triggerbedingungen angewendet.

Zusätzlich zu der im realen Detektor umgesetzten Triggerlogik wurden zwei weitere Bedingungen aufgestellt, die eine Referenz für die Effizienzbestimmung des Triggers bieten sollten. Für die erste Bedingung wurde lediglich die Effizienz der RPC/XPC-Lagen auf 100% gesetzt. Für die zweite Bedingung wurde statt der 2-of-3-Majority eine Oderschaltung über die drei RPC/XPC-Lagen verwendet und ebenfalls die Effizienz der Lagen auf 100% gesetzt.

3.3.2 Erkennung von Ereignissen mit Schauern

Zusätzlich wurde eine Bedingung aufgestellt, die es erlaubt, für die Driftzeitmessung schädliche Schauer zu erkennen. Die simulierten Daten enthalten – anders als eine reale Messung – noch die Information, wie viele Teilchen eine Röhre getroffen haben. Durch visuelle Überprüfung von Ereignissen mit zwei oder mehr Treffern pro Driftröhre konnte abgeschätzt werden, dass nahezu alle dieser

Ereignisse Schauer beinhalten. Dadurch wurde eine komplizierte Analyse mit Mustererkennung etc. vermieden. Stattdessen werden alle Ereignisse, bei denen mindestens eine Driftröhre mehrfach getroffen wurde, als Schauerereignisse angesehen. Mehrfachtreffer in einer Driftröhre zerstören in den meisten Fällen ohnehin die Ortsinformation und müssen somit als „schädliche" Ereignisse angesehen werden. Für die zeitliche Genauigkeit des Triggers spielen Schauer keine Rolle, da die üblichen Driftzeiten groß gegen mögliche zeitliche Störungen durch Schauer sind.

Um genauere Aussagen über die Beeinträchtigung der Funktion des PT durch Schauerereignisse zu erhalten, müsste der Rekonstruktionsalgorithmus aus dem Softwarepaket *OpRec* auf die simulierten Daten der Schauerereignisse angewendet werden, dies ist jedoch Gegenstand einer Promotionsarbeit an der Universität Hamburg. Bei der in der vorliegenden Arbeit verwendeten Methode werden zum Beispiel auch Ereignisse, bei denen die Trajektorie vom Myon räumlich deutlich getrennt vom Schauer ist, als nicht rekonstruierbar charakterisiert, obwohl in diesem Fall eine Rekonstruktion der Myonspur möglich wäre. Solche Ereignisse sind jedoch weitgehend unrealistisch, da die Richtungen von Myon und Schauer miteinander gekoppelt sind.

3.3.3 Rekonstruierbarkeit der Myonspur

Ein weiteres Kriterium bei der Klassifikation der Ereignisse ist der zur Rekonstruktion der Myonenspur nötige Informationsgehalt eines Ereignisses im PT. Da jeder Treffer nur Informationen über den Abstand zum Draht liefert, ist eine eindeutige Rekonstruktion nur möglich, wenn mindestens drei Röhren angesprochen haben. Jede Wand besitzt deshalb und aus Redundanzgründen vier Lagen. Um die nötige Winkelauflösung zu erreichen, muss diese Bedingung in einem Wändepaar gleichzeitig erfüllt sein. Sprechen in einer Wand nur zwei Röhren an, können die beiden Teilspuren nicht zuverlässig miteinander verbunden werden. Auch hier muss für eine genauere Aussage der tatsächlich verwendete Rekonstruktionsalgorithmus angewendet werden.

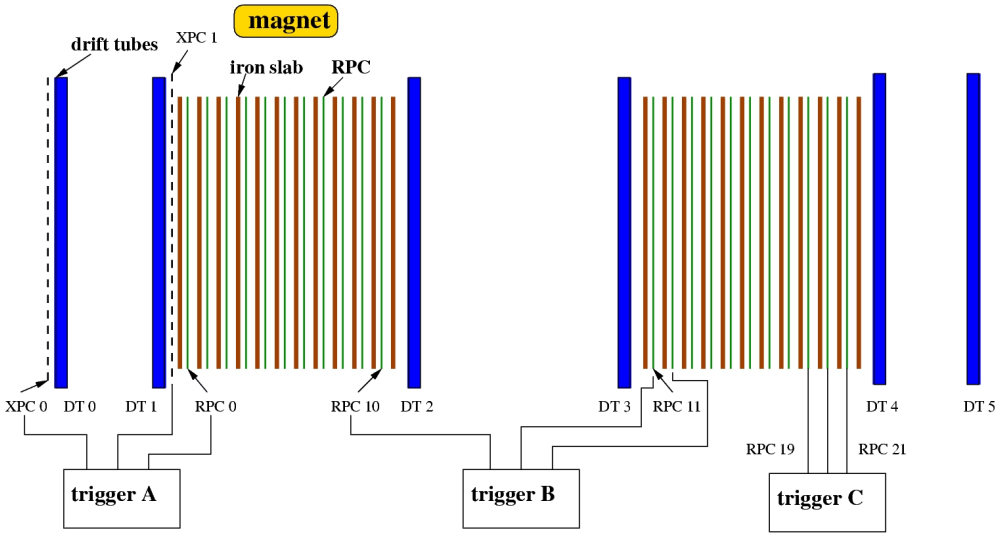


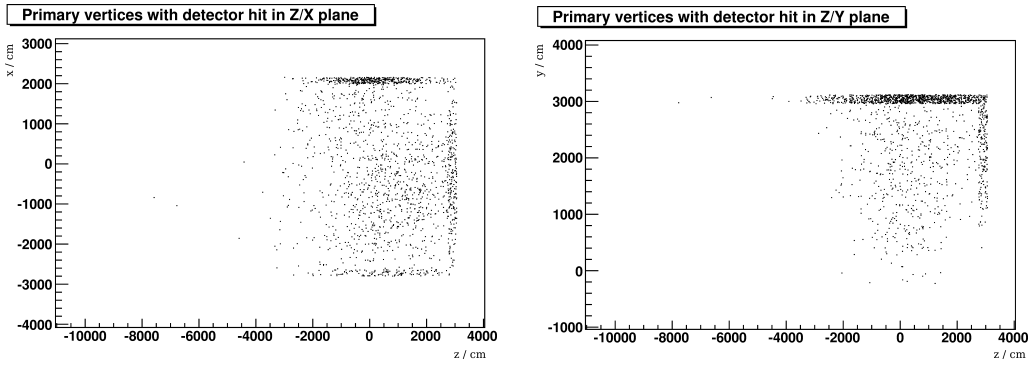
Abbildung 2.7: Stationen des Triggers für den PT. Pro Supermodul gibt es drei Stationen, die jeweils für ein Paar PT-Wände zuständig sind. [Fel05]

3.4 Ergebnis der Analyse

Die Verteilung der primären Vertizes ist in Abbildung 3.3 dargestellt. Die Vertizes liegen auf der Oberfläche eines Quaders, der die gesamte Halle umfasst. Da der OPERA-Detektor nicht in der Hallenmitte aufgestellt ist, ergibt sich eine asymmetrische Verteilung.

In Tabelle 3.1 sind die Triggerraten- und -effizienzen für die Ereignisse mit kosmischen Myonen angegeben. Auffällig ist zunächst einmal, dass eine Effizienz der RPCs/XPCs von 96% die Effizienz des Triggers nicht beeinträchtigt, da die Raten sich nicht signifikant von denen mit 100% RPC/XPC-Effizienz unterscheiden.

Des Weiteren ist das Verhältnis $N_{\text{maj}}^{96\%}/N_{\text{oder}}^{100\%}$ interessant, da dies die Wahrscheinlichkeit angibt, mit der ein Myon den Trigger bei Durchquerung auslöst. Auffällig ist, dass bei den Triggerstationen 2 und 5 dieses Verhältnis erheblich höher ist als bei den anderen Stationen. Das ist ein geometrischer Effekt: bei diesen beiden Stationen werden drei benachbarte RPC-Ebenen verwendet, während für die anderen Stationen zwei benachbarte RPCs und ein auf der anderen Seite der zugehörigen PT-Doppelwand liegender XPC verwendet werden (siehe Abbildung 2.7). Myonen, die die PT-Doppelwand nicht vollständig durchqueren, lösen deshalb die Triggerstationen 0, 1, 3 und 4 mit einer geringeren Wahrscheinlichkeit aus, da sie nicht alle Ebenen des Triggers treffen. Bei den Stationen 2 und 5 ist diese



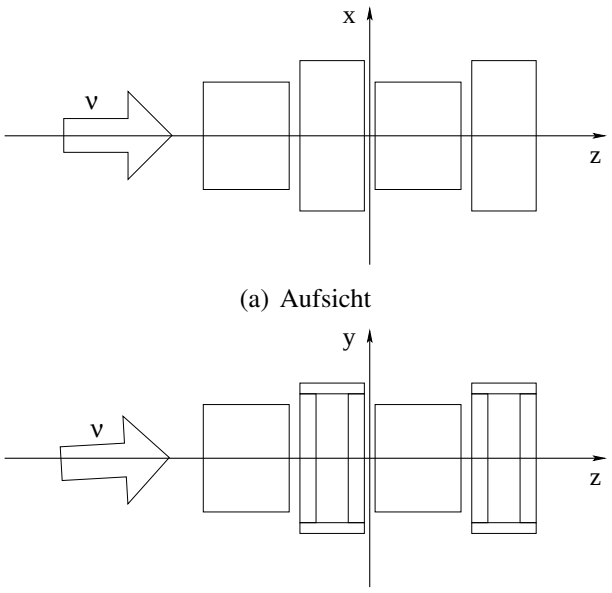
(a) Aufsicht (b) Seitenansicht

Abbildung 3.3: Verteilung der primären Vertizes simulierter kosmischer Myonen mit Treffern im OPERA-Detektor. Die Vertizes liegen auf der Oberfläche eines Quaders. Die dichteren Ränder entstehen durch die Projektion der Seitenwände des Quaders auf die jeweilige Ebene. Die Dicke der Ränder hingegen ist ein Artefakt der Darstellung (Größe des *binnings*). Der Detektor steht nicht in der Hallenmitte, wodurch sich eine asymmetrische Verteilung ergibt. Zur Definition des Koordinatensystems siehe Abbildung 3.4.

Selektivität nicht vorhanden. Solche Ereignisse sind aber nicht oder nur ungenau rekonstruierbar (siehe Abschnitt 3.3.3).

Ereignisse, bei denen ein Myon eine PT-Doppelwand vollständig durchquert, können am genauesten rekonstruiert werden. Deshalb erhält man bei Betrachtung dieser Ereignisse einen aussagekräftigeren Wert für die Effizienz. 673 Ereignisse haben eine PT-Doppelwand durchquert, von denen haben 667 den Trigger ausgelöst (siehe Tabelle 3.1). Damit beträgt die Effizienz des Triggers $\eta = \frac{667}{673} = (99.1 \pm 0.5)\%$. Die angegebene Unsicherheit wurde aus Abweichungen von Simulationsläufen mit anderen Bedingungen bestimmt. Ereignisse mit Mehrfachtreffern in den Driftröhren wurden hierbei nicht berücksichtigt, da diese nicht rekonstruierbar wären.

Die Azimutverteilung der primären Myonen (Abbildung 3.6) wird durch die Form des Gran Sasso Massivs hervorgerufen. An der Erdoberfläche ist der Azimut der kosmischen Myonen gleichverteilt. Im Untergrundlabor treffen jedoch besonders wenige Myonen aus Richtung des CNGS Neutrinostrahls (0°) ein, da hier die Abschirmung am größten ist. Auch das Minimum bei -150° kann so erklärt werden (Abbildung 3.12). Der Trigger unterdrückt aufgrund seiner geometrischen Anordnung noch zusätzlich Ereignisse mit Myonen orthogonal zum Neutrinostrahl ($\pm 90^\circ$).



(a) Aufsicht

(b) Seitenansicht

Abbildung 3.4: Definition des Koordinatensystems. Im oberen Bild ist eine Aufsicht, im unteren eine Seitenansicht des Detektors. Der Neutrinostrahl trifft unter einer Inklination von 3.22° [Fer02] zur z-Achse auf den Detektor.

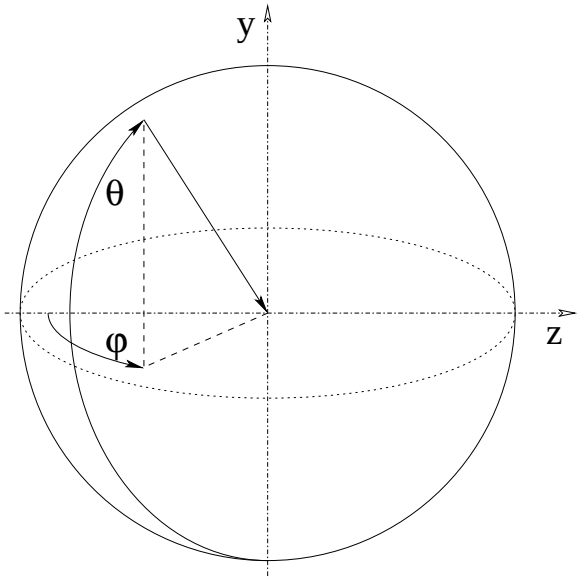


Abbildung 3.5: Definition des Azimuts φ und des Polarwinkels Θ in der Analyse. Zur Definition des Koordinatensystems siehe Abbildung 3.4.

Triggerstation	$N_{\text{maj}}^{96\%}$	$N_{\text{maj}}^{100\%}$	$N_{\text{oder}}^{100\%}$	$N_{\text{maj}}^{96\%}/N_{\text{maj}}^{100\%}$	$N_{\text{maj}}^{96\%}/N_{\text{oder}}^{100\%}$
0	443	444	625	0.998	0.709
1	464	465	664	0.998	0.699
2	491	491	521	1.000	0.942
3	439	439	607	1.000	0.723
4	469	469	700	1.000	0.670
5	475	475	516	1.000	0.921

Getriggerte Ereignisse insgesamt:	1401
Schauer insgesamt:	331
Getriggerte Schauer:	235
Ereignisse mit getroffenen PT-Doppelwänden:	
ohne Schauer:	673
getriggert, ohne Schauer:	667
getriggert, nur Schauer:	139

Tabelle 3.1: Triggerraten und -effizienzen der einzelnen Triggerstationen (Angaben in Ereignissen pro Tag). Die Stationen 0 bis 2 sind im ersten, die Stationen 3 bis 5 im zweiten Supermodul. $N_{\text{maj}}^{96\%}$ bezeichnet die Rate des Trigger mit einer 2-of-3-Majority und 96% Effizienz der RPC/XPC-Ebenen, was die experimentellen Bedingungen entspricht. Für die Berechnung von $N_{\text{maj}}^{100\%}$ wurde lediglich die Effizienz der RPCs/XPCs auf 100% gesetzt, während für $N_{\text{oder}}^{100\%}$ noch zusätzlich die 2-of-3-Majority durch ein logisches Oder ersetzt wurde. $N_{\text{maj}}^{100\%}$ und $N_{\text{oder}}^{100\%}$ dienen lediglich als Referenz zur Berechnung der Effizienz.

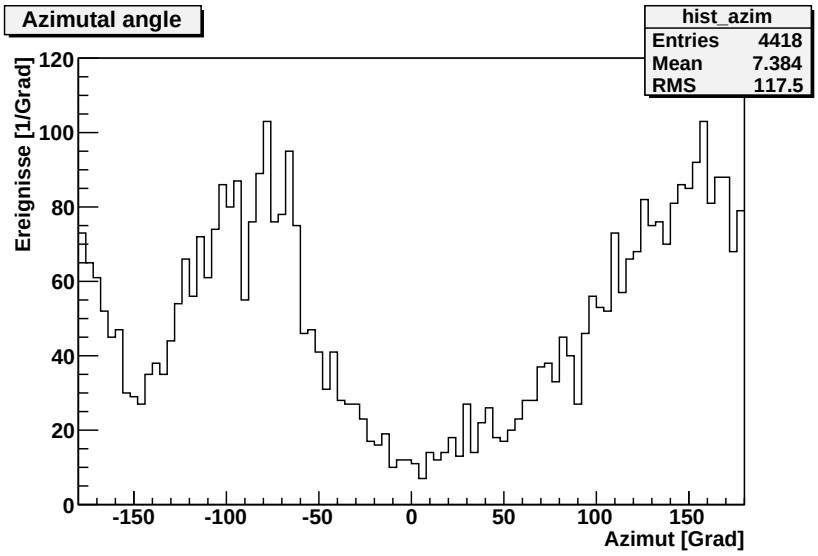
Auch die Polarwinkelverteilung (Abbildung 3.7) wird durch die Abschirmung beeinflusst. An der Erdoberfläche ist der Polarwinkel gemäß $\cos^2(180^\circ - \Theta)$ verteilt. Die Polarwinkelverteilung hätte an der Erdoberfläche damit sein Maximum bei 180°. Durch die Form des Felses werden aber steil einfallende Myonen stärker abgeschirmt, wodurch sich die gezeigte Verteilung ergibt. Zusätzlich werden diese Myonen noch durch die Geometrie des Triggers unterdrückt, da sie fast parallel zu den Triggerebenen auf den Detektor treffen.

Betrachtet man den Winkel zur z-Achse (Abbildung 3.8), wird deutlich, dass keine Verwechslungsgefahr mit Ereignissen aus dem Neutrinostrahl besteht. Innerhalb von 15° zur Strahlachse liegen keine Ereignisse aus kosmischen Myonen.

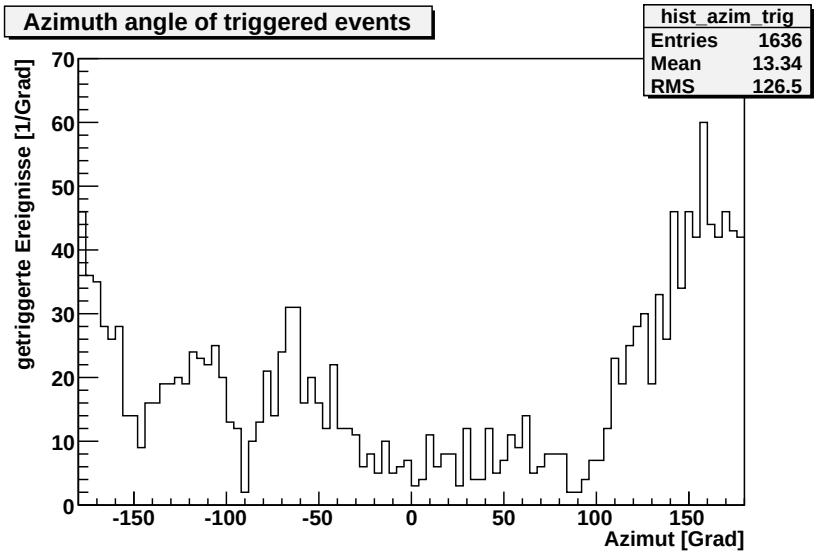
In Abbildung 3.9 ist noch eine 2D-Ansicht der Winkel (φ, Θ) der getriggerten primären Myonen gezeigt. In Strahlrichtung $(0^\circ, 90^\circ)$ befinden sich keine Ereignisse. Bei den Azimutwinkeln von -90° und $+90^\circ$ ist eine Ausdünnung der Ereignisdichte aufgrund der Geometrie des Triggers zu sehen.

Ereignisse mit Mehrfachtreffern im PT werden als „schädliche" Schauerereignisse angesehen (siehe Abschnitt 3.3.2). In Abbildung 3.10 ist die maximale Zahl der Treffer im PT pro Ereignis dargestellt. 321 Ereignisse haben zwei und mehr Treffer, von diesen haben 235 den Trigger ausgelöst (siehe auch Tabelle 3.1). In Abbildung 3.11 ist das Energiespektrum der Ereignisse mit bzw. ohne Schauer dargestellt. Deutlich zu erkennen ist, dass Ereignisse mit Schauern im Mittel eine höhere Energie des primären Myons aufweisen.

Für eine exaktere Analyse müsste der Ausgabedatensatz der Simulation mit dem Softwarepaket *OpRec* rekonstruiert werden. Dann könnten genaue Aussagen getroffen werden, welche Ereignisse rekonstruiert werden können, und zum Beispiel wie lange die Kalibrationsmessung zur Bestimmung der Driftzeit-Orts-Beziehung dauern wird. Mit den bisherigen Daten (Tabelle 3.1) kann hier nur gesagt werden, dass über 650 geeignete Ereignisse pro Tag auftreten. Die Driftzeit-Orts-Beziehung sollte man damit in wenigen Tagen erhalten können. Einige tausend Ereignisse sind ausreichend, um diese Beziehung für die benötigte Auflösung genau genug zu messen.

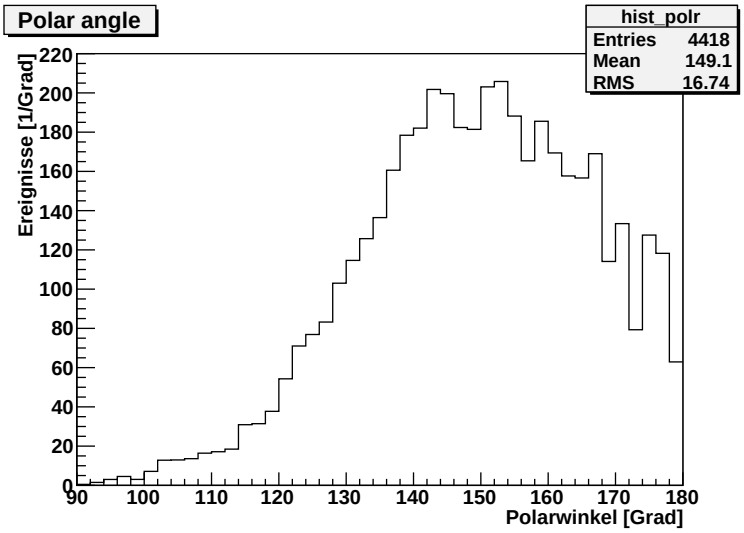


(a) Azimut aller Ereignisse

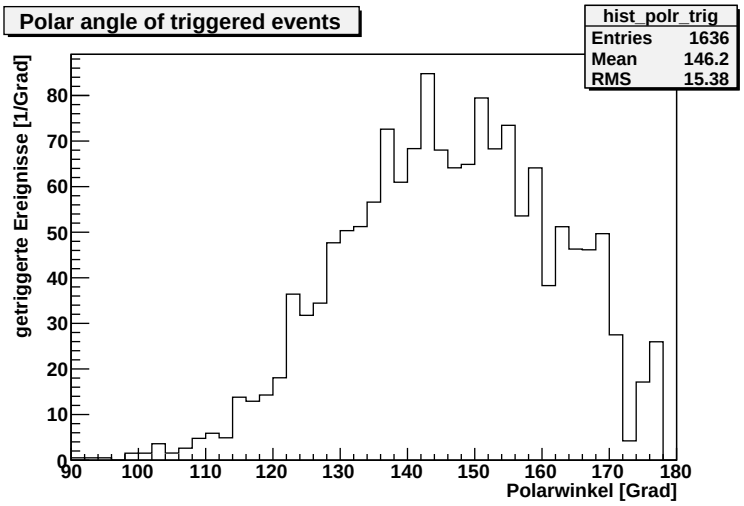


(b) Azimut getriggerter Ereignisse

Abbildung 3.6: Azimut von allen (a) und den getriggerten Ereignissen (b). Die Verteilung ist durch die Form des abschirmenden Gran-Sasso-Massivs bestimmt, siehe Abbildung 3.12.

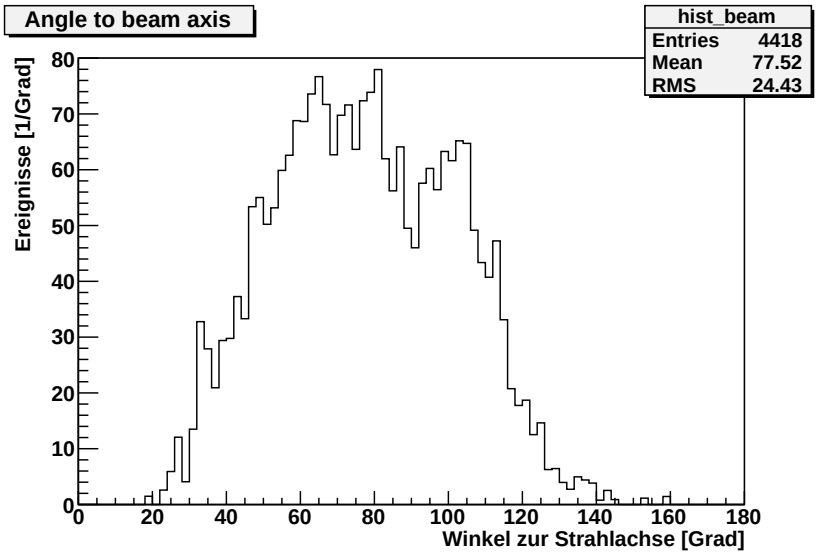


(a) Polarwinkel aller Ereignisse

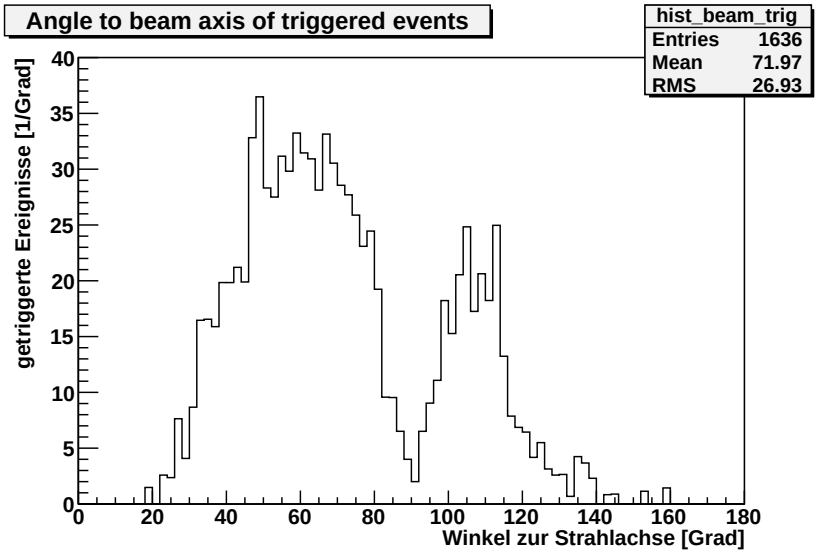


(b) Polarwinkel getriggerter Ereignisse

Abbildung 3.7: Polarwinkel von allen (a) und den getriggerten Ereignissen (b). Steil von oben eintreffende Myonen werden von dem Trigger aufgrund seiner Ausrichtung unterdrückt. Wegen der Form des Gran Sasso Massiv sind diese Ereignisse noch zusätzlich unterdrückt. Die Raten wurden auf eine feste Raumwinkelgröße normiert.



(a) Winkel zur z-Achse aller Ereignisse



(b) Winkel zur z-Achse getriggerter Ereignisse

Abbildung 3.8: Winkel zur z-Achse von allen (a) und den getriggerten Ereignissen (b). Ein Winkel von 0° würde bedeuten, dass das Myon in etwa aus derselben Richtung kommt, wie die Strahlneutrinos.

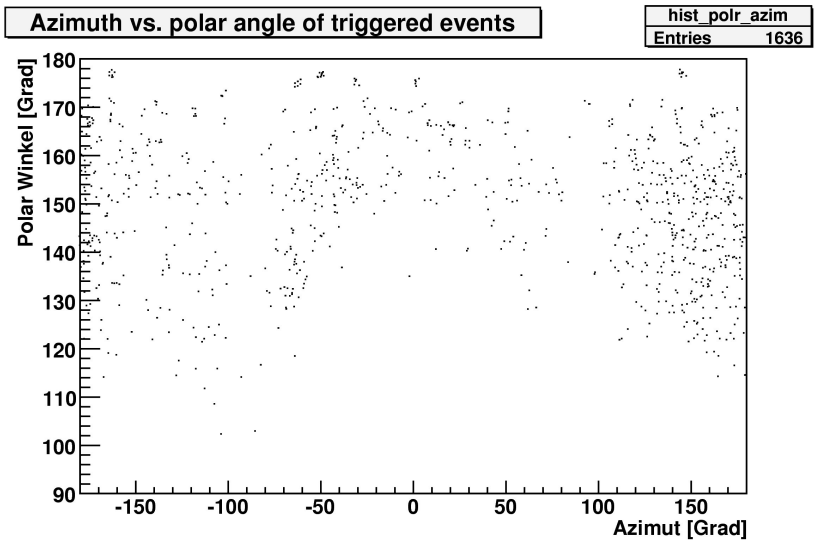


Abbildung 3.9: Polarwinkel über dem Azimut der getriggerten Ereignisse. Bei $\varphi = \pm 90^\circ$ sind Ausdünnungen zu erkennen, die durch die geometrische Ausrichtung des Triggers entstehen.

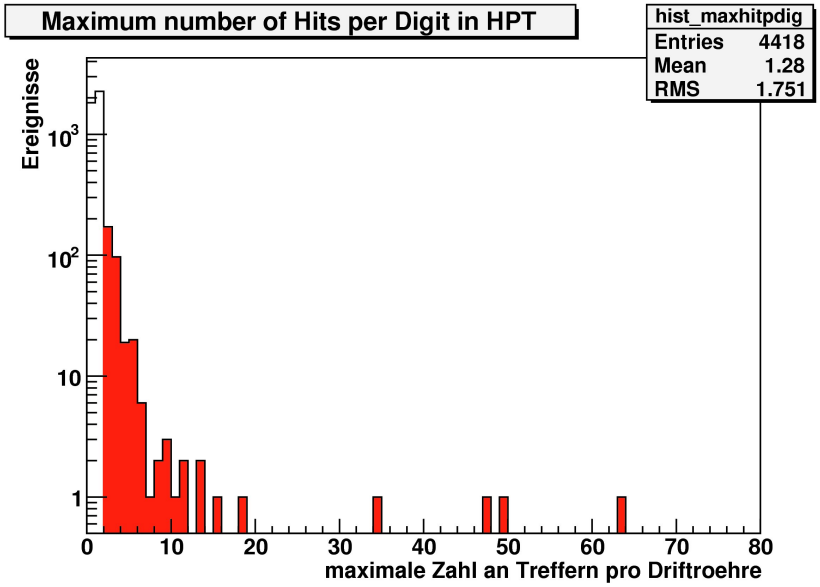
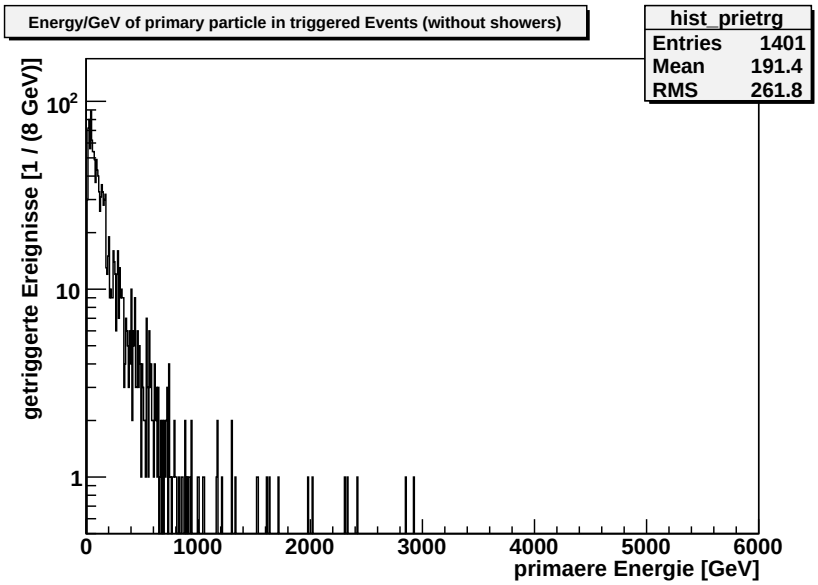
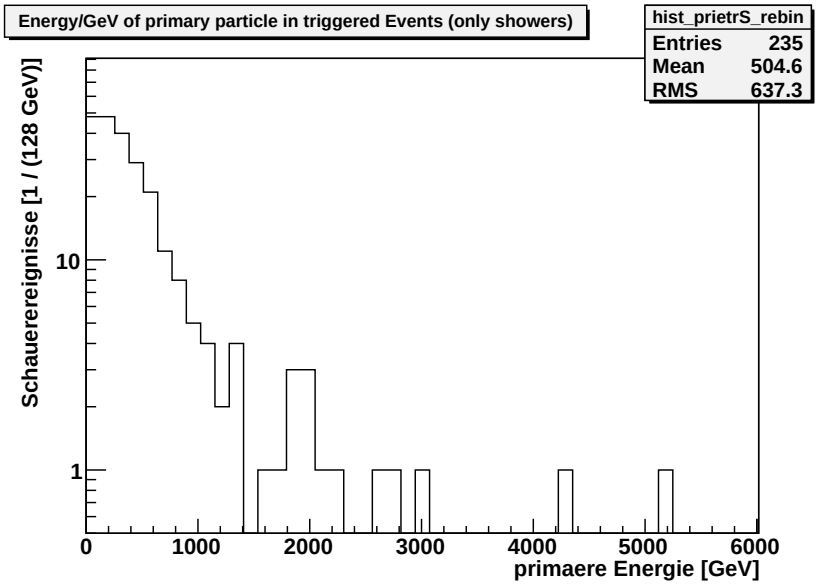


Abbildung 3.10: Maximale Zahl von Treffern pro Röhre. Alle Ereignisse mit mindestens zwei Treffern in einer Röhre werden als problematische Schauerereignisse angesehen (rot gekennzeichnet).



(a) Nur Ereignisse ohne Schauer



(b) Nur Ereignisse mit Schauer

Abbildung 3.11: Energie des primären Teilchens von getriggerter Ereignisse. Ereignisse mit Schauern haben im Mittel eine höhere Energie: der Mittelwert der Energie bei Ereignissen ohne Schauer liegt bei 191 GeV, während der Mittelwert bei Ereignissen mit Schauen bei 505 GeV liegt.

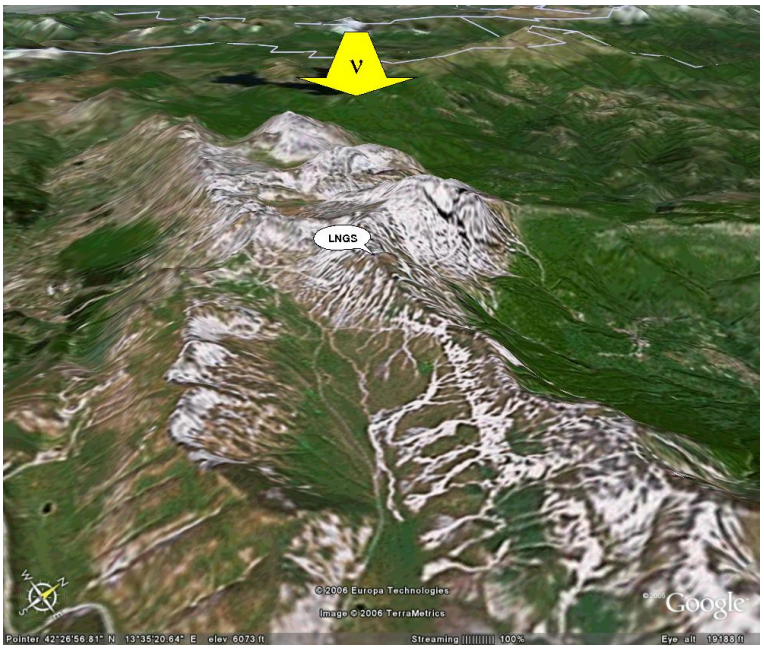


Abbildung 3.12: 3D-Ansicht des Gran Sasso Massivs. Die Markierung befindet sich an dem Punkt der Oberfläche, unter dem sich das LNGS-Labor befindet. Die Blickrichtung ist in etwa entgegen dem Neutrinostrahl. In Strahlrichtung ist besonders viel Fels zur Abschirmung der Myonen vorhanden (Vergleiche Minimum bei 0° in der Azimutverteilung Abbildung 3.6). Auch der Gebirgskamm, im Bild nach rechts unten, ist in der Azimutverteilung bei -150° als Minimum wiederzufinden. [Www03]

Kapitel 4

Slow Control Datenbank des Precision Trackers

4.1 Struktur und Funktionsweise

Zum Aufzeichnen der äußeren, langsam veränderlichen Parameter des *precision tracker*s wird eine *slow control* Datenbank benötigt. Die Gliederung der *slow control* des PT ist in Abbildung 4.1 dargestellt. Sie gliedert sich in mehrere Subsysteme, die Daten an die Datenbank liefern oder aus ihr abfragen. Die eigentliche Übermittlung der Daten läuft jedoch über zwei zentrale Programme, *hptSCcollect*[1] und *hptDBhub*, wobei Letzteres im Rahmen dieser Diplomarbeit entwickelt wurde. *hptSCcollect* sammelt von jedem Subsystem die Daten ein und leitet sie an *hptDBhub* weiter. *hptDBhub* schreibt auf Anfrage von *hptSCcollect* die immer noch den Subsystemen zugeordneten Daten in die Datenbank. Sollwerte werden (zum Beispiel nach einem Stromausfall) von *hptDBhub* wieder ausgelesen und über *hptSCcollect* an die jeweiligen Subsysteme weitergeleitet.

Die Struktur der Datenbank wird durch die Datei *hptDB.xml* beschrieben (siehe Anhang D.3). Festgelegt werden müssen Typ und Name der Spalten jeder Tabelle. Für jedes Subsystem können mehrere Tabellen definiert werden, bei den meisten Subsystemen genügt aber eine einzige Tabelle. *hptDBhub* legt die Datenbank gemäß dieser Informationen automatisch an. Das Programm ist so organisiert, dass es (im Rahmen definierter Regeln, siehe Anhang D.3) keine Annahmen über die benötigten Daten und die Struktur der Datenbank macht. Alle benötigten Informationen werden aus der Datei *hptDB.xml* gelesen. Dadurch kann

[1] Das Programm *hptSCcollect* wird zur Zeit an der Universität Hamburg entwickelt; bisher existieren nur provisorische Programme, die teilweise die Aufgaben von *hptSCcollect* übernehmen.

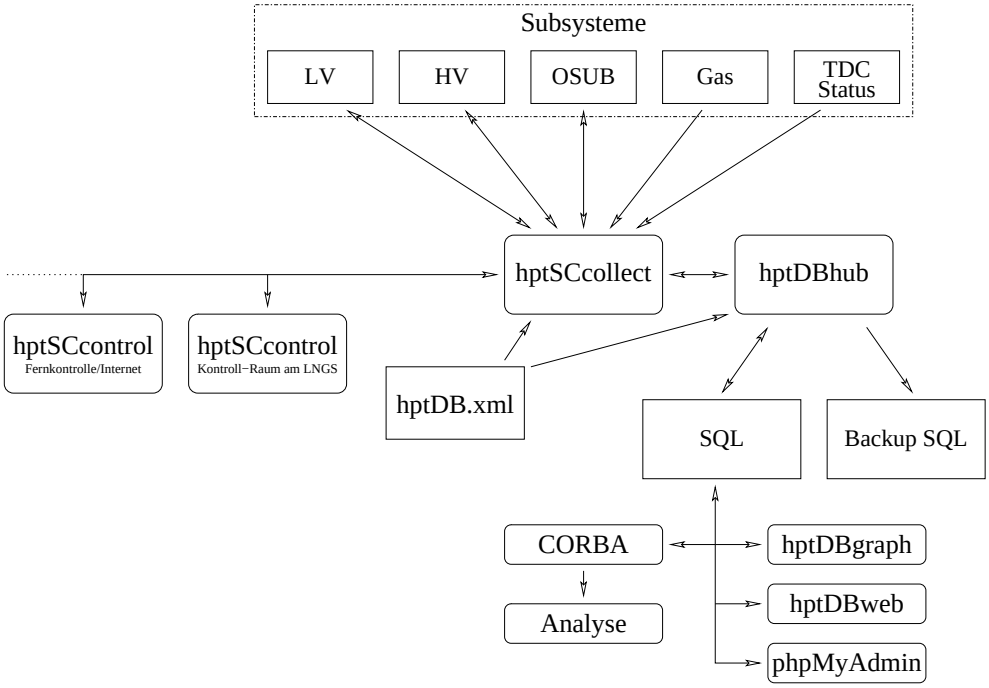


Abbildung 4.1: Datenfluss der *slow control* des PT. Das Programm *hptDBhub* empfängt seine Daten von *hptSCcollect*, das seine Daten aktiv von den verschiedenen Subsystemen abruft. Die Daten werden in eine SQL-Datenbank[a] geschrieben, von der sie später zur Analyse über ein standardisiertes CORBA-*Interface* wieder ausgelesen werden können (siehe Abschnitt 4.4). Die Programme *hptDBgraph* und *hptDBweb* wurden ebenfalls im Rahmen dieser Diplomarbeit entwickelt und dienen der schnellen Visualisierung der Daten über das Internet. Im Gegensatz dazu kann *hptSCcontrol* auch Slow-Control-Parameter ändern.

[a] *structured query language*, eine standardisierte Sprache zur Kommunikation mit Datenbank-Software. Für die *slow control* des PT wird die Open-Source-Software MySQL verwendet.

die Datenbank jederzeit während der Entwicklung der anderen Komponenten der *slow control* an die geänderten Anforderungen angepasst werden.

Von *hptSCcollect* können im späteren Betrieb mehrere Instanzen gleichzeitig auf verschiedenen PCs laufen, die jeweils für verschiedene Subsysteme zuständig sind. *hptDBhub* läuft jedoch nur in einer einzigen Instanz auf dem Datenbankserver. Die Daten müssen daher über ein Netzwerk übertragen werden, wofür im Rahmen dieser Diplomarbeit ein Übertragungsprotokoll definiert wurde. Nach einem Datenkopf[2] werden die Nutzdaten übertragen. Diese müssen in dem

[2] Für eine genaue Beschreibung des Protokolls und insbesondere des Kopfes siehe Anhang D.1

Datentyp und in der Reihenfolge, wie sie in der Datei *hptDB.xml* definiert sind, im Netzwerk übertragen werden. *hptSCcollect* hat ebenfalls Zugriff auf dieselbe *hptDB.xml*, wodurch jederzeit eine konsistente Datenübertragung gewährleistet ist.

hptSCcollect sammelt alle 10 s die aktuellen Messdaten von den Subsystemen und leitet diese an *hptDBhub* weiter. Zwecks Reduktion der Datenmenge schreibt *hptDBhub* nur bei Änderung die Daten tatsächlich in die Datenbank. Dazu wird ein Schwellwert in der Datei *hptDB.xml* festgelegt, um den sich der neue Wert vom zuletzt geschriebenen unterscheiden muss. Zusätzlich gibt es noch eine maximale Zeitspanne, nach der die Daten auch ohne Veränderung neu geschrieben werden. Diese wird ebenfalls in der Datei *hptDB.xml* festgelegt und sollte im Normalfall 10 min betragen.

Sollwerte werden nur bei Änderung an *hptDBhub* gesendet. Ein Schwellwert zur Datenreduktion wie bei den Messdaten wird deshalb hier nicht verwendet. Abgesehen davon werden Sollwerte aber von *hptDBhub* identisch wie Messwerte behandelt. Bei einem Stromausfall oder Neustart eines Subsystems fragt *hptSCcollect* die zuletzt gesetzten Sollwerte über *hptDBhub* aus der Datenbank ab.

4.2 Abschätzung des Speicherbedarfs

Zur Dimensionierung des Datenbankservers wurde eine grobe Abschätzung des Speicherbedarfs dieser Datenbank vorgenommen. Dazu wurde zunächst für jedes Subsystem die Zahl der Messwerte abgeschätzt (siehe Tabelle 4.1). Die dort angegebene Anzahl der Messwerte ist eine Vorhersage über die Anforderungen der einzelnen Subsysteme und kann von der tatsächlich benötigten Anzahl abweichen. Nach diesen Vorhersagen können maximal 4778 Byte pro 10 s-Zyklus an Messdaten anfallen. Das sind zirka 70 GByte für eine Messzeit von fünf Jahren. Die verwendete Datenbanksoftware *MySQL* belegt an Festplattenspeicher zirka 3.5% mehr als die reinen Nutzdaten[3], wodurch maximal zirka 72 GByte Speicherplatz belegt werden. Auch wenn noch erheblich mehr Daten in diese Datenbank aufgenommen werden müssen, bleibt die Datenmenge im Bereich des heute technisch Realisierbaren. Aufgrund der in Abschnitt 4.1 beschriebenen Datenreduktion wird die Datenbank weniger Speicherplatz benötigen. Nicht berücksichtigt sind hierbei die Sollwerte, da sie nur wenige Male gesetzt werden müssen. Um genügend Speicherplatz auch im Fall einer deutlich größeren Datenbank und eauch für andere Zwecke zur Verfügung zu haben, wurde der Datenbankserver mit einer 300 GByte großen Festplatte ausgestattet.

[3]Dieser Wert basiert auf einem Vergleich der in die Datenbank geschriebenen Menge an Nutzdaten mit dem auf der Festplatte belegten Speicher.

Subsystem	**Zahl der Messwerte**	**Zahl der Bytes**		
LV	4 Strom, Spannung, Status	$(3 \cdot 4) \cdot 4$	=	48
HV	48 Strom, Spannung, Status	$(3 \cdot 48) \cdot 4$	=	576
OSUB	120 Temperaturen, 100 Status	$(120 + 100) \cdot 4$	=	880
Gas (5 Tabellen)	72 Sauerstoff, 15 Druck, 18 Temperatur, 24 Fluss, 200 Ventile, 167 Status	$(72 + 15 + 18 + 24 + 200 + 167) \cdot 4$	=	1984
TDC	100 Strom, Spannung, Status	$(3 \cdot 100) \cdot 4$	=	1200
Summe	ohne Kopfdaten		=	4688
Kopfdaten	9 Tabellen, je 10 Bytes	$9 \cdot 10$	=	90
Summe	**mit Kopfdaten**		**=**	**4778**

Tabelle 4.1: Datenmenge der *slow control* Messwerte des PT. Jeder Messwert wird einheitlich mit 4 Byte abgespeichert, zusätzlich kommen noch Kopfdaten (Zeitstempel etc.) mit 10 Byte pro Datenbanktabelle hinzu, die in der obigen Tabelle nicht berücksichtigt sind (jedes Subsystem hat eine Datenbanktabelle, wenn nicht anders angegeben). Damit ergibt sich eine Gesamtgröße von 4778 Byte pro Datensatz.

4.3 Geschwindigkeit

Die Datenrate der *slow control* Datenbank ist mit maximal $285.9 \frac{\text{Bytes}}{\text{s}}$ nicht besonders hoch. Trotzdem wurde ein Test bezüglich Geschwindigkeit und Stabilität der Datenbank durchgeführt. Dazu wurden Zufallsdaten mit deutlich höherer Geschwindigkeit als im Regelbetrieb vorgesehen an *hptDBhub* gesendet. Bei diesem Test wurden Datensatzraten von zirka $30 \frac{\text{Datensätze}}{\text{s}}$ erreicht[4]. Bei dauerhaft höheren Geschwindigkeiten gehen bereits einige Datensätze verloren. Das liegt daran, dass gemäß Protokoll für jeden gesendeten Datensatz eine eigene Verbindung auf- und wieder abgebaut wird. Aufgrund von Beschränkungen des zu Grunde liegenden TCP/IP-Protokolls bleibt eine Verbindung noch einige Zeit nach Beendigung des Datentransfers geöffnet, und es können nur eine bestimmte Anzahl von Verbindungen gleichzeitig existieren. Dadurch wird die maximale Zahl von TCP/IP-Verbindungen ab der gemessenen Datensatzrate von zirka $30 \frac{\text{Datensätze}}{\text{s}}$ überschritten. Kurzzeitig hohe Datensatzraten, die durch zufälliges Zusammentreffen von mehreren Datentransfers auftreten können, sind dadurch nicht betroffen, da hier nicht die maximale Zahl von Verbindungen überschritten wird.

[4] Diese Messung wurde auf einem 1.7 GHz Pentium IV System mit 512 MByte Arbeitsspeicher durchgeführt. Da es sich aber, wie im Folgenden dargelegt, hauptsächlich um ein grundsätzliches Limit handelt, dürfte der Wert auch auf schnelleren Maschinen nicht höher ausfallen.

Im Falle der *slow control* des PT ist eine Begrenzung der Datensatzrate aber unproblematisch, da die zu erwartende Rate im Regelbetrieb um zwei Größenordnungen unter der maximalen Rate liegt. Außerdem wäre eine geringe Anzahl verlorener Datensätze unkritisch, zumal es auf der Seite des Absenders eine entsprechende Fehlermeldung gibt. Im Falle von Sollwerten muss *hptSCcollect* bei einer Fehlermeldung trotzdem sichergestellen, dass die Daten ohne Fehler erneut gesendet werden. In *hptDBhub* wurden Vorkehrungen getroffen, um die Datensatzrate zu limitieren und auch bei Fehlverhalten von *hptSCcollect* einen Absturz durch den sogenannten *denial of service*[5] zu verhindern.

4.4 CORBA-Interface

Zur Extraktion der aufgezeichneten Parameter und Messdaten wurde ein CORBA-Interface entwickelt. CORBA ist eine objektorientierte Zwischenanwendung, die es programmiersprachen- und systemunabhängig erlaubt, Programme zu schreiben, die über ein Netzwerk verteilt sind. Zweckmäßigerweise legt man für größere Projekte ein generisches *Interface* fest, das heißt eine Schnittstellendefinition, die als gemeinsame Grundlage für verschiedene konkrete Schnittstellen dient.

Dieses Interface sollte die Datenauslese für die Analyse vereinheitlichen. Das dazu nötige generische Interface wurde leider nicht bereitgestellt, so dass im Rahmen dieser Arbeit das Interface für den PT nur nicht-standardgemäß entwickelt werden konnte. Wenn das generische Interface aber bereitsteht, ist eine Anpassung auf dieses möglich. Für die exakte Definition des CORBA-Interfaces des PT siehe Anhang D.4.

Auf Basis dieses CORBA-Interfaces wurde zu Testzwecken auch ein Java-Applet[6] geschrieben, das Daten aus der Datenbank abfragen und tabellarisch darstellen kann (Abbildung 4.2).

Aufgrund von Problemen im Zusammenspiel von Java und CORBA wurde jedoch das Programm zur einfachen Darstellung der Daten, *hptDBweb*, nicht auf Basis von CORBA und Java entwickelt. Stattdessen wird ein „Server-seitig" laufendes Programm in der Programmiersprache PHP verwendet, welches eine Internetseite mit einer grafischen Darstellung der Messwerte dynamisch erzeugt.

[5] *denial of service*, engl. Dienstverweigerung. Dieser Begriff bezeichnet einen Zustand, in dem aufgrund von einer Überflutung mit Verbindungen keine neuen Verbindungen zu einem Rechner aufgebaut werden können. Dadurch wird z.B. auch eine Fernwartung zur Behebung des Fehlers unmöglich.

[6] Ein Java-Applet ist ein in einem Internet-Browser laufendes Programm, das in der Programmiersprache Java geschrieben wurde. Solche Programme sollen mit möglichst geringen Voraussetzungen auf jedem Rechner ohne Installation lauffähig sein.

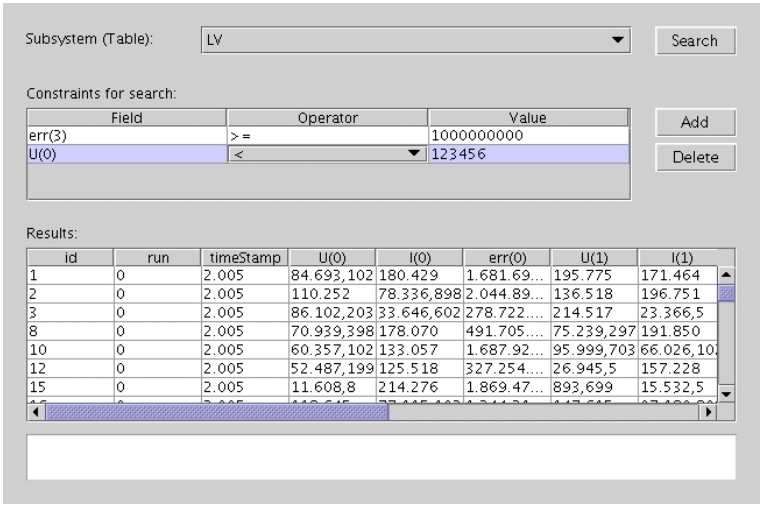


Abbildung 4.2: Java-Applet zur tabellarischen Darstellung der Slow-Control-Daten im Internet-Browser. Dieses Applet wurde im Wesentlichen zum Test des CORBA-Interfaces geschrieben. Die in der Tabelle ausgegebenen Werte sind Zufallszahlen, die testweise in die Datenbank geschrieben wurden.

Dadurch ergeben sich in jedem Fall geringere Voraussetzungen für den zur Darstellung verwendeten Internet-Browser.

Kapitel 5

Zusammenfassung

Die vorliegende Diplomarbeit beschreibt die Simulation kosmischer Myonen sowie die Slow-Control-Datenbank für den *precision tracker* des OPERA-Experiments.

Kosmische Myonen können zur Kalibration des *precision tracker*s verwendet werden. Der Simulation entsprechend ist ihre Rate hoch genug, um in wenigen Tagen die Driftzeit-Orts-Beziehung zu messen. Pro Tag gibt es zirka 650 hierfür geeignete Ereignisse, die sowohl den Trigger auslösen als auch eine Doppelwand des Precision Trackers durchqueren.

Des Weiteren musste geprüft werden, ob kosmische Myonen einen Untergrund für die Messung der Strahlneutrinos darstellen. Aufgrund der Form des abschirmenden Gran-Sasso-Massivs gibt es nahezu keine kosmischen Myonen, die in Strahlrichtung auf den Detektor treffen. Dadurch besteht schon wegen der Winkelverteilung keine Verwechslungsgefahr von kosmischen Myonen mit Strahlereignissen. Insbesondere während der Aufbauphase, in der das Blei-Target noch nicht installiert ist, können so Ereignisse von Strahlneutrinos, die bereits im Fels wechselwirken, identifiziert werden. Ein Vergleich der Winkelverteilungen mit gemessenen Daten ist aufgrund der noch unbekannten exakten Detektorpositionen nicht möglich.

Die Slow-Control-Datenbank zeichnet alle Umgebungsparameter, Fehlermeldungen und Sollwerte des Precision Trackers während der gesamten Messzeit auf. Die dazu nötige Software wurde im Rahmen dieser Diplomarbeit entwickelt und getestet und befindet sich bereits im Einsatz. Die voraussichtliche Größe der Datenbank bei fünf Jahren Messzeit bleibt mit zirka 72 GByte im Rahmen des technisch Realisierbaren. Zusätzlich zur eigentlichen Datenbanksoftware wurden weitere Programme erstellt, die zum Beispiel zur graphischen Darstellung der Messdaten dienen. Alle Programme können an die sich ändernden Anforderungen (zum Beispiel Menge und Art der zu speichernden Informationen) angepasst werden.

Anhang A

Änderungen an der OPERA-Software

A.1 Paket *OpCosmics* (v2r1)

A.1.1 Datei *steeringF.F*

Statt den beiden DATA-Statements

```
      DATA BOX/934.000,1000.0,1117.8/ !dimensioni in cm
      DATA SHIFT/0.01,0.01,0.01/
```

wurden folgende Zeilen eingefügt:

```
!
!     *** These changes have been made to allow interactions with rock (showers) ***
!         (by Martin Hierholzer, 16.11.2006)
!

      parameter         (showerDepth   = 1300.00)              ! all dimensions in cm
      parameter         (showerDepth2  =  showerDepth*2.0)
      parameter         (Z_CLIP_UPSTRM =    0.00)              ! clip hall in upstream direction (to reduce opsim
cpu time)
      parameter         (SIZE_Z_GALC   = 9991.71)
      parameter         (SIZE_X_BASE   = 2153.40)
      parameter         (SIZE_Y_SHIELD =  930.00)
      parameter         (SIZE_R_ROCK   = 1157.00)
!
      parameter         (CPOS_X_OPDY   =  300.00)
      parameter         (CPOS_Y_OPDY   =  411.84)
      parameter         (CPOS_Z_OPDY   = 3273.16)
!
      parameter         (X_BOX = SIZE_X_BASE                +              &
     &                         showerDepth*2.0)
      parameter         (Y_BOX = SIZE_Y_SHIELD + SIZE_R_ROCK +             &
     &                         showerDepth)
      parameter         (Z_BOX = SIZE_Z_GALC - Z_CLIP_UPSTRM +             &
     &                         showerDepth*2.0)

      data BOX          / Z_BOX, X_BOX, Y_BOX /

      parameter         (X_SHIFT = -CPOS_X_OPDY - X_BOX/2.0)
      parameter         (Y_SHIFT = 0.5 * (SIZE_Y_SHIELD + SIZE_R_ROCK)     &
     &                  - CPOS_Y_OPDY + showerDepth/2 - Y_BOX/2.0)
      parameter         (Z_SHIFT = CPOS_Z_OPDY-Z_CLIP_UPSTRM+Z_BOX/2.0)    &
```

```
      data SHIFT        / -Z_SHIFT, X_SHIFT, Y_SHIFT /
```

Durch diese Änderungen wird eine Vergrößerung des Quaders, auf dessen Oberfläche die Myonen erzeugt werden, auf die gesamte Halle ausgedehnt.

Um eine Verschiebung des Quaders zu erlauben, wurde in der Routine *ggpbox* folgende Änderung durchgeführt:

```
c     Commented by Martin Hierholzer, 06.09.2006
c      X = X !+ SHIFT(1)
c      Y = Y !+ SHIFT(2)
c      Z = Z !+ SHIFT(3)
c     Inserted by Martin Hierholzer, 06.09.2006
      X = X + SHIFT(1)
      Y = Y + SHIFT(2)
      Z = Z + SHIFT(3)
```

Des Weiteren wurden eine Änderung genenüber der Version v2 rückgängig gemacht, die das Koordinatensystem inkompatibel zum Rest der OPERA-Software verändert hatten. Dazu wurde Folgendes aus der Routine *steeringf* ersatzlos entfernt:

```
c MS230806: commented 3 lines
c         X(1) = (X(1)-BOX(1)/2. + 313.8)
c         X(2) = (X(2)-BOX(2)/2.)
c         X(3) = (X(3)-BOX(3)/2.)
c MS230806: added 3 lines
         X(1) = X(1)/100.
         X(2) = X(2)/100.
         X(3) = X(3)/100.
```

A.2 Paket *OpSim* (v7)

A.2.1 Datei *opsim.cxx*

Das Hauptprogramm von *OpSim* wurde so modifiziert, dass eine automatisierte Ausführung möglich wird. Einige Parameter können statt in der Konfigurationsdatei nun als Kommandozeilenargumente angegeben werden.

...

```
//
//  Initialize the Root environment
//
extern void InitGui();
VoidFuncPtr_t initfuncs[] = { InitGui, 0 };
TROOT gopsim("gopsim","The OPERA/ROOT Interface for Event Simulation", initfuncs);

//
//  Signal handler for clean exit
//  (by Martin Hierholzer 23.03.2006)
//
void cleanexit(int iarg) {
    opsim->Finalize();
    exit(0);
}
```

```
//
//  Main routine
//
int main(int argc, char **argv)
{
   TApplication theApp("theApp",NULL,NULL);    // Needs to be NULL pointers, or argv cannot be accessed
   // TApplication theApp("theApp",&argc,argv);
//
   opsim = new OpSim("OpSim","The OPERA Simulation");
//
//  Catch some signals to ensure clean shutdown
//  (by Martin Hierholzer 23.03.2006)
//
   signal(SIGINT,cleanexit);
   signal(SIGTERM,cleanexit);
   signal(SIGHUP,cleanexit);
//
//  Set filenames and event numbers by command line (optional, can still be done by ConfigMC.C)
//  (by Martin Hierholzer, 23.03.2006)
//
   if(argc > 1 && argc < 5) {
     std::cout << std::endl
       << " *** Usage: opsim firstEvent numberOfEvents inputFileName outputFileName [volOfPriVert]"
       << std::endl;
     exit(1);
   }
   else if(argc > 1) {
     opsim->GetDataStore()->SetInputFile(argv[3]);
     opsim->GetDataStore()->SetInputTree("TreeMC");
     opsim->GetDataStore()->SetNumberOfEvents(atoi(argv[2])+atoi(argv[1]));
     opsim->GetDataStore()->SetFirstEvent(atoi(argv[1]));
//
     opsim->GetDataStore()->SetOutputFile(argv[4]);
     opsim->GetDataStore()->SetOutputTree("TreeMCH");
//
     if(argc > 5) opsim->GetGeom()->SetVolOfPrimVert(argv[5]);
   }
//
//  Initialize OpSim
//

                                                    ...

}
```

A.2.2 Datei *OpSimDataStore.cxx*

Diese Modifikationen verhindern, dass Ereignisse ohne Treffer im Detektor in die Ausgabedatei geschrieben werden. Hierdurch ergibt sich eine enorme Zeitersparnis, da ansonsten *OpDigit* erheblich mehr Ereignisse verarbeiten müsste.

Weitere Änderungen in dieser Datei dienten lediglich dem schnellen Erstellen von einigen Histogrammen während der Simulation. Dadurch kann ein weiterer (zeitintensiver) Durchlauf zur schnellen Kontrolle der Simulation (insbesondere der Lage der primären Vertizes) vermieden werden.

```
                                                    ...

//_____________________________________________________________________________
OpSimDataStore::OpSimDataStore()
  : TObject() {
  firstEvent=0;
  numberOfEvents=0;
  inputFile=0;
  outputFile=0;
```

```
  treeMC=0;
  treeMCH=0;
  treeManager=0;
  doHistPriVert=0;
  delZeroHitEvt=0;
}
```

...

```
void OpSimDataStore::Finalize() {
//
    cout << "OpSim::Finalize: writing output tree...";
//
//  Output histograms if requested
//  (by Martin Hierholzer, 24.03.2006)
//
    if(doHistPriVert) {
      outputFile->cd();
      hist_zx->Write();
      hist_zy->Write();
      hist_xy->Write();
      hist_zx_hit->Write();
      hist_zy_hit->Write();
      hist_xy_hit->Write();
    }
//
//  Finalize file
//
    outputFile->Write();
//
    cout << " done!" << endl;
}
```

...

```
//
//  Enable histogramming of primary vertices
//  (by Martin Hierholzer, 30.03.2006)
//
void OpSimDataStore::EnableHistPriVert() {
//
    doHistPriVert = 1;
//
//  Create histograms
//  hist_??_hit will contain only those events which hit the detector
//
    hist_zx = new TH2F("hist_zx","Primary vertices in Z/X plane",
                     1300, -10000, 3000, 450, -2000, 2500);
    hist_zy = new TH2F("hist_zy","Primary vertices in Z/Y plane",
                     1300, -10000, 3000, 280,  -800, 2000);
    hist_xy = new TH2F("hist_xy","Primary vertices in X/Y plane",
                      280,   -800, 2000, 450, -2000, 2500);
//
    hist_zx_hit = new TH2F("hist_zx_hit","Primary vertices with detector hit in Z/X plane",
                         1300, -10000, 3000, 450, -2000, 2500);
    hist_zy_hit = new TH2F("hist_zy_hit","Primary vertices with detector hit in Z/Y plane",
                         1300, -10000, 3000, 280,  -800, 2000);
    hist_xy_hit = new TH2F("hist_xy_hit","Primary vertices with detector hit in X/Y plane",
                          280,   -800, 2000, 450, -2000, 2500);
//
    hist_zx->SetBit(TH2::kCanRebin);
    hist_zy->SetBit(TH2::kCanRebin);
    hist_xy->SetBit(TH2::kCanRebin);
    hist_zx_hit->SetBit(TH2::kCanRebin);
    hist_zy_hit->SetBit(TH2::kCanRebin);
    hist_xy_hit->SetBit(TH2::kCanRebin);
//
    cout << "Histogramming of primary vertices enabled!" << endl;
}
```

...

```
void OpSimDataStore::WriteEventOnTree() {
```

...

```
  // Update event header
  m_EvtHeader->SetNVertex(m_MCVertexList->GetSize());
  m_EvtHeader->SetNTrack(gOpSim->GetStack()->GetNtrack());

  //
  // Optionally skip events without any hit
  // (by Martin Hierholzer, 29.03.2006)
  //
  int ic = 0;
  if(delZeroHitEvt || doHistPriVert) {
    if(m_EMULHitList->GetSize()  != 0) ic++;
    if(m_TSCINHitList->GetSize() != 0) ic++;
    if(m_SRPCHitList->GetSize()  != 0) ic++;
    if(m_SDTHitList->GetSize()   != 0) ic++;
    if(m_XPCHitList->GetSize()   != 0) ic++;
    if(m_VETOHitList->GetSize()  != 0) ic++;
    if(ic == 0 && delZeroHitEvt) return;
  }

  //
  // Do optional histogramming
  // (by Martin Hierholzer, 24.03.2006)
  //
  if(doHistPriVert && ic != 0) {
    RVertex* rvertex = (RVertex*) (m_MCVertexList->At(0));
    hist_zx_hit->Fill(rvertex->Z(),rvertex->X());
    hist_zy_hit->Fill(rvertex->Z(),rvertex->Y());
    hist_xy_hit->Fill(rvertex->X(),rvertex->Y());
  }

  // Fill the Tree

                                                                    ...
}
                                                                    ...
```

A.2.3 Datei *OpSimDataStore.h*

```
                                                                    ...
class OpSimDataStore : public TObject
{

                                                                    ...
 private:

                                                                    ...
  //
  // Optional histogramming of primary vertices, see EnableHistPriVert() and Finalize()
  // (by Martin Hierholzer, 30.03.2006)
  //
  int doHistPriVert;
  TH2F *hist_zx, *hist_zy, *hist_xy;
  TH2F *hist_zx_hit, *hist_zy_hit, *hist_xy_hit;

  //
  // Optionally delete (i.e. do not write to output tree) events without any hit in OPERA detector
  // (by Martin Hierholzer, 29.03.2006)
  //
  int delZeroHitEvt;


  ClassDef(OpSimDataStore,1)

};

#endif //OPSIMDATASTORE_H
```

A.3 Konfigurationsdateien

A.3.1 Datei *ConfigCosmics.C*

Hier wurde lediglich die Laufzeit auf 1 Tag gesetzt.

```
#include <iostream>

void ConfigCosmics() {

  // Set Run Number
  // default is 1
  gOpCosmics->SetRunNumber(0);

  // Set equivalent time of cosmics generation,
  // unit=year
  // default is 0.001
  gOpCosmics->SetLiveTime(1./365.);

  // Set charge ratio (mu+/mu_),
  //default is 1.2
  gOpCosmics->SetChargeRatio(1.2);
}
```

A.3.2 Datei *ConfigMC.C*

In dieser Datei wurden zwecks Automatisierung alle Dateinamen und Ereignisnummern entfernt. Diese Angaben erfolgten dann an der Kommandozeile von *OpSim*. Außerdem wurde der Geometriemodus auf *FULL*, d.h. volle Geometrie inkl. Halle, gesetzt.

```
//#include<iostream>

void ConfigMC() {

  ////////////////////////////////
  // Set Input options
  ////////////////////////////////

  // Done by command line

  ////////////////////////////////
  // Set Output options
  ////////////////////////////////

  // gOpSim->GetDataStore()->SetOutputFile("simcosmics.root");
  // gOpSim->GetDataStore()->SetOutputTree("TreeMCH");

  gOpSim->GetDataStore()->EnableHistPriVert();
  gOpSim->GetDataStore()->EnableDelZeroHitEvt();

  ////////////////////////////////
  // Set geometry options
  ////////////////////////////////

  gOpSim->GetGeom()->SetGeometryMode("FULL");
  // gOpSim->GetGeom()->SetGeometryMode("OPERA");

  gOpSim->GetGeom()->SetBricksEmpty();
  gOpSim->GetGeom()->SetFieldMode("CONST");

  gOpSim->GetGeom()->SetModeOfPrimVert("FILE");
  // gOpSim->GetGeom()->SetModeOfPrimVert("RANDOM");

  // gOpSim->GetGeom()->SetVolOfPrimVert("WRLD");
  // gOpSim->GetGeom()->SetVolOfPrimVert("ROCK");
```

```
  // gOpSim->GetGeom()->SetVolOfPrimVert("OPDY");
  // gOpSim->GetGeom()->SetVolOfPrimVert("PBPL");
  // gOpSim->GetGeom()->SetVolCopyNoOfPrimVert(1);

  gOpSim->GetGeom()->SetBrickOn();
  gOpSim->GetGeom()->SetTscinOn();
  gOpSim->GetGeom()->SetSpectroRPCOn();
  gOpSim->GetGeom()->SetDriftTubeOn();
  gOpSim->GetGeom()->SetXPCOn();
  gOpSim->GetGeom()->SetVetoOn();

  ////////////////////////////////////////
  // Choose and init a simulator
  ////////////////////////////////////////

  ConfigG3();

}
```

...

Anhang B

Quellcode der Analyse der Simulation

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//
// TriggerAnalyse.cpp - Trigger Simulation für OPERA HPT - Martin Hierholzer - 06.10.2006
//
// Benötigt initialisiertes OpRelease und (z.B.) OpCosmic-Package (mit setup.sh)
//
////////////////////////////////////////////////////////////////////////////////////////////////////
#include "TSystem.h"
#include "TFile.h"
#include "TTree.h"
#include "TString.h"
#include <TTree.h>
#include <TLeaf.h>
#include <TBranch.h>
#include <TCanvas.h>
#include <TH1.h>
#include <TH2.h>
#include <TH3.h>
#include "TParticlePDG.h"

#include "OpData/SDTDigit.h"

#include "OpRData/TreeManager.h"
#include "OpRData/RRunHeader.h"
#include "OpRData/REvtHeader.h"
#include "OpRData/RParticle.h"
#include "OpRData/RVertex.h"
#include "OpRData/RSRPCHit.h"
#include "OpRData/RSDTHit.h"
#include "OpRData/RXPCHit.h"
#include "OpRData/RSRPCDigit.h"
#include "OpRData/RSDTDigit.h"
#include "OpRData/RXPCDigit.h"

#include <stdlib.h>
//
//  Konstanten (Achtung, teilweise nicht ohne Weiteres zu verändern!)
//
const int   nplnpsm = 11;                           // RPC-Planes pro Sub-Modul
const int   nsubmod = 2;                            // Submodule pro Super-Modul
const int   nmoduls = 2;                            // Module
const int   nplanes = nplnpsm*nsubmod*nmoduls;      // RPC-Planes insgesamt
const int   nxpcpln = 2;                            // 2 XPC-Planes pro Supermodul
const int   nhptpln = 6;                            // HPT planes pro Supermodul
const int   nhptlay = 4;                            // HPT layers pro plane
//
const int   npossbl = 6;                            // Zahl der unabh. Trigger
//const int   npossbl = 3;                                  // Zahl der unabh. Trigger
const int   nmajrty = 2;                            // n-of-m-Majority
const int   mmajrty = 3;                            // n-of-m-Majority
```

```
//
const float effrpcs = 0.96;                          // RPC efficiency [prüfen !]
const float effxpcs = 0.96;                          // XPC efficiency [erraten !!!]
//
const int   strtidx[npossbl] = {0, 12, 21, 24, 36, 45}; // Startindizes der Triggerplanes
//const int   strtidx[npossbl] = {0, 12, 21};             // Startindizes der Triggerplanes
//
const int   mbranchs = 1;                            // Maximale Zahl der Branches
//
//  Globale Variablen
//
TH1D           *hist_polr, *hist_azim, *hist_beam, *hist_polr_trig, *hist_azim_trig, *hist_beam_trig;
TH2D           *hist_enrg_polr, *hist_enrg_azim, *hist_enrg_beam, *hist_polr_azim;
TH1D           *hist_maxtime, *hist_prietrg, *hist_prietrS;
TH1I           *hist_maxdens, *hist_hpthits, *hist_hitpdig, *hist_maxhitpdig;
TH1I           *hist_layptrg, *hist_layptrS, *hist_dblptrg, *hist_dblptrS;
//
////////////////////////////////////////////////////////////////////////////////////////////////////
//
// Hauptprogramm
//
int main() {
   int         ntrig0[npossbl][mbranchs],             // Zähler für Trigger-Events
               ntrig1[npossbl][mbranchs],
               ntrig2[npossbl][mbranchs],
               ntrig3[npossbl][mbranchs],
               ntrigS[npossbl][mbranchs],
               nevnts[mbranchs],                     // Zahl der Events im Branch
               nmuons[mbranchs];                     // Zahl der Myonen insgesamt
   int         muster0[nplanes+nmoduls*nxpcpln],      // Muster der getroffenen Planes
               muster1[nplanes+nmoduls*nxpcpln],
               musterD[nhptlay*nhptpln*nmoduls];
//
   int         ipl,ism,index,index2, val, ibranch,nbranchs, notrig;
   int         n0=0,n1=0,n2=0,n3=0;                  // Getriggerte primaries überhaupt, mit E >= 23,8, E >= 30
bzw. E >= 45 GeV
   int         nS=0,nSt=0;                           // Schauer und getriggerte Schauer
   int         nR=0,nRs=0,nRu=0;                     // Getriggerte Events mit getroffenen double planes (je mind.
3 Hits/Plane)
//
   float       pi = 3.141592654;
   float       polr,azim,beam;
   float       px,py,pz, w, energy;
//
   FILE        *fproto, *fliste;
   char        filna[1024];
//
//  Initialisierung
//  ***************
//
   srand(time(0));
//
//  Arrays löschen
//
   for(int i=0;i<mbranchs;i++) {
     for(int k=0;k<npossbl;k++) {
       ntrig0[k][i] = 0;
       ntrig1[k][i] = 0;
       ntrig2[k][i] = 0;
       ntrig3[k][i] = 0;
       ntrigS[k][i] = 0;
     }
     nevnts[i] = 0;
     nmuons[i] = 0;
   }
   nbranchs = 1;
//
//  Dateiliste öffnen
//
   fliste = fopen("TriggerAnalyse.lst","r");
//
//  Ausgabedatei öffnen
//
   TFile* outfile = new TFile("TriggerAnalyse.root","RECREATE");
   outfile->cd();
//
//  Create histograms
//
   hist_polr = new TH1D("hist_polr","Polar angle",                                  45, 90, 180);
   hist_azim = new TH1D("hist_azim","Azimutal angle",                               90,  0, 360);
```

```
    hist_beam = new TH1D("hist_beam","Angle to beam axis",                              90,  0, 180);
    hist_polr_trig = new TH1D("hist_polr_trig","Polar angle of triggered events",       45, 90, 180);
    hist_azim_trig = new TH1D("hist_azim_trig","Azimuth angle of triggered events",     90,  0, 360);
    hist_beam_trig = new TH1D("hist_beam_trig","Angle to beam axis of triggered events", 90,  0, 180);
    hist_enrg_polr = new TH2D("hist_enrg_polr","Energy of primaries over polar angle of triggered events",
                              45, 90, 180,  100, 0, 100);
    hist_enrg_azim = new TH2D("hist_enrg_azim","Energy of primaries over azimuth angle of triggered events",
                              90,  0, 360,  100, 0, 100);
    hist_enrg_beam = new TH2D("hist_enrg_beam","Energy of primaries over angle to beam axis of triggered events",
                              90,  0, 180,  100, 0, 100);
    hist_polr_azim = new TH2D("hist_polr_azim","Azimuth vs. polar angle of triggered events",
                              90,  0, 360,  45, 90, 180);
//
    hist_polr->SetBit(TH2::kCanRebin);
    hist_azim->SetBit(TH2::kCanRebin);
    hist_beam->SetBit(TH2::kCanRebin);
    hist_polr_trig->SetBit(TH2::kCanRebin);
    hist_azim_trig->SetBit(TH2::kCanRebin);
    hist_beam_trig->SetBit(TH2::kCanRebin);
    hist_enrg_polr->SetBit(TH2::kCanRebin);
    hist_enrg_azim->SetBit(TH2::kCanRebin);
    hist_enrg_beam->SetBit(TH2::kCanRebin);
    hist_polr_azim->SetBit(TH2::kCanRebin);
//
    hist_maxtime = new TH1D("hist_maxtime","Time of last hit [ns]", 100, 0, 1000);
    hist_maxtime->SetBit(TH1::kCanRebin);
    hist_maxdens = new TH1I("hist_maxdens","Maximum particle density [arb. units]", 200, 0, 200);
    hist_maxdens->SetBit(TH1::kCanRebin);
    hist_hpthits = new TH1I("hist_hpthits","Number of Hits per Event in HPT", 500, 0, 500);
    hist_hpthits->SetBit(TH1::kCanRebin);
    hist_hitpdig = new TH1I("hist_hitpdig","Number of Hits per Digit in HPT", 500, 0, 500);
    hist_hitpdig->SetBit(TH1::kCanRebin);
    hist_layptrg = new TH1I("hist_layptrg","Number of HPT layers hit in triggered Events (without showers)", 500, 0, 500);
    hist_layptrS = new TH1I("hist_layptrS","Number of HPT layers hit in triggered Events (only showers)", 500, 0, 500);
    hist_dblptrg = new TH1I("hist_dblptrg","Number of HPT double planes hit in triggered Events (without showers)", 500, 0,
500);
    hist_dblptrS = new TH1I("hist_dblptrS","Number of HPT double planes hit in triggered Events (only showers)", 500, 0, 500);
    hist_prietrg = new TH1D("hist_prietrg","Energy/GeV of primary particle in triggered Events (without showers)", 2000, 0,
2000);
    hist_prietrg->SetBit(TH1::kCanRebin);
    hist_prietrS = new TH1D("hist_prietrS","Energy/GeV of primary particle in triggered Events (only showers)", 2000, 0, 2000);
    hist_prietrS->SetBit(TH1::kCanRebin);
    hist_maxhitpdig = new TH1I("hist_maxhitpdig","Maximum number of Hits per Digit in HPT", 500, 0, 500);
    hist_maxhitpdig->SetBit(TH1::kCanRebin);
//
//  Schleife, bis alle Trees aus Liste verarbeitet
//  **********************************************
//
    while(!feof(fliste)) {
//
//    Öffnen des Trees
//    ================
//
//    Dateinamen aus Liste lesen
//
     fscanf(fliste,"%s\n",filna);
     cout << "Processing input file " << filna << endl;
//
//    Daten-Datei öffnen
//
     TFile* treefile = TFile::Open(filna);
     ibranch = 1;
     TTree* theTree = (TTree*)(gROOT->FindObject("TreeMCD"));
     TreeManager *treeManager = TreeManager::giveThis(theTree,"read");
//
     Int_t numberOfEvents = theTree->GetEntries();
     cout << "Number of Events: " << numberOfEvents << endl;
//
//    Event-Schleife
//    **************
//
     for(int ievent=0; ievent<numberOfEvents; ievent++) {
//
//      Listen für Event holen
//
       treeManager->clear();
       TList *SRPCDigitsList  = treeManager->SRPCDigitList(ievent);
       TList *XPCDigitsList   = treeManager->XPCDigitList(ievent);
       TList *SDTDigitsList   = treeManager->SDTDigitList(ievent);
       TList *SDTHitsList     = treeManager->SDTHitList(ievent);
```

```
      TList *SDTHitsUsedList = treeManager->SDTHitUsedList(ievent);
      TList *PartList        = treeManager->MCParticleList(ievent);
      TList *PartUsedList    = treeManager->MCParticleUsedList(ievent);
//
//       Längen der Listen holen
//
      Int_t numberOfSRPCDigits = SRPCDigitsList->GetSize();
      Int_t numberOfXPCDigits  = XPCDigitsList->GetSize();
      Int_t numberOfSDTDigits  = SDTDigitsList->GetSize();
//
//       Muster-Arrays löschen
//
      for(Int_t l=0;l<nplanes+2*nxpcpln;l++) { muster0[l] = 0; muster1[l] = 0; }
      for(Int_t l=0;l<nhptlay*nhptpln*nmoduls;l++) musterD[l] = 0;
//
//       DT-Hits nach Schauern untersuchen
//       ---------------------------------
//
      TH1I *hist_hitpdig2 = new TH1I("hist_hitpdig2","Number of Hits per Digit in HPT", 500, 0, 500);
      hist_hitpdig2->SetBit(TH1::kCanRebin);

      RParticle *primaryParticle = (RParticle*) PartList->At(0);
      float primaryEnergy = primaryParticle->Energy();

      int iMaxHits = 0, isSchauer = 0;

      for(int iSDTDigit=0; iSDTDigit<numberOfSDTDigits; iSDTDigit++) {
        RSDTDigit     *rSDTDigit = (RSDTDigit*)(SDTDigitsList->At(iSDTDigit));
        hist_hitpdig->Fill(rSDTDigit->Hit()->GetEntriesFast());
        hist_hitpdig2->Fill(rSDTDigit->Hit()->GetEntriesFast());
        if(iMaxHits < rSDTDigit->Hit()->GetEntriesFast()) iMaxHits = rSDTDigit->Hit()->GetEntriesFast();
//          RParticle     *localParticle = rSDTDigit->Hit(0)->LocalParticle();
//
        int iHptPlane = (rSDTDigit->Plane()-1) + (rSDTDigit->SM()-1)*nhptlay*nhptpln;
        if(iHptPlane < 0 || iHptPlane > nhptlay*nhptpln*nmoduls-1) {
          cout << "*** Falsche HPT plane: " << iHptPlane << endl;
        }
        else {
          musterD[iHptPlane]++;
        }
      }

      hist_maxhitpdig->Fill(iMaxHits);

      int iHptLayersHit = 0;
      int iHptLayersHitPerPlaneLast = 0;
      int iHptDoublePlanesHit = 0;
      for(Int_t l=0;l<nhptpln*nmoduls;l++) {
        int iHptLayersHitPerPlane = 0;
        for(Int_t m=0;m<nhptlay;m++) {
          if(musterD[l*nhptlay + m] > 0) {
            iHptLayersHitPerPlane++;
            iHptLayersHit++;
          }
        }
        if(l % 2 == 0 && iHptLayersHitPerPlane >= 3 && iHptLayersHitPerPlaneLast >= 3) {
          iHptDoublePlanesHit++;
        }
        iHptLayersHitPerPlaneLast = iHptLayersHitPerPlane;
      }

      if(iMaxHits > 1) isSchauer = 1;
      if(isSchauer == 1) nS++;
      if(isSchauer == 0 && iHptDoublePlanesHit > 0) nRu++;

      hist_hitpdig2->Clear();
      hist_hitpdig2->Delete();
//
//       RPCs in Muster-Arrays eintragen
//       -------------------------------
//
      for(int iSRPCDigit=0; iSRPCDigit<numberOfSRPCDigits; iSRPCDigit++) {
        RSRPCDigit *rSRPCDigit = (RSRPCDigit*)(SRPCDigitsList->At(iSRPCDigit));
        ipl = rSRPCDigit->Plane();
        ism = rSRPCDigit->SM();
//
        if( (ipl < 1) || (ipl > nsubmod*nplnpsm) ) {
          cout << "Ungültige RPC-Plane " << ipl << "!" << endl;
          exit(1);
        }
```

```
        if( (ism < 1) || (ism > nmoduls) ) {
          cout << "Ungültige Supermodul-Nummer " << ism << "!" << endl;
          exit(1);
        }
//
        index = ipl-1 + nxpcpln + (ism-1)*(nplnpsm*nsubmod+nxpcpln);
//          cout << "RPC: " << index << endl;
//
        muster0[index]++;
        if( ((float) rand()) / ((float) RAND_MAX) <= effrpcs ) muster1[index]++;
      }
//
//      XPCs in Muster-Arrays eintragen
//      -------------------------------
//
      for(int iXPCDigit=0; iXPCDigit<numberOfXPCDigits; iXPCDigit++) {
        RXPCDigit *rXPCDigit = (RXPCDigit*)(XPCDigitsList->At(iXPCDigit));
        ipl = rXPCDigit->Plane();
        ism = rXPCDigit->SM();
//
        if( (ipl < 1) || (ipl > nsubmod*nxpcpln) ) {
          cout << "Ungültige XPC-Plane " << ipl << "!" << endl;
          exit(1);
        }
        if( (ism < 1) || (ism > nmoduls) ) {
          cout << "Ungültige Supermodul-Nummer " << ism << "!" << endl;
          exit(1);
        }
//
        index = ipl-1 + (ism-1)*(nplnpsm*nsubmod + nxpcpln);
//          cout << "XPC: " << index << endl;
//
        muster0[index]++;
        if( ((float) rand()) / ((float) RAND_MAX) <= effxpcs ) muster1[index]++;
      }
//
//      Winkel berechnen
//      ----------------
//
      RParticle *part = (RParticle*) PartList->At(0);
      if(!part) {
        cout << "*** part == NULL" << endl;
        continue;
      }
      px = part->Pz();
      py = part->Px();
      pz = part->Py();
      beam = (pi/2. - atan(-px/sqrt(py*py + pz*pz)));
      polr = (pi/2. - atan(pz/sqrt(px*px + py*py)));
      if(py < 0) {
        azim = 2.*pi - acos(px/sqrt(px*px + py*py));
      } else {
        azim = acos(px/sqrt(px*px + py*py));
      }
      w = 0.5 / sin(pi-polr);             // 0.5 kommt aus Integral über sin im Winkelbereich 0 bis pi
      hist_polr->Fill(polr*180./pi, w);
      hist_azim->Fill(azim*180./pi);
      w = 0.5 / sin(pi-beam);
      hist_beam->Fill(beam*180./pi,w);
//
//      Energie berechnen
//
      energy = sqrt( px*px + py*py + pz*pz );
//
//      Triggerbedingungen anwenden
//      ---------------------------
//
      notrig = 1;
      int histoDone = 0;
      for(index=0; index<npossbl; index++) {
        int nhit = 0;
        val = 0;
        for(index2=strtidx[index]; index2<(strtidx[index]+mmajrty); index2++) {
          val += (muster0[index2] > 0) ? 1 : 0;
          if(muster0[index2] > nhit) nhit = muster0[index2];
        }
        if(val >= nmajrty) {
          notrig = 0;
          ntrig0[index][ibranch-1]++;
        }
```

```
        if(val >= 1) ntrig2[index][ibranch-1]++;
//
        val = 0;
        for(index2=strtidx[index]; index2<(strtidx[index]+mmajrty); index2++) {
          val += (muster1[index2] > 0) ? 1 : 0;
        }
        if(val >= nmajrty) {
          ntrig1[index][ibranch-1]++;
          if(isSchauer) ntrigS[index][ibranch-1]++;
//
//          Winkelverteilung, Zahl der getroffenen Planes, Energie
//
          if(histoDone == 0) {
            w = 0.5 / sin(pi-polr);
            hist_polr_trig->Fill(polr*180./pi, w);
            hist_azim_trig->Fill(azim*180./pi);
            w = 0.5 / sin(pi-beam);
            hist_beam_trig->Fill(beam*180./pi, w);
//
            w = 0.5 / sin(pi-polr);
            hist_enrg_polr->Fill(polr*180./pi, energy, w);
            hist_enrg_azim->Fill(azim*180./pi, energy);
            w = 0.5 / sin(pi-beam);
            hist_enrg_beam->Fill(beam*180./pi, energy, w);
//
            w = 0.5 / sin(pi-polr);
            hist_polr_azim->Fill(azim*180./pi, polr*180./pi, w);
//
            if(!isSchauer) {
              hist_layptrg->Fill(iHptLayersHit);
              hist_dblptrg->Fill(iHptDoublePlanesHit);
              hist_prietrg->Fill(primaryEnergy);
              n0++;
              if(primaryEnergy >= 23.8) n1++;
              if(primaryEnergy >= 30.0) n2++;
              if(primaryEnergy >= 45.0) n3++;
              if(iHptDoublePlanesHit > 0) nR++;
            }
            else {
              hist_layptrS->Fill(iHptLayersHit);
              hist_dblptrS->Fill(iHptDoublePlanesHit);
              hist_prietrS->Fill(primaryEnergy);
              nSt++;
              if(iHptDoublePlanesHit > 0) nRs++;
            }
//
            histoDone = 1;
          }
        }
//
        val = 0;
        for(index2=strtidx[index]; index2<(strtidx[index]+2); index2++) {
          val += (muster1[index2] > 0) ? 1 : 0;
        }
        if(val == 2) {
          ntrig3[index][ibranch-1]++;
        }
      }
//
//      Events mitzählen
//
      nevnts[ibranch-1]++;
    }
//
//    Datei schließen
//
    treefile->Close();
    TreeManager::releaseThis();
  }
//
//  Zählungen ausgeben
//  *****************
//
  fproto = fopen("TriggerAnalyse.txt","w");
  printf(        "Events  Trig  ntrig0  ntrig1  ntrig2  ntrig3  t1/t0   t1/t2\n");
  fprintf(fproto,"Events  Trig  ntrig0  ntrig1  ntrig2  ntrig3  t1/t0   t1/t2\n");
//
  for(int k=1;k<=nbranchs;k++) {
    float   ea,eb;
    ea = (ntrig0[0][k-1] != 0 ? (float) ntrig1[0][k-1] / (float) ntrig0[0][k-1] : -1);
```

```
      eb = (ntrig2[0][k-1] != 0 ? (float) ntrig1[0][k-1] / (float) ntrig2[0][k-1] : -1);
      printf(        "%6d  %4d  %6d  %6d  %6d  %6d  %6.4f  %6.4f\n",
                     nevnts[k-1],0,
                     ntrig0[0][k-1],ntrig1[0][k-1],ntrig2[0][k-1],ntrig3[0][k-1],ea,eb);
      fprintf(fproto,"%6d  %4d  %6d  %6d  %6d  %6d  %6.4f  %6.4f\n",
                     nevnts[k-1],0,
                     ntrig0[0][k-1],ntrig1[0][k-1],ntrig2[0][k-1],ntrig3[0][k-1],ea,eb);
      for(int i=1;i<npossbl;i++) {
        ea = (ntrig0[i][k-1] != 0 ? (float) ntrig1[i][k-1] / (float) ntrig0[i][k-1] : -1);
        eb = (ntrig2[i][k-1] != 0 ? (float) ntrig1[i][k-1] / (float) ntrig2[i][k-1] : -1);
        printf(         "        %4d  %6d  %6d  %6d  %6d  %6.4f  %6.4f\n",
                       i,ntrig0[i][k-1],ntrig1[i][k-1],ntrig2[i][k-1],ntrig3[i][k-1],ea,eb);
        fprintf(fproto,"        %4d  %6d  %6d  %6d  %6d  %6.4f  %6.4f\n",
                       i,ntrig0[i][k-1],ntrig1[i][k-1],ntrig2[i][k-1],ntrig3[0][k-1],ea,eb);
      }
    }
//
    cout << "Getriggerte primaries insgesamt:                                  " << n0  << endl;
    cout << "Schauer insgesamt:                                                " << nS  << endl;
    cout << "Getriggerte Schauer:                                              " << nSt << endl;
    cout << "Getriggerte primaries mit E >= 23.8 GeV:                          " << n1  << endl;
    cout << "Getriggerte primaries mit E >= 30.0 GeV:                          " << n2  << endl;
    cout << "Getriggerte primaries mit E >= 45.0 GeV:                          " << n3  << endl;
    cout << "Getriggerte Evts mit getr. HPT double planes (ohne Schauer):       " << nR  << endl;
    cout << "Getriggerte Evts mit getr. HPT double planes (nur Schauer):        " << nRs << endl;
    cout << "Alle Evts mit getr. HPT double planes (ohne Schauer):              " << nRu << endl;
//
//  Ende
//  ****
//
    outfile->cd();
    hist_polr->Write();
    hist_azim->Write();
    hist_beam->Write();
    hist_polr_trig->Write();
    hist_azim_trig->Write();
    hist_beam_trig->Write();
    hist_enrg_polr->Write();
    hist_enrg_azim->Write();
    hist_enrg_beam->Write();
    hist_polr_azim->Write();
    hist_maxtime->Write();
    hist_maxdens->Write();
//    hist_hpthits->Write();
    hist_hitpdig->Write();
    hist_maxhitpdig->Write();
    hist_layptrg->Write();
    hist_layptrS->Write();
    hist_dblptrg->Write();
    hist_dblptrS->Write();
    hist_prietrg->Write();
    hist_prietrS->Write();
//
    fclose(fproto);
    fclose(fliste);
    return 0;
}
```

Anhang C

Quellcode der Simulation zum Triggerrauschen

```
////////////////////////////////////////////////////////////////////////////////////////
//                                                                                    //
//  noise.c - 17.01.2006 - Martin Hierholzer                                          //
//                                                                                    //
//  Noise-Rate bei 2-of-3-Majority-Trigger                                            //
//                                                                                    //
////////////////////////////////////////////////////////////////////////////////////////
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <time.h>

float rnd() {
  return ((float) rand()) / ((float) RAND_MAX);
}

int main() {
    const float     fnois = 1000.0;                      // Frequenz des Rauschens [Hz]
    const float     tsig = 0.000000200;                  // Länge eines Signals [s]
//
    const int       nruns = 100000;                      // Zahl der Runs
    const int       nmc = 100;                           // Zahl der MC-Daten
    const int       npl = 3;                             // Zahl der Trigger-Planes
    float          mc[npl][nmc];                        // MC-Daten (Zeiten der Signale)
//
//  Initialisieren
//  ==============
//
    int     ntrig = 0;
//
    float   tmc;                                        // Länge der MC [s]
    tmc = (float) nmc / fnois;
//
    srand(time(0));
//
    printf("RAND_MAX = %d\n",RAND_MAX);
    printf("Relative Granularität: %f\n",tmc/(float)RAND_MAX/tsig);
//
//  Schleife für Runs
//  =================
//
    for(int irun=0;irun<nruns;irun++) {
//
//    MC-Daten generieren
//
      for(int i=0;i<nmc;i++) {
        for(int k=0;k<npl;k++) {
          mc[k][i] = rnd() * tmc;
        }
```

```
    }
//
//    Überlappungen zählen
//
    for(int i1=0;i1<nmc;i1++) {
      int ic = 0;
      for(int k1=0;k1<npl-1;k1++) {
        for(int i2=0;i2<nmc;i2++) {
          for(int k2=k1+1;k2<npl;k2++) {
            if( ( mc[k1][i1] + tsig > mc[k2][i2]) && (mc[k2][i2] > mc[k1][i1] - tsig) ) ic++;
          }
        }
      }
      if(ic > 0) ntrig++;
    }
  }
//
//  Ende
//  ====
//
   printf("Länge der Daten: %f\n",tmc*(float)nruns);
   printf("Trigger-Rate: %f +- %f\n",(float) ntrig / (tmc*(float)nruns),1.0 / sqrt((double) ntrig)*(float) ntrig /
(tmc*(float)nruns));
   return 0;
}
```

Anhang D

Dokumentation der Slow Control Datenbank

D.1 Protokoll für die Datenübertragung

Für die Übertragung der Daten zwischen *hptDBhub* und *hptSCcollect* wird ein einfaches Protokoll verwendet. Der eigentlichen Datenübertragung liegen TCP/IP-Sockets zugrunde, wie sie unter Linux und Windows kompatibel zueinander vorhanden sind. Für jeden Datensatz, der zwischen den Programmen übertragen wird, wird eine neue Verbindung (das heißt auch ein neuer Socket) aufgebaut und nach der Übertragung wieder geschlossen. Dadurch werden Probleme zum Beispiel nach einem Stromausfall von vornherein vermieden.

Das Protokoll besitzt zunächst einen Kopf:

```
typedef struct _tHeader tHeader;
typedef tHeader *ptHeader;
struct _tHeader {
  int           iVersion;          // Version of protocol
  int           iSource;           // ID of data source (i.e. table number)
  int           iFlags;            // Bitmask of valid fields (i.e. columns).
                                   // The lowest bit represents the first data field
                                   // in all channels!
  int           iTimeStamp;        // Time stamp of measurement
  int           iRun;              // Current run number
  unsigned int  nBytes;            // Number of data bytes following this header
};
```

Auf diesen Kopf folgen die Datenbytes, exakt in der Reihenfolge und mit dem Typ, wie in der Datei *hptDB.xml* festgelegt. Alle Daten (auch im Kopf) werden in der beim PC üblichen *little-endian* Schreibweise notiert, d.h. das niederwertigste Byte liegt zuerst im Speicher.

D.2 Funktionen für die Datenübertragung

Die Umsetzung dieses Protokolls erfolgt in speziellen C-Funktionen, für die auch eine VisualBasic-Schnittstelle geschrieben wurde. Dies ist für eine Integration mit dem unter anderem für das Gassystem verwendete Automationssystem der Firma Beckhoff nötig. Da die kompletten Funktionen den Rahmen dieser Arbeit sprengen würden, wird im Folgenden nur die Funktionsweise der beiden Funktionen der VisualBasic-Schnittstelle anhand des Kommentarblocks aus ihrem Dateikopf dokumentiert.

D.2.1 ipcSendData

```
// ============================================================================
//
//  ipcSendData.c - 14.12.2005 - Martin Hierholzer
//
//  Inter process communication: Send data package
//  See ipcsend for additional details!
//
//  This routine can be called from Visual Basic with the following decleration:
//
//  Private Declare Sub ipcSendData Lib "ipcdll.dll" Alias "_ipcSendData@44"
//    (ByVal target As String, ByVal iSource As Long,
//     ByVal iRun As Long, ByVal mchn As Long,
//     ByVal nF As Long, ByVal nI As Long, ByRef mF As Long, ByRef mI As Long,
//     ByRef fval As Single, ByRef ival As Long, ByRef icode As Long)
//
//  fval and ival needs to be the first elements of the arrays!
//
//  Arguments:
//    target      [in]  String identifying the target (hostname:port)
//    iSource     [in]  Source ID (= Number of Table)
//    iRun        [in]  Run number
//    mchn        [in]  Maximum number of channels, i.e. max(mF,mI)
//    nF, nI      [in]  Number of fields of appr. type
//    mF, mI      [in]  Arrays of number of channels per field
//    fval        [in]  Array with "Single" data of dimension (nF,mchn)
//    ival        [in]  Array with "Long" data of dimension (nI,mchn)
//    icode       [out] (-inf, 0(     connection error code
//                      (0, +inf(     reply code of remote
//
// ============================================================================
```

D.2.2 ipcRequestData

```
// ============================================================================
//
//  ipcRequestData.c - 01.03.2006 - Martin Hierholzer
//
//  Inter process communication: Request data package
//  See ipcrequest for additional details!
```

```
//
//  This routine can be called from Visual Basic 6 with the following decleration:
//
//  Private Declare Sub ipcRequestData Lib "ipcdll.dll" Alias "_ipcRequestData@40"
//    (ByVal target As String, ByVal iSource As Long, ByVal mchn As Long,
//     ByVal nF As Long, ByVal nI As Long, ByRef mF As Long, ByRef mI As Long,
//     ByRef fval As Single, ByRef ival As Long, ByRef icode As Long)
//
//  fval and ival needs to be the first elements of the arrays!
//
//  Arguments:
//    target      [in]  String identifying the target (hostname:port)
//    iSource     [in]  Source ID (= Number of Table)
//    mchn        [in]  Maximum number of channels, i.e. max(mF,mI)
//    nF, nI      [in]  Number of fields of appr. type
//    mF, mI      [in]  Arrays of number of channels per field
//    fval        [out] Array with "Single" data of dimension (nF,mchn)
//    ival        [out] Array with "Long" data of dimension (nI,mchn)
//    icode       [out] (-inf, 0(     connection error code
//                      (0, +inf(     reply code of remote
//
// ================================================================================
```

D.3 XML-Datei zur Datenbankbeschreibung

Die XML-Datei zur Beschreibung der Datenbankstruktur beinhaltet für jede Tabelle eine eigenes `<table>`-Element. Dieses gibt zunächst mit den Attributen `name` und `tmax` den Tabellennamen sowie den maximalen Zeitabstand zwischen zwei Schreibvorgängen in Sekunden an. Innerhalb des `<table>`-Elements gibt es für jede Messgröße ein `<col>`-Unterelement. Die Attribute `name`, `type`, `nchn` und `thrs` geben den Variablennamen und -typ (`float` oder `int`), sowie die Zahl der Kanäle und den Schwellwert, um den der Wert maximal abweichen darf ohne neu geschrieben zu werden, an. Das Attribut `bckhfname` wird von *hptSCcollect* verwendet und gibt ggf. den zugehörigen Variablennamen im Beckhoff-System an.

Die Reihenfolge der Felder und Tabellen ist relevant, da hieraus die Reihenfolge bei der Übertragung und die Tabellennummer (vgl. Argument `iSource` bei den beiden Funktionen in Abschnitt D.2) abgeleitet wird.

```
<database>
<!--

  !!! Always all float fields first, than all int fields !!!
  !!! Order of fields AND tabels is important !!!

  This version is for the test setups only!
  Database host is neutrino.selfip.org:3000

  Last change: 02.03.2006

  Latest version can be downloaded from:
  http://opera.neutrino.selfip.org/hptDB.xml

-->
```

```
  <table name="LV"         tmax="600.0">
    <col name="U"      type="float" nchn="4"     thrs="0.1"   bckhfname=""/>
    <col name="I"      type="float" nchn="4"     thrs="0.1"   bckhfname=""/>
    <col name="err"    type="int"   nchn="4"     thrs="0"     bckhfname=""/>
  </table>
  <table name="HV"         tmax="600.0">
    <col name="U"      type="float" nchn="48"    thrs="1.0"   bckhfname=""/>
    <col name="I"      type="float" nchn="48"    thrs="1.0"   bckhfname=""/>
    <col name="err"    type="int"   nchn="48"    thrs="0"     bckhfname=""/>
  </table>
  <table name="HVnominal"  tmax="30758400.0">
    <col name="U"      type="float" nchn="48"    thrs="1.0"   bckhfname=""/>
    <col name="I"      type="float" nchn="48"    thrs="1.0"   bckhfname=""/>
  </table>
  <table name="gas"        tmax="600.0">
    <col name="O2"     type="float" nchn="4"     thrs="0.1"   bckhfname=""/>
    <col name="p"      type="float" nchn="5"     thrs="0.001" bckhfname="PRESSURE.APRESSURE"/>
    <col name="T"      type="float" nchn="4"     thrs="0.1"   bckhfname=""/>
    <col name="flow"   type="float" nchn="16"    thrs="0.1"   bckhfname="MFC.AMFC"/>
    <col name="valve"  type="int"   nchn="1"     thrs="0"     bckhfname=""/>
    <col name="O2err"  type="int"   nchn="4"     thrs="0"     bckhfname=""/>
  </table>

[...]

</database>
```

D.4 CORBA-Interface

Ein CORBA-Interface wird in einer speziellen Sprache beschrieben, der sogenannten *interface definition language* (IDL). Die Realisierung des Interfaces erfolgt dann in einer beliebigen Programmiersprache, in diesem Fall wurde dafür C++ gewählt. Da eine vollständige Wiedergabe der Implementation des Interfaces den Rahmen dieser Arbeit sprengen würde, sei hier nur die Interface Definition wiedergegeben:

```
//============================================================================
//
//  hptDBcorba.idl - 23.01.2006 - Martin Hierholzer
//
//  IDL for database access to OPERA HPT slow control data
//
//============================================================================

module hptDBcorba {

//
//============================================================================
// Data types etc.
//

  exception InvalidArgument {
    long    errorCode;
    string  description;
  };

  enum tSubsystem { LV, HV, GAS, SUPPORT, TDC, BECKHOFF, TESTPULSE, THRESHOLDS };

//----------------------------------------------------------------------------
// tRow struct
//

  enum tDataType { FRACTIONAL, INTEGER };

  union tValue switch(tDataType) {
    case FRACTIONAL: float   fVal;
    case INTEGER:    long    lVal;
  };
  typedef sequence <tValue>        tValues;
```

```
  struct tField {
    string      name;
    tDataType   dataType;
    tValues     values;
  };
  typedef sequence <tField>         tFields;

  struct tFieldTitle {
    string      name;
    tDataType   dataType;
    long        nChannels;
  };
  typedef sequence <tFieldTitle>    tFieldTitles;

  struct tRow {
    long        id;
    short       run;
    string      time;
    tFields     fields;
  };
  typedef sequence <tRow>           tRows;

//---------------------------------------------------------------------------
// tRequest struct
//

  enum tSortDirection { ASCENDING, DESCENDING };
  enum tOperator { LESS, LESSEQUAL, EQUAL, GREATEREQUAL, GREATER, NOTEQUAL };

  struct tConstraint {
    string      column;
    tOperator   operator;
    string      value;
  };
  typedef sequence <tConstraint>    tConstraints;

  struct tRequest {
    tSubsystem                      subsystem;
    string                          sortColumn;
    tSortDirection                  sortDirection;
    tConstraints                    constraints;
  };

//
//===========================================================================
// Interfaces
//

  interface hptDB {
    void getFieldTitles(in tSubsystem subsystem, out tFieldTitles fieldTitles)
         raises(InvalidArgument);
    void getData(in tRequest request, out tRows rows)
         raises(InvalidArgument);
  };
};
```

Literaturverzeichnis

[Ash04] Y. Ashie, *Evidence for an Oscillatory Signature in Atmospheric Neutrino Oscillations*, Phys. Rev. Lett. 93, 101801 (2004).

[Amb95] M. Ambrosio et al. [MACRO Collaboration.],*Vertical muon intensity measured with MACRO at the Gran Sasso Laboratory*, Phys. Rev. D 52 (1995) 3793.

[Aok00] S. Aoki et al., *Nuclear emulsions in a large, hybrid experiment (CHORUS) to search for* $\nu_\mu \rightarrow \nu_\tau$ *oscillations*, NIM A447, (2000), 361-376.

[Bah76] J.N. Bahcall, R. Davis et. al., *Solar Neutrinos: A Scientific Puzzle.*, Jr. Science 191, 264-267 (1976).

[Bec95] R. Becker-Szendy et. al., *Neutrino measurements with the IMB director*, Nuclear Physics B, Volume 38, Issues 1-3 (1995), p. 331-336.

[Cow56] C.L. Cowan, Jr., F. Reines et. al., *Detection of the Free Neutrino: A Confirmation*, Science 124, 103 (1956).

[Crn00] *Al appearance experiment to search for* $\nu_\mu \leftrightarrow \nu_\tau$ *oscillations in the CNGS beam*, Experiment Proposal, CERN-SPSC 2000-028 SPSC/P318 LNGS P25/2000 (10 July 2000).

[Dra04] M. Dracos, *The OPERA Experiment*, Physics of Atomic Nuclei, Vol. 67, No. 6, 2004, 1092-1096.

[Esk97] E. Eskut et al., *The CHORUS Experiment to search for* $\nu_\mu \rightarrow \nu_\tau$ *oscillations*, CERN-PPE/97-33 (Mar 1997).

[Esk98] E. Eskut et al. [CHORUS collaboration], *Search for* ν_μ *to* ν_τ *oscillation using the* τ *decay modes into single charged particle*, Phys. Lett. B 434 (1998) 205-213.

[Fel05] G. Felici, A. Paolini, R. van Staa, R. Zimmermann, *Concept of the Trigger System for the Precision Tracker*, Internal note / OPERA (07 Dec 2005).

[Fer02] A. Ferrari, A. Rubbia, C. Rubbia, P.R. Sala, *Proton driver optimization for new-generation neutrino superbeams to search for sub-leading* $\nu_\mu \rightarrow \nu_e$ *oscillations (*θ_{13} *angle)*, New J. Phys. 4 (2002) 88.

[Fog05] G.L. Fogli, *Neutrino mass and mixing parameters: A short review*, arXiv:hep-ph/0506307 v1 (29 Jun 2005).

[Fuk98] Y. Fukuda et al. [Super-Kamiokande Collaboration], *Study of the atmospheric neutrino flux in the multi-GeV energy range*, arXiv:hep-ex/9805006 v3 (13 Jul 1998).

[Fuk01] S. Fukada et. al, *Solar* ^8B *and hep Neutrino Measurements from 1258 Days of Super-Kamiokande Data*, Phys. Rev. Lett. 86, Issue 25, 5651-5655 (2001).

[Goo95] M.C. Goodman [Soudan 2 Collaboration], *The atmospheric neutrino anomaly in Soudan 2*, Nuclear Physics B, Volume 38, Issues 1-3 (1995), p. 337-342.

[Hir88] K.S. Hirata, T. Kajita, M. Koshiba et. al., *Experimental study of the atmospheric neutrino flux*, Physics Letters B, Volume 205, Issue 2-3, p. 416-420.

[Kod01] K. Kodama et. al., *Observation of tau neutrino interactions*, Physics Letters, Section B: Nuclear, Elementary Particle and High-Energy Physics, Vol. 504 (2001) Nr. 3, S. 218-224.

[Led63] L. Lederman et. al., *The two-neutrino experiment*, Scientific American, March 1963, S. 60.

[Pau61] W. Pauli, W. Westphal (Hrsg.): *Aufsätze und Vorträge über Physik und Erkenntnistheorie*, Vieweg & Sohn, Braunschweig (1961), p. 156.

[Pdg06] W.-M. Yao et al., J. Phys. G 33, 1 (2006).

[Per78] M.L. Perl et. al., *The tau heavy lepton: A recently discovered elementary particle*, Nature, Vol. 275 (1978) Nr. 5678, S. 273-278.

[Pon58] B. Pontecorvo, *Inverse beta process and nonconservation of lepton charge*, Sov. Phys. JETP 7, (1958), 172-173.

[Sch97] N. Schmitz, *Neutrinophysik*, B. G. Teubner Stuttgart 1997.

[Sew06] J. Sewing, *Entwicklung und Bestimmung der Nachweiseigenschaften des Myon-Detektors für das OPERA-Experiment*, Dissertation, Hamburg 2006.

[Rep05] *Installation of the OPERA experiment: Status report*, CERN-SPSC-2005-025 / SPSC-M-738.

[Rol88] Rolfs, Rodney: *Cauldrons in the Cosmos*, University of Chicago Press, 1988.

[Www01] *OPERA Main Page*, http://operaweb.web.cern.ch/operaweb/index.shtml

[Www02] Nach: *Solar Neutrino Flux Deficits*, http://www.ps.uci.edu/∼smy/solproblem.html

[Www03] Erstellt mit *Google Earth*, http://earth.google.com/

Danksagung

Zum Abschluss möchte ich allen danken, die mich bei dieser Arbeit unterstützt haben.

Insbesondere gilt mein Dank Herrn Prof. Dr. Dieter Frekers und Frau Prof. Dr. Caren Hagner. Beide haben es mir ermöglicht, an diesem interessanten Gebiet zu arbeiten. Herr Prof. Frekers finanzierte mir auch zahlreiche Fahrten nach Hamburg, durch die meine Zusammenarbeit mit Frau Prof. Hagners Arbeitsgruppe erst möglich wurde. Des Weiteren ermöglichte er mir die Teilnahme an Konferenzen in Hamburg, Bad Honnef und Italien.

Ganz besonders möchte ich mich bedanken bei Björn Wonsak, Eike Grewe und Dr. Raoul Zimmermann, die mich bei meiner Diplomarbeit betreut und mir stets bei Fragen weitergeholfen haben. Ebenso gilt mein Dank beiden Arbeitsgruppen, insbesondere Henrik Dohmann und Jan Thies für die Aufmunterung beim Kaffee, Ole Ross, Thorben Ferber, Christoph Göllnitz sowie Benjamin Orth für die fruchtbare Zusammenarbeit bei der *slow control*. Sönke Hollstein danke ich ebenfalls für die Beantwortung vieler Fragen. Auch den hier nicht namentlich Erwähnten möchte ich für die gute Atmosphäre bei der Zusammenarbeit, und allen, die diese Arbeit Korrektur gelesen haben, danken.

Zuletzt möchte ich mich bei meiner Mutter und meinem verstorbenen Vater bedanken. Sie haben mich für die Naturwissenschaften und insbesondere für die Physik begeistert. Meine Mutter hat mich während dieser Arbeit stets unterstützt und motiviert.

Hiermit versichere ich, die vorliegende Arbeit selbstständig angefertigt und keine anderen als die angegebenen Hilfsmittel verwendet zu haben.

Münster, 02.01.2007