ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

CARACTERIZACIÓN DE LOS DETECTORES CHERENKOV DE AGUA DE LAGO-MÉXICO MEDIANTE SIMULACIÓN, PARA LA DETERMINACIÓN DEL ÁREA EFECTIVA DE PARTÍCULAS SECUNDARIAS GENERADAS POR UN GAMMA PRIMARIO DE ENERGÍAS ENTRE 200 GEV Y 1.1 TEV

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE FÍSICO

PROYECTO DE INVESTIGACIÓN

ANDRÉS GABRIEL DELGADO GILER
andres.delgado@epn.edu.ec

DIRECTOR: NICOLÁS ALEJANDRO VÁSQUEZ PAZMIÑO, PH.D.
vasqpaz@gmail.com

Quito, Mayo 2017

DECLARACIÓN

Yo, ANDRÉS GABRIEL DELGADO GILER, declaro bajo juramento que el trabajo aquí escrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la Normatividad Institucional vigente.

Andrés Gabriel Delgado Giler

CERTIFICACIÓN

Certifico que el siguiente trabajo fue desarrollado por ANDRÉS GABRIEL DELGADO GILER, bajo mi supervisión.

Nicolás Vásquez, Ph.D.
DIRECTOR

AGRADECIMIENTO ESPECIAL

Quiero dejar constancia de que el presente trabajo es parte del Proyecto "Estudio de Rayos Cósmicos con el Observatorio HAWC" que lo dirige el Ph.D. Ibrahim Torres en el Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE) en México, por ello agradezco de manera especial al Ph.D. Ibrahim Torres.

Andrés Gabriel Delgado Giler

AGRADECIMIENTOS

Quiero agradecer a mis padres: Gabriel Delgado y Sonia Giler, que me dieron la libertad y opurtunidad de escoger la carrera de física y poder cumplir uno de mis objetivos. Agradezco a mi hermana Gabriela Delgado y a su esposo Gene Alarcón, por el apoyo y aliento que me dieron para cumplir mi sueño de estudiar física. Quiero agradecerle a mi amiga y pareja Michelle Moreno por el apoyo incondicional que me brindó.

Agradezco también al PhD. Ibrahim Torres y al INAOE por la opurtinidad que me dieron para realizar la tesis durante mi estancia en México, así como también al PhD. Nicolás Vaśquez que permitió desarrollar el tema de titulación en colaboración con el INAOE, para el proyecto LAGO-Ecuador. Agradezco a Aline Galindo, que me brindó su confianza y conocimiento para el desarrollo de este trabajo.

Andrés Gabriel Delgado Giler

DEDICATORIA

Dedico mi trabajo de tesis a mi futuro en la Física, esperando que este trabajo sirva como motivación para seguir con estudios de postgrado. La dedico también a mi familia que formó parte indispensable durante mi vida en Quito, y que me brindaron todos los recursos para estudiar en la Escuela Politécnica Nacional.

Andrés Gabriel Delgado Giler

Índice de Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Los rayos cósmicos son partículas (90 % protones, 9 % partículas alfa, 1 % núcleos pesados) que golpean la atmósfera terrestre a una tasa de 1000 por metro cuadrado por segundo, y cuyo valor de energía se extiende hasta 10^{20} eV [9]. Tanto los rayos cósmicos como los rayos gamma, al incidir sobre la atmósfera, generan cascadas de partículas secundarias (muones, electrones, gammas, antimuones, positrones) debido a la interacción con los componentes de la atmósfera. Al conjunto de estas cascadas de partículas secundarias se les llama cascadas atmosféricas extendidas (EAS, por sus siglas en inglés) [13].

Uno de los estudios de los EAS se lleva a cabo mediante LAGO [19], que es un proyecto latinoamericano. LAGO propone estudiar indirectamente los rayos gamma y los rayos cósmicos a través de los EAS, mediante detectores Cherenkov de agua (WCD) [4], ubicados a nivel del suelo. Los WCDs son tanques cilíndricos de aluminio recubiertos internamente con Tyvek, un material altamente reflectante. Existe un arreglo de WCDs en cada latitud y altitud considerable de Latinoamérica [5, 6]; uno de esos arreglos es LAGO-México que está ubicado a 4530 m.s.n.m. en las laderas del volcán Sierra Negra, Puebla, México y consta de tres WCDs, ubicados en los vértices de un triángulo de lados 24,74 m, 28,7 m, y 30 m. Los WCDs tienen 7,3 m de diámetro y 1 m de alto. Cada WCD contiene cuatro tubos fotomultiplicadores (PMTs) de 8 pulgadas de diámetro en su parte superior, con el fotocátodo hacia abajo. Los detectores están llenos con agua libre de impurezas. Además hay un WCD central de 4,2 m de altura y 7,3 m de diámetro, ubicado en el baricentro del triángulo.

Actualmente, los tanques están en proceso de construcción. Sin embargo, no se ha determinado la posición de los PMTs en el detector central y se desea conocer si el WCD central podrá diferenciar entre una detección generada por un rayo gamma o un rayo cósmico. Además, se desea calcular el área efectiva (A_{eff}) del experimento que es la fracción de cascadas que se esperan detectar según la energía del rayo cósmico primario o del rayo gamma primario. Este trabajo tiene como objetivo: estudiar la energía y proporción de las partículas secundarias a nivel del suelo mediante el software CORSIKA producidos por un protón de 10 TeV y un gamma de 10 TeV, generar un código en C++ utilizando el software

GEANT4 para la simulación de los WCDs, determinar la estructura y función del WCD central, y calcular el área efectiva del experimento para un gamma primario que incide verticalmente en la atmósfera con energía entre 200 GeV y 1.1 TeV.

Este trabajo está organizado de la siguiente manera. En el capítulo 2, se hace una definición de los rayos cósmicos y rayos gammas en relación a la producción de EAS, el proceso de radiación Cherenkov en el método de detección de EAS y el proyecto LAGO de México para el cual se realiza este trabajo. En el capítulo 3 se describe los paquetes computacionales (software) utilizados para la simulación de los EAS (CORSIKA) el diseño de los detectores (Geant4). En el capítulo 4 se describe la metodología empleada para la ejecución y generación de los EAS, la creación y modificación de los detectores WCD y PMTs, los procesos físicos involucrados y la conexión de datos CORSIKA-Geant4. En el capítulo 5 se presentan los resultados de las simulaciones: posición de los PMTs en el tanque central, rise time de secundarios para la validación en la diferenciación de muones en el tanque central y gráfica de área efectiva en función de la energía del primario. En el capítulo 6 se escribe las conclusiones de los resultados y se presenta la estructura final de los detectores WCD. En la parte de los anexos están todos los códigos editados en Geant4.

Además, los códigos desarrollados en este trabajo estarán almacenados en un repositorio al cual podrán acceder los participantes de la colaboración LAGO, entre los cuales está incluido Ecuador. En este sentido, los códigos desarrollados, y el método empleado en las simulaciones darán una guía para realizar simulaciones con los datos de Ecuador (altitud, campo magnético), como parte de futuros proyectos.

Capítulo 2

Marco Teórico

2.1. Rayos cósmicos

La mayoría de los rayos cósmicos (RC) son relativistas ya que tienen energías comparables o superiores a sus masas en reposo. La radiación de origen galáctico corresponde a energías por debajo de $10^{16} - 10^{17} eV$, mientras la componente con energías por encima de $10^{20} eV$ es de origen extragaláctico. Ésta última se denomina radiación ultra energética (UHE), y es poco conocida debido a dificultades técnicas de medición [9, 13].

Las principales preguntas que el estudio intenta responder son: ¿De dónde vienen los RC? y ¿cuáles son los mecanismos de aceleración que les permiten alcanzar altas energías?. Aunque no sean del todo conocidos, se sabe que los RC menos energéticos provienen fuera del Sistema Solar, pero de nuestra galaxia. Los RC altamente energéticos tienen radios de giro más grandes que el tamaño de la galaxia, por lo que estos podrían tener origen extragalácticos.

2.1.1. Estudio de RC

A pesar del avance en los aceleradores de partículas, la información que generan los RC es importante en cuanto a los datos de altas energías que proporciona, superando en el orden de 10^7 a los generados en el LHC. Hoy en día, hay un número importante de áreas en la cual el conocimiento de la interacción entre partículas es necesaria para entender las implicaciones astrofísicas de los datos de rayos cósmicos [10]:

- Producción de rayos cósmicos secundarios como antiprotones producidos por rayos cósmicos primarios, cuando los RC primarios colisionan con núcleos en el medio interestelar. De las cantidades relativas de los RC secundarios, se puede aprender de la naturaleza de la materia y campos que conforman el medio interestelar y como se propagan los rayos cósmicos a través de dichos campos.
- Producción de fotones, neutrinos y otras partículas en colisiones de rayos cósmicos con materia cercana en donde se aceleran las partículas de rayos cósmicos. Permite

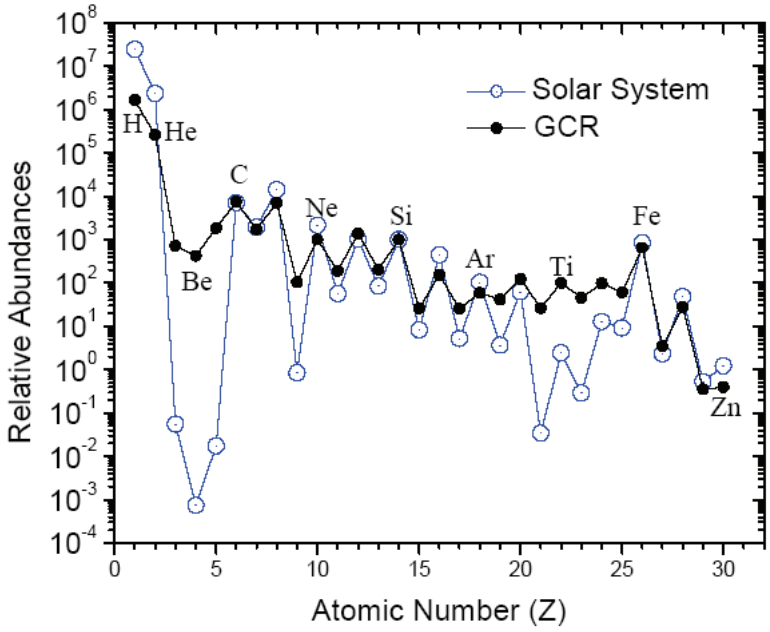


Figura 2.1: Abundancias relativas elementales de los rayos cósmicos que llegan a la Tierra comparadas con las del Sistema Solar, medidas con los instrumentos ACE/CRIS, normalizada respecto al Silicio (imagen tomada de Katharina Lodders, 2003). La línea azul (con círculos blancos) muestra las abundancias relativas elementales en el Sistema Solar, mientras la línea negra (con círculos negros) representa la abundancia elemental de RC en la galaxia.

estudiar los mecanismos de aceleración y generación de rayos cósmicos.

- Penetración de rayos cósmicos en la Tierra; y la detección de muones y neutrinos mediante detectores localizados debajo de la superficie terrestre.
- La relación entre las lluvias atmosféricas generadas por los rayos cósmicos. Los rayos cósmicos UHE son raros y no pueden ser observados directamente con los pequeños detectores encima de la atmósfera, pero pueden ser estudiados indirectamente por grandes detectores de lluvias de partículas ubicados a nivel del suelo. Esto permite inferir la naturaleza del primario por medio las partículas secundarias.

2.1.2. Composición

Aunque la composición química de los RC no está del todo definida, se sabe que son 99% núcleos de partículas conocidas, mientras el 1% son núcleos pesados. Del 99%, el 90% son protones (núcleos de hidrógeno) y el 9% corresponde a partículas alfa (núcleos de helio). Existe también una pequeña fracción de antimateria como positrón-antipositrón, por lo que es una área de investigación actual. La abundancia relativa de los rayos cósmicos son comparadas con las abundancias de los elementos del Sistema Solar, como se observa en la Figura 2.1 [21], donde los círculos blancos y negros representan la abundancia relativa de elementos en el Sistema Solar y de los RC, respectivamente.

2.1.3. Espectro de energía

La Figura 2.2 muestra el espectro de RC para energías por encima de $10^{11}eV$. El flujo de RC está descrito por una ley de potencia, de tal manera que el diferencial del flujo es proporcional a una potencia de la energía (E),

$$\frac{dN}{dE} \alpha\ E^{-(\delta)} \tag{2.1}$$

donde δ es el índice espectral. En dicha figura se ha multiplicado el flujo cósmico por E^2 para enfatizar e indicar la cantidad de energía transportada por los rayos cósmicos. La Figura 2.2, del flujo de RC contiene tres características generales: 1) los rayos cósmicos de la rodilla por encima de $10^{15}eV$, 2) los rayos cósmicos del tobillo por encima de $10^{18}eV$, y 3) el corte de los rayos cósmicos alrededor de $10^{19}eV$. Por debajo de la rodilla, el índice espectral de la ley de potencia es $\delta = 2{,}7$ y por encima de la rodilla el índice espectral se incrementa en $\Delta\delta = 0{,}3$ [20]. Para valores superiores a los del tobillo la ley de potencia es más plana y similar a aquella antes de la rodilla. donde δ es el índice espectral. Para valores de energía E $\sim 10^6$ GeV, el índice espectral es $\delta \approx 1.7$. Para valores superiores de energía, $\delta \approx 2.0$.

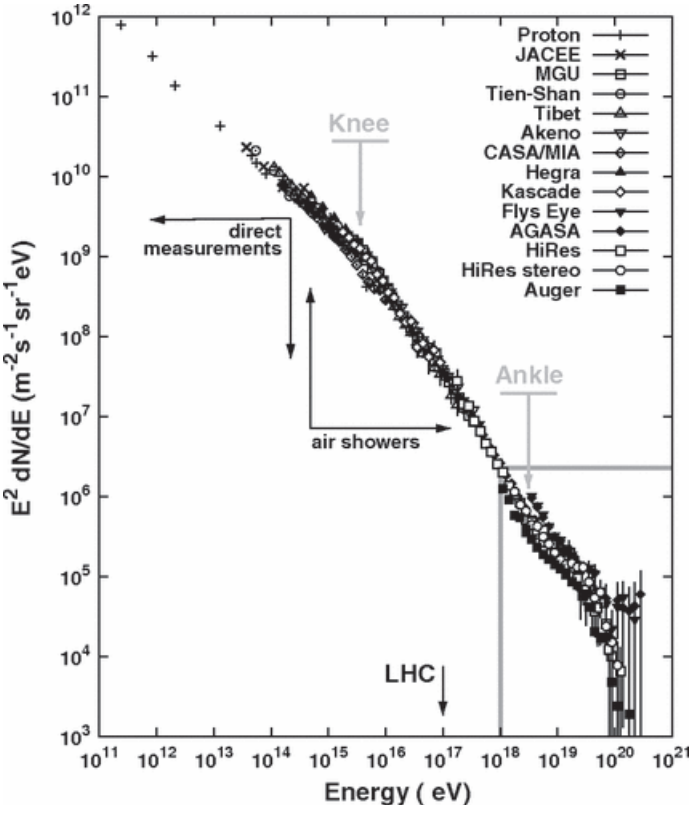


Figura 2.2: En el eje vertical, flujo de los rayos cósmicos multiplicado por E^2; y en el eje horizontal, energía de los RC por encima de $10^{11}eV$. (imagen tomada de Antoine Letessier-Selvon y Todor Stanev, *Reviews of Modern Physics*, Vol. 83, 2011.)

2.2. Rayos Gamma (GRBs)

Los Gamma Ray Bursts (GRBs, por sus siglas en inglés) son destellos cortos e intensos de rayos gamma, y son otra manera de generar RC. De 1969 a 1973 se empieza a detectar fotones en el rango de energía 0,2 a 1,5 MeV, en intervalos de tiempo de 0,1s a 30s [18]. Los GRBs permiten probar las propiedades relacionadas con el desplazamiento al rojo en el universo: expansión cósmica, tasa de formación estelar. De acuerdo a GRBs de larga duración, se cree que se originan por el colapso de estrellas masivas [23]. Las curvas de luz de los GRBs varían entre una y otro destello. En la Figura 2.3 mostramos las curvas de luz de los GRBs: GRB790613 en la parte superior, GRB780918 en la mitad y GRB820313 en la parte inferior. En el eje vertical se muestra la intensidad en cuentas normalizadas para cada GRB, mientras en el eje horizontal está la escala temporal en segundos. Las formas son diferentes al igual que las escalas temporales que van de pocos milisegundos a cientos de segundos [3, 24]. Los perfiles temporales pueden presentar múltiples picos, o simplemente no tener una estructura fina. Los GRBs son clasificados de acuerdo al intervalo de tiempo en que se producen: en la Figura 2.3 el GRB790613 se produce en 2 ms, a diferencia del GRB820313 que presenta una distribución bimodal de dos picos en 0.3 sy 10-20s [3, 24]. Cuando nos referimos a la radiación gamma, se puede distinguir entre rayos gamma de

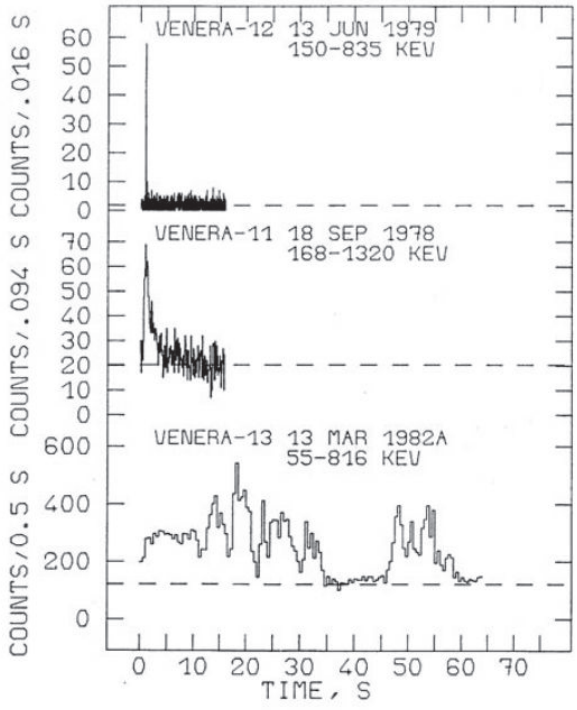


Figura 2.3: Diversidad de curvas de luz de GRBs: GRB790613 (superior), GRB780918 (mitad) y GRB820313 (inferior) detectados por la collaboración Franco-Soviet SIGNE. En el eje vertical se muestra la intensidad en cuentas normalizadas para cada GRB, mientras en el eje horizontal está la escala temporal en segundos. Cada GRB fue detectado en un año distinto y cubre un rango de energía distinto en el orden de los KeV. (Imagen tomada de C. Barat, G. Chambon, *Astrophysics and Space Science*, Vol. 75, 1981.)

fuentes puntuales y la radiación gamma difusa [13]. En particular la mayor cantidad de radiación observada en el cielo es referida a fuentes extendidas. Las fuentes puntuales emisores de rayos gamma a menudo son asociadas a objetos astronómicos que ya han sido observados en otras longitudes de onda, como en la banda del radio o del óptico.

2.3. Lluvias atmosféricas extendidas

Cuando un rayo cósmico o rayo gamma (partícula primaria) entra en la atmósfera terrestre, produce un gran número de partículas secundarias debido a una serie de colisiones con los núcleos de los componentes atmosféricos como N_2, O_2, Ar. Estas partículas secundarias se las denomina lluvias o cascadas atmosféricas extendidas (EAS, por sus siglas en inglés) o simplemente lluvias atmosféricas (AS, por sus siglas en inglés). En la Figura 2.4 se observa el esquema del desarrollo lateral y longitudinal de un EAS en la atmósfera, generado por un protón primario. En dicha figura se muestra las componentes secundarias detectables a distintos niveles las cuales pueden producir efectos (Cherenkov atmosférico, fluorescencia atmosférica) a lo largo de la atmósfera. El número de partículas secundarias que se producen en un EAS a un cierto nivel en la atmósfera se conoce como tamaño de la lluvia, N. Por lo general, N solo incluye partículas cargadas (electrones, positrones, muones, antimuones, etc). N depende de tres parámetros: energía del primario E_0, ángulo cenital θ (o de incidencia del primario) y la altura de la primera interacción h_1 [14].

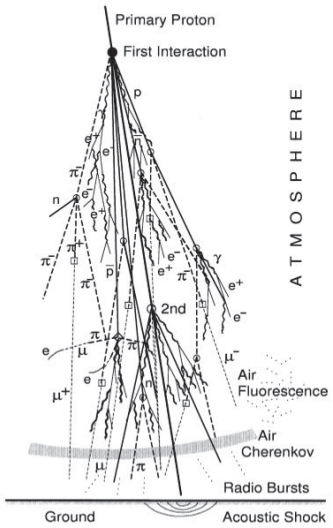


Figura 2.4: Esquema del desarrollo lateral y longitudinal de un EAS en la atmósfera, generado por un protón primario que interacciona con los componentes de la atmósfera. En el eje vertical representamos la altura de la atmósfera y en el eje horizontal el nivel del suelo o de detección. Se puede observar los distintas partículas secundarias tales como muones, antimuones, piones, electrones, positrones, gammas. El paso de estas partículas a través de la atmósfera pueden producir efectos como fluorescencia atmosférica y Cherenkov atmosférico. (Imagen tomada de Peter K.F. Grieder, *Extensive Air Showers: High Energy Phenomena and Astrophysical Aspects. A Tutorial, Reference Manual and Data Book*, 2010.)

En un EAS se tienen tres componentes de partículas: electromagnética, muónica y hadrónica. Los EAS están caracterizados por varios parámetros, tales como el núcleo de la lluvia que es la región central que contiene el 90% de los secundarios, el desarrollo máximo de la lluvia medida en [g/cm2] que da la profundidad a la que se tiene mayor número de partículas, la edad de la lluvia, s, que se refiere al desarrollo máximo de la lluvia, siendo:

- $s < 1$, lluvia joven, no se ha alcanzado el máximo desarrollo de la cascada,
- $s > 1$, lluvia vieja, se ha alcanzado más allá del máximo desarrollo de la cascada,
- $s = 1$, se está en el desarrollo máximo,
- $s = 2$, la lluvia atmosférica ha terminado en una sola partícula.

2.3.1. Lluvias iniciadas por hadrones

Una lluvia atmosférica consiste de la superposición de dos tipos de cascadas: cascada hadrónica y cascada electromagnética. La cascada hadrónica es el proceso responsable del transporte de energía dentro de una lluvia. Es el resultado de numerosas colisiones del primario con núcleos de nitrógeno, oxígeno y ocasionalmente con trazas de gases a lo largo de la trayectoria en la atmósfera.
Los muones (μ^+, μ^-) son producidos numerosamente en las cascadas hadrónicas, con un 10% de la contribución del total de flujo de partículas en una lluvia atmosférica a nivel del mar. Principalmente, se producen como productos de una variedad de partículas inestables que emergen de colisiones de altas energías, tales como kaones, piones entre otras:

$$p + A \rightarrow p + A + n\pi^{\pm,0} + X \tag{2.2}$$

$$\begin{aligned} \pi^+ &\rightarrow \mu^+ + \nu_\mu, \\ \pi^- &\rightarrow \mu^- + \overline{\nu}_\mu. \end{aligned} \tag{2.3}$$

$$\begin{aligned} \mu^+ &\rightarrow e^+ + \nu_e + \nu_\mu, \\ \mu^- &\rightarrow e^- + \overline{\nu}_e + \overline{\nu}_\mu. \end{aligned} \tag{2.4}$$

donde X representa cualquier hadrón, A es el número másico de un determinado núcleo, p son protones, (e^+, e^-) son electrones y $(\nu_\mu, \overline{\nu}_\mu)$ son neutrinos y antineutrinos tipo muón, respectivamente. Los principales contribuidores a la componente muónica en los EAS son los piones cargados (π^+, π^-) y kaones (K^+, K^0, K^-), pero también partículas charmed tales como $D^\pm, D^0$ y J/ψ.

2.3.2. Lluvias iniciadas por electrones y rayos gamma

Los EAS iniciados por rayos gamma ($E_\gamma \geq 10TeV$) son de particular interés porque estos no son deflectados por los campos magnéticos en la galaxia, a diferencia de las partículas cargadas. Dado que la dirección de arribo de un EAS generado por fotones gamma puede ser determinada por arreglo de detectores de Cherenkov atmosférico o mediante arreglos de detectores Cherenkov de agua (WCD), es posible identificar fuentes de rayos gamma, las cuales pueden ser fuentes potenciales de rayos cósmicos (hadrónicos). Todas las partículas cargadas están sujetas a interacciones electromagnéticas. Generalmente, los procesos fundamentales se puede resumir en el Cuadro 2.1. Los procesos más relevantes

Procesos iniciados por fotones	Procesos iniciados por electrón
- Dispersión Rayleigh	- Dispersión Coulomb
- Efecto fotoeléctrico	- Bremsstrahlung (Coulomb)
- Efecto Compton	- Radiación Cherenkov
- Producción de pares	- Radiación Sincrotrón
- Interacción foto-nuclear	- Efecto Compton inverso
- Interacción fotón-fotón	- Aniquilación positrón

Cuadro 2.1: Procesos fundamentales involucrados en el desarrollo de una lluvia atmosférica iniciada por electrones y por fotones.

en el desarrollo de una cascada electromagnética son la *producción de pares* para fotones y *bremsstrahlung* para electrones en el campo Coulómbico de un núcleo:

- **Producción de pares**: se da cuando la energía de un fotón alcanza el umbral de energía (para electrones, $E_{umbral} = 1{,}022MeV$) y produce un electrón e^- y un positrón e^+,

$$\gamma \rightarrow e^+ + e^- \tag{2.5}$$

- **Bremsstrahlung**: producción de radiación electromagnética generada por la desaceleración de una partícula cargada causada por el campo eléctrico de otra partícula cargada:

$$e^\pm + (nucleo) \rightarrow e^\pm + \gamma \tag{2.6}$$

2.4. Métodos de detección de lluvias atmosféricas

Una lluvia atmosférica se caracteriza por tener un disco de partículas muy fino que se propaga a lo largo del eje de la lluvia aproximadamente a la velocidad de la luz. El disco de la partícula tiene una alta densidad de secundarios en el centro y decrece exponencialmente con el aumento de la distancia radial. El lugar de la máxima densidad de partículas define experimentalmente la posición en el suelo del eje de la lluvia. Las partículas cargadas (secundarios) producen luz Cherenkov altamente polarizada y emisión de radio, en su

propagación a través de la atmósfera, así como también fluorescencia en el aire. Cada uno de estos fenómenos representa una forma de estudiar los EAS. Sin embargo, hay características del disco de partículas que no han sido del todo explorados. Hoy en día, las lluvias atmosféricas son estudiadas por detectores Cherenkov de agua, telescopios de Cherenkov atmosférico y fluorescencia atmosférica [1, 2].

2.4.1. Arreglo de detectores de partículas

Este método se basa en la detección de los secundarios del disco de partículas que llegan a nivel del suelo mediante arreglo de detectores. Lo que se obtiene es información del arribo temporal, tipo y proporción de partículas [1]. Lo que se obtiene es una imagen bidimensional (incompleta) del EAS en un momento particular y con la información del tiempo de arribo y la densidad de partículas, se reconstruye y analiza la lluvia atmosférica. En la Figura 2.5 se observa que si el arreglo de detectores es lo suficientemente grande, el disco de partículas puede activar cada uno de los detectores y registrar el paso de secundarios.

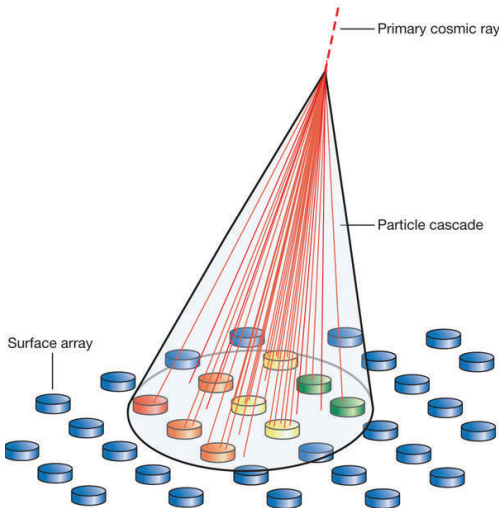


Figura 2.5: Esquema de un rayo cósmico primario (línea roja punteada) que experimenta interacción con los componentes de la atmósfera y produce los EAS (líneas rojas dentro del cono). La cascada de partículas deja una huella sobre el suelo que cubre una área de unos cientos de kilómetros cuadrados. Los detectores que son activados (cilindros de colores amarillo y verde), registran el paso de partículas. Imagen obtenida de (Pablo M. Bauleo, Julio Rodríguez Martino, 2009).

El observatorio Pierre Auger es un arreglo de tanques llenos de agua ultrapura con tubos fotomulplicadores (PMTs, por sus siglas en inglés) ubicados en la parte superior con el fotocátodo hacia abajo, y su objetivo es estudiar rayos cósmicos a través de los EAS [1, 8]. Auger registra el paso de las partículas secundarias a través de su interacción con el agua situada dentro de los tanques.

[1] En algunas aplicaciones solo se cubre un cierta ángulo sólido del cielo, como es el caso del Observatario Auger localizado en Sierra Negra, Puebla, México.

2.4.2. Respuesta de un arreglo de detectores

Para convertir una observación de lluvia atmosférica en un espectro de energía del primario de una fuente puntual, uno necesita conocer la aceptancia o área efectiva del detector. Esto es crucial para convertir una tasa observada a un flujo medido [11]:

$$Flux[cm^{-2}s^{-1}sr^{-1}] = \frac{Tasa}{A(E,\theta)\ x\ T\ x\ \Delta\Omega} \tag{2.7}$$

donde la $Tasa$ es el número de partículas detectadas en un intervalo de tiempo, $A(E,\theta)$ es el área efectiva del arreglo de detectores, dependiente de la energía y ángulo de incidencia del primario, T es el tiempo de detección y $\Delta\Omega$ es el ángulo solido o de detección del arreglo. Para una condición de triggering[2] dada, el área efectiva se incrementa con la energía. Con las lluvias atmosféricas hay dos casos: pequeñas lluvias que pueden penetrar profundamente y cubrir área favorables activando el arreglo; y lluvias extremas que activan el arreglo de detectores incluso cuando su núcleo se encuentra fuera del perímetro físico del arreglo. Para el cálculo del área efectiva A_{eff} se utiliza el método de Monte Carlo, que toma en cuenta las fluctuaciones en el desarrollo de la lluvia atmosférica. Para calcular el área efectiva se utiliza la siguiente fórmula [16]:

$$A_{eff}(E,\theta) = A_{thrown} N_{PMT}(E,\theta) \frac{N_{obs}(E,\theta)}{N_{thrown}(E,\theta)}, \tag{2.8}$$

donde A_{thrown} es el área donde se arrojan los EAS, N_{PMT} es el número de detecciones o hits en el arreglo, $N_{thrown}(E,\theta)$ es el número de EAS simuladas con un energía y ángulo determinado para el primario, y $N_{obs}(E,\theta)$ es el número de EAS observadas o detectadas por el arreglo.

2.4.3. Dirección de arribo

La dirección de arribo de la lluvia atmosférica es perpendicular al disco de partículas y está dado por el ángulo cenital θ y el ángulo azimutal ϕ. Estos ángulos están determinados por el retraso en el tiempo de arribo del frente del EAS a diferentes lugares a lo largo del arreglo de detectores. Para un arreglo de 5 detectores, como se muestra en la Figura 2.6,

$$\tan\phi = -\frac{\Delta t_{N,C}}{\Delta t_{W,C}} \tag{2.9}$$

y

$$\sin\theta = -\frac{c}{d}\frac{\Delta t_{N,C}}{\sin\phi} \tag{2.10}$$

donde $\Delta t_{[i,j]}$ es la diferencia de tiempo de arribo en [ns] entre los detectores i y j ($i = E,W,N,S; j = C$), c es la velocidad de la luz, y d es la mitad de la diagonal del cuadrado

[2]Condición que se establece en los detectores, para que una detección del PMT equivalga a un fotón Cherenkov real.

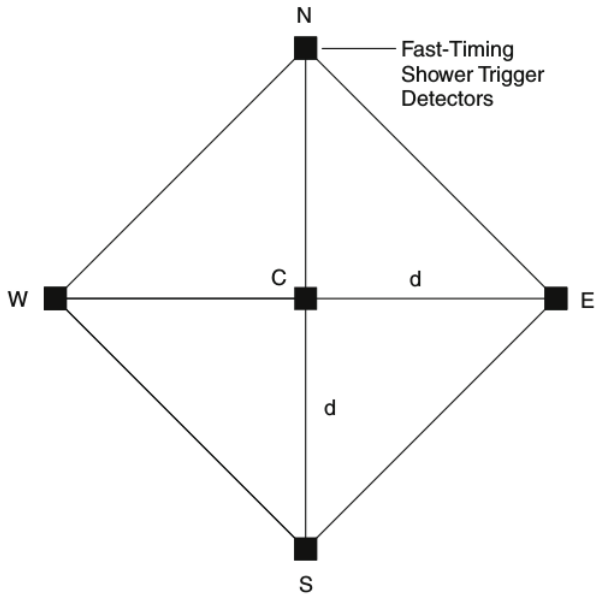


Figura 2.6: Ejemplo del diseño de un sistema de detección de lluvias atmosféricas. Cada vértice correponde a un detector en coincidencia con los puntos cardinales. C es un detector ubicado en el centro geométrico del arreglo. (Imagen tomada de Peter K.F. Grieder, *Extensive Air Showers: High Energy Phenomena and Astrophysical Aspects. A Tutorial, Reference Manual and Data Book*, 2010.)

formado por los cuatro detectores.

2.5. Procesos de Interacción

2.5.1. Radiación Cherenkov

Cuando una partícula cargada se mueve a altas velocidades en un medio dieléctrico causa una polarización simétrica en el plano azimutal, mas no a lo largo del eje de movimiento. Se obtiene un campo dipolar que se extiende a lo largo de la trayectoria de la partícula. Las ondas de la radiación electromagnética emitidas por las transiciones dipolares, se esparcen sobre una banda de frecuencias correspondientes a las componentes de Fourier del pulso de polarización. Por lo general, se genera una interferencia destructiva, de tal manera que en cualquier punto el campo resultante es cero.

Sin embargo, si la velocidad de la partícula cargada excede la velocidad de fase de la luz en el medio, las ondas de todos los puntos a lo largo de la trayectoria de la partícula estarán en fase unas con otras bajo un ángulo de emisión particular, θ, medido con respecto a la dirección de movimiento de la carga y se combinarán para formar una onda plana. A este tipo de luz se le llama radiación Cherenkov. Debido a la simetría azimutal, la emisión de un elemento de la trayectoria se propaga a lo largo de un cono, cuyo ángulo de apertura es θ y cuyo vértice está sobre la trayectoria de la partícula, tal como se observa en la figura

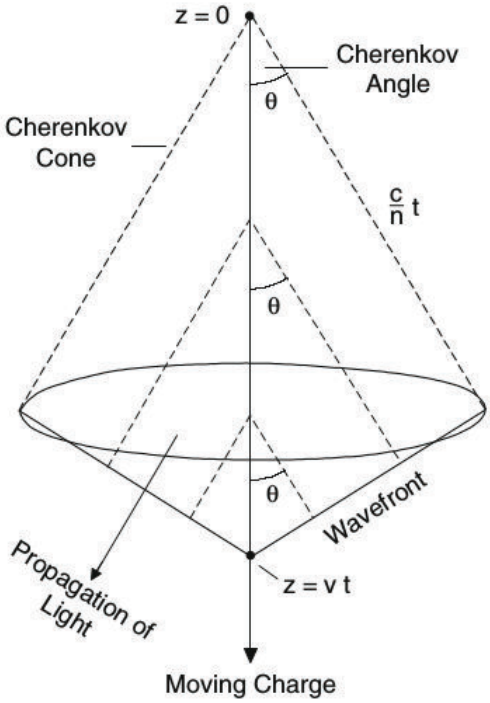


Figura 2.7: Geometría básica del fenómeno de radiación Cherenkov. Se muestra el ángulo de emisión θ de una partícula relativista moviendose a lo largo del eje z, la posición instantanea de un frente de onda y la propagación de la luz Cherenkov. (Imagen tomada de Peter K.F. Grieder, *Extensive Air Showers: High Energy Phenomena and Astrophysical Aspects. A Tutorial, Reference Manual and Data Book*, 2010.)

2.7. Si $\beta = v/c$, donde v es la velocidad de la partícula cargada, c la velocidad de la luz en el vacio y n el índice de refracción del medio en el cual se mueve la partícula, la condición de coherencia o *relación de Cherenkov* se escribe como:

$$\cos(\theta) = \frac{1}{\beta n} = \frac{c}{vn} \tag{2.11}$$

La acción de retroceso del fotón en el movimiento de la partícula cargada causa una ligera deflección de la emisión, por lo que (2.11) se ve modificada:

$$\cos(\theta) = \frac{1}{\beta n} + \frac{\hbar k}{2p}\left(1 - \frac{1}{n^2}\right) \tag{2.12}$$

donde $\hbar k$ es el momento del fotón, p es el momento de la partícula cargada y θ es el ángulo de emisión entre la dirección de la partícula incidente y el fotón emitido. De la relación (2.11) se pueden obtener las siguientes conclusiones:

- Para un índice de refracción dado n, hay una velocidad umbral:

$$\beta = (1/n) \tag{2.13}$$

por debajo de la cual no se produce la radiación Cherenkov. En el valor umbral, el ángulo de emisión, θ, es cero.

- Para cada medio existe un *ángulo máximo de emisión*, θ_{max}, llamado *ángulo Cherenkov*, en el que se tiene $\beta = 1$, es decir en el límite ultrarelativista:

$$\theta_{max} = \arccos(1/n), \tag{2.14}$$

- Dado que $|\cos(\theta)| \leq 1$, de (refrelacioncherenkov) se tiene que $1/\beta n \leq 1$. Reescribiendo se obtiene $n \geq 1/\beta$, por tanto la condición para que tome lugar el proceso Cherenkov es:

$$n > 1. \tag{2.15}$$

  Para $\beta = 1$ también se cumple la relación.

Por tanto, la radiación Cherenkov es producida solo en medios y a frecuencias donde $n(\omega) > 1$. Esto implica que la radiación Cherenkov predomina en la región del visible y solo una pequeña fracción cerca de la región infrarroja.

2.5.2. Tasa de radiación Cherenkov

La energía perdida de una partícula cargada debido a la radiación Cherenkov generada al atravesar un medio, puede ser obtenida clasicamente o a partir de la mecánica cuántica. Ambos métodos generan un resultado similar, con pequeñas diferencias. Para la derivación clásica se obtiene que la energía perdida dE (por efecto Cherenkov) por unidad de longitud dl está dada por:

$$\left(\frac{dE}{dl}\right)_{Ch} = 4\pi\left(\frac{Ze}{c}\right)^2 \int_{\beta n>1}\left(1 - \frac{1}{\beta^2 n^2}\right).\omega d\omega \ [eV/m], \tag{2.16}$$

donde Ze es la carga de la partícula, y ω la frecuencia angular de la radiación emitida. Sin un rango de frecuencia la integral diverge, por lo que se utiliza la desigualdad $\beta n > 1$. La Eq. (2.16) puede ser convertida al número de fotones, $N_{fotones}$, radiados a lo largo de una trayectoria de longitud l:

$$N_{fotones} = 2\pi z^2 \alpha l\left(\frac{1}{\lambda_1} - \frac{1}{\lambda_2}\right).\left(1 - \frac{1}{\beta^2 n^2}\right), \tag{2.17}$$

donde α es la constante de estructura fina ($\alpha = e^2/\hbar c = 1/137$) y n es el índice de refracción del medio Cherenkov. Reemplazando la Eq. (2.11) en (2.17), se obtiene:

$$N_{fotones} = 2\pi z^2 \alpha l\left(\frac{1}{\lambda_1} - \frac{1}{\lambda_2}\right)\sin^2(\theta). \tag{2.18}$$

En el caso de un electrón relativista en el aire al nivel del mar, la tasa de 30 fotones/m entre los 350 nm y 500 nm. Dado que dE/dl es proporcional a $1/\lambda$ la energía perdida por unidad

de longitud es proporcional a $1/\lambda^2$. Esto explica el hecho de que la radiación Cherenkov sea predominante a longitudes de onda corta, dándole una tonalidad azulada. El aire al nivel de mar, tiene un índice de refracción n=1,00029. Con este valor se puede obtener el umbral de energía Cherenkov de un electrón E_{umbral} = 21 MeV y el ángulo máximo de emisión Cherenkov θ_{max} = 1,3°. La tasa de producción de fotones Cherenkov de un electrón con energía por encima del valor umbral, en el aire, es de 0,3 fotones/cm en el intervalo de longitud de onda entre 400 y 500 nm. Para el electrón en el agua, se tiene n=1,33 con lo que: E_{umbral} = 260 keV, θ_{max} = 41° y dN/dl = 250 fotones/cm. El valor del umbral de energía Cherenkov en el agua para muones es ∼ 53 MeV, para piones ∼ 70 MeV y para protones ∼ 475 MeV.

2.6. Proyecto LAGO

Latin American Giant Observatory (LAGO) es un observatorio de astropartículas, que se extiende a escala global y está formado por 10 países. El proyecto nació como una alternativa para evitar los elevados costos de experimentos de rayos cósmicos. La red de detección LAGO consiste de uno o varios detectores de partículas llamados detectores Cherenkov de agua, localizados en sitios con diferentes altitudes y latitudes. Actualmente, los sitios que forman parte de la colaboración LAGO y que cuentan con la instrumentación son: Chacaltaya en Bolivia, Mérida en Venezuela; Marcapomacocha, en Perú y Sierra Negra en México. Además, países como Colombia y Ecuador también forma parte de esta colaboración, y tienen instalados prototipos de detectores individuales en Universidades como la Escuela Politécnica Nacional (EPN) y San Francisco de Quito en Ecuador, y en la UIS de Colombia. El objetivo principal del proyecto LAGO es la detección de chubascos atmosféricos extendidos (EAS) generados por rayos gamma del orden de 100 GeV y rayos cósmicos del orden de 10 GeV, utilizando detectores Cherenkov de agua, mediante la técnica de la partícula simple. Uno de los requisitos fundamentales es que la altitud del lugar supere los 3000 msnm, ya que esto evita que parte de las partículas secundarias generadas por las cascadas se absorban en la atmósfera.

2.6.1. LAGO-México en Sierra Negra

El proyecto LAGO-México está ubicado sobre el volcán Sierra Negra, en Puebla. Se encuentra a una altitud de 4500 msnm, lo que lo hace un lugar óptimo para detectar EAS [6]. Su interés radica en la detección de GRBs para conocer el ángulo de arribo y poder asociarlo a una fuente puntual, y de trazador de partículas para conocer el primario que inició la cascada de partículas. Otros experimentos, como HAWC y GTM se encuentra también en el volcán. En la Figura 2.8 se observa el arreglo de tanques. LAGO-México consta de tres WCDs[3], ubicados en los vértices de un triángulo de lados 24,74 m, 28,7 m, y 30 m. Los WCDs tienen 7,3 m de diámetro y 1 m de alto. Cada WCD contiene cuatro

[3]WCD, Water Cherenkov Detector: detectores Cherenkov de agua, para la detección de EAS.

Figura 2.8: Vista del sitio LAGO-México, en Sierra Negra, Puebla. Arreglo de WCD de 7.3 m de diámetro: 3 tanques laterales de 1m de alto, y 1 tanque central de 7.3 m de alto.

tubos foto-multiplicadores (PMTs) de 8 pulgadas de diámetro en su parte superior, con el fotocátodo hacia abajo. Los detectores están llenos con agua libre de impurezas. Además hay un WCD central de 4,2 m de altura y 7,3 m de diámetro, ubicado en el baricentro del triángulo. Cada tanque esta recubierto internamente con una lona para contener el agua, y una segunda capa de material Tyvek[4], para tener reflectividad y difusión dentro del tanque. El Tyvek tiene como función mejorar la eficiencia en la detección de fotones Cherenkov (producidos por los secundarios en el agua) en los PMTs, y generar una difusión de los fotones para no saturar los PMTs.

Los tres WCD laterales están segmentados internamente por cuatro paredes en partes iguales, con la finalidad de aumentar la capacidad de detección en cada sección. Esto se puede ver en la Figura 2.9. En cada sección hay un PMT ET Enterprises 9354KFLB, colocado a 2 m a partir del centro del tanque y a 1 m sobre la base con el fotocátodo hacia abajo.

Detector Cherenkov de agua

Un WCD es un tanque recubierto internamente con Tyvek y lleno de agua libre de impurezas, al cual se le ha implementado PMTs, y tiene un sistema de adquisión de datos que permite almacenar los pulsos de luz registrados.

2.6.2. Técnica de la partícula individual

El objetivo de LAGO-México es el estudio de RC. La técnica de la partícula individual (SPT, por sus siglas en inglés) se basa en el estudio de los RC mediante la detección de los EAS o de algunas partículas generadas en la lluvia atmosférica. La detección del paso de partículas se realiza mediante la detección de radiación Cherenkov en los WCDs. Cuando

[4]Tyvek: es un material tipo banner utilizado en los tanques para aumentar la eficiencia de detección en los PMTs.

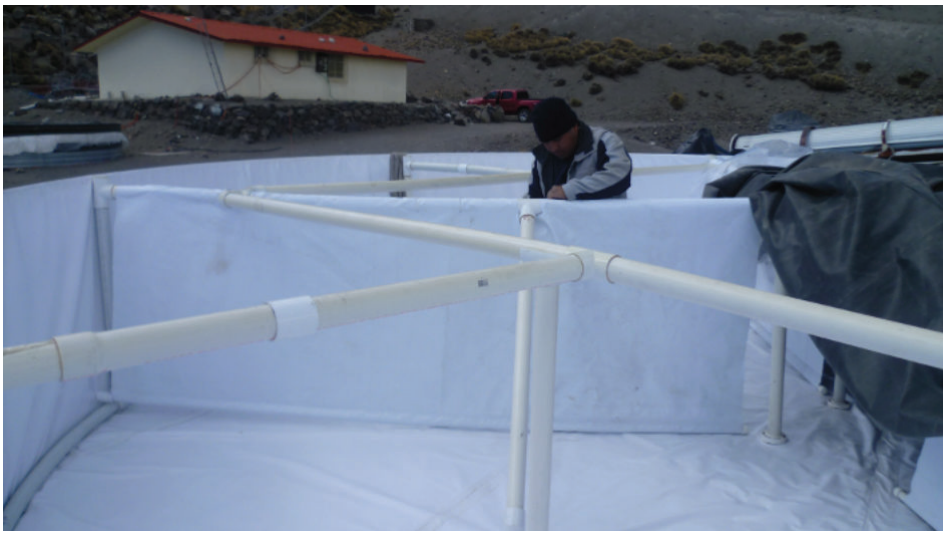


Figura 2.9: Vista interna de los detectores WCD con Tyvek. Para referenciar las dimensiones en el interior se ve a una persona haciendo arreglos.

una partícula que atraviesa el agua se mueve a una velocidad mayor que la velocidad de la luz en el agua, produce radiación Cherenkov, como se explicó en la Sección 2.5.1. Es decir, se estudia los GRBs de manera indirecta, a través de los EAS. El conjunto de WCDs permite la detección de la lluvias atmosféricas, generadas por rayos cósmicos de entre 10-100 TeV. La dirección de arribo del primario se obtiene comparando el tiempo de detección del frente de partículas de la lluvia atmosférica en diferentes detectores del arreglo. La energía del primario se puede obtener por la distribución y el tipo de partículas detectadas en el arreglo. Sin embargo, es importante mencionar que la técnica de la partícula individual no permite la medición de la energía y la dirección de los rayos gamma, ya que el número de partículas es muy pequeño para reconstruir los parámetros de la lluvia atmosférica [25].

Capítulo 3

Software de altas energías

En este Capítulo se detalla las herramientas utilizadas para las simulaciones de los EAS y de los WCDs. Para este trabajo se utilizó CORSIKA y Geant4, dos paquetes computacionales para simulaciones de altas energías.

3.1. CORSIKA

CORSIKA (COsmic Ray SImulations for KAscade) es un programa basado en el método Monte Carlo para la simulación detallada de lluvias atmosféricas extendidas iniciadas por partículas de rayos cósmicos de alta energía tales como protones, núcleos ligeros hasta hierro, fotones, entre otras. Describe fenómenos energéticos de hasta 100 EeV o $10^{20} eV$. En un inicio se desarrolló para el experimento KASCADE [17], y hoy en día se utiliza como software de altas energías para el estudio de EAS y las interacciones hadrónicas de alta y bajas energías. El desarrollo de CORSIKA está guiado por la idea de no solo predecir los valores promedios correctos con este código de simulación, sino también reproducir las fluctuaciones correctas alrededor de los valores promedios. CORSIKA utiliza un script en shell para correr. En este script se deben ingresar parámetros como la partícula primaria, rango de energía del primario, número de lluvias a ser generadas, angulo cenital y azimutal del primario, altitud sobre el nivel del mar del lugar, componentes magnéticas del lugar, módelo atmosférico, entre otros.

3.1.1. Sistema de coordenadas

Las coordenadas en CORSIKA se definen con respecto al sistema de coordenadas cartesianas. En el eje x positivo en la dirección del norte magnético, el eje y positivo con el este, y el eje z positivo hacia arriba. El origen está localizado a nivel del mar. Esta definición es necesaria debido a que se toma en cuenta el campo magnético terrestre en la deflección de la trayectoria de las partículas que impactan la Tierra. El ángulo cenital θ de una trayectoria de partícula es medido entre el vector momento de la partícula y el eje z negativo, y

el ángulo azimutal ϕ entre el eje x positivo y la componente $x-y$ del vector momento de la partícula medida en sentido anti-horaria.

3.1.2. Deflección en el campo magnético terrestre

El campo magnético terrestre es un parámetro a ser considerado en la simulación de los EAS, dado que al ser los rayos cósmicos partículas cargadas, estas se ven desviadas si su valor energético es considerablemente bajo. El campo magnético terrestre está caracterizado por su intensidad B_E, su ángulo de declinación δ, y su ángulo de inclinación ν. Para la locación KASKADE, estos valores son:

$$B_E = 47{,}80\mu T, \quad \delta = -9, \quad y \quad \nu = 64°44',$$

correspondientes con las componentes

$$B_x = 20{,}40\mu T \quad ; \quad B_z = -43{,}23\mu T,$$

mientras $B_y = 0$ por definición. Valores para otros lugares pueden ser obtenidos con el programa Geomag, el cual está disponible en la web[1].

3.1.3. Modelo atmosférico

En CORSIKA, la atmósfera consiste de 78,1 % de N_2, 21,0 % de O_2 y 0,9 % de Ar [22]. La variación de la densidad en la atmósfera está modelada por 5 capas. En las cuatro capas inferiores, la relación entre la variación de densidad con la altura está dada por:

$$T(h) = a_i + b_i e^{-h/c_i}, \quad i = 1, ..., 4, \tag{3.1}$$

donde T es la presión atmosférica, a_i, b_i y c_i son parámetros ajustables de tal manera que $T(h)$ se una función continua. En la quinta capa la variación de la densidad decrece linealmente con la altura,

$$T(h) = a_5 - b_5 \frac{h}{c_5}. \tag{3.2}$$

La densidad de masa atmosférica desaparece a $h = 112,8$ km. CORSIKA consta con un modelo estandar atmosférico y 7 modelos atmosféricos los cuales están definidos para cierto periodo del año. En esta simulación se utilizo el modelo estandar cuyos parámetros estan en la Cuadro 3.1.

3.2. Geant4

Geant4 (GEometry ANd Tracking) es un software computacional utilizado para simular el paso de partículas a través de la materia, usando el método de Monte Carlo. La

[1] https://www.ngdc.noaa.gov/geomag-web/

Capa i	Altitud h [km]	$a_i[g/cm^2]$	$b_i[g/cm^2]$	$c_i[g/cm^2]$
1	0...4	-186.5562	1222.6562	994186.38
2	4...10	-94.919	1144.9069	878153.55
3	10...40	0.61289	1305.5948	636143.04
4	40...100	0.0	540.1778	772170.16
5	100	0.01128292	1	10^9

Cuadro 3.1: Parámetros de la átmosfera estandar de Estados Unidos. La atmósfera está dividida en capas enumeradas del 1 al 5, con una determinada altitud. a_i, b_i y c_i son parámetros de ajuste para modelar la variación de densidad en la atmósfera descritos en el texto.

plataforma incluye herramientas para modificar [7]:

- Geometría del sistema,
- materiales involucrados,
- partículas fundamentales,
- generación de eventos primarios,
- el paso de partículas a través de materiales y campos electromagnéticos,
- procesos físicos en interacciones,
- respuestas de los componentes de un detector,
- generación de datos de un evento,
- almacenamiento de eventos y tracks,
- visualización del detector y trayectorias de partículas.

La Figura 3.2 muestra el diagrama de la categoria de clases utilizada en Geant4. La categoría "Global" cubre el sistema de unidades, constantes y manejo de número aleatorios. Las categorias "Materials and Particles" implementan facilidades para describir las propiedades de los materiales y de las partículas en la simulación de interacciones partícula-materia. El módulo "Geometry" ofrece la habilidad para describir una estructura geométrica y la propagación de las partículas. Las categorías superiores sirven para describir el paso de las partículas y los procesos físicos que experimentan. La categoría track contiene las clases las trayectorias y los pasos, utilizada por la categoría "processes" la cual contiene los modelos de las interacciones físicas: interacciones electromagnéticas de leptones, fotones, hadrones e iones, e interacciones hadrónicas.

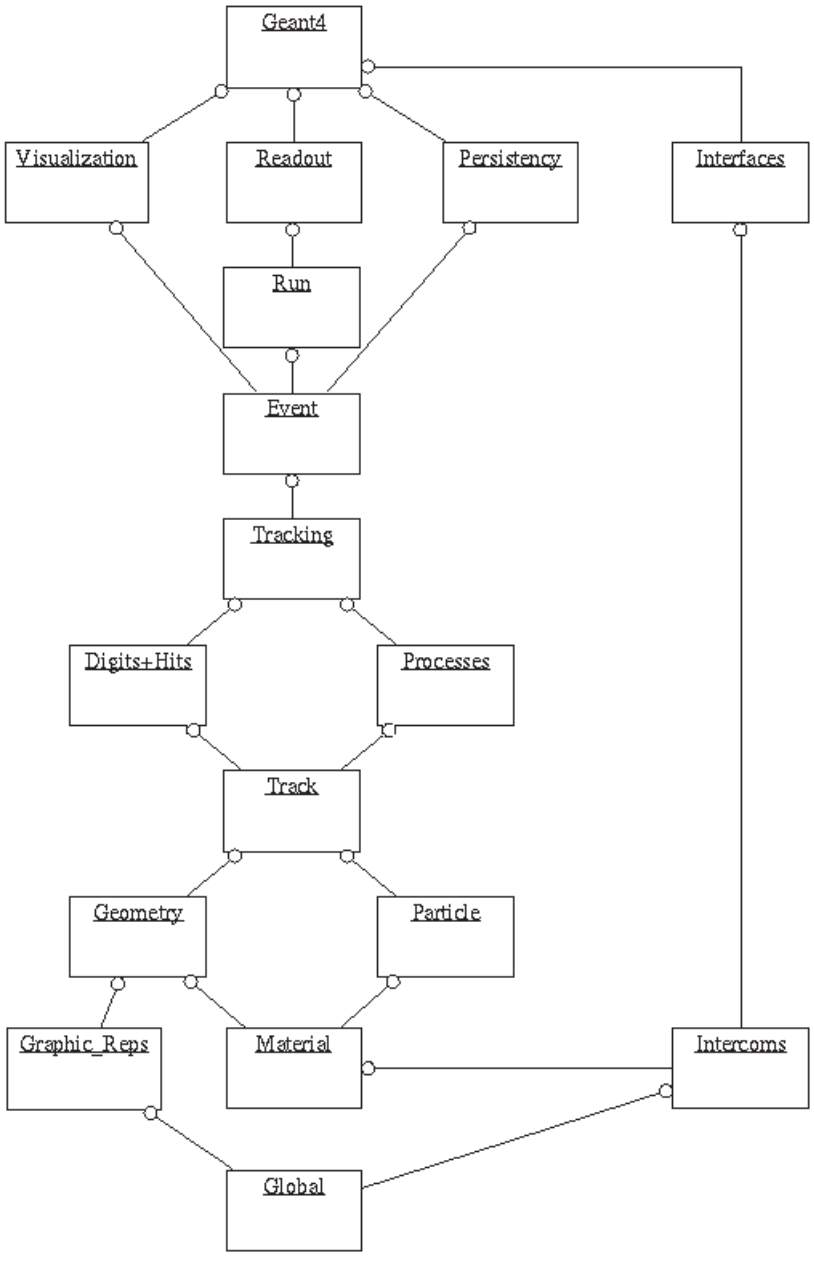


Figura 3.1: Diagrama de la categoria de clases en Geant4.

Capítulo 4

Metodología

En este Capítulo se describe el proceso para simular los EAS en CORSIKA y los detectores WCD en Geant4. Además, se describe el proceso utilizado para determinar la respuesta del detector o área efectiva de los detectores WCD de LAGO-México.

4.1. Procedimiento en CORSIKA

Para la simulación de los EAS es necesario seguir tres pasos: descargar e instalar el ejecutable de CORSIKA, elaborar el script de entrada de CORSIKA y ejecución de la simulación.

4.1.1. Instalación del ejecutable

El programa CORSIKA se descarga en https://www.ikp.kit.edu/corsika/[1]. Para crear el ejecutable se debe acceder en la terminal a la dirección de la carpeta y escribir[2] :

```
./coconut
```

A continuación, se debe elegir las opciones de modelo de interacción hadrónica para baja y altas energías, la opción de detector plano, y la opción de fotones Cherenkov:

- Modelo de interacción hadrónico a altas energías: QGSJET (Quark Gluon String model with JETs) en el intercambio supercrítico de Pomerones, los cuales son los cortes de acuerdo a la regla de Abramovskii-Gribov-Kancheli y formas dos cuerdas cada uno. Con este algoritmo se alcanza energías mayores a $10^{11} GeV$.
- A bajas energías se utiliza el paquete GHEISHA, que soporta energías en el sistema laboratorio de unos pocos GeV. Se utiliza solo para energías por debajo de los 80 GeV para interacciones de proyectiles hadrónicos con núcleos atmosféricos. Dentro de GHEISHA se considera la captura neutrónica para neutrones con $E_{lab} \leq 0{,}033 GeV$.

[1] Aunque es de libre distribución, para descargar el programa se debe solicitar una cuenta de acceso al correo del director.

[2] El cuadro de color oscuro indica que se debe escribir el comando en la terminal de linux.

- Opción detector plano: se escoge esta opción porque solo se quieren la información de las partículas que llegan a nivel del suelo .
- Opción de fotones Cherenkov: es necesario escoger esta opción porque permite ingresar el área donde va a caer el núcleo de los EAS (es decir el 90% de las partículas secundarias producidas por un primario).

Al final de estos pasos se crea un ejecutable con el nombre *corsika75600Linux_QGSJET_geisha*.

4.1.2. Archivo de entrada

La simulación de los chubascos atmosféricos se realiza mediante un archivo *all-inputs* que contiene los parámetros de entrada. En cada línea del archivo de entrada se escribe un parámetro y a continuación los valores que va a tomar ese parámetro, tal como se observa a continuación:

```
RUNNR 1                 number of run
EVTNR 1                 number of the first shower event
SEED 100 0 0            seed for hadronic part
SEED 200 0 0            seed for EGS4 part
SEED 300 0 0            seed for Cherenkov part
NSHOW 1000000           no of showers to simulate
PRMPAR 1                primary particle code (gamma)
ERANGE 0.5 10.E3        energy range of the primary (GeV)
ESLOPE -2.7             slople od energy spectrum
THETAP 0. 60.           range zenith angle (deg)
PHIP 0. 360.            range azimuth angle (deg)
QGSJET T 0              QSGJET for high energy and debug level
QGSSIG T                QSGJET cross-section enabled
HADFLG 0 0 0 0 0 2      HDPM interact. flags and fragmentation flag
ELMFLG T T              elmagn. interaction flags NKG, EGS4
STEPFC 1.               multiple scattering step length factor
RADNKG 200.E5           outer radious (cm) of NKG elect. distrib.
MAGNET 27.7 29.7        magnetic field in Sierra Negra (microT)
ATMOD 1                 atmospheric model
ECUTS 0.3 0.3 0.003 0.003 energy cuts: hadr. muon. elect. phot (GeV)
LONGI T 1. T F          longitud, step size (g/cm2), fit, out
MUMULT T                muon multiple scattering by Moliere
MUADDI T                aditional muon information
OBSLEV 4530.E2          observation level (cm)
MAXPRT 10               max number of printed events
CSCAT 1 35000. 35000.   scatter Cherenkov events
ECTMAP 1.E4             printout gamma factor cut
DIRECT ./               path of the particle output
USER user               user name for the data base file
HOST user               host name fot the data base file
DEBUG F 6 F 9999999     debug flag, log. unit, delayed debug
EXIT
```

Para correr este archivo de entrada en CORSIKA, se debe acceder mediante la terminal al directorio *run* donde se instaló CORSIKA y escribir en la consola:

```
./corsika75600Linux_QGSJET_geisha <all-inputs> salida.txt
```

El archivo *salida.txt* contiene información acerca de los EAS, como número de partículas secundarias al nivel de suleo, etc.

4.1.3. Archivo de salida

Este archivo es un binario con nombre “DAT0000N". N corresponde al número de corrida, especificada en el archivo de entrada *all-inputs*. Para transformar el binario en un archivo de datos, utilizamos el programa corsikaread.f que se encuentra en el directorio */src/utils* dentro del directorio de instalación del corsika. Escribimos en la terminal:

```
gfortran -fbounds-check corsikaread.f -o corsikaread
```

El comando creará un ejecutable llamado *corsikaread*. Ejecutamos escribiendo:

```
corsikaread <direccion >output
```

donde *direccion* es un archivo que contiene la dirección completa donde esta el archivo binario DAT000N, y se genera un *output* con la estructura del archivo de salida. El archivo de salida se llama fort.7 y contiene toda la información de las partículas codificada en bloques.

4.2. Geant4

Para la simulación de los detectores WCD se modificaron los archivos correspondientes a los ejemplos LXe y OpNovice, que vienen en la carpeta ejemplos, categoria “optical” de Geant. OpNovice simula el transporte y generación de fotones ópticos. Incluye procesos físicos ópticos (Cherenkov, Centelleo, absorción, Rayleigh, etc). Además se usa el mecanismo de conteo de secundarios generados. LXe es un proyecto que implementa el uso de material centellador y PMTs. Se modificaron los códigos de estos dos ejemplos para simular la geometría de los detectores Cherenkov de agua.

4.2.1. Descripción del WCD en Geant4

En Geant4, el diseño de un objeto está definido por capas, de tal forma que cada capa o geometría, representa una forma definida (cilindro, esfera, cubo, etc) y material (agua, metal, etc). Esto significa que cada geometría en Geant4 debe ser descompuesta en figuras geométricas más simples para ser simuladas. Un WCD es un tanque cilíndrico de metal, recubierto internamente de Tyvek, lleno de agua. La descomposición se puede observar en la Figura 4.1.

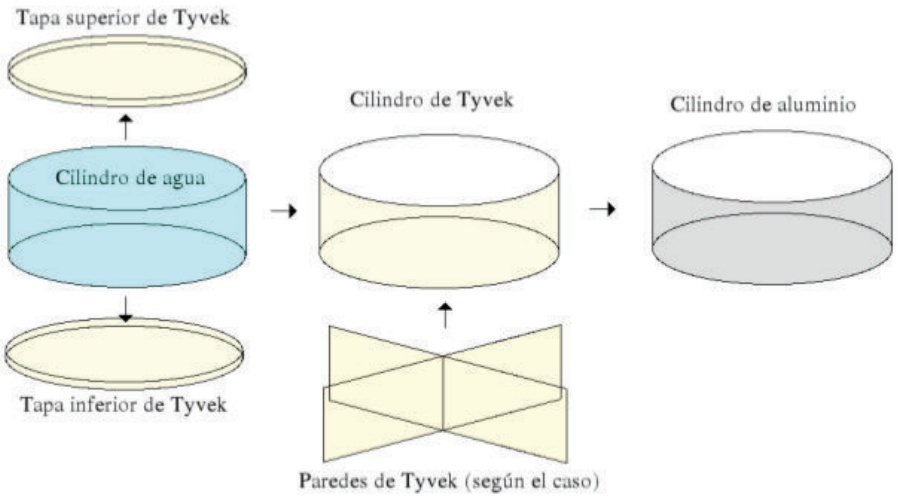


Figura 4.1: Composición de volúmenes de un detector WCD simulado en Geant4.

4.2.2. Definición de volúmenes

Un WCD simulado consiste de un tanque cilíndrico de aluminio lleno de agua, cuya cavidad cilíndrica interna, tapa y fondo fueron declaradas con las propiedades del Tyvek [3], tal como se ve en la Figura 4.1. El recubrimiento interno tiene propiedades de reflexión. El diámetro del tanque es de 7,3 m y su altura es de 1 m. La descripción de la geometría del detector fue definida en *OpNoviceDetectorConstruction.cc*.Por conveniencia, se colocó algunos valores para la descripción del detector al inicio del código. Los valores move-x2, move-y2, move-x3, move-y3, move-x4, move-y4, son las posiciones donde están el segundo, tercer y cuarto (central) tanque respecto del primero. A continuación se detalla lo antes mencionado:

```
1  OpNoviceDetectorConstruction::OpNoviceDetectorConstruction()
2   : G4VUserDetectorConstruction()
3  {
4    fExpHall_x = fExpHall_y =  100.0*m;
5    fExpHall_z = 20.0*m;
6    fTank_x    = fTank_y    = fTank_z    =  5.0*m;
7    fBubble_x  = fBubble_y  = fBubble_z  =  0.5*m;
8
9    scint_x = scint_y = 7.3*m;    //Diametro
10   scint_z = 1.0*m;             //Artura
11   Scint_z = 4.2*m;
12   outerRadius_pmtL = 10.3*cm;    //Radio de los fototubos laterales 8
        pulgadas
13
14 //Desplazamiento de posicion del tanque tank 2 respecto del tanque 1
```

[3]Las propiedades del Tyvek se pueden encontrar en la página web http://www.dupont.com

```
15    move_x2 = 24.74*m;    //Desplazamiento en x
16    move_y2 = 0.0*m;  //Desplazamiento en y
17
18  //Desplazamiento de posicion del tanque tank 3 respecto del tanque 1
19    move_x3 = 17.56*m;    //Desplazamiento en x
20    move_y3 = -21.64*m;  //Desplazamiento en y
21
22  //desplazamiento tanque central
23    move_x4 = 14.10*m;
24    move_y4 = -7.22*m;
25
26    wtyvek=0.2*mm;   //grosor tyvek=200 micras
27    d_mtl=0.635*cm; //width aluminum cylinder
28  }
```

Primero se construye el volumen mundo que por lo general es una caja, que contiene todo el experimento, la cual se ha llamado *The Experimental hall*. Para crearla se define el objeto geométrico con la clase *G4Box*, que crea un cubo. Después se le asigna un material a dicho objeto con la clase *G4LogicalVolumen*. Finalmente, se sitúa el objeto en el espacio físico con la clase *G4VPhysicalVolume*. A continuación se detalla lo antes mencionado:

```
1  // The experimental Hall
2    G4Box* expHall_box = new G4Box("World",fExpHall_x,fExpHall_y,
         fExpHall_z);
3
4    G4LogicalVolume* expHall_log
5      = new G4LogicalVolume(expHall_box,Vacuum,"World",0,0,0);
6
7    G4VPhysicalVolume* expHall_phys
8      = new G4PVPlacement(0,G4ThreeVector(),expHall_log,"World",0,false,0)
           ;
```

Esto se debe hacer con cada parte de la Figura 4.1. Para la sección del agua del tanque se procede de la misma manera. Se crea el objeto cilíndrico con *G4Tubs*. Después se le asigna el material que en este caso es agua al objeto antes creado. Luego se sitúa el volumen lógico en el espacio. Dado que los tres tanques exteriores son iguales, es suficiente con replicar los tanques con la clase *G4VPhysicalVolume* cambiando al final el número de réplica como se observa en el código. El tanque central al tener una altura distinta debe ser creado desde el inicio. A continuación se detalla lo antes mencionado:

```
1  // The Water Tank
2    G4Tubs* waterTank_box = new  G4Tubs("Tank",           //Nombre
3                                        0.,             //radio interno
4                                        scint_x/2.,     //radio externo
5                                        scint_z/2.,     //mitad de
```

```
                                          longitud en z
6                                       0.*deg,         //comienzo en
                                          phi
7                                       360.*deg);      //segmento del
                                          angulo
8
9     G4LogicalVolume* waterTank_log
10      = new G4LogicalVolume(waterTank_box,water,"Tank",0,0,0);
11  //Tanque 1
12    G4VPhysicalVolume* waterTank_phys
13      = new G4PVPlacement(0,G4ThreeVector(),waterTank_log,"Tank1",
14                          expHall_log,false,1);
15  //Tanque 2
16    G4VPhysicalVolume* waterTank_phys2
17      = new G4PVPlacement(0,G4ThreeVector(move_x2,move_y2,0.0*m),
          waterTank_log,"Tank2",
18                          expHall_log,false,2);
19  //Tanque 3
20    G4VPhysicalVolume* waterTank_phys3
21      = new G4PVPlacement(0,G4ThreeVector(move_x3,move_y3,0.0*m),
          waterTank_log,"Tank3",
22                          expHall_log,false,3);
23  //tanque central
24    G4Tubs* waterTank_box_c = new  G4Tubs("Tank_c",              //Nombre
25                                       0.,             //radio interno
26                                       scint_x/2.,     //radio externo
27                                       Scint_z/2.,     //mitad de
                                          longitud en z
28                                       0.*deg,         //comienzo en
                                          phi
29                                       360.*deg);      //segmento del
                                          angulo
30
31    G4LogicalVolume* waterTank_log_c
32      = new G4LogicalVolume(waterTank_box_c,water,"Tank_c",0,0,0);
33    G4VPhysicalVolume* waterTank_phys4
34      = new G4PVPlacement(0,G4ThreeVector(move_x4,move_y4,1.6*m),
          waterTank_log_c,"Tank4",
35                          expHall_log,false,4);
```

4.2.3. Definición de los PMTs

Los PMTs se diseñaron dándole la forma de un casquete esférico al fotocátodo, a partir de una esfera. Se lo definió con Aluminio, y fueron situados a una distancia de 2.03 m en cada eje respecto del centro. Dado que los PMTs van situados en el volumen lógico del

agua, solo se deben colocar los cuatro PMTs allí, y estarán situados en cada réplica de los detectores. A continuación se detalla lo antes mencionado:

```
//PMTs
G4double sphere_w = 0.5*mm;     //sphere_width
G4double cone_w = 0.0*mm;       //cone_width
G4double long_cone = 30*cm;     //long cone
G4double sshift=outerRadius_pmtL*0.5;  //desplazamiento

G4Sphere* photocathL = new G4Sphere("photocathL",outerRadius_pmtL-
    sphere_w,outerRadius_pmtL,0.*deg,360.*deg,0.*deg,60.*deg);

G4LogicalVolume* photocath_logL= new G4LogicalVolume(photocathL,
    G4Material::GetMaterial("Al"),"photocath_logL");

G4VPhysicalVolume* photocatht1_phys = new G4PVPlacement(rm,G4ThreeVector
    (203.0*cm,203.0*cm,scint_z/2.+sshift), photocath_logL,"photocatht1",
     waterTank_log, false, 10);

G4VPhysicalVolume* photocatht2_phys = new G4PVPlacement(rm,G4ThreeVector
    (-203.0*cm,203.0*cm,scint_z/2.+sshift), photocath_logL,"photocatht2"
    , waterTank_log, false, 11);

G4VPhysicalVolume* photocatht3_phys = new G4PVPlacement(rm,G4ThreeVector
    (203.0*cm,-203.0*cm,scint_z/2.+sshift), photocath_logL,"photocatht3"
    , waterTank_log, false, 12);

G4VPhysicalVolume* photocatht4_phys = new G4PVPlacement(rm,G4ThreeVector
    (-203.0*cm,-203.0*cm,scint_z/2.+sshift), photocath_logL,"photocatht4
    ", waterTank_log, false, 13);
```

4.2.4. Definición de los procesos físicos

Reflexión del Tyvek

La reflexión en el Tyvek está definida en el archivo *OpNoviceDetectorConstruction.cc*. Este proceso está definido en función de las propiedades del Tyvek, por lo que se debe caracterizar el material. La forma de ingresar el proceso es asociando un índice de reflexión del Tyvek en función de la energía de los fotones Cherenkov. Estos valores se obtuvieron experimentalmente, por la colaboración de LAGO-México, midiendo con láseres la cantidad de luz reflejada en el Tyvek para diferentes longitudes de onda. Los valores de energía del fotón y del índice de reflexión son ingresados en el código en arreglos:

```
  G4double Ephoton[34] = {
4.960*eV,4.769*eV,4.428*eV,4.133*eV,
```

```
3.875*eV,3.647*eV,3.444*eV,3.351*eV,
3.263*eV,3.170*eV,3.100*eV,3.024*eV,
2.952*eV,2.883*eV,2.818*eV,2.755*eV,
2.695*eV,2.583*eV,2.530*eV,2.480*eV,
2.384*eV,2.296*eV,2.138*eV,2.066*eV,
2.00*eV,1.938*eV,1.879*eV,1.823*eV,
1.771*eV,1.722*eV,1.675*eV,1.631*eV,
1.590*eV,1.550*eV};

  G4double TRef[34] ={
0.82,0.86,0.89,0.92,0.94,
0.95,0.95,0.95,0.96,0.96,
0.97,0.97,0.97,0.97,0.97,
0.97,0.97,0.97,0.97,0.97,
0.97,0.97,0.97,0.97,0.97,
0.97,0.97,0.97,0.97,0.97,
0.97,0.97,0.97,0.97};
```

Luego, se crea un objeto de la clase *G4MaterialPropertiesTable* al cual se le añade la propiedad de reflexión con el método *AddProperty* donde se ingresa como parámetros los valores de energía de fotón e índice de reflexión. Luego se define la superficie óptica como un metal dieléctrico mediante la clase *G4OpticalSurface*, a la cual se le añade la propiedad de reflexión. Finalmente, se añade la superficie óptica a cada parte del WCD: tapa inferior y superior, cilindro y paredes internas. A continuación se detalla lo antes mencionado:

```
G4MaterialPropertiesTable* optyvek = new G4MaterialPropertiesTable();
optyvek->AddProperty("REFLECTIVITY", Ephoton, TRef, num);
G4OpticalSurface* OpTyvekSurface = new G4OpticalSurface("TyvekSurface",
    unified,polished,dielectric_metal);
OpTyvekSurface->SetMaterialPropertiesTable(optyvek);
new G4LogicalSkinSurface("tyvekw_surface",tyvek_log,OpTyvekSurface);
new G4LogicalSkinSurface("tyvekb_surface",tyvekbottom_log,OpTyvekSurface
    );
new G4LogicalSkinSurface("tyvekt_surface",tyvektop_log,OpTyvekSurface);
new G4LogicalSkinSurface("tyvekwlx_surface",tyvekwallshortx1_log,
    OpTyvekSurface);
new G4LogicalSkinSurface("tyvekwlx_surface",tyvekwallshorty1_log,
    OpTyvekSurface);
```

Radiación Cherenkov

La radiación Cherenkov se define en el archivo *LXePhysicsList.cc*. En este archivo se incluye en la cabacera las librerias *"G4Cerenkov.hh"* y *"G4OpticalPhysics.hh"*. Se crea un objeto de la clase *G4OpticalPhysics* y utlizando el método *SetTrackSecondariesFirst* se activa

los fotones Cherenkov. A la vez se desactiva el proceso de centelleo, con false, o colocando 0 en el factor de producción de centelleo:

```
G4OpticalPhysics* opticalPhysics = new G4OpticalPhysics();
RegisterPhysics( opticalPhysics );
opticalPhysics->SetWLSTimeProfile("delta");
opticalPhysics->SetScintillationYieldFactor(0);
opticalPhysics->SetScintillationExcitationRatio(0);
opticalPhysics->SetMaxNumPhotonsPerStep(100);
opticalPhysics->SetMaxBetaChangePerStep(10.0);
opticalPhysics->SetTrackSecondariesFirst(kCerenkov,true);
opticalPhysics->SetTrackSecondariesFirst(kScintillation,false);
```

4.2.5. Conexión de datos Corsika-geant

La conexión de datos de corsika con geant4 se realiza en el archivo *OpNovicePrimaryGeneratorAction.cc*, que lee la información de los EAS contenida en el archivo ya.dat. Por tanto, se debe descomponer el archivo fort.7 en los archivos ya.dat con la información de cada una de las lluvias. Esto se realiza con el ejecutable *calculo_n_archivos* que lee el archivo fort.7 y genera N archivos ya(n).dat cada uno con información de los secundarios producidos por cada lluvia, donde N corresponde al número de lluvias simuladas en Corsika (n = 1, 2, ..., N). La estructura de los archivos ya(n).dat se observa en el Cuadro 4.1. En la primera fila se encuentra la información de la lluvia: área donde se arrojaron los EAS, energía del primario en (GeV), ángulo θ del primario, ángulo ϕ del primario, posición en el eje x y y donde impacta el núcleo de la lluvia. La segunda fila del archivo ya(N).dat contiene la información de los secundarios generados por esa lluvia: tipo de partícula, momento lineal en x, y, z del secundario, posición en (x, y) del secundario, tiempo de arribo del secundario en [ns] y energía del secundario en [GeV].

1era línea: Cabecera de datos							
Área	Energía del primario [GeV]	Ángulo θ primario [°]	Ángulo ϕ primario [°]	Posición del Core X [cm]	Posición del Core Y [cm]	0	0
2da línea: Información de partícula secundaria (si es que la tiene)							
ID-Partícula	Momento en X [GeV/c]	Momento en Y [GeV/c]	Momento en Z [GeV/c]	Posición en la coordenada X [cm]	Posición en la coordenada Y [cm]	Tiempo desde la primera interacción [ns]	Energía de la partícula [GeV]
Continuación de líneas de partículas secundarias (si es que la tiene)							
...	...	...	...	...	...	...	...

Cuadro 4.1: Estructura del archivo ya(n).dat.

La información del archivo ya(n).dat es ingresada en geant a traves del archivo *Op-*

NovicePrimaryGeneratorAction.cc, donde se ingresan los tipos de secundarios, asi como su energía.

4.2.6. Datos de salida de Geant4

El archivo de salida que entrega geant4 se llama *dato.dat* y tiene la estructura que se muestra en el Cuadro 4.2.

1era línea									
Área	Energía del primario [GeV]	Ángulo θ primario [$\circ$]	Ángulo ϕ primario [$\circ$]	Posición del Core X [cm]	Posición del Core Y [cm]	0	0	0	0
2da línea									
ID-partícula	Momento en X [GeV/c]	Momento en Y [GeV/c]	Momento en Z [GeV/c]	Posición en X [cm]	Posición en Y [cm]	0	0	Código PMT	Radio PMT

Cuadro 4.2: Estructura del archivo de salida dato.dat

Capítulo 5

Resultados y Análisis

En este Capítulo se presentan los resultados de la simulaciones: distribución de las partículas secundarias sobre los tanques, dimensiones del observatorio LAGO-México, geometría de tanque central, distribuciones temporales de los fotones Cherenkov detectados por los PMTs, Integral de los fotones Cherenkov en el detector central, gráfica del área efectiva para un gamma que incide verticalmente en la atmósfera, entre otras. Se describe, cada una de las gráficas, y se hace comparaciones en la distribución temporal de fotones Cherenkov para el detector central con y sin Tyvek.

5.1. Simulación de EAS

Se simularon 1000 cascadas producidas por un protón y por un gamma de 10 TeV. La Figura 5.1 presenta la distribución de energía de las partículas secundarias generadas por un gamma de 10 TeV que incide verticalmente en la atmósfera, y la Figura 5.2 muestra la distribución de energía de las partículas secundarias generadas por un protón de 10 TeV que incide verticalmente en la atmósfera .

La diferencia entre la Figura 5.1 y 5.2, radica en la cantidad de componente muónica. La distribución de las partículas en función de la distancia al eje de la cascada para una muestra de 10 lluvias se observa en la Figura 5.3 para un gamma de 10 TeV y en la Figura 5.4 para un protón de 10 TeV.

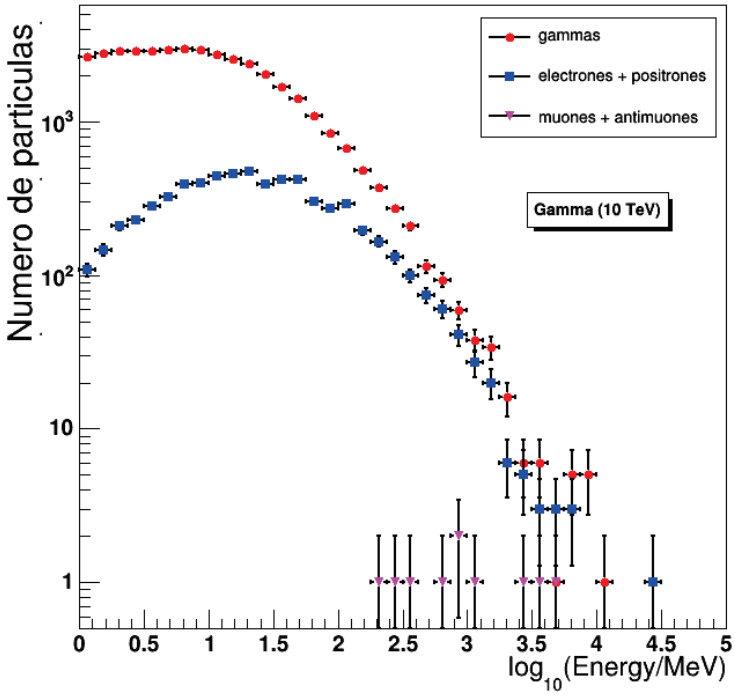


Figura 5.1: Distribución energética de las partículas secundarias generadas por un gamma de 10 TeV con $\theta = 0°$.

La Figura 5.4 muestra una cantidad de muones considerable distribuidas en una área de 99 923.39 $[m^2]$, en comparación con las de la Figura 5.3. Se demuestra que la diferencia entre una cascada hadrónica y una cascada electromagnética es la componente muónica. Se determinó que la tasa de muones producidos a partir de un hadrón con respecto de las producidas por un gamma es:

$$\frac{\mu^{\pm}_{hadron}}{\mu^{\pm}_{\gamma}} \approx \frac{20}{1}. \tag{5.1}$$

En los Cuadros 5.1 y 5.2, se presentan la composición de partículas secundarias generadas, por un gamma y protón, respectivamente. Se determina que la proporción de abundancias de partículas al nivel de 4530 m.s.n.m. es aproximadamente $\gamma : e^{\pm} : \mu^{\pm} = 10^4 : 10^3 : 1$ para gammas de 10 TeV, y $\gamma : e^{\pm} : \mu^{\pm} = 10^3 : 10^2 : 1$ para protones de 10 TeV. La energía media de los muones a una altidud de 4530 m.s.n.m. es $E_{\mu^{\pm}} \approx 4$ [GeV] producidos por un gamma primario, y $E_{\mu^{\pm}} \approx 8$ [GeV] producidos por un protón primario.

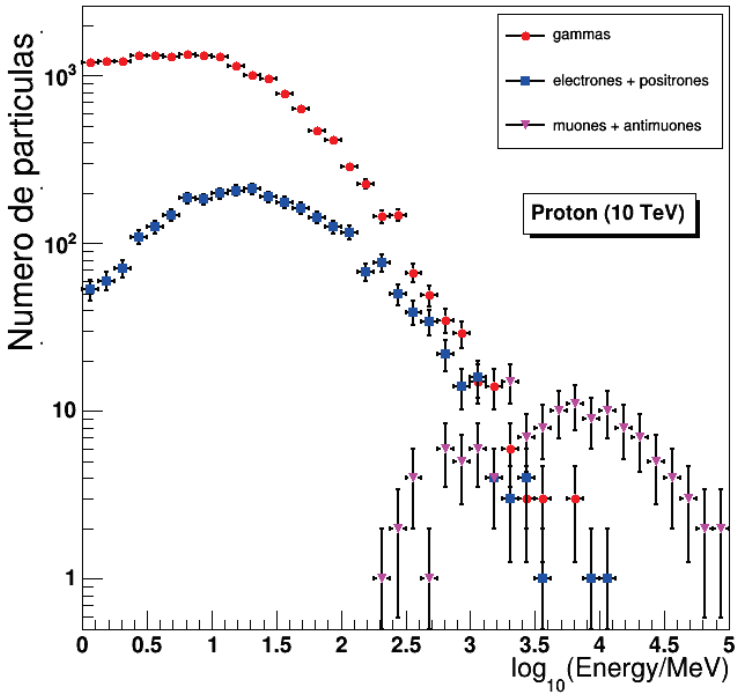


Figura 5.2: Distribución energética de las partículas secundarias generadas por un protón de 10 TeV con $\theta = 0°$.

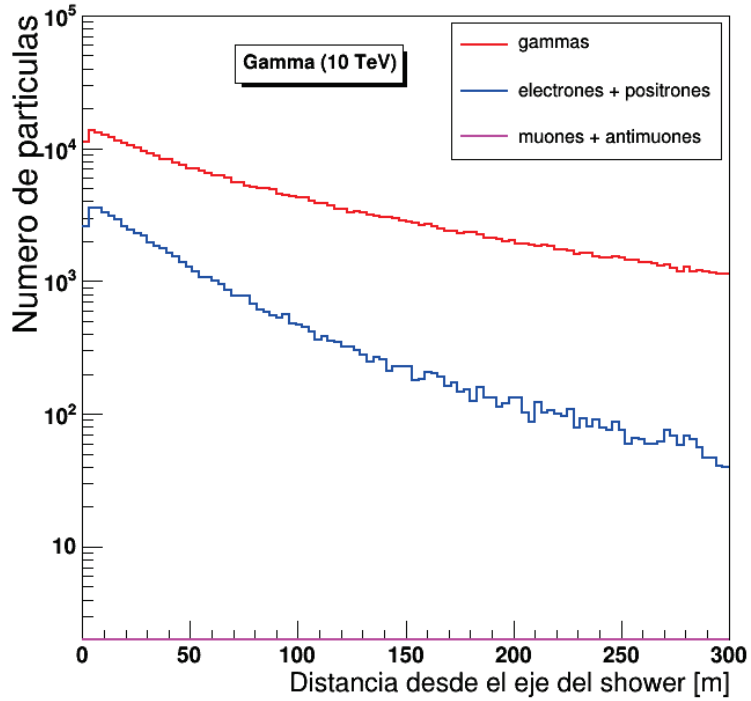


Figura 5.3: Distribución lateral de las partículas secundarias generadas por un gamma de 10 TeV con $\theta = 0°$.

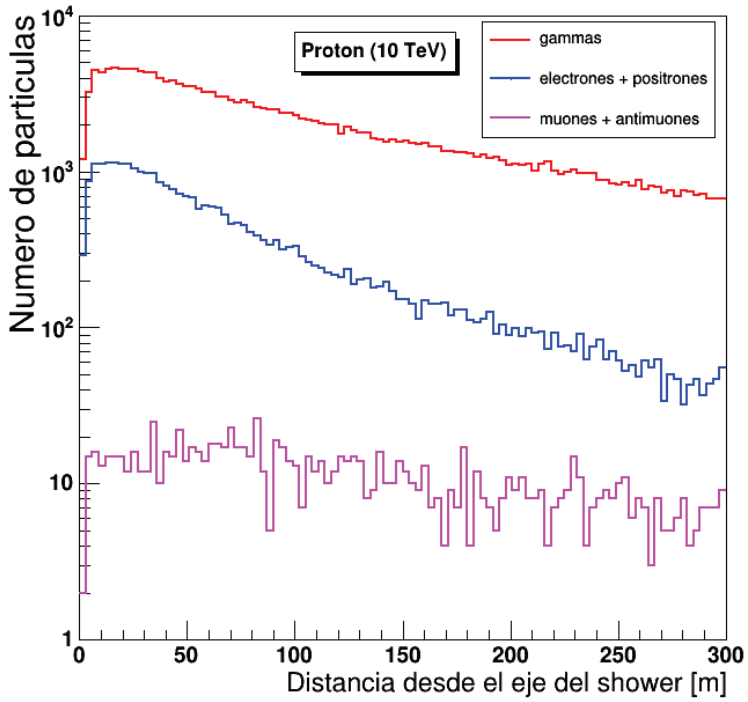


Figura 5.4: Distribución lateral de las partículas secundarias generadas por un proton de 10 TeV con $\theta = 0°$.

Partícula secundaria	Nro. de secundarios	Energía media [MeV]
γ	48 483 806	34.08 ± 0.08117
e^-	4 089 802	98.38 ± 1.07
e^+	2 329 954	154.1 ± 0.5593
μ^-	4 914	3771 ± 116.1
μ^+	4 943	3853 ± 119

Cuadro 5.1: Composición de un EAS de 1000 lluvias generadas por un gamma de 10 TeV con $\theta = 0°$.

Partícula secundaria	Nro. de secundarios	Energía media [MeV]
γ	2 4261 997	65.89 ± 0.09324
e^-	2 093 712	139.1 ± 0.5639
e^+	1 207 233	204.9 ± 0.9432
μ^-	106 259	7863 ± 40.4
μ^+	108 620	7956 ± 40.23

Cuadro 5.2: Composición de un EAS de 1000 lluvias generadas por un protón de 10 TeV con $\theta = 0°$.

5.2. Simulación del detector central.

La función del detector central es de trazador de partículas, lo que significa que debe determinar la naturaleza de las partículas que lo atraviesan. Para determinar si el detector central podrá diferenciar entre gammas, electrones y muones, se debe encontrar el "rise time" en la integral temporal de cada partícula. El detector central tiene 7,3 m de diámetro y 4,5 m de alto.

Primero, se determinó la posición de los PMTs respecto al centro del detector. Se realizó una primera prueba colocando 10 PMTs a distintos radios para contar el número de fotones, producidos por radiación Cherenkov. La Figura 5.5 muestra el tanque con los PMTs colocados en la parte inferior. La Figura 5.6 muestra los fotones Cherenkov detectados en función de la distancia al centro del WCD (r) para un muón de 10 TeV que incide verticalmente en el centro del WCD, con Tyvek y sin Tyvek.

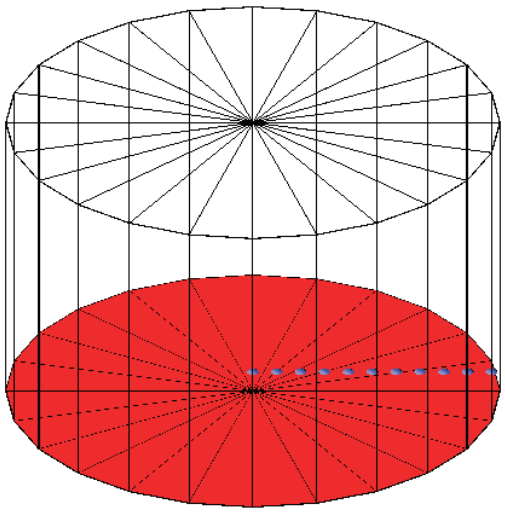


Figura 5.5: Vista 3D del detector central simulado en Geant4. Los PMTs (en color azul) se encuentran distribuidos a lo largo de un solo eje sobre la base del detector (en color rojo).

En una segunda prueba preliminar, se introdujeron 100 muones verticales de 10 GeV distribuido aleatoriamente sobre un área de 3.65x3.65 m^2 (ver Figura 5.7), cubriendo al tanque donde se localiza un arreglo de 20 PMTs en el eje x, y 20 PMTs en el eje Y, sin contar el PMT en la posición (0,0). La geometría se puede ver en la Figura 5.8. Se contaron los fotones Cherenkov registrados por los grupos de PMTs ubicados a la misma distancia del centro del detector. Se hizo la simulación con y sin Tyvek. El 76 % de los muones generados cayeron dentro del detector, con 37036 fotones Cherenkov que golpean los PMTs sin Tyvek, y 162817 fotones Cherenkov que golpean los PMTs con Tyvek. El número de fotones Cherenkov registrados en función de la distancia al centro del tanque está en la Figura 5.9.

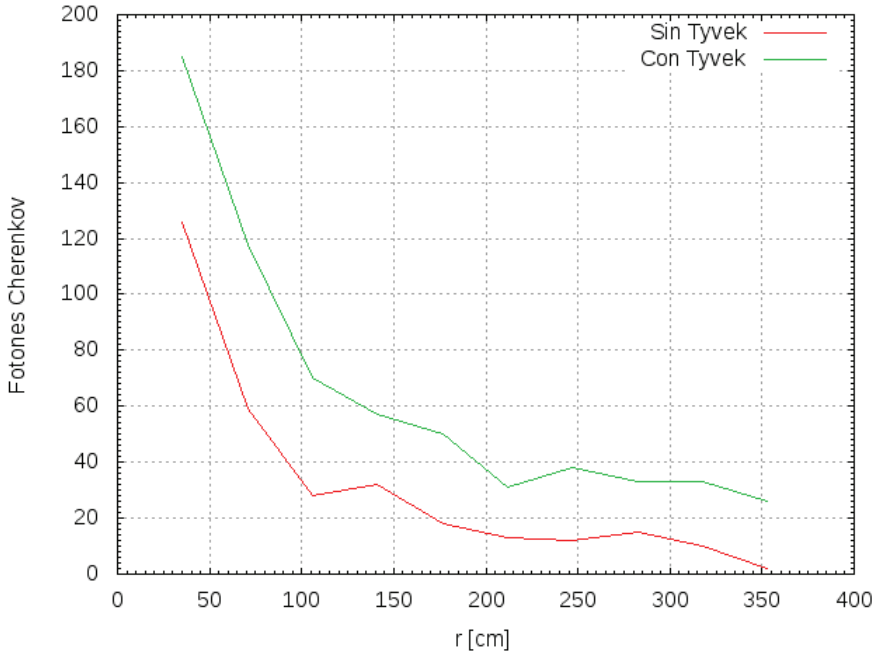


Figura 5.6: Número de fotones Cherenkov detectados (en el eje y), producidos por un muón vertical de 10 TeV, en función de la distancia al centro del WCD (r) en [cm]. Se muestra en color verde y rojo, los fotones Cherenkov detectados con y sin Tyvek, respectivamente.

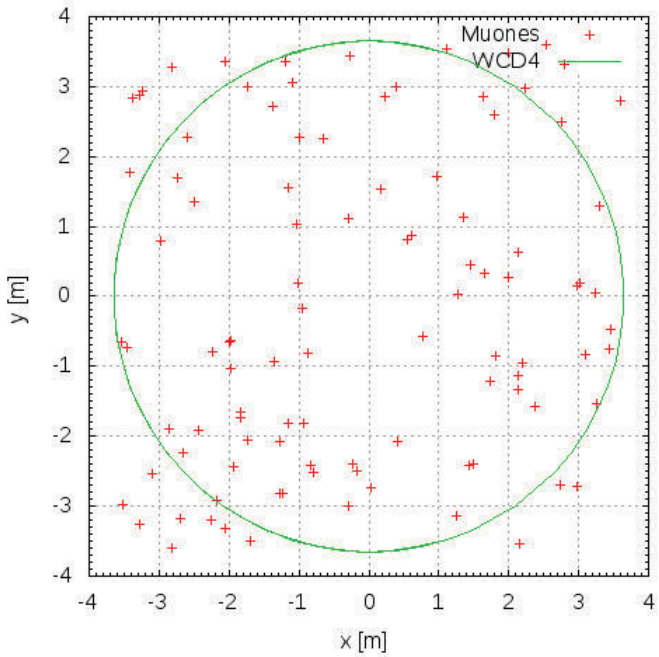


Figura 5.7: Distribución de 100 muones en un área de 3,65x3,65 m^2. El plano-xy representa la proyección sobre el suelo. El círculo de color verde representa el detector central denotado por WCD4, y las cruces rojas representan la posición donde impactan los muones arrojados.

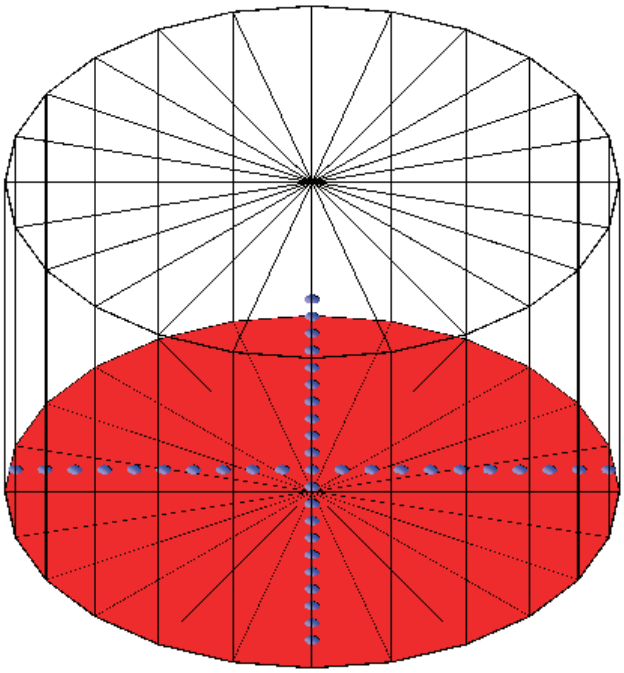


Figura 5.8: Vista 3D del detector central simulado en Geant4. Los PMTs (en color azul) se encuentran distribuidos a lo largo de los ejes x, y sobre la base del detector (en color rojo).

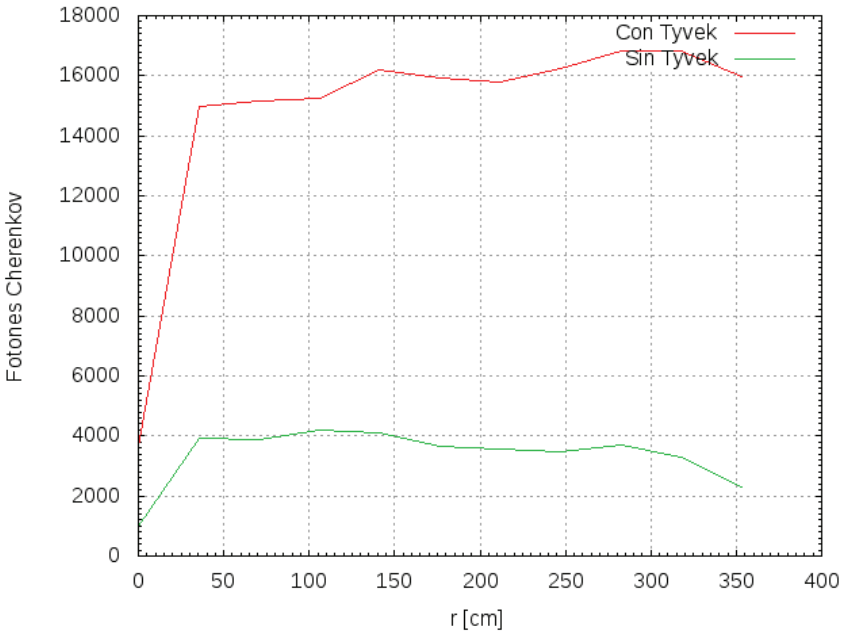


Figura 5.9: Número de fotones Cherenkov detectados (en el eje y), producidos por 100 muones verticales de 100 GeV, en función de la distancia al centro del WCD (r) en [cm]. Se muestra en color rojo y verde, los fotones Cherenkov detectados con y sin Tyvek, respectivamente.

5.2.1. Determinación de la posición de los PMTs

Para determinar la posición de los PMTs en el detector central, se colocaron cuatro PMTs equidistantes respecto al centro del tanque, con determinadas distancias. Primero, se arrojaron 1000 muones verticales con una energía de 10 GeV, y una distribución aleatoria, como se indica en la Figura 5.10. El 79,7 % de los muones cayeron dentro del tanque. Se produjeron 162 765 284 fotones Cherenkov, de los cuales 1 654 998 impactaron en los PMTs. En esta prueba se contabilizó el número de fotones producidos por cada muón vertical, que golpearon los PMTs. En la Figura 5.11 se observa los fotones Cherenkov generados por cada muón, que impactaron en los PMTs ubicados a 70,6 cm del centro del detector.

Se realizaron los histogramas de los fotones Cherenkov que impactaron en cada grupo de PMTs, los cuales se visualizan en las Figuras 5.12, 5.13 y 5.14. En la Figura 5.15 se presenta el número promedio de fotones Cherenkov detectados por los PMTs a diferentes distancias del centro del detector, para el tanque cubierto con material Tyvek y sin Tyvek. Se observa que la mejor posición se da entre 150 [cm] y 200 [cm], tanto con Tyvek como sin Tyvek.

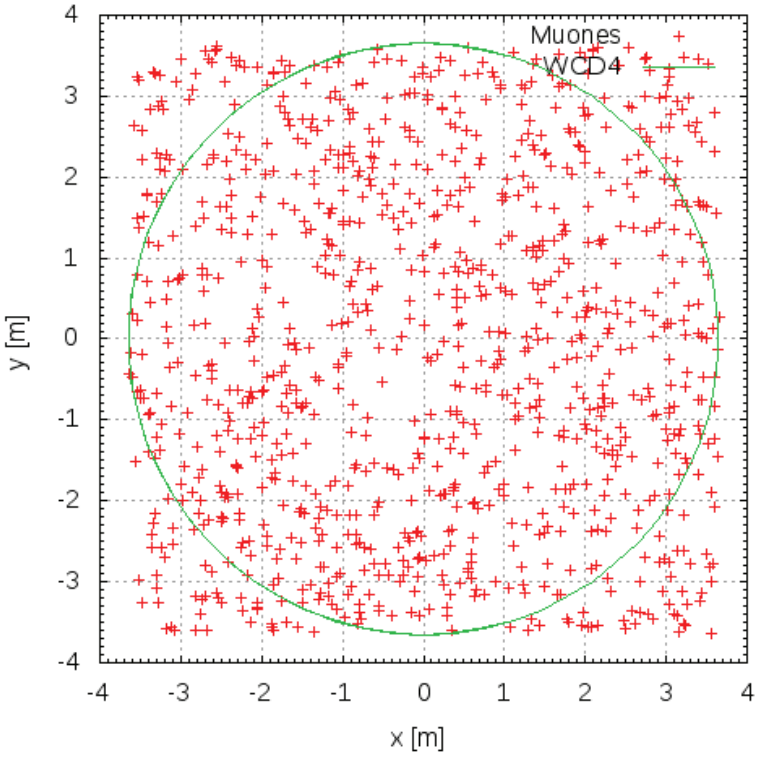


Figura 5.10: Distribución de 1000 muones verticales de 10 GeV en un área de 3,65x3,65 m^2. El plano-xy representa la proyección sobre el suelo. El círculo de color verde representa el detector central denotado por WCD4, y las cruces rojas representan la posición donde impactan los muones arrojados.

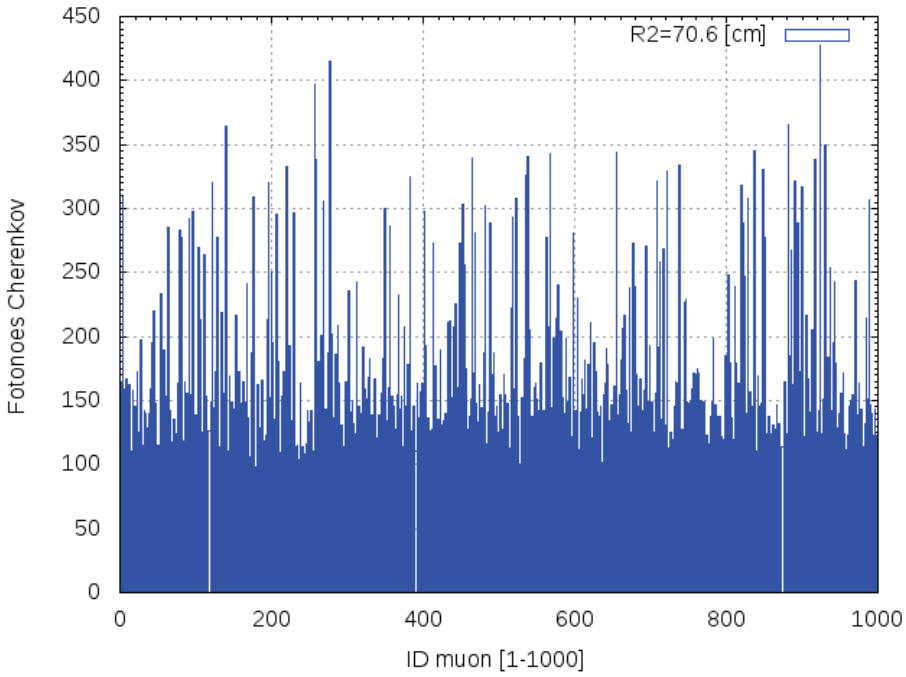


Figura 5.11: En el eje y se representa el número de fotones Cherenkov (detectados por los PMTs ubicados a 70,6 cm del centro del detector) que fueron generados por cada uno de los 1000 muones verticales. En el eje x, se encuentran distribuidos los 1000 muones que se arrojaron, identificados con un número (ID-muon) del 1 al 1000.

5.2.2. Determinación del Rise time

Se simuló un EAS de 1000 chubascos producido por un protón de 10 TeV de energía que incide verticalmente en la atmósfera. Se determinó la composición del EAS y la energía media de las partículas secundarias, que se muestra en el Cuadro 5.3.

Partícula Secundaria	Nro. de secundarios	Energía media [GeV]
γ	2 4261 997	0.07 ± 0.00
e^-	2 093 712	0.14 ± 0.00
e^+	1 207 233	0.20 ± 0.00
μ^-	106 259	7.86 ± 0.04
μ^+	108 620	7.96 ± 0.04

Cuadro 5.3: Composición de un EAS de 1000 chubascos generado por un protón de 10 TeV con un ángulo cenital de 0°.

Se determinó la distribución temporal para 500 muones con energía de 8 GeV, 1000 electrones de 140 MeV y 10000 gammas de 70 MeV, los cuales ingresan al tanque central verticalmente. Se hizo el análisis con y sin Tyvek. La comparación se muestra en las Figuras 5.16, 5.17, 5.18.

En las Figuras 5.16, 5.17 y 5.18 se observan múltiples picos en el caso que el detector está

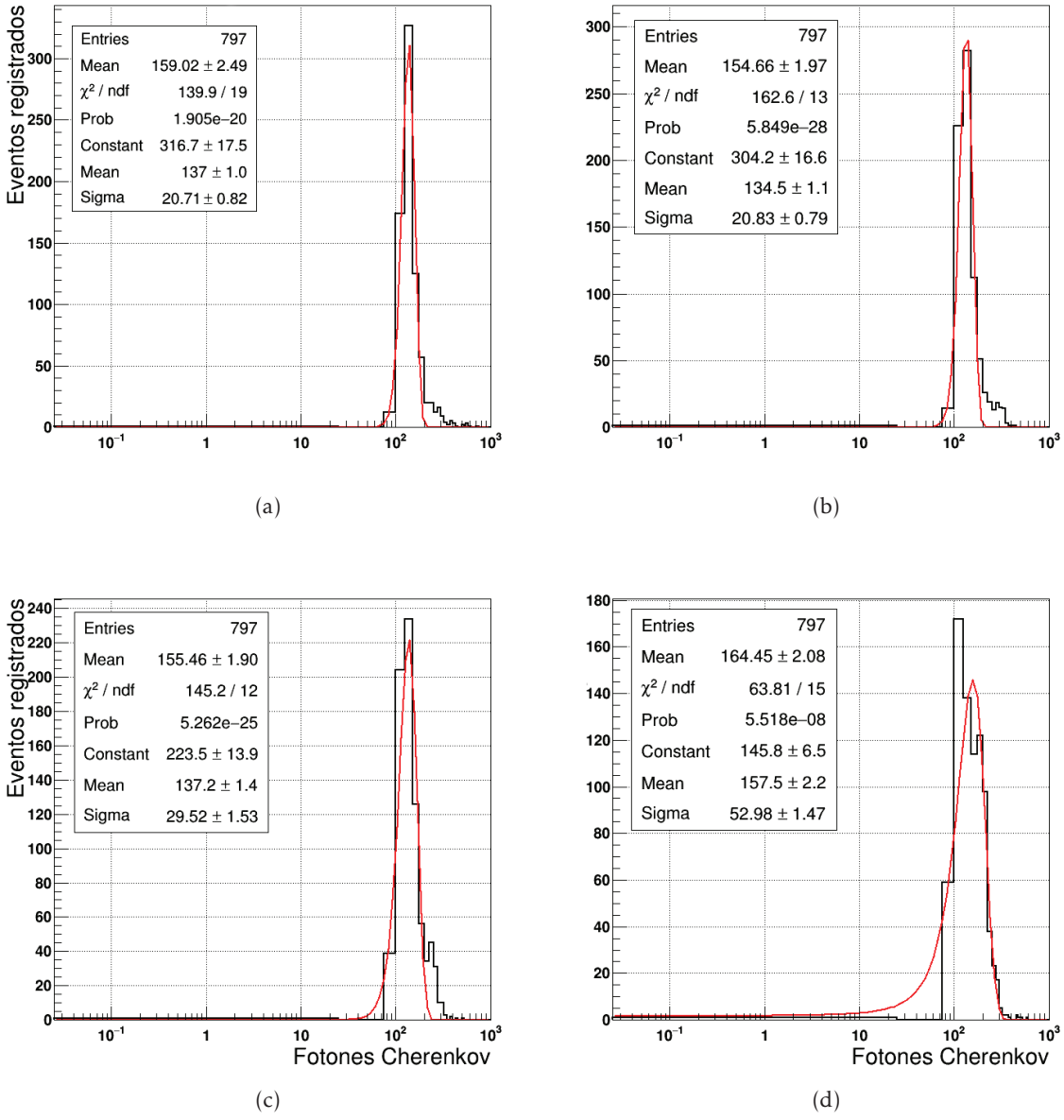


Figura 5.12: Histograma de fotones Cherenkov detectados para PMTs ubicados entre 35.3 cm y 141.2 cm respecto al centro del detector: (a) PMTs ubicados 35.3 cm, (b) PMTs ubicados 70.6 cm, (c) PMTs ubicados 105.9 cm, y (d) PMTs ubicados 141.2 cm. La línea negra gruesa es el histograma de los fotones Cherenkov dados por la simulación, y la línea roja delgada es el ajuste gaussiano.

con Tyvek. Cuando está sin Tyvek, se tiene solamente un pico. El primer pico corresponde a los fotones Cherenkov detectados directamente, mientras que los demás picos corresponden a los fotones Cherenkov detectados por los PMTs tras varias reflexiones en las paredes.

Para diferenciar las partículas secundarias a nivel del suelo, se debe analizar el rise time, que corresponde al tiempo en que la señal de los PMTs se incrementa del 10 % al 90 %.

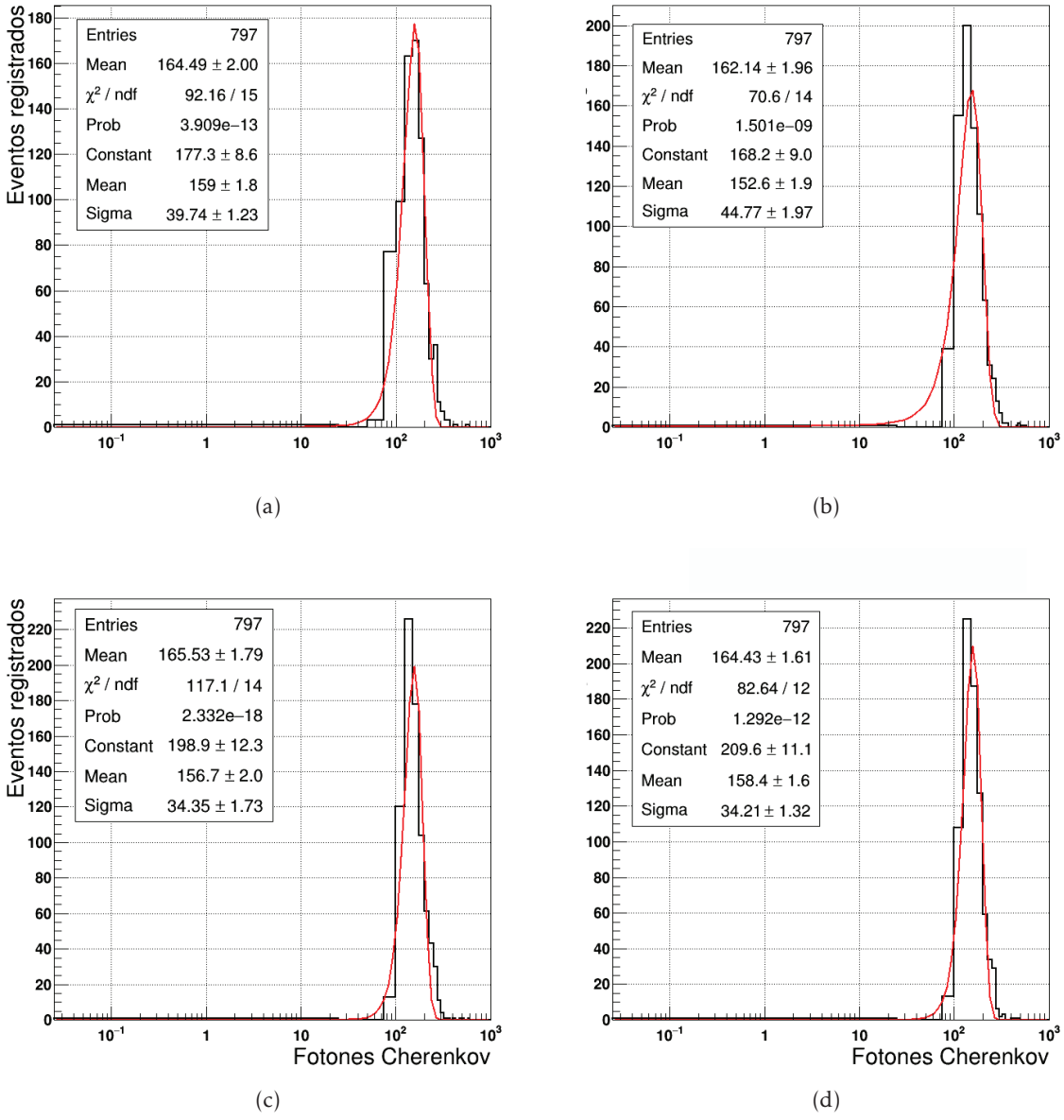
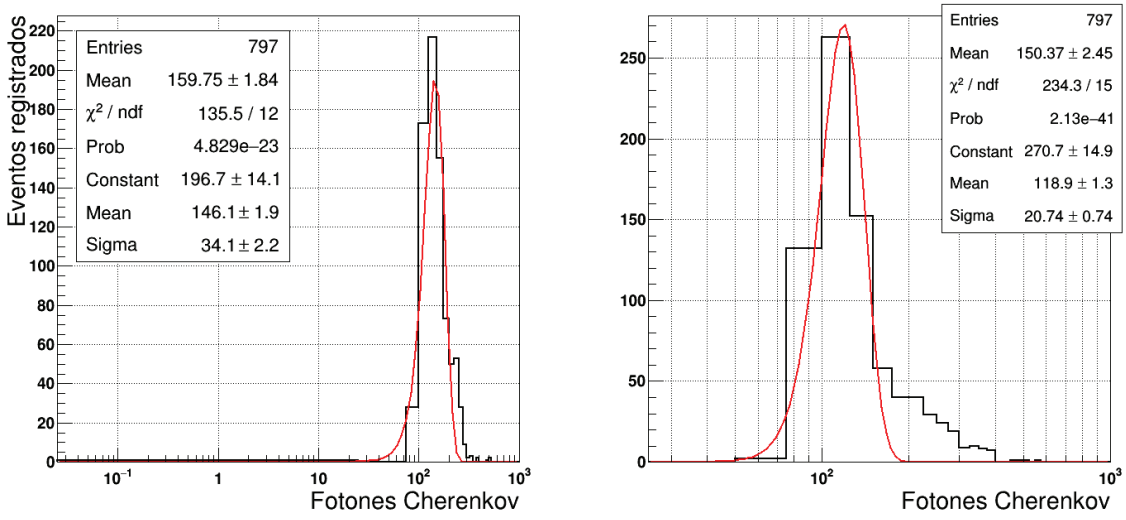


Figura 5.13: Histograma de fotones Cherenkov detectados para PMTs ubicados entre 176.5 cm y 282.4 cm respecto al centro del detector: (a) PMTs ubicados 176.5 cm (b) PMTs ubicados 211.8 cm (c) PMTs ubicados 247.1 cm, y (d) PMTs ubicados 282.4 cm. La línea negra gruesa es el histograma de los fotones Cherenkov dados por la simulación, y la línea roja delgada es el ajuste gaussiano.

Se realizó la integral de los histogramas de tiempo de las Figuras 5.16, 5.17, 5.18. La integral temporal para gammas, muones y electrones se presenta en la figura 5.19. Los valores de rise time[1] para el muón, gamma y electrón son de $9 \pm 0{,}1$ [ns], $7{,}5 \pm 0{,}1$ [ns] y $7{,}3 \pm 0{,}1$ [ns], respectivamente.

[1] El rise time es la diferencia de tiempo que una intensidad de señal pasa del 10% al 90% de la intensidad máxima de la señal.



(a) (b)

Figura 5.14: Histograma de fotones Cherenkov detectados para PMTs ubicados entre 317.7 cm y 353 cm respecto al centro del detector: (a) PMTs ubicados 317.7 cm, y (b) PMTs ubicados 353 cm. La línea negra gruesa es el histograma de los fotones Cherenkov dados por la simulación, y la línea roja delgada es el ajuste gaussiano.

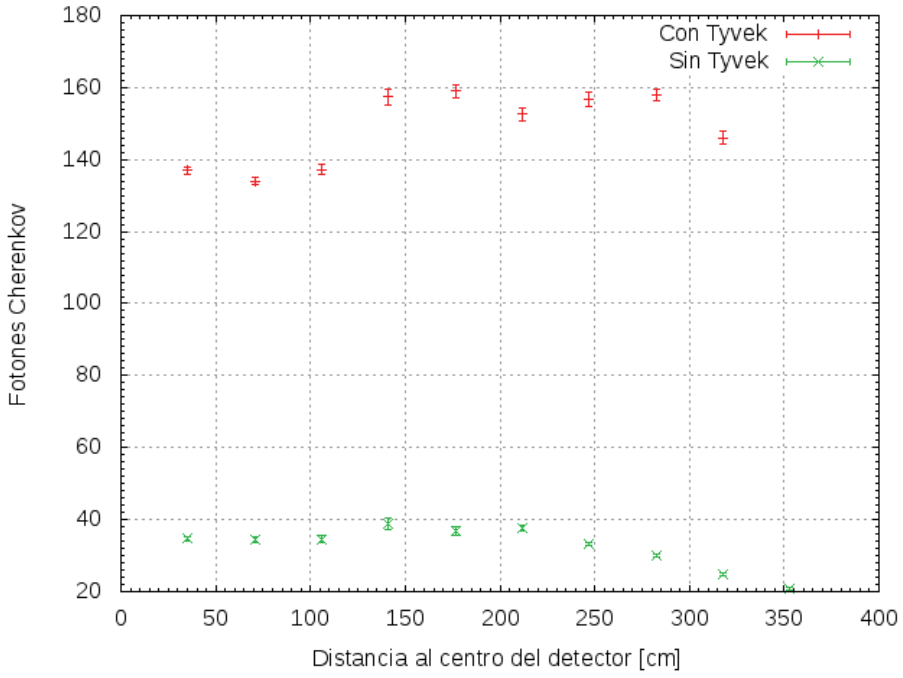


Figura 5.15: Número máximo de fotones Cherenkov detectados, en función de la distancia de los PMTs al centro del detector. En color rojo y verde, se muestran el máximo numero de fotones Cherenkov detectados con y sin Tyvek, respectivamente.

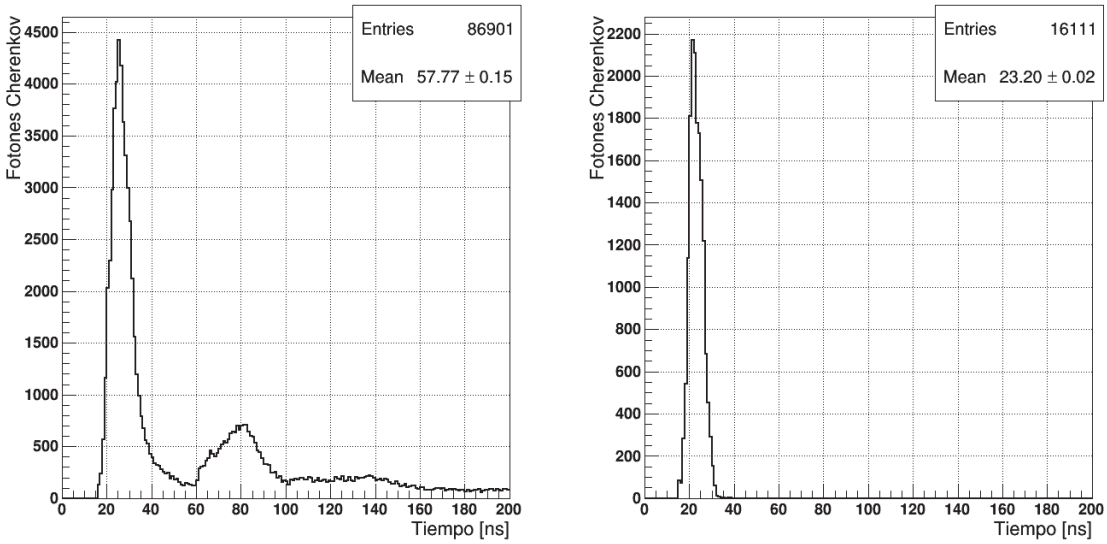


Figura 5.16: Distribución temporal de fotones Cherenkov (detectados) generados por 10000 gammas verticales de 70 MeV, que inciden en el detector central. La gráfica de la izquierda corresponde a detecciones con Tyvek y la gráfica de la derecha corresponde a detecciones sin Tyvek.

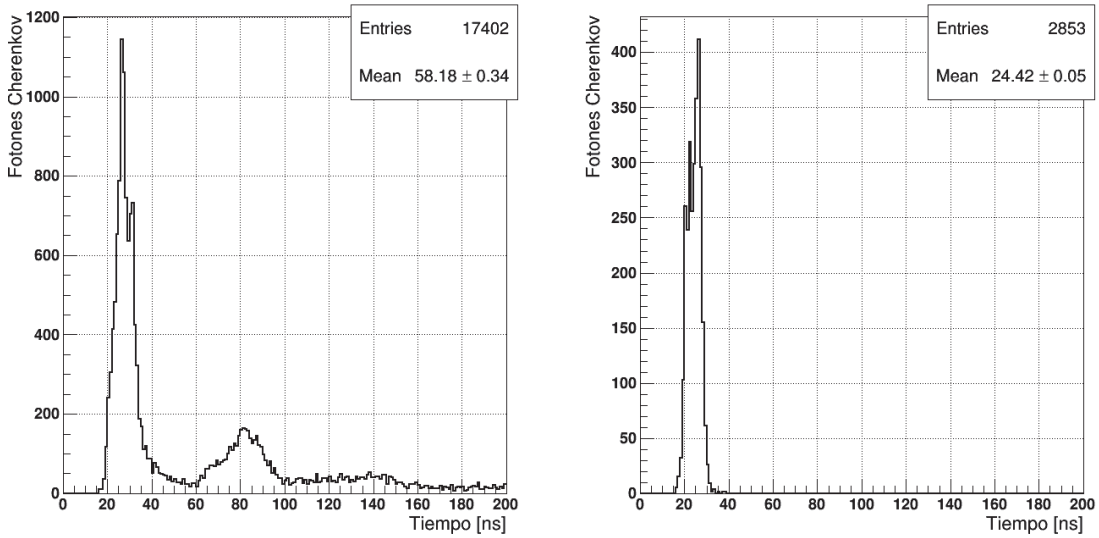


Figura 5.17: Distribución temporal de fotones Cherenkov (detectados) generados por 1000 electrones verticales de 140 MeV, que inciden en el detector central. La gráfica de la izquierda corresponde a detecciones con Tyvek y la gráfica de la derecha corresponde a detecciones sin Tyvek.

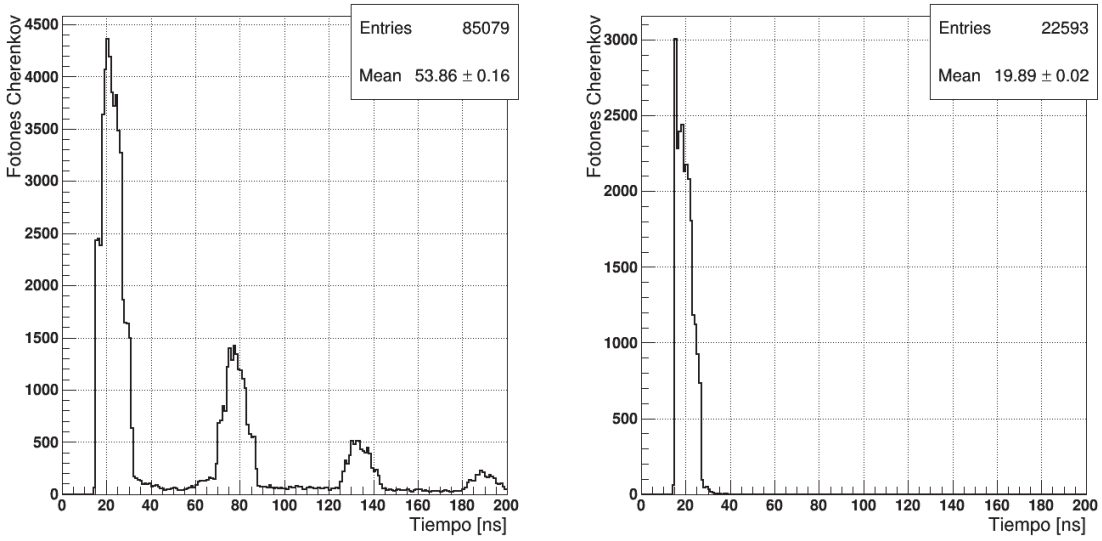


Figura 5.18: Distribución temporal de fotones Cherenkov (detectados) generados por 500 muones verticales de 8 GeV, que inciden en el detector central. La gráfica de la izquierda corresponde a detecciones con Tyvek y la gráfica de la derecha corresponde a detecciones sin Tyvek.

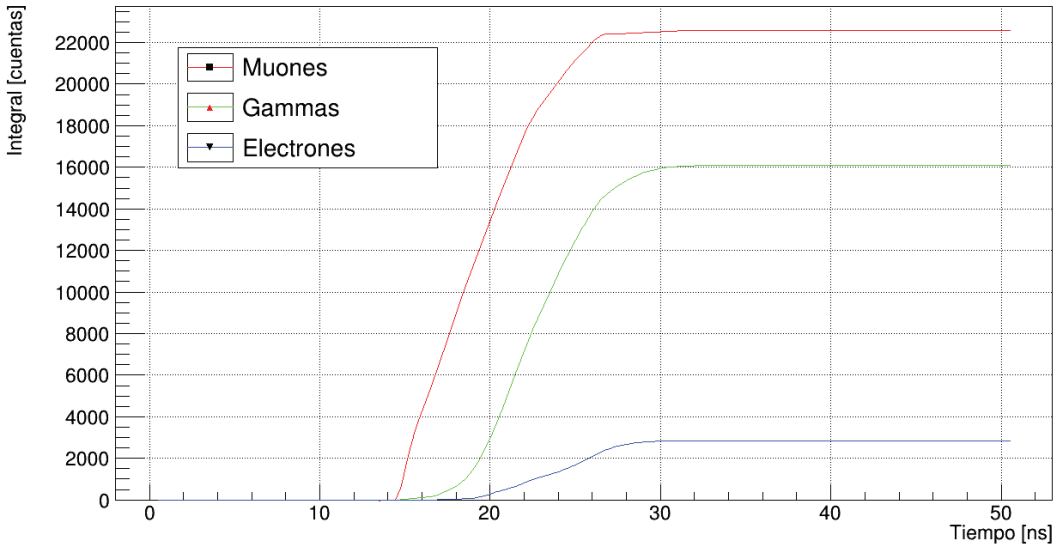


Figura 5.19: Integral temporal de los fotones Cherenkov generados por muones, electrones y gammas provenientes de un protón primario de 10 TeV que incide verticalmente en la atmósfera. Las curvas corresponden a la suma de la distribución temporal de fotones Cherenkov de las Figuras 5.16, 5.17, 5.18.

5.3. Área efectiva

Para determinar el área efectiva se simularon EAS producidos por un γ que incide verticalmente en la atmósfera con energías de 200, 500, 800 y 1100 GeV. Para cada energía del gamma en la simulación de los EAS se arrojaron 1000000 de chubascos. Las partículas secundarias consideradas para la producción de radiación Cherenkov en los WCD fueron: gammas, electrones, positrones, muones y antimuones. La conexión de los EAS simulados en CORSIKA que corresponde a datos en los archivos ya.dat que contiene información de las partículas secundarias de cada chubasco, con GEANT4 se realizó mediante la siguiente sección del archivo OpNovicePrimaryGenerator.cc:

```
void OpNovicePrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
    G4double fTEnergy;
    G4int nl=0;
    G4int nl2=0;
    ifstream inc("ya.dat",ios::in);
    ofstream dat("dato.dat",ios::app);
    if ( !inc ){
    G4cout<<"Error al abrir archivo de datos de CORSIKA"<<G4endl;
    }

    else {
    G4cout<<"Archivo de datos de CORSIKA abierto exitosamente"<<G4endl;
    G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable()
       ;
    G4String particleName;

    while (inc>>fPartId>>fUx>>fUy>>fUz>>fX>>fY>>fTimeDelay>>fTEnergy)
    {
    if ((nl2==0) && (fPartId == 999233900)){
    fPrimEnergy = fUx;
    fangle = fUy;
    fPhi = fUz;
    fxCore = fX;
    fyCore = fY;
    dat<<"999233900"<<" "<<fPrimEnergy<<" "<<fangle<<" "<<fPhi<<" "
                <<fxCore<<" "<<fyCore<<" 0 0 0 0"<<endl;}
    nl2++;
    nl++;
    if ( fPartId == 1)
            {
              particleName = "gamma";
            }
```

```
33      else if ( fPartId == 2 )
34              {                       //positron
35                particleName = "e+";
36              }
37      else if ( fPartId == 3 )
38              {                       //electron
39                particleName = "e-";
40              }
41      else if ( fPartId == 5 )
42              {                       //antimuon
43                particleName = "mu+";
44              }
45      else if ( fPartId == 6 )
46              {                       //muon
47                particleName = "mu-";
48              }
49      else  //no simula otras particulas
50            continue;
51
52      fZ=450; //posicion inicial de la particula en z
53      G4double ttime;
54      ttime = fTimeDelay;
55      G4ThreeVector direction ( fUx*GeV, fUy*GeV, -fUz*GeV );
56      G4ThreeVector position ( fX*cm, fY*cm, fZ*cm );
57      fParticleGun->SetParticleDefinition ( particleTable
58                          ->FindParticle ( particleName ) );
59      fParticleGun->SetParticleMomentumDirection ( direction );
60      fParticleGun->SetParticlePosition ( position );
61      fParticleGun->SetParticleEnergy ( fTEnergy*GeV );
62      fParticleGun->SetParticleTime ( ttime*ns );
63      fParticleGun->GeneratePrimaryVertex(anEvent);
64      }
65      G4cout<<"Total de eventos "<<nl2<<G4endl;
66      G4cout<<"Eventos dentro del Hall "<<nl<<G4endl;
67    }
68  }
```

El archivo de salida *dato.dat* contiene los fotones Cherenkov registrados por los PMTs en la simulación y su estructura es de 8 columnas: área, energía del primario, ángulo azimutal, ángulo cenital, código de PMT y distancia respecto al centro del PMT a la que impacta el fotón. Para la determinación del área efectiva se utilizó la Eq. (2.8), donde $A_{thrown} = 99923{,}39\ [m^2]$ donde se arrojaron los chubascos en la simulación. El procesamiento de los datos se realizó en root. La Figura 5.20 muestra los puntos simulados de las detecciones de los WCD de LAGO-México para un gamma que incide verticalmente en la atmósfera, en el rango de energía [200,1100] GeV, la cual fue ajustada con un curva gaussiana. Los valores

de los parámetros de ajuste se encuentran en el Cuadro 5.4.

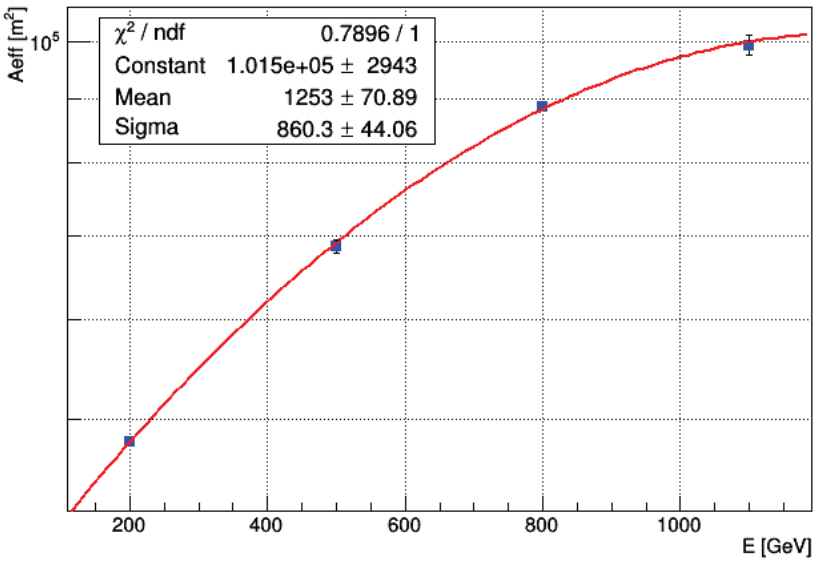


Figura 5.20: Área efectiva (A_{eff}) vs Energía (E) del primario de los detectores LAGO-México para un gamma que incide verticalmente en la atmósfera.

Parámetro	Valor	Error
Constant	1.01532e+05	2.94254e+03
Mean	1.25314e+03	7.08893e+01
Sigma	8.60276e+02	4.40631e+01

Cuadro 5.4: Parámetros del ajuste gaussiano para los datos del área efectiva de un gamma primario que incide verticalmente en la atmósfera.

Capítulo 6

Conclusiones

Se determinó que la posición óptima de los PMTs en el tanque central del arreglo está entre 141,2 cm y 176,5 cm en el caso que lleve Tyvek, y 141,2 cm en el caso sin Tyvek. Se establece que el tanque central debe ir sin Tyvek, puesto que se encuentra una diferencia apreciable en los rise time en la curva de la integral temporal de los fotones Cherenkov generados por cada tipo de partícula. Se simuló EAS generados por un protón de 10 TeV y por un gamma de 10 TeV; y se encontró la proporción de partículas secundarias: gammas, muones, antimuones, electrones y positrones que llegan al nivel de los detectores. Se determinó que existe diferencia en la componente muónica entre una cascada hadrónica y electromagnética, en 20 a 1, para 10 TeV. Esto permitirá diferenciar entre una cascada generada por un protón o un gamma. Es importante mencionar que el método utilizado es más eficiente en cuestión de tiempo, a comparación de otros métodos utilizados. Se determinó la respuesta de los detectores Cherenkov de agua de LAGO-México para un gamma con ángulo cenital 0° en el rango de [200, 1100] GeV. La curva presentada en la Figura 5.20 sirve como función de respuesta del detector, y en caso de detectar GRBs que inciden verticalmente debe ser utilizada para hallar el flujo de energía y partículas de dicho evento. El procedimiento utilizado para determinar el área efectiva toma tiempo, y depende del número de EAS simulados. También depende del número de nodos que contenga el cluster donde se corren los programas. En este caso, se determinó unicamente para gammas (rayos gamma) que inciden verticalmente sobre la atmósfera, para reducir el tiempo de simulación dado que este trabajo está orientado a validar la construcción del experimento.

Para futuros trabajos, se recomienda utilizar un rango de ángulo cenital más extenso (de $\theta = 0°$ a $\theta = 60°$) para la partícula primaria. Esto se puede hacer modificando el script de parámetros de entrada de CORSIKA. Sin embargo, se debe mencionar que el número de lluvias a simular debe ser mucho mayor que 1000000 (valor que se utilizó en nuestra simulación) ya que ese número de lluvias se repartirá en bines de ángulos cenitales, por lo que en el procesamiento de datos los errores estadísticos pueden ser grandes. El rango de energía de la partícula primaria también puede ser modificado.

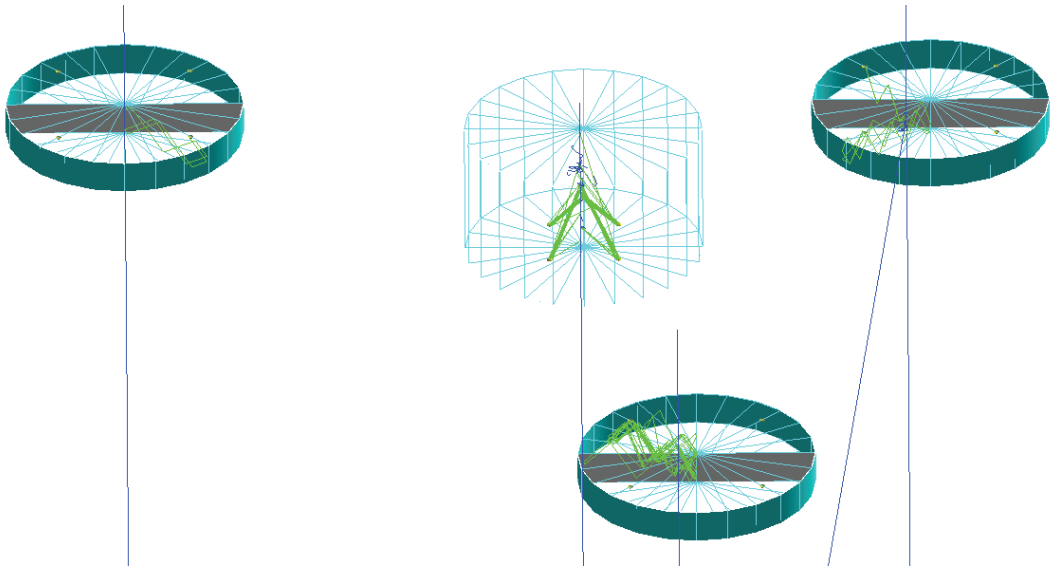


Figura 6.1: Diseño final en 3D de los WCDs de LAGO-México con Geant4. De izquierda a derecha en la imagen, el primero, tercero y cuarto cilindro corresponde a los WCDs laterales, mientras el segundo cilindro es el detector central. En color amarillo se representan los PMTs, ubicados en la parte superior en los WCD laterales y en la parte inferior en el WCD central. En línea azul se observa el paso de muones y en línea verde los fotones Cherenkov generados por los muones.

Los códigos en C++ y los scripts utilizados para hallar el área efectiva están guardados en un repositorio y servirá como parte de la colaboración LAGO. En la Figura 6.1 se observa el diseño final de los tanques simulados con geant4. Específicamente, para el caso que en Ecuador se quiera instalar detectores, se recomienda que se haga un estudio de la proporción de muones que llegan a nivel del suelo, para diferentes valores de energía del primario, y en base a ello estudiar la necesidad de plantar un observatorio. Además es importante mencionar que Ecuador, ya se tiene instalado un WCD, de manera aislada. Se podría implementar un arreglo, e incluir simulaciones para obtener la respuesta del detector en base a la electrónica, lo cual implicaría simular además el pico de la señal en los PMTs.

Referencias

[1] The pierre auger collaboration. *Astropart. Phys.*, 35:266, 2011.

[2] A. A. Abdo, B. T. Allen, D. Berley, E. Blaufuss, S. Casanova, B. L. Dingus, R. W. Ellsworth, M. M. Gonzalez, J. A. Goodman, and E. Hays. Milagro constraints on very high energy emission from short-duration gamma-ray bursts. *The Astrophysical Journal*, 666:361–367, 2007.

[3] C. Barat, G. Chambon, K. Hurley, M. Niel, G. Vedrenne, I. V. Estulin, A. V. Kuznetsov, and V. M. Zenchenko. A review of recent results from the franco-soviet signe gamma-burst experiments. *Astrophysics and Space Science*, 75:83–91, 1981.

[4] Xavier Bertou and Denis Allard, editors. *Detection of GRB with Water Cherenkov Detectors*. RICH2004 Proceedings Nucl. Instr. and Methods., 2005.

[5] D. Allard et al., editor. *The Large Aperture GRB Observatory*. PROCEEDINGS OF THE 31st ICRC, 2009.

[6] D. Allard et al., editor. *Operating Water Cherenkov Detectors in high altitude sites for the Large Aperture GRB Observatory*. PROCEEDINGS OF THE 31st ICRC, Junio 2009.

[7] S. Agostinelli et al. Geant4 simulation toolkit. *Nuclear Instruments and Methods in Physics Research*, 506:240–300, 2003.

[8] I. Valiño for The Pierre Auger Collaboration. Measurements of the muon content of air showers at the pierre auger observatory. *Journal of Physics: Conference Series*, 632, 2015.

[9] Thomas K. Gaisser. *Cosmic Rays and Particle Physics*, chapter Cosmic rays. Press Syndicate of the University of Cambridge, 1 edition, 1990.

[10] Thomas K. Gaisser. *Cosmic Rays and Particle Physics*, chapter Cosmic rays, pages 1–3. Press Syndicate of the University of Cambridge, 1 edition, 1990.

[11] Thomas K. Gaisser. *Cosmic Rays and Particle Physics*, chapter Simulation Techniques:Acceptanceof of an air shower array, pages 249–260. Press Syndicate of the University of Cambridge, 1 edition, 1990.

[12] Peter K.F. Grieder. *Extensive Air Showers: High Energy Phenomena and Astrophysical Aspects. A Tutorial, Reference Manual and Data Book.*, chapter Shower Detection Methods and Basic Event Reconstruction, pages 33–58. Springer, 1 edition, 2010.

[13] Peter K.F. Grieder. *Extensive Air Showers: High Energy Phenomena and Astrophysical Aspects. A Tutorial, Reference Manual and Data Book.*, chapter Gamma Radiation. Springer, 1 edition, 2010.

[14] Peter K.F. Grieder. *Extensive Air Showers: High Energy Phenomena and Astrophysical Aspects. A Tutorial, Reference Manual and Data Book.*, chapter Hadron Initiated Air Showers, Gamma Ray and Electron Initiated Air Showers, pages 3–27. Springer, 1 edition, 2010.

[15] Peter K.F. Grieder. *Extensive Air Showers: High Energy Phenomena and Astrophysical Aspects. A Tutorial, Reference Manual and Data Book.*, chapter Atmospheric Cherenkov Radiation, pages 835–840. Springer, 1 edition, 2010.

[16] HAWC. *HAWC effective area update*, 2011. HAWC Technical Note TN008.

[17] D. Heck and T. Pierog. Extensive air shower simulation with corsika: A user's guide. 2006.

[18] Klebesadel, Strong, and Olson. Observations of gamma-ray bursts of cosmic origin. *Astrophysical Journal*, 182:L85–L88, 1973.

[19] LAGO Collaboration, http://lagoproject.org. *The Latin American Giant Observatory*.

[20] Antoine Letessier-Selvon and Todor Stanev. Ultrahigh energy cosmic rays. *Reviews of Modern Physics*, 83:907–942, 2011.

[21] Katharina Lodders. Solar system abundances and condensation temperatures of the elements. *The Astrophysical Journal*, 591:1220–1247, 2003.

[22] W. A. Miziolek, V. Palleschi, and I. Schechter. *Handbook of Chemestry and Physics*. R.C. Weast ed, 67 edition, 2003-2004.

[23] Patrick Petitjean, F. Y. Wang, X. F. Wu, and J. J. Wei. Grbs and fundamental physics. *Space Science Reviews*, 202:195–234, 2016.

[24] Gilbert Vedrenne and Jean-Luc Atteia. *Gamma-Ray Bursts: The Brightest Explosions in the Universe*. Praxis Publishing Ltd, 1 edition, 2009.

[25] Silvia Vernetto. Detection of gamma-ray bursts in the 1 gev–1 tev energy range by ground-based experiments. *Astroparticle Physics*, 13:75–86, 2000.

Trabajos asociados

Nota técnica asociada al trabajo desarrollado:

- I. Torres, A. Galindo, **A. Delgado** for the LAGO Collaboration. **"Manual para obtener el área efectiva del experimento LAGO en Sierra Negra"**. LAGO TN-2016-001. Octubre 2016

Apéndice A

Codigos de Geant4

A continuacion se presentan los codigos editados para la simulacion[1]. Se presenta los archivos.hh y los codigos.cc:

A.1. Archivo LXeEMPhysics.hh

```
#ifndef LXeEMPhysics_h
#define LXeEMPhysics_h 1
#include "globals.hh"
#include "G4ios.hh"
#include "G4VPhysicsConstructor.hh"
#include "G4PhotoElectricEffect.hh"
#include "G4ComptonScattering.hh"
#include "G4GammaConversion.hh"
#include "G4eMultipleScattering.hh"
#include "G4eIonisation.hh"
#include "G4eBremsstrahlung.hh"
#include "G4eplusAnnihilation.hh"
class LXeEMPhysics : public G4VPhysicsConstructor
{
  public:
    LXeEMPhysics(const G4String& name ="EM");
    virtual ~LXeEMPhysics();
  public:
    virtual void ConstructParticle();
    virtual void ConstructProcess();
};
#endif
```

[1]Los códigos también se pueden encontrar en el repositorio GitHub en la dirección web: https://github.com/andresgd17/LAGO-Mexico/tree/master/LAGO

A.2. Archivo LXeEMPhysics.cc

```
#include "LXeEMPhysics.hh"
#include "globals.hh"
#include "G4ios.hh"
#include <iomanip>
LXeEMPhysics::LXeEMPhysics(const G4String& name) : G4VPhysicsConstructor
    (name){}
LXeEMPhysics::~LXeEMPhysics() {}
#include "G4ParticleDefinition.hh"
#include "G4ParticleTable.hh"
#include "G4Gamma.hh"
#include "G4Electron.hh"
#include "G4Positron.hh"
#include "G4NeutrinoE.hh"
#include "G4AntiNeutrinoE.hh"
void LXeEMPhysics::ConstructParticle()
{
  // gamma
  G4Gamma::GammaDefinition();
  // electron
  G4Electron::ElectronDefinition();
  G4Positron::PositronDefinition();
  G4NeutrinoE::NeutrinoEDefinition();
  G4AntiNeutrinoE::AntiNeutrinoEDefinition();
}
#include "G4ProcessManager.hh"
void LXeEMPhysics::ConstructProcess()
{
  G4PhotoElectricEffect* fPhotoEffect = new G4PhotoElectricEffect();
  G4ComptonScattering* fComptonEffect = new G4ComptonScattering();
  G4GammaConversion* fPairProduction = new G4GammaConversion();
  // Electron physics
  G4eMultipleScattering* fElectronMultipleScattering = new
      G4eMultipleScattering();
  G4eIonisation* fElectronIonisation = new G4eIonisation();
  G4eBremsstrahlung* fElectronBremsStrahlung = new G4eBremsstrahlung();
    //Positron physics
  G4eMultipleScattering* fPositronMultipleScattering = new
      G4eMultipleScattering();
  G4eIonisation* fPositronIonisation = new G4eIonisation();
  G4eBremsstrahlung* fPositronBremsStrahlung = new G4eBremsstrahlung();
  G4eplusAnnihilation* fAnnihilation = new G4eplusAnnihilation();
  G4ProcessManager* pManager = 0;
  // Gamma Physics
```

```
41    pManager = G4Gamma::Gamma()->GetProcessManager();
42    pManager->AddDiscreteProcess(fPhotoEffect);
43    pManager->AddDiscreteProcess(fComptonEffect);
44    pManager->AddDiscreteProcess(fPairProduction);
45    // Electron Physics
46    pManager = G4Electron::Electron()->GetProcessManager();
47    pManager->AddProcess(fElectronMultipleScattering, -1, 1, 1);
48    pManager->AddProcess(fElectronIonisation,         -1, 2, 2);
49    pManager->AddProcess(fElectronBremsStrahlung,     -1, 3, 3);
50    //Positron Physics
51    pManager = G4Positron::Positron()->GetProcessManager();
52    pManager->AddProcess(fPositronMultipleScattering, -1, 1, 1);
53    pManager->AddProcess(fPositronIonisation,         -1, 2, 2);
54    pManager->AddProcess(fPositronBremsStrahlung,     -1, 3, 3);
55    pManager->AddProcess(fAnnihilation,                0,-1, 4);
56  }
```

A.3. Archivo LXeEventAction.hh

```
1   #ifndef LXeEventAction_h
2   #define LXeEventAction_h 1
3   #include "LXeEventMessenger.hh"
4   #include "G4UserEventAction.hh"
5   #include "globals.hh"
6   #include "G4ThreeVector.hh"
7   class G4Event;
8   class LXeEventAction : public G4UserEventAction
9   {
10    public:
11      LXeEventAction();
12      virtual ~LXeEventAction();
13    public:
14      virtual void BeginOfEventAction(const G4Event*);
15      virtual void EndOfEventAction(const G4Event*);
16      void SetSaveThreshold(G4int);
17      void SetEventVerbose(G4int v){fVerbose=v;}
18      void SetPMTThreshold(G4int t){fPMTThreshold=t;}
19      void SetForceDrawPhotons(G4bool b){fForcedrawphotons=b;}
20      void SetForceDrawNoPhotons(G4bool b){fForcenophotons=b;}
21    private:
22      LXeEventMessenger* fEventMessenger;
23      G4int              fSaveThreshold;
24      G4int              fScintCollID;
25      G4int              fPMTCollID;
```

```
26      G4int               fVerbose;
27      G4int               fPMTThreshold;
28      G4bool fForcedrawphotons;
29      G4bool fForcenophotons;
30  };
31  #endif
```

A.4. Archivo LXeEventAction.cc

```
1   #include "LXeEventAction.hh"
2   #include "LXeScintHit.hh"
3   #include "LXePMTHit.hh"
4   #include "LXeUserEventInformation.hh"
5   #include "LXeTrajectory.hh"
6   #include "G4EventManager.hh"
7   #include "G4SDManager.hh"
8   #include "G4RunManager.hh"
9   #include "G4Event.hh"
10  #include "G4EventManager.hh"
11  #include "G4TrajectoryContainer.hh"
12  #include "G4Trajectory.hh"
13  #include "G4VVisManager.hh"
14  #include "G4ios.hh"
15  #include "G4UImanager.hh"
16  #include "G4SystemOfUnits.hh"
17  #include "globals.hh"
18  #include <iostream>
19  #include <fstream>
20  #include <iomanip>
21  #include <stdlib.h>
22  #include <string>
23  using namespace std;
24  LXeEventAction::LXeEventAction()
25    : fSaveThreshold(0),fScintCollID(-1),fPMTCollID(-1),fVerbose(0),
        fPMTThreshold(1),fForcedrawphotons(false),fForcenophotons(false)
26  {
27    fEventMessenger = new LXeEventMessenger(this);
28  }
29  LXeEventAction::~LXeEventAction(){}
30  void LXeEventAction::BeginOfEventAction(const G4Event* anEvent){
31    G4cout<<"GetEventID: "<<anEvent->GetEventID()<<G4endl;
32    G4EventManager::
33    GetEventManager()->SetUserInformation(new LXeUserEventInformation);
34    G4SDManager* SDman = G4SDManager::GetSDMpointer();
```

```
35    if(fPMTCollID<0)
36      fPMTCollID=SDman->GetCollectionID("pmtHitCollection");
37  }
38  void LXeEventAction::EndOfEventAction(const G4Event* anEvent){
39    LXeUserEventInformation* eventInformation =(LXeUserEventInformation*)
          anEvent->GetUserInformation();
40    G4TrajectoryContainer* trajectoryContainer=anEvent->
          GetTrajectoryContainer();
41    G4int n_trajectories = 0;
42    if (trajectoryContainer) n_trajectories = trajectoryContainer->entries
          ();
43    if (G4VVisManager::GetConcreteInstance()){
44      for (G4int i=0; i<n_trajectories; i++){
45        LXeTrajectory* trj = (LXeTrajectory*)
46          ((*(anEvent->GetTrajectoryContainer()))[i]);
47        if(trj->GetParticleName()=="opticalphoton"){
48          trj->SetForceDrawTrajectory(fForcedrawphotons);
49          trj->SetForceNoDrawTrajectory(fForcenophotons);
50        }
51        trj->DrawTrajectory();
52      }
53    }
54    LXeScintHitsCollection* scintHC = 0;
55    LXePMTHitsCollection* pmtHC = 0;
56    G4HCofThisEvent* hitsCE = anEvent->GetHCofThisEvent();
57    if(hitsCE){
58      if(fScintCollID>=0)scintHC = (LXeScintHitsCollection*)(hitsCE->GetHC
            (fScintCollID));
59      if(fPMTCollID>=0)pmtHC = (LXePMTHitsCollection*)(hitsCE->GetHC(
            fPMTCollID));
60    }
61    if(scintHC){
62      int n_hit = scintHC->entries();
63      G4ThreeVector  eWeightPos(0.);
64      G4double edep;
65      G4double edepMax=0;
66      for(int i=0;i<n_hit;i++){ //gather info on hits in scintillator
67        edep=(*scintHC)[i]->GetEdep();
68        eventInformation->IncEDep(edep); //sum up the edep
69        eWeightPos += (*scintHC)[i]->GetPos()*edep;//calculate energy
              weighted pos
70        if(edep>edepMax){
71          edepMax=edep;//store max energy deposit
72          G4ThreeVector posMax=(*scintHC)[i]->GetPos();
73          eventInformation->SetPosMax(posMax,edep);
74        }
```

```
75      }
76      if(eventInformation->GetEDep()==0.){
77        if(fVerbose>0)G4cout<<"No hits in the scintillator this event."<<
              G4endl;
78      }
79      else{
80        //Finish calculation of energy weighted position
81        eWeightPos/=eventInformation->GetEDep();
82        eventInformation->SetEWeightPos(eWeightPos);
83        if(fVerbose>0){
84          G4cout << "\tEnergy weighted position of hits in LXe : "
85                 << eWeightPos/mm << G4endl;
86        }
87      }
88      if(fVerbose>0){
89      G4cout << "\tTotal energy deposition in scintillator : "
90             << eventInformation->GetEDep() / keV << " (keV)" << G4endl;
91      }
92    }
93    if(pmtHC){
94      G4ThreeVector reconPos(0.,0.,0.);
95      G4int pmts=pmtHC->entries();
96      //Gather info from all PMTs
97      for(G4int i=0;i<pmts;i++){
98        eventInformation->IncHitCount((*pmtHC)[i]->GetPhotonCount());
99        reconPos+=(*pmtHC)[i]->GetPMTPos()*(*pmtHC)[i]->GetPhotonCount();
100       if((*pmtHC)[i]->GetPhotonCount()>=fPMTThreshold){
101         eventInformation->IncPMTSAboveThreshold();
102       }
103       else{//wasnt above the threshold, turn it back off
104         (*pmtHC)[i]->SetDrawit(false);
105       }
106     }
107     if(eventInformation->GetHitCount()>0){//dont bother unless there
            were hits
108       reconPos/=eventInformation->GetHitCount();
109       if(fVerbose>0){
110         G4cout << "\tReconstructed position of hits in LXe : "
111                << reconPos/mm << G4endl;
112       }
113       eventInformation->SetReconPos(reconPos);
114     }
115     pmtHC->DrawAllHits();
116   }
117   // ofstream dat("dato.dat",ios::app);
118   // dat<<eventInformation->GetHitCount()<<G4endl;
```

```
119     if(fVerbose>0){
120    // G4cout<<"NHit"<<eventInformation->GetHitCount()<<G4endl;
121       //End of event output. later to be controlled by a verbose level
122       G4cout << "\tNumber of photons that hit PMTs in this event : "
123              << eventInformation->GetHitCount() << G4endl;
124       G4cout << "\tNumber of PMTs above threshold("<<fPMTThreshold<<") : "
125              << eventInformation->GetPMTSAboveThreshold() << G4endl;
126       G4cout << "\tNumber of photons produced by scintillation in this
             event : "
127              << eventInformation->GetPhotonCount_Scint() << G4endl;
128       G4cout << "\tNumber of photons produced by cerenkov in this event :
             "
129              << eventInformation->GetPhotonCount_Ceren() << G4endl;
130       G4cout << "\tNumber of photons absorbed (OpAbsorption) in this event
              : "
131              << eventInformation->GetAbsorptionCount() << G4endl;
132       G4cout << "\tNumber of photons absorbed at boundaries (OpBoundary)
             in "
133              << "this event : " << eventInformation->
                 GetBoundaryAbsorptionCount()
134              << G4endl;
135       G4cout << "Unacounted for photons in this event : "
136              << (eventInformation->GetPhotonCount_Scint() +
137                  eventInformation->GetPhotonCount_Ceren() -
138                  eventInformation->GetAbsorptionCount() -
139                  eventInformation->GetHitCount() -
140                  eventInformation->GetBoundaryAbsorptionCount())
141              << G4endl;
142     }
143     if(fSaveThreshold&&eventInformation->GetPhotonCount() <=
          fSaveThreshold)
144       G4RunManager::GetRunManager()->rndmSaveThisEvent();
145   }
146   void LXeEventAction::SetSaveThreshold(G4int save){
147     fSaveThreshold=save;
148     G4RunManager::GetRunManager()->SetRandomNumberStore(true);
149     G4RunManager::GetRunManager()->SetRandomNumberStoreDir("random/");
150   }
```

A.5. Archivo LXeEventMessenger.hh

```
1   #ifndef LXeEventMessenger_h
2   #define LXeEventMessenger_h 1
3   #include "G4UImessenger.hh"
```

```
#include "globals.hh"
class LXeEventAction;
class G4UIcmdWithAnInteger;
class G4UIcmdWithABool;
class LXeEventMessenger: public G4UImessenger
{
  public:
    LXeEventMessenger(LXeEventAction*);
    virtual ~LXeEventMessenger();
    virtual void SetNewValue(G4UIcommand*, G4String);
  private:
    LXeEventAction*        fLXeEvent;
    G4UIcmdWithAnInteger*  fSaveThresholdCmd;
    G4UIcmdWithAnInteger*  fVerboseCmd;
    G4UIcmdWithAnInteger*  fPmtThresholdCmd;
    G4UIcmdWithABool*      fForceDrawPhotonsCmd;
    G4UIcmdWithABool*      fForceDrawNoPhotonsCmd;
};
#endif
```

A.6. Archivo LXeEventMessenger.cc

```
#include "LXeEventMessenger.hh"
#include "LXeEventAction.hh"
#include "G4UIcmdWithABool.hh"
#include "G4UIcmdWithAnInteger.hh"
LXeEventMessenger::LXeEventMessenger(LXeEventAction* event)
 : fLXeEvent(event)
{
  fSaveThresholdCmd = new G4UIcmdWithAnInteger("/LXe/saveThreshold",this
      );
  fSaveThresholdCmd->SetGuidance("Set the photon count threshold for
      saving the random number seed");
  fSaveThresholdCmd->SetParameterName("photons",true);
  fSaveThresholdCmd->SetDefaultValue(4500);
  fSaveThresholdCmd->AvailableForStates(G4State_PreInit,G4State_Idle);
  fVerboseCmd = new G4UIcmdWithAnInteger("/LXe/eventVerbose",this);
  fVerboseCmd->SetGuidance("Set the verbosity of event data.");
  fVerboseCmd->SetParameterName("verbose",true);
  fVerboseCmd->SetDefaultValue(0);
  fPmtThresholdCmd = new G4UIcmdWithAnInteger("/LXe/pmtThreshold",this);
  fPmtThresholdCmd->SetGuidance("Set the pmtThreshold (in # of photons)"
      );
```

```
19    fForceDrawPhotonsCmd=new G4UIcmdWithABool("/LXe/forceDrawPhotons",this
          );
20    fForceDrawPhotonsCmd->SetGuidance("Force drawing of photons.");
21    fForceDrawPhotonsCmd->SetGuidance("(Higher priority than /LXe/
          forceDrawNoPhotons)");
22    fForceDrawNoPhotonsCmd=new G4UIcmdWithABool("/LXe/forceDrawNoPhotons",
          this);
23    fForceDrawNoPhotonsCmd->SetGuidance("Force no drawing of photons.");
24    fForceDrawNoPhotonsCmd->SetGuidance("(Lower priority than /LXe/
          forceDrawPhotons)");
25  }
26  LXeEventMessenger::~LXeEventMessenger(){
27    delete fSaveThresholdCmd;
28    delete fVerboseCmd;
29    delete fPmtThresholdCmd;
30    delete fForceDrawPhotonsCmd;
31    delete fForceDrawNoPhotonsCmd;
32  }
33  void LXeEventMessenger::SetNewValue(G4UIcommand* command, G4String
       newValue){
34    if( command == fSaveThresholdCmd ){
35      fLXeEvent->SetSaveThreshold(fSaveThresholdCmd->GetNewIntValue(
            newValue));
36    }
37    else if( command == fVerboseCmd ){
38      fLXeEvent->SetEventVerbose(fVerboseCmd->GetNewIntValue(newValue));
39    }
40    else if( command == fPmtThresholdCmd ){
41      fLXeEvent->SetPMTThreshold(fPmtThresholdCmd->GetNewIntValue(newValue
            ));
42    }
43    else if(command == fForceDrawPhotonsCmd){
44      fLXeEvent->SetForceDrawPhotons(fForceDrawPhotonsCmd->GetNewBoolValue
            (newValue));
45    }
46    else if(command == fForceDrawNoPhotonsCmd){
47      fLXeEvent->SetForceDrawNoPhotons(fForceDrawNoPhotonsCmd->
            GetNewBoolValue(newValue));
48    }
49  }
```

A.7. Archivo LXeGeneralPhysics.hh

```
1  #ifndef LXeGeneralPhysics_h
```

```
#define LXeGeneralPhysics_h 1
#include "globals.hh"
#include "G4ios.hh"
#include "G4VPhysicsConstructor.hh"
class LXeGeneralPhysics : public G4VPhysicsConstructor
{
  public:
    LXeGeneralPhysics(const G4String& name = "general");
    virtual ~LXeGeneralPhysics();
    virtual void ConstructParticle();
    virtual void ConstructProcess();
};
#endif
```

A.8. Archivo LXeGeneralPhysics.cc

```
#include "LXeGeneralPhysics.hh"
#include "globals.hh"
#include "G4ios.hh"
#include <iomanip>
#include "G4Decay.hh"
LXeGeneralPhysics::LXeGeneralPhysics(const G4String& name)
                    :  G4VPhysicsConstructor(name) {}
LXeGeneralPhysics::~LXeGeneralPhysics() {
  //fDecayProcess = NULL;
}
#include "G4ParticleDefinition.hh"
#include "G4ProcessManager.hh"
#include "G4Geantino.hh"
#include "G4ChargedGeantino.hh"
#include "G4GenericIon.hh"
#include "G4Proton.hh"
void LXeGeneralPhysics::ConstructParticle()
{
  G4Geantino::GeantinoDefinition();
  G4ChargedGeantino::ChargedGeantinoDefinition();
  G4GenericIon::GenericIonDefinition();
}
void LXeGeneralPhysics::ConstructProcess()
{
  G4Decay* fDecayProcess = new G4Decay();
  aParticleIterator->reset();
  while( (*aParticleIterator)() ){
    G4ParticleDefinition* particle = aParticleIterator->value();
```

```
29      G4ProcessManager* pmanager = particle->GetProcessManager();
30      if (fDecayProcess->IsApplicable(*particle)) {
31        pmanager ->AddProcess(fDecayProcess);
32        pmanager ->SetProcessOrdering(fDecayProcess, idxPostStep);
33        pmanager ->SetProcessOrdering(fDecayProcess, idxAtRest);
34      }
35    }
36  }
```

A.9. Archivo LXeMuonPhysics.hh

```
1   #ifndef LXeMuonPhysics_h
2   #define LXeMuonPhysics_h 1
3   #include "globals.hh"
4   #include "G4ios.hh"
5   #include "G4VPhysicsConstructor.hh"
6   #include "G4MuMultipleScattering.hh"
7   #include "G4MuBremsstrahlung.hh"
8   #include "G4MuPairProduction.hh"
9   #include "G4MuIonisation.hh"
10  #include "G4hIonisation.hh"
11  #include "G4MuonMinusCapture.hh"
12  class LXeMuonPhysics : public G4VPhysicsConstructor
13  {
14    public:
15      LXeMuonPhysics(const G4String& name="muon");
16      virtual ~LXeMuonPhysics();
17      virtual void ConstructParticle();
18      virtual void ConstructProcess();
19  };
20  #endif
```

A.10. Archivo LXeMuonPhysics.cc

```
1   #include "LXeMuonPhysics.hh"
2   #include "globals.hh"
3   #include "G4ios.hh"
4   #include "G4PhysicalConstants.hh"
5   #include <iomanip>
6   LXeMuonPhysics::LXeMuonPhysics(const G4String& name)
7                     :  G4VPhysicsConstructor(name) {
```

```
8   }
9   LXeMuonPhysics::~LXeMuonPhysics() {}
10  #include "G4ParticleDefinition.hh"
11  #include "G4ParticleTable.hh"
12  #include "G4MuonPlus.hh"
13  #include "G4MuonMinus.hh"
14  #include "G4NeutrinoMu.hh"
15  #include "G4AntiNeutrinoMu.hh"
16  #include "G4Neutron.hh"
17  #include "G4Proton.hh"
18  #include "G4PionZero.hh"
19  #include "G4PionPlus.hh"
20  #include "G4PionMinus.hh"
21  void LXeMuonPhysics::ConstructParticle()
22  {
23    // Mu
24    G4MuonPlus::MuonPlusDefinition();
25    G4MuonMinus::MuonMinusDefinition();
26    G4NeutrinoMu::NeutrinoMuDefinition();
27    G4AntiNeutrinoMu::AntiNeutrinoMuDefinition();
28    //These are needed for the mu- capture
29      G4Neutron::Neutron();
30      G4Proton::Proton();
31      G4PionMinus::PionMinus();
32      G4PionZero::PionZero();
33      G4PionPlus::PionPlus();
34  }
35  #include "G4ProcessManager.hh"
36  void LXeMuonPhysics::ConstructProcess()
37  {
38    G4MuIonisation* fMuPlusIonisation = new G4MuIonisation();
39    G4MuMultipleScattering* fMuPlusMultipleScattering = new
        G4MuMultipleScattering();
40    G4MuBremsstrahlung* fMuPlusBremsstrahlung = new G4MuBremsstrahlung();
41    G4MuPairProduction* fMuPlusPairProduction = new G4MuPairProduction();
42    G4MuIonisation* fMuMinusIonisation = new G4MuIonisation();
43    G4MuMultipleScattering* fMuMinusMultipleScattering = new
        G4MuMultipleScattering();
44    G4MuBremsstrahlung* fMuMinusBremsstrahlung = new G4MuBremsstrahlung();
45    G4MuPairProduction* fMuMinusPairProduction = new G4MuPairProduction();
46    G4MuonMinusCapture* fMuMinusCaptureAtRest = new G4MuonMinusCapture();
47    G4ProcessManager * pManager = 0;
48    // Muon Plus Physics
49    pManager = G4MuonPlus::MuonPlus()->GetProcessManager();
50    pManager->AddProcess(fMuPlusMultipleScattering,-1,  1, 1);
51    pManager->AddProcess(fMuPlusIonisation,        -1,  2, 2);
```

```
52    pManager->AddProcess(fMuPlusBremsstrahlung,    -1,  3, 3);
53    pManager->AddProcess(fMuPlusPairProduction,    -1,  4, 4);
54    // Muon Minus Physics
55    pManager = G4MuonMinus::MuonMinus()->GetProcessManager();
56    pManager->AddProcess(fMuMinusMultipleScattering,-1,  1, 1);
57    pManager->AddProcess(fMuMinusIonisation,        -1,  2, 2);
58    pManager->AddProcess(fMuMinusBremsstrahlung,    -1,  3, 3);
59    pManager->AddProcess(fMuMinusPairProduction,    -1,  4, 4);
60    pManager->AddRestProcess(fMuMinusCaptureAtRest);
61  }
```

A.11. Archivo LXePhysicsList.hh

```
1   #ifndef LXePhysicsList_h
2   #define LXePhysicsList_h 1
3   #include "G4VModularPhysicsList.hh"
4   #include "globals.hh"
5   class LXePhysicsList: public G4VModularPhysicsList
6   {
7     public:
8       LXePhysicsList();
9       virtual ~LXePhysicsList();
10    public:
11      virtual void SetCuts();
12  };
13  #endif
```

A.12. Archivo LXePhysicsList.cc

```
1   #include "LXePhysicsList.hh"
2   #include "LXeGeneralPhysics.hh"
3   #include "LXeEMPhysics.hh"
4   #include "LXeMuonPhysics.hh"
5   #include "G4OpticalPhysics.hh"
6   #include "G4OpticalProcessIndex.hh"
7   #include "G4SystemOfUnits.hh"
8   LXePhysicsList::LXePhysicsList() : G4VModularPhysicsList()
9   {
10    defaultCutValue = 1.0*mm;
11    // General Physics
12    RegisterPhysics( new LXeGeneralPhysics("general") );
```

```
13    // EM Physics
14    RegisterPhysics( new LXeEMPhysics("standard EM"));
15    // Muon Physics
16    RegisterPhysics( new LXeMuonPhysics("muon"));
17    // Optical Physics
18    G4OpticalPhysics* opticalPhysics = new G4OpticalPhysics();
19    RegisterPhysics( opticalPhysics );
20    opticalPhysics->SetWLSTimeProfile("delta");
21    opticalPhysics->SetScintillationYieldFactor(0);
22    opticalPhysics->SetScintillationExcitationRatio(0);
23    opticalPhysics->SetMaxNumPhotonsPerStep(100);
24    opticalPhysics->SetMaxBetaChangePerStep(10.0);
25    opticalPhysics->SetTrackSecondariesFirst(kCerenkov,true);
26    opticalPhysics->SetTrackSecondariesFirst(kScintillation,false);
27  }
28  LXePhysicsList::~LXePhysicsList() {}
29  void LXePhysicsList::SetCuts(){
30    SetCutsWithDefault();
31  }
```

A.13. Archivo LXePMTHit.hh

```
1   #ifndef LXePMTHit_h
2   #define LXePMTHit_h 1
3   #include "G4VHit.hh"
4   #include "G4THitsCollection.hh"
5   #include "G4Allocator.hh"
6   #include "G4ThreeVector.hh"
7   #include "G4LogicalVolume.hh"
8   #include "G4Transform3D.hh"
9   #include "G4RotationMatrix.hh"
10  #include "G4VPhysicalVolume.hh"
11  #include "tls.hh"
12  class G4VTouchable;
13  class LXePMTHit : public G4VHit
14  {
15    public:
16      LXePMTHit();
17      virtual ~LXePMTHit();
18      LXePMTHit(const LXePMTHit &right);
19      const LXePMTHit& operator=(const LXePMTHit &right);
20      G4int operator==(const LXePMTHit &right) const;
21      inline void *operator new(size_t);
22      inline void operator delete(void *aHit);
```

```
23      virtual void Draw();
24      virtual void Print();
25      inline void SetDrawit(G4bool b){fDrawit=b;}
26      inline G4bool GetDrawit(){return fDrawit;}
27      inline void IncPhotonCount(){fPhotons++;}
28      inline G4int GetPhotonCount(){return fPhotons;}
29      inline void SetPMTNumber(G4int n) { fPmtNumber = n; }
30      inline G4int GetPMTNumber() { return fPmtNumber; }
31      inline void SetPMTPhysVol(G4VPhysicalVolume* physVol){this->fPhysVol
           =physVol;}
32      inline G4VPhysicalVolume* GetPMTPhysVol(){return fPhysVol;}
33      inline void SetPMTPos(G4double x,G4double y,G4double z){fPos=
           G4ThreeVector(x,y,z);}
34      inline G4ThreeVector GetPMTPos(){return fPos;}
35      inline void SetTime(G4double t){time=t;} ;
36      inline G4double GetTime() const {return time;} ;
37    private:
38      G4int fPmtNumber;
39      G4int fPhotons;
40      G4ThreeVector fPos;
41      G4VPhysicalVolume* fPhysVol;
42      G4bool fDrawit;
43      G4double time;
44  };
45  typedef G4THitsCollection<LXePMTHit> LXePMTHitsCollection;
46  extern G4ThreadLocal G4Allocator<LXePMTHit>* LXePMTHitAllocator;
47  inline void* LXePMTHit::operator new(size_t){
48    if(!LXePMTHitAllocator)
49        LXePMTHitAllocator = new G4Allocator<LXePMTHit>;
50    return (void *) LXePMTHitAllocator->MallocSingle();
51  }
52  inline void LXePMTHit::operator delete(void *aHit){
53    LXePMTHitAllocator->FreeSingle((LXePMTHit*) aHit);
54  }
55  #endif
```

A.14. Archivo LXePMTHit.cc

```
1  #include "LXePMTHit.hh"
2  #include "G4ios.hh"
3  #include "G4VVisManager.hh"
4  #include "G4Colour.hh"
5  #include "G4VisAttributes.hh"
6  #include "G4LogicalVolume.hh"
```

```
#include "G4VPhysicalVolume.hh"
G4ThreadLocal G4Allocator<LXePMTHit>* LXePMTHitAllocator=0;
LXePMTHit::LXePMTHit() : fPmtNumber(-1),fPhotons(0),fPhysVol(0),fDrawit(
    false) {}
LXePMTHit::~LXePMTHit() {}
LXePMTHit::LXePMTHit(const LXePMTHit &right) : G4VHit()
{
  fPmtNumber=right.fPmtNumber;
  fPhotons=right.fPhotons;
  fPhysVol=right.fPhysVol;
  fDrawit=right.fDrawit;
}
const LXePMTHit& LXePMTHit::operator=(const LXePMTHit &right){
  fPmtNumber = right.fPmtNumber;
  fPhotons=right.fPhotons;
  fPhysVol=right.fPhysVol;
  fDrawit=right.fDrawit;
  return *this;
}
G4int LXePMTHit::operator==(const LXePMTHit &right) const{
  return (fPmtNumber==right.fPmtNumber);
}
void LXePMTHit::Draw(){
  if(fDrawit&&fPhysVol){
    G4VVisManager* pVVisManager = G4VVisManager::GetConcreteInstance();
    if(pVVisManager){//Make sure that the VisManager exists
      G4VisAttributes attribs(G4Colour(1.,0.,0.));
      attribs.SetForceSolid(true);
      G4RotationMatrix rot;
      if(fPhysVol->GetRotation())//If a rotation is defined use it
        rot=*(fPhysVol->GetRotation());
      G4Transform3D trans(rot,fPhysVol->GetTranslation());//Create
          transform
      pVVisManager->Draw(*fPhysVol,attribs,trans);//Draw it
    }
  }
}
void LXePMTHit::Print() {}
```

A.15. Archivo LXePMTSD.hh

```
#ifndef LXePMTSD_h
#define LXePMTSD_h 1
#include "G4DataVector.hh"
```

```
#include "G4VSensitiveDetector.hh"
#include "LXePMTHit.hh"
#include <vector>
class G4Step;
class G4HCofThisEvent;
class LXePMTSD : public G4VSensitiveDetector
{
  public:
    LXePMTSD(G4String name);
    virtual ~LXePMTSD();
    void Initialize(G4HCofThisEvent* );
    G4bool ProcessHits(G4Step* aStep, G4TouchableHistory* );
    G4bool ProcessHits_constStep(const G4Step* ,G4TouchableHistory* );
    virtual void EndOfEvent(G4HCofThisEvent* );
    virtual void clear();
    void DrawAll();
    void PrintAll();
    inline void InitPMTs(G4int nPMTs){
      if(fPMTPositionsX)delete fPMTPositionsX;
      if(fPMTPositionsY)delete fPMTPositionsY;
      if(fPMTPositionsZ)delete fPMTPositionsZ;
      fPMTPositionsX=new G4DataVector(nPMTs);
      fPMTPositionsY=new G4DataVector(nPMTs);
      fPMTPositionsZ=new G4DataVector(nPMTs);
    }
    //Store a pmt position
    void SetPmtPositions(const std::vector<G4ThreeVector>& positions);
  private:
    LXePMTHitsCollection* fPMTHitCollection;
    G4DataVector* fPMTPositionsX;
    G4DataVector* fPMTPositionsY;
    G4DataVector* fPMTPositionsZ;
};
#endif
```

A.16. Archivo LXePMTSD.cc

```
#include "LXePMTSD.hh"
#include "LXePMTHit.hh"
#include "OpNoviceDetectorConstruction.hh"
#include "LXeUserTrackInformation.hh"
#include "OpNovicePrimaryGeneratorAction.hh"
#include "G4VPhysicalVolume.hh"
#include "G4LogicalVolume.hh"
```

```
8  #include "G4Track.hh"
9  #include "G4Step.hh"
10 #include "G4VTouchable.hh"
11 #include "G4TouchableHistory.hh"
12 #include "G4ios.hh"
13 #include "G4ParticleTypes.hh"
14 #include "G4ParticleDefinition.hh"
15 #include "G4ThreeVector.hh"
16 #include "G4SystemOfUnits.hh"
17 #include <iostream>
18 #include <fstream>
19 #include <iomanip>
20 #include <stdlib.h>
21 #include <string>
22 using namespace std;
23 LXePMTSD::LXePMTSD(G4String name)
24   : G4VSensitiveDetector(name),fPMTHitCollection(0),fPMTPositionsX(0)
25   ,fPMTPositionsY(0),fPMTPositionsZ(0)
26 {
27   collectionName.insert("pmtHitCollection");
28 }
29 LXePMTSD::~LXePMTSD() {}
30 void LXePMTSD::SetPmtPositions(const std::vector<G4ThreeVector>&
      positions)
31 {
32   for (G4int i=0; i<G4int(positions.size()); ++i) {
33     if(fPMTPositionsX)fPMTPositionsX->push_back(positions[i].x());
34     if(fPMTPositionsY)fPMTPositionsY->push_back(positions[i].y());
35     if(fPMTPositionsZ)fPMTPositionsZ->push_back(positions[i].z());
36   }
37 }
38 void LXePMTSD::Initialize(G4HCofThisEvent* hitsCE){
39   fPMTHitCollection = new LXePMTHitsCollection(SensitiveDetectorName,
        collectionName[0]);
40   static G4int hitCID = -1;
41   if(hitCID<0){
42     hitCID = GetCollectionID(0);
43   }
44   hitsCE->AddHitsCollection( hitCID, fPMTHitCollection );
45 }
46 G4bool LXePMTSD::ProcessHits(G4Step* ,G4TouchableHistory* ){
47   return false;
48 }
49 //Generates a hit and uses the postStepPoint's mother volume replica
      number
50 //PostStepPoint because the hit is generated manually when the photon is
```

```
51  //absorbed by the photocathode
52  G4bool LXePMTSD::ProcessHits_constStep(const G4Step* aStep,
53                                         G4TouchableHistory* ){
54    //need to know if this is an optical photon
55    if(aStep->GetTrack()->GetDefinition()
56       != G4OpticalPhoton::OpticalPhotonDefinition()) return false;
57    //User replica number 1 since photocathode is a daughter volume
58    //to the pmt which was replicated
59    G4double t1= aStep->GetPostStepPoint()->GetGlobalTime();
60    G4int pmtNumber=
61      aStep->GetPostStepPoint()->GetTouchable()->GetReplicaNumber();
62    G4VPhysicalVolume* physVol=
63      aStep->GetPostStepPoint()->GetTouchable()->GetVolume(1);
64    //How far is the PE from the PMT's center
65    G4int repNo =
66      aStep->GetPostStepPoint()->GetTouchable()->GetReplicaNumber();
67    G4int tankNo =
68      aStep->GetPostStepPoint()->GetTouchable()->GetVolume(1)->GetCopyNo()
          ;
69    double posX =
70      aStep->GetPostStepPoint()->GetPosition().x();
71   double posY =
72      aStep->GetPostStepPoint()->GetPosition().y();
73  //in cm
74  posX = posX/10.;
75  posY = posY/10.;
76  //move PE from tank 2 & 3
77   if(tankNo==2){posX = posX - 2474.;}
78   if(tankNo==3){posX = posX - 1756.;
79                 posY = posY + 2164.;}
80   if(tankNo==4){posX = posX -1410.;
81                 posY = posY + 722.;}
82  //get distance from PE's position to PMT's center
83  posX = abs(posX);
84  posY = abs(posY);
85  double radi;
86  if (tankNo==4)
87  {
88      cout<<posX<<" " <<posY<<endl;
89      radi = sqrt((posX-106.)*(posX-106.)+(posY-106.)*(posY-106.));
90  }
91  else {radi = sqrt((posX-203.)*(posX-203.)+(posY-203.)*(posY-203.));}
92    ifstream inc("ya.dat",ios::in);
93    ofstream dat("dato.dat",ios::app);
94           if (inc.good())
95          {  string sLine;
```

```
            getline(inc, sLine);
             dat<<sLine;}
  dat<<" "<<tankNo<<repNo<<" "<<radi<<G4endl;
  //Find the correct hit collection
  G4int n=fPMTHitCollection->entries();
  LXePMTHit* hit=NULL;
  //ofstream cosa("time.txt",ios::app);
  //G4double test = t1/ns;
  //cosa<<tankNo<<" "<<repNo<<" "<<test<<G4endl;
  for(G4int i=0;i<n;i++){
    if((*fPMTHitCollection)[i]->GetPMTNumber()==pmtNumber){
      hit=(*fPMTHitCollection)[i];
      break;
    }
  }
  if(hit==NULL){//this pmt wasnt previously hit in this event
    hit = new LXePMTHit(); //so create new hit
    hit->SetPMTNumber(pmtNumber);
    hit->SetPMTPhysVol(physVol);
    hit->SetTime(t1);
    fPMTHitCollection->insert(hit);
    hit->SetPMTPos((*fPMTPositionsX)[pmtNumber],(*fPMTPositionsY)[
        pmtNumber],
                   (*fPMTPositionsZ)[pmtNumber]);
  }
  hit->IncPhotonCount(); //increment hit for the selected pmt
  return true;
}
void LXePMTSD::EndOfEvent(G4HCofThisEvent* ) {}
void LXePMTSD::clear() {}
void LXePMTSD::DrawAll() {}
void LXePMTSD::PrintAll() {}
```

A.17. Archivo LXeScintHit.hh

```
#ifndef LXeScintHit_h
#define LXeScintHit_h 1
#include "G4VHit.hh"
#include "G4THitsCollection.hh"
#include "G4Allocator.hh"
#include "G4ThreeVector.hh"
#include "G4LogicalVolume.hh"
#include "G4Transform3D.hh"
#include "G4RotationMatrix.hh"
```

```
#include "G4VPhysicalVolume.hh"
#include "tls.hh"
class LXeScintHit : public G4VHit
{
  public:
    LXeScintHit();
    LXeScintHit(G4VPhysicalVolume* pVol);
    virtual ~LXeScintHit();
    LXeScintHit(const LXeScintHit &right);
    const LXeScintHit& operator=(const LXeScintHit &right);
    G4int operator==(const LXeScintHit &right) const;
    inline void *operator new(size_t);
    inline void operator delete(void *aHit);
    virtual void Draw();
    virtual void Print();
    inline void SetEdep(G4double de) { fEdep = de; }
    inline void AddEdep(G4double de) { fEdep += de; }
    inline G4double GetEdep() { return fEdep; }
    inline void SetPos(G4ThreeVector xyz) { fPos = xyz; }
    inline G4ThreeVector GetPos() { return fPos; }
    inline const G4VPhysicalVolume * GetPhysV() { return fPhysVol; }
  private:
    G4double fEdep;
    G4ThreeVector fPos;
    const G4VPhysicalVolume* fPhysVol;
};
typedef G4THitsCollection<LXeScintHit> LXeScintHitsCollection;
extern G4ThreadLocal G4Allocator<LXeScintHit>* LXeScintHitAllocator;
inline void* LXeScintHit::operator new(size_t)
{
  if(!LXeScintHitAllocator)
      LXeScintHitAllocator = new G4Allocator<LXeScintHit>;
  return (void *) LXeScintHitAllocator->MallocSingle();
}
inline void LXeScintHit::operator delete(void *aHit)
{
  LXeScintHitAllocator->FreeSingle((LXeScintHit*) aHit);
}
#endif
```

A.18. Archivo LXeScintHit.cc

```
#include "LXeScintHit.hh"
#include "G4ios.hh"
```

```
#include "G4VVisManager.hh"
#include "G4Colour.hh"
#include "G4VisAttributes.hh"
#include "G4LogicalVolume.hh"
#include "G4VPhysicalVolume.hh"
G4ThreadLocal G4Allocator<LXeScintHit>* LXeScintHitAllocator=0;
LXeScintHit::LXeScintHit() : fEdep(0.), fPos(0.), fPhysVol(0) {}
LXeScintHit::LXeScintHit(G4VPhysicalVolume* pVol) : fPhysVol(pVol) {}
LXeScintHit::~LXeScintHit() {}
LXeScintHit::LXeScintHit(const LXeScintHit &right) : G4VHit()
{
  fEdep = right.fEdep;
  fPos = right.fPos;
  fPhysVol = right.fPhysVol;
}
const LXeScintHit& LXeScintHit::operator=(const LXeScintHit &right){
  fEdep = right.fEdep;
  fPos = right.fPos;
  fPhysVol = right.fPhysVol;
  return *this;
}
G4int LXeScintHit::operator==(const LXeScintHit&) const{
  return false;
}
void LXeScintHit::Draw() {}
void LXeScintHit::Print() {}
```

A.19. Archivo LXePMTHit.hh

```
#ifndef LXePMTHit_h
#define LXePMTHit_h 1
#include "G4VHit.hh"
#include "G4THitsCollection.hh"
#include "G4Allocator.hh"
#include "G4ThreeVector.hh"
#include "G4LogicalVolume.hh"
#include "G4Transform3D.hh"
#include "G4RotationMatrix.hh"
#include "G4VPhysicalVolume.hh"
#include "tls.hh"
class G4VTouchable;
class LXePMTHit : public G4VHit
{
  public:
```

```
16      LXePMTHit();
17      virtual ~LXePMTHit();
18      LXePMTHit(const LXePMTHit &right);
19
20      const LXePMTHit& operator=(const LXePMTHit &right);
21      G4int operator==(const LXePMTHit &right) const;
22
23      inline void *operator new(size_t);
24      inline void operator delete(void *aHit);
25
26      virtual void Draw();
27      virtual void Print();
28
29      inline void SetDrawit(G4bool b){fDrawit=b;}
30      inline G4bool GetDrawit(){return fDrawit;}
31
32      inline void IncPhotonCount(){fPhotons++;}
33      inline G4int GetPhotonCount(){return fPhotons;}
34
35      inline void SetPMTNumber(G4int n) { fPmtNumber = n; }
36      inline G4int GetPMTNumber() { return fPmtNumber; }
37
38      inline void SetPMTPhysVol(G4VPhysicalVolume* physVol){this->fPhysVol
           =physVol;}
39      inline G4VPhysicalVolume* GetPMTPhysVol(){return fPhysVol;}
40
41      inline void SetPMTPos(G4double x,G4double y,G4double z){
42        fPos=G4ThreeVector(x,y,z);
43      }
44      inline G4ThreeVector GetPMTPos(){return fPos;}
45      inline void SetTime(G4double t){time=t;} ;
46      inline G4double GetTime() const {return time;} ;
47
48    private:
49
50      G4int fPmtNumber;
51      G4int fPhotons;
52      G4ThreeVector fPos;
53      G4VPhysicalVolume* fPhysVol;
54      G4bool fDrawit;
55      G4double time;
56
57  };
58  typedef G4THitsCollection<LXePMTHit> LXePMTHitsCollection;
59  extern G4ThreadLocal G4Allocator<LXePMTHit>* LXePMTHitAllocator;
60  inline void* LXePMTHit::operator new(size_t){
```

```
61    if(!LXePMTHitAllocator)
62        LXePMTHitAllocator = new G4Allocator<LXePMTHit>;
63    return (void *) LXePMTHitAllocator->MallocSingle();
64  }
65  inline void LXePMTHit::operator delete(void *aHit){
66    LXePMTHitAllocator->FreeSingle((LXePMTHit*) aHit);
67  }
```

A.20. Archivo LXePMTHit.cc

```
1   #include "LXeScintHit.hh"
2   #include "G4ios.hh"
3   #include "G4VVisManager.hh"
4   #include "G4Colour.hh"
5   #include "G4VisAttributes.hh"
6   #include "G4LogicalVolume.hh"
7   #include "G4VPhysicalVolume.hh"
8   G4ThreadLocal G4Allocator<LXeScintHit>* LXeScintHitAllocator=0;
9   LXeScintHit::LXeScintHit() : fEdep(0.), fPos(0.), fPhysVol(0) {}
10  LXeScintHit::LXeScintHit(G4VPhysicalVolume* pVol) : fPhysVol(pVol) {}
11  LXeScintHit::~LXeScintHit() {}
12  LXeScintHit::LXeScintHit(const LXeScintHit &right) : G4VHit()
13  {
14    fEdep = right.fEdep;
15    fPos = right.fPos;
16    fPhysVol = right.fPhysVol;
17  }
18  const LXeScintHit& LXeScintHit::operator=(const LXeScintHit &right){
19    fEdep = right.fEdep;
20    fPos = right.fPos;
21    fPhysVol = right.fPhysVol;
22    return *this;
23  }
24  G4int LXeScintHit::operator==(const LXeScintHit&) const{
25    return false;
26  }
27  void LXeScintHit::Draw() {}
28  void LXeScintHit::Print() {}
```

A.21. Archivo OpNoviceDetectorConstruction.cc

```
#include "OpNoviceDetectorConstruction.hh"
#include "LXePMTSD.hh"
#include "LXeScintSD.hh"
#include "G4Material.hh"
#include "G4Element.hh"
#include "G4LogicalBorderSurface.hh"
#include "G4LogicalSkinSurface.hh"
#include "G4OpticalSurface.hh"
#include "G4Box.hh"
#include "G4LogicalVolume.hh"
#include "G4ThreeVector.hh"
#include "G4PVPlacement.hh"
#include "G4SystemOfUnits.hh"
#include "G4VSensitiveDetector.hh"
#include "G4Tubs.hh"
#include "G4Sphere.hh"
#include "G4PhysicalConstants.hh"
#include "G4RotationMatrix.hh"
#include "G4Cons.hh"
#include "G4VisAttributes.hh"
#include "G4Colour.hh"
#include "G4SDManager.hh"
#include "G4VTouchable.hh"
#include "G4TouchableHistory.hh"
#include "G4Track.hh"
#include "G4Step.hh"
#include "G4ios.hh"
#include "G4VProcess.hh"

OpNoviceDetectorConstruction::OpNoviceDetectorConstruction()
 : G4VUserDetectorConstruction()
{
  fExpHall_x = fExpHall_y =  100.0*m;
  fExpHall_z = 20.0*m;
  fTank_x    = fTank_y    = fTank_z    =  5.0*m;
  fBubble_x  = fBubble_y  = fBubble_z  =  0.5*m;

  scint_x = scint_y = 7.3*m;    //Diametro
  scint_z = 1.0*m;             //Artura
  Scint_z = 4.2*m;
  outerRadius_pmtL = 10.3*cm;    //Radio de los fototubos laterales 8
     pulgadas

//Desplazamiento de posicion del tanque tank 2 respecto del tanque 1
  move_x2 = 24.74*m;    //Desplazamiento en x
  move_y2 = 0.0*m;  //Desplazamiento en y
```

```
46
47  //Desplazamiento de posicion del tanque tank 3 respecto del tanque 1
48    move_x3 = 17.56*m;    //Desplazamiento en x
49    move_y3 = -21.64*m;  //Desplazamiento en y
50
51  //desplazamiento tanque central
52    move_x4 = 14.10*m;
53    move_y4 = -7.22*m;
54
55    wtyvek=0.2*mm;   //grosor tyvek=200 micras
56    d_mtl=0.635*cm; //width aluminum cylinder
57  }
58
59  OpNoviceDetectorConstruction::~OpNoviceDetectorConstruction(){;}
60  G4VPhysicalVolume* OpNoviceDetectorConstruction::Construct()
61  {
62  // ------------- Materials -------------
63    G4double a, z, density;
64    G4int nelements;
65  //Aluminum
66  G4Material* Al = new G4Material("Al",z=13.,a=26.98*g/mole,density=2.7*g/
      cm3);
67  //Vacuum
68  G4Material* Vacuum = new G4Material("Vacuum", z=1., a=1.01*g/mole,
      density=universe_mean_density, kStateGas, 0.1*kelvin, 1.e-19*pascal)
      ;
69  //Elements
70    G4Element* N = new G4Element("Nitrogen", "N", z=7 , a=14.01*g/mole);
71    G4Element* O = new G4Element("Oxygen"  , "O", z=8 , a=16.00*g/mole);
72    G4Element* C = new G4Element("C", "C", z=6., a=12.01*g/mole);
73  // Air
74    G4Material* air = new G4Material("Air", density=1.29*mg/cm3, nelements
        =2);
75    air->AddElement(N, 70.*perCent);
76    air->AddElement(O, 30.*perCent);
77  // Water
78    G4Element* H = new G4Element("Hydrogen", "H", z=1 , a=1.01*g/mole);
79    G4Material* water = new G4Material("Water", density= 1.0*g/cm3,
        nelements=2);
80    water->AddElement(H, 2);
81    water->AddElement(O, 1);
82  //Tyvek
83    G4Material* Tyvek = new G4Material("Tyvek",density=0.935*g/cm3,2);
84    Tyvek->AddElement(C,2);
85    Tyvek->AddElement(H,4);
86  // ------------ Generate & Add Material Properties Table ------------
```

```
87      G4double photonEnergy[] =
88                { 2.034*eV, 2.068*eV, 2.103*eV, 2.139*eV,
89                  2.177*eV, 2.216*eV, 2.256*eV, 2.298*eV,
90                  2.341*eV, 2.386*eV, 2.433*eV, 2.481*eV,
91                  2.532*eV, 2.585*eV, 2.640*eV, 2.697*eV,
92                  2.757*eV, 2.820*eV, 2.885*eV, 2.954*eV,
93                  3.026*eV, 3.102*eV, 3.181*eV, 3.265*eV,
94                  3.353*eV, 3.446*eV, 3.545*eV, 3.649*eV,
95                  3.760*eV, 3.877*eV, 4.002*eV, 4.136*eV };
96
97      const G4int nEntries = sizeof(photonEnergy)/sizeof(G4double);
98
99    //
100   // Water
101   //
102     G4double refractiveIndex1[] =
103               { 1.3435, 1.344,  1.3445, 1.345,  1.3455,
104                 1.346,  1.3465, 1.347,  1.3475, 1.348,
105                 1.3485, 1.3492, 1.35,   1.3505, 1.351,
106                 1.3518, 1.3522, 1.3530, 1.3535, 1.354,
107                 1.3545, 1.355,  1.3555, 1.356,  1.3568,
108                 1.3572, 1.358,  1.3585, 1.359,  1.3595,
109                 1.36,   1.3608};
110
111     assert(sizeof(refractiveIndex1) == sizeof(photonEnergy));
112
113     G4double absorption[] =
114              {3.448*m,  4.082*m,  6.329*m,  9.174*m, 12.346*m, 13.889*m,
115              15.152*m, 17.241*m, 18.868*m, 20.000*m, 26.316*m, 35.714*m,
116              45.455*m, 47.619*m, 52.632*m, 52.632*m, 55.556*m, 52.632*m,
117              52.632*m, 47.619*m, 45.455*m, 41.667*m, 37.037*m, 33.333*m,
118              30.000*m, 28.500*m, 27.000*m, 24.500*m, 22.000*m, 19.500*m,
119              17.500*m, 14.500*m };
120
121     assert(sizeof(absorption) == sizeof(photonEnergy));
122
123     G4double scintilFast[] =
124               { 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
125                 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
126                 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
127                 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
128                 1.00, 1.00, 1.00, 1.00 };
129
130     assert(sizeof(scintilFast) == sizeof(photonEnergy));
131
132     G4double scintilSlow[] =
```

```
133             { 0.01, 1.00, 2.00, 3.00, 4.00, 5.00, 6.00,
134               7.00, 8.00, 9.00, 8.00, 7.00, 6.00, 4.00,
135               3.00, 2.00, 1.00, 0.01, 1.00, 2.00, 3.00,
136               4.00, 5.00, 6.00, 7.00, 8.00, 9.00, 8.00,
137               7.00, 6.00, 5.00, 4.00 };
138
139    assert(sizeof(scintilSlow) == sizeof(photonEnergy));
140
141    G4MaterialPropertiesTable* myMPT1 = new G4MaterialPropertiesTable();
142
143    myMPT1->AddProperty("RINDEX",       photonEnergy, refractiveIndex1,
          nEntries)
144          ->SetSpline(true);
145    myMPT1->AddProperty("ABSLENGTH",    photonEnergy, absorption,
          nEntries)
146          ->SetSpline(true);
147    myMPT1->AddProperty("FASTCOMPONENT",photonEnergy, scintilFast,
          nEntries)
148          ->SetSpline(true);
149    myMPT1->AddProperty("SLOWCOMPONENT",photonEnergy, scintilSlow,
          nEntries)
150          ->SetSpline(true);
151
152    myMPT1->AddConstProperty("SCINTILLATIONYIELD",50./MeV);
153    myMPT1->AddConstProperty("RESOLUTIONSCALE",1.0);
154    myMPT1->AddConstProperty("FASTTIMECONSTANT", 1.*ns);
155    myMPT1->AddConstProperty("SLOWTIMECONSTANT",10.*ns);
156    myMPT1->AddConstProperty("YIELDRATIO",0.8);
157
158    G4double energy_water[] = {
159       1.56962*eV, 1.58974*eV, 1.61039*eV, 1.63157*eV,
160       1.65333*eV, 1.67567*eV, 1.69863*eV, 1.72222*eV,
161       1.74647*eV, 1.77142*eV, 1.7971 *eV, 1.82352*eV,
162       1.85074*eV, 1.87878*eV, 1.90769*eV, 1.93749*eV,
163       1.96825*eV, 1.99999*eV, 2.03278*eV, 2.06666*eV,
164       2.10169*eV, 2.13793*eV, 2.17543*eV, 2.21428*eV,
165       2.25454*eV, 2.29629*eV, 2.33962*eV, 2.38461*eV,
166       2.43137*eV, 2.47999*eV, 2.53061*eV, 2.58333*eV,
167       2.63829*eV, 2.69565*eV, 2.75555*eV, 2.81817*eV,
168       2.88371*eV, 2.95237*eV, 3.02438*eV, 3.09999*eV,
169       3.17948*eV, 3.26315*eV, 3.35134*eV, 3.44444*eV,
170       3.54285*eV, 3.64705*eV, 3.75757*eV, 3.87499*eV,
171       3.99999*eV, 4.13332*eV, 4.27585*eV, 4.42856*eV,
172       4.59258*eV, 4.76922*eV, 4.95999*eV, 5.16665*eV,
173       5.39129*eV, 5.63635*eV, 5.90475*eV, 6.19998*eV
174    };
```

```
175
176     const G4int numentries_water = sizeof(energy_water)/sizeof(G4double);
177
178     //assume 100 times larger than the rayleigh scattering for now.
179     G4double mie_water[] = {
180        167024.4*m, 158726.7*m, 150742  *m,
181        143062.5*m, 135680.2*m, 128587.4*m,
182        121776.3*m, 115239.5*m, 108969.5*m,
183        102958.8*m, 97200.35*m, 91686.86*m,
184        86411.33*m, 81366.79*m, 76546.42*m,
185        71943.46*m, 67551.29*m, 63363.36*m,
186        59373.25*m, 55574.61*m, 51961.24*m,
187        48527.00*m, 45265.87*m, 42171.94*m,
188        39239.39*m, 36462.50*m, 33835.68*m,
189        31353.41*m, 29010.30*m, 26801.03*m,
190        24720.42*m, 22763.36*m, 20924.88*m,
191        19200.07*m, 17584.16*m, 16072.45*m,
192        14660.38*m, 13343.46*m, 12117.33*m,
193        10977.70*m, 9920.416*m, 8941.407*m,
194        8036.711*m, 7202.470*m, 6434.927*m,
195        5730.429*m, 5085.425*m, 4496.467*m,
196        3960.210*m, 3473.413*m, 3032.937*m,
197        2635.746*m, 2278.907*m, 1959.588*m,
198        1675.064*m, 1422.710*m, 1200.004*m,
199        1004.528*m, 833.9666*m, 686.1063*m
200     };
201
202     assert(sizeof(mie_water) == sizeof(energy_water));
203
204     // gforward, gbackward, forward backward ratio
205     G4double mie_water_const[3]={0.99,0.99,0.8};
206
207     myMPT1->AddProperty("MIEHG",energy_water,mie_water,numentries_water)
208           ->SetSpline(true);
209     myMPT1->AddConstProperty("MIEHG_FORWARD",mie_water_const[0]);
210     myMPT1->AddConstProperty("MIEHG_BACKWARD",mie_water_const[1]);
211     myMPT1->AddConstProperty("MIEHG_FORWARD_RATIO",mie_water_const[2]);
212
213     G4cout << "Water G4MaterialPropertiesTable" << G4endl;
214     myMPT1->DumpTable();
215
216     water->SetMaterialPropertiesTable(myMPT1);
217
218     // Set the Birks Constant for the Water scintillator
219
220     water->GetIonisation()->SetBirksConstant(0.126*mm/MeV);
```

```
221
222   //
223   // Air
224   //
225     G4double refractiveIndex2[] =
226               { 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
227                 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
228                 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
229                 1.00, 1.00, 1.00, 1.00, 1.00, 1.00, 1.00,
230                 1.00, 1.00, 1.00, 1.00 };
231
232     G4MaterialPropertiesTable* myMPT2 = new G4MaterialPropertiesTable();
233     myMPT2->AddProperty("RINDEX", photonEnergy, refractiveIndex2, nEntries
          );
234
235     G4cout << "Air G4MaterialPropertiesTable" << G4endl;
236     myMPT2->DumpTable();
237
238     air->SetMaterialPropertiesTable(myMPT2);
239   // ------------- Volumes --------------
240   // The experimental Hall
241     G4Box* expHall_box = new G4Box("World",fExpHall_x,fExpHall_y,
          fExpHall_z);
242
243     G4LogicalVolume* expHall_log
244       = new G4LogicalVolume(expHall_box,Vacuum,"World",0,0,0);
245
246     G4VPhysicalVolume* expHall_phys
247       = new G4PVPlacement(0,G4ThreeVector(),expHall_log,"World",0,false,0)
            ;
248
249   // The Water Tank
250     G4Tubs* waterTank_box = new  G4Tubs("Tank",           //Nombre
251                                         0.,              //radio interno
252                                         scint_x/2.,      //radio externo
253                                         scint_z/2.,      //mitad de
                                              longitud en z
254                                         0.*deg,          //comienzo en
                                              phi
255                                         360.*deg);       //segmento del
                                              angulo
256
257     G4LogicalVolume* waterTank_log
258       = new G4LogicalVolume(waterTank_box,water,"Tank",0,0,0);
259   //Tank 1
260     G4VPhysicalVolume* waterTank_phys
```

```
261      = new G4PVPlacement(0,G4ThreeVector(),waterTank_log,"Tank1",
262                          expHall_log,false,1);
263  //Tank 2
264    G4VPhysicalVolume* waterTank_phys2
265      = new G4PVPlacement(0,G4ThreeVector(move_x2,move_y2,0.0*m),
           waterTank_log,"Tank2",
266                          expHall_log,false,2);
267  //Tank 3
268    G4VPhysicalVolume* waterTank_phys3
269      = new G4PVPlacement(0,G4ThreeVector(move_x3,move_y3,0.0*m),
           waterTank_log,"Tank3",
270                          expHall_log,false,3);
271   //tanque central
272    G4Tubs* waterTank_box_c = new  G4Tubs("Tank_c",              //Nombre
273                                          0.,             //radio interno
274                                          scint_x/2.,     //radio externo
275                                          Scint_z/2.,     //mitad de
                                               longitud en z
276                                          0.*deg,         //comienzo en
                                               phi
277                                          360.*deg);      //segmento del
                                               angulo
278
279    G4LogicalVolume* waterTank_log_c
280      = new G4LogicalVolume(waterTank_box_c,water,"Tank_c",0,0,0);
281    G4VPhysicalVolume* waterTank_phys4
282      = new G4PVPlacement(0,G4ThreeVector(move_x4,move_y4,1.6*m),
           waterTank_log_c,"Tank4",
283                          expHall_log,false,4);
284
285  //The Tyvek cylinder
286
287  G4Tubs* tivek_cylinder = new G4Tubs("tivek_cylinder",
288                                      scint_x/2.,
289                                      scint_x/2.+ wtyvek,
290                                      scint_z/2.,
291                                      0.*deg,
292                                      360.*deg);//cilindro completo
293
294  G4LogicalVolume* tyvek_log = new G4LogicalVolume(tivek_cylinder,
       G4Material::GetMaterial("Tyvek"),"tivek_log",0,0,0);
295
296  //Tank 1
297  G4VPhysicalVolume* tyvek_phys = new G4PVPlacement(0,  //sin rotacion
298                                    G4ThreeVector(0.,0.,0.),//en (x, y, z
                                         )
```

```
                                    tyvek_log,    //su volumen logico
                                    "tivek_cylinder1",      //su nombre
                                    expHall_log,    //su volumen madre
                                    false,   //sin operaciones booleanas
                                    1);     //numero de copia
  //Tank 2
  G4VPhysicalVolume* tyvek_phys2 = new G4PVPlacement(0,
                                    G4ThreeVector(move_x2,move_y2,0.0*m)
                                        ,//en (x, y, z)
                                    tyvek_log,
                                    "tivek_cylinder2",
                                    expHall_log,
                                    false,
                                    2);
  //Tank 3
  G4VPhysicalVolume* tyvek_phys3 = new G4PVPlacement(0,      //sin
      rotacion
                                    G4ThreeVector(move_x3,move_y3,0.0*m)
                                        ,//en (x, y, z)
                                    tyvek_log,
                                        //su volumen logico
                                    "tivek_cylinder3",
                                                                  //su
                                        nombre
                                    expHall_log,
                                    false,
                                    3);
  //tanque central
  /*G4Tubs* tivek_cylinder_c = new G4Tubs("tivek_cylinder_c",
                                     scint_x/2.,
                                     scint_x/2.+ wtyvek,
                                     Scint_z/2.,
                                     0.*deg,
                                     360.*deg);//cilindro completo

  G4LogicalVolume* tyvek_log_c = new G4LogicalVolume(tivek_cylinder_c,
      G4Material::GetMaterial("Tyvek"),"tivek_log",0,0,0);

  G4VPhysicalVolume* tyvek_phys4 = new G4PVPlacement(0,
                                    G4ThreeVector(move_x4,move_y4,1.6*m),
                                    tyvek_log_c,
                                    "tivek_cylinder4",
                                    expHall_log,
                                    false,
                                    4);
  //Tyvek top
```

```
338
339 G4Tubs* tyvektop = new G4Tubs("tyvektop",
340                          0,
341                          scint_x/2.+wtyvek,
342                          wtyvek/2.,
343                          0.*deg,
344                          360.*deg);//tapa
345 G4LogicalVolume* tyvektop_log    = new G4LogicalVolume(tyvektop,
       G4Material::GetMaterial("Tyvek"),"tyvektop_log",0,0,0);
346
347 //Tank 1
348 G4VPhysicalVolume* tyvektop_phys =  new G4PVPlacement(0,
349                                        G4ThreeVector(0,0,scint_z/2.+
                                           wtyvek/2.),
350                                        tyvektop_log,
351                                        "tyvektop1",
352                                        expHall_log,
353                                        false,
354                                        1);
355 //Tank 2
356 G4VPhysicalVolume* tyvektop_phys2 =  new G4PVPlacement(0,
357                                        G4ThreeVector(move_x2,move_y2,
                                           scint_z/2.+wtyvek/2.),
358                                        tyvektop_log,
359                                        "tyvektop2",
360                                        expHall_log,
361                                        false,
362                                        2);
363 //Tank 3
364 G4VPhysicalVolume* tyvektop_phys3 =  new G4PVPlacement(0,
365                                        G4ThreeVector(move_x3,move_y3,
                                           scint_z/2.+wtyvek/2.),
366                                        tyvektop_log,
367                                        "tyvektop3",
368                                        expHall_log,
369                                        false,
370                                        3);
371
372 //tanque central
373 /*G4Tubs* tyvektop_c = new G4Tubs("tyvektop_c", 0, scint_x/2.+wtyvek,
       wtyvek/2., 0.*deg, 360.*deg);//tapa
374 G4LogicalVolume* tyvektop_log_c    = new G4LogicalVolume(tyvektop_c,
       G4Material::GetMaterial("Tyvek"),"tyvektop_log_c",0,0,0);
375 G4VPhysicalVolume* tyvektop_phys4 =  new G4PVPlacement(0,
376                                        G4ThreeVector(move_x4,move_y4,
                                           Scint_z/2.+wtyvek/2.+1.6*m),
```

```
                                          tyvektop_log_c,
                                          "tyvektop4",
                                          expHall_log,
                                          false,
                                          4);
*/

//Tyvek bottom

G4Tubs* tyvekbottom = new G4Tubs("tyvekbottom",
                         0.,
                         scint_x/2.+wtyvek,
                         wtyvek/2.,
                         0.*deg,
                         360.*deg);//base
G4LogicalVolume* tyvekbottom_log = new G4LogicalVolume(tyvekbottom,
   G4Material::GetMaterial("Tyvek"),"tyvekbottom_log",0,0,0);
//Tank 1
G4VPhysicalVolume* tyvekbottom_phys = new G4PVPlacement(0,
                                          G4ThreeVector(0,0,-scint_z/2.-
                                             wtyvek/2.),
                                          tyvekbottom_log,
                                          "tyvekbottom1",
                                          expHall_log,
                                          false,
                                          1);
//Tank 2
G4VPhysicalVolume* tyvekbottom_phys2 = new G4PVPlacement(0,
                                          G4ThreeVector(move_x2,move_y2,-
                                             scint_z/2.-wtyvek/2.),
                                          tyvekbottom_log,
                                          "tyvekbottom2",
                                          expHall_log,
                                          false,
                                          2);
//Tank 3
G4VPhysicalVolume* tyvekbottom_phys3 = new G4PVPlacement(0,
                                          G4ThreeVector(move_x3,move_y3,-
                                             scint_z/2.-wtyvek/2.),
                                          tyvekbottom_log,
                                          "tyvekbottom3",
                                          expHall_log,
                                          false,
                                          3);
//tanque central
```

```
418  /*G4Tubs* tyvekbottom_c = new G4Tubs("tyvekbottom_c", 0., scint_x/2.+
         wtyvek, wtyvek/2., 0.*deg, 360.*deg);//base
419  G4LogicalVolume* tyvekbottom_log_c = new G4LogicalVolume(tyvekbottom_c,
         G4Material::GetMaterial("Tyvek"),"tyvekbottom_log_c",0,0,0);
420  G4VPhysicalVolume* tyvekbottom_phys4 = new G4PVPlacement(0,
421                                          G4ThreeVector(move_x4,move_y4,-
                                                scint_z/2.-wtyvek/2.),
422                                          tyvekbottom_log_c,
423                                          "tyvekbottom1_c",
424                                          expHall_log,
425                                          false,
426                                          4);
427
428  //TyvekWall x1
429  G4Box* tyvekwallshortx1 = new G4Box("tyvekwallx", scint_x/2., wtyvek/2.,
          scint_z/2.);
430
431  G4LogicalVolume* tyvekwallshortx1_log  = new G4LogicalVolume(
         tyvekwallshortx1, G4Material::GetMaterial("Tyvek"), "
         tyvekwallshortx1_log",0,0,0);
432
433  G4VPhysicalVolume* tyvekwallshortx1_phys = new G4PVPlacement(0,
434                                       G4ThreeVector(0, 0, 0),
435                                       tyvekwallshortx1_log,
436                                       "tyvekwallshortx1",
437                                       waterTank_log,
438                                       false,
439                                       0);
440
441  //TyvekWallshort y1
442
443  G4Box* tyvekwallshorty1 = new G4Box("tyvekwally1", wtyvek/2., scint_x
         /4.-wtyvek/4, scint_z/2.);
444
445  G4LogicalVolume* tyvekwallshorty1_log  = new G4LogicalVolume(
         tyvekwallshorty1, G4Material::GetMaterial("Tyvek"), "
         tyvekwallshorty1_log",0,0,0);
446
447  //First y wall
448  //
449  G4VPhysicalVolume* tyvekwallshorty1_phys = new G4PVPlacement(0,
450                                       G4ThreeVector(0, -scint_y/4.-
                                             wtyvek/2, 0),
451                                       tyvekwallshorty1_log,
452                                       "tyvekwallshorty1",
453                                       waterTank_log,
```

```
454                                         false,
455                                         0);
456
457
458   //Second y wall
459
460   G4VPhysicalVolume* tyvekwallshorty2_phys = new G4PVPlacement(0,
461                                         G4ThreeVector(0, scint_y/4.+
                                                wtyvek/2, 0),
462                                         tyvekwallshorty1_log,
463                                         "tyvekwallshorty2",
464                                         waterTank_log,
465                                         false,
466                                         1);
467
468   //PMTs spere
469   //
470   G4double sphere_w = 0.5*mm;     //sphere_width
471   G4double cone_w = 0.0*mm;       //cone_width
472   G4double long_cone = 30*cm;//long cone
473   G4RotationMatrix* rm = new G4RotationMatrix();
474   rm->rotateY(180*deg);
475   G4double sshift=outerRadius_pmtL*0.5;
476
477   G4Sphere* photocathL = new G4Sphere("photocathL",outerRadius_pmtL-
          sphere_w,outerRadius_pmtL,0.*deg,360.*deg,0.*deg,60.*deg);
478
479   G4LogicalVolume* photocath_logL= new G4LogicalVolume(photocathL,
480                                          G4Material::GetMaterial("Al"),
481                                          "photocath_logL");
482
483   //PMTs first tank
484
485   G4VPhysicalVolume* photocatht1_phys = new G4PVPlacement(rm,G4ThreeVector
          (203.0*cm,203.0*cm,scint_z/2.+sshift), photocath_logL,"photocatht1",
           waterTank_log,false,10);
486
487   G4VPhysicalVolume* photocatht2_phys = new G4PVPlacement(rm,G4ThreeVector
          (-203.0*cm,203.0*cm,scint_z/2.+sshift), photocath_logL,"photocatht2"
          , waterTank_log,false,11);
488
489   G4VPhysicalVolume* photocatht3_phys = new G4PVPlacement(rm,G4ThreeVector
          (203.0*cm,-203.0*cm,scint_z/2.+sshift),  photocath_logL,"photocatht3
          ",waterTank_log,false,12);
490
```

```
491  G4VPhysicalVolume* photocatht4_phys = new G4PVPlacement(rm,G4ThreeVector
         (-203.0*cm,-203.0*cm,scint_z/2.+sshift), photocath_logL,"photocatht4
         ", waterTank_log,false,13);
492
493  //Aluminum cylinder
494
495  G4Tubs* AlCylinder = new G4Tubs("AlCylinder",
496                          scint_x/2.+wtyvek,
497                          scint_x/2.+wtyvek + d_mtl,
498                          scint_z/2.,
499                          0.*deg,
500                          360.*deg);//base
501  G4LogicalVolume* AlCylinder_log = new G4LogicalVolume(AlCylinder,
         G4Material::GetMaterial("Al"),"AlCylinder_log",0,0,0);
502  //Tank 1
503  G4VPhysicalVolume* AlCylider_phys = new G4PVPlacement(0,
504                                          G4ThreeVector(0,0,0),
505                                          AlCylinder_log,
506                                          "AlCylinder1",
507                                          expHall_log,
508                                          false,
509                                          1);
510
511  //Tank 2
512  G4VPhysicalVolume* AlCylider_phys2 = new G4PVPlacement(0,
513                                          G4ThreeVector(move_x2,move_y2
                                                ,0),
514                                          AlCylinder_log,
515                                          "AlCylinder2",
516                                          expHall_log,
517                                          false,
518                                          2);
519
520  //Tank 3
521  G4VPhysicalVolume* AlCylider_phys3 = new G4PVPlacement(0,
522                                          G4ThreeVector(move_x3,move_y3
                                                ,0),
523                                          AlCylinder_log,
524                                          "AlCylinder3",
525                                          expHall_log,
526                                          false,
527                                          3);
528  //tanque central
529  G4Tubs* AlCylinder_c = new G4Tubs("AlCylinder_c",
530                          scint_x/2.+wtyvek,
531                          scint_x/2.+wtyvek + d_mtl,
```

```
532                             Scint_z/2.,
533                             0.*deg,
534                             360.*deg);//base
535   G4LogicalVolume* AlCylinder_log_c = new G4LogicalVolume(AlCylinder_c,
          G4Material::GetMaterial("Al"),"AlCylinder_log_c",0,0,0);
536   G4VPhysicalVolume* AlCylider_phys4 = new G4PVPlacement(0,
537                                               G4ThreeVector(move_x4,move_y4
                                                    ,1.6*m),
538                                               AlCylinder_log_c,
539                                               "AlCylinder4",
540                                               expHall_log,
541                                               false,
542                                               4);
543
544   //PMTs en tanque central
545
546   G4VPhysicalVolume* photocatht1_phys_c = new G4PVPlacement(0,
          G4ThreeVector(106.0*cm,106.0*cm, - 1.98*m), photocath_logL,"
          photocatht1", waterTank_log_c,false,10);
547   G4VPhysicalVolume* photocatht2_phys_c = new G4PVPlacement(0,
          G4ThreeVector(-106.0*cm,106.0*cm, - 1.98*m), photocath_logL,"
          photocatht1", waterTank_log_c,false,11);
548   G4VPhysicalVolume* photocatht3_phys_c = new G4PVPlacement(0,
          G4ThreeVector(106.0*cm,-106.0*cm, - 1.98*m), photocath_logL,"
          photocatht1", waterTank_log_c,false,12);
549   G4VPhysicalVolume* photocatht4_phys_c = new G4PVPlacement(0,
          G4ThreeVector(-106.0*cm,-106.0*cm, - 1.98*m), photocath_logL,"
          photocatht1", waterTank_log_c,false,13);
550
551
552
553   // PMT-Sensitive detector
554
555     // PMT SD
556
557     if (!fPmt_SD.Get()) {
558       //Created here so it exists as pmts are being placed
559       G4cout << "Construction /LXeDet/pmtSD" << G4endl;
560       LXePMTSD* pmt_SD = new LXePMTSD("/LXeDet/pmtSD");
561       fPmt_SD.Put(pmt_SD);
562       pmt_SD->InitPMTs(16); //let pmtSD know # of pmts
563       //pmt_SD->SetPmtPositions(fMainVolume->GetPmtPositions());
564     }
565
566       SetSensitiveDetector(photocath_logL, fPmt_SD.Get());
567
```

```
568     // Scint SD
569
570     if (!fScint_SD.Get()) {
571       G4cout << "Construction /LXeDet/scintSD" << G4endl;
572       LXeScintSD* scint_SD = new LXeScintSD("/LXeDet/scintSD");
573       fScint_SD.Put(scint_SD);
574     }
575     SetSensitiveDetector(waterTank_log, fScint_SD.Get());
576
577   // ------------- Surfaces --------------
578   //
579
580   //
581   // Generate & Add Material Properties Table attached to the optical
        surfaces
582   //
583     const G4int num = 2;
584     G4double Ephoton[34] = {
585   4.960*eV,4.769*eV,4.428*eV,4.133*eV,
586   3.875*eV,3.647*eV,3.444*eV,3.351*eV,
587   3.263*eV,3.170*eV,3.100*eV,3.024*eV,
588   2.952*eV,2.883*eV,2.818*eV,2.755*eV,
589   2.695*eV,2.583*eV,2.530*eV,2.480*eV,
590   2.384*eV,2.296*eV,2.138*eV,2.066*eV,
591   2.00*eV,1.938*eV,1.879*eV,1.823*eV,
592   1.771*eV,1.722*eV,1.675*eV,1.631*eV,
593   1.590*eV,1.550*eV};
594     //** Tyvek surface properties
595
596     G4double TRef[34] ={
597   0.82,0.86,0.89,0.92,0.94,
598   0.95,0.95,0.95,0.96,0.96,
599   0.97,0.97,0.97,0.97,0.97,
600   0.97,0.97,0.97,0.97,0.97,
601   0.97,0.97,0.97,0.97,0.97,
602   0.97,0.97,0.97,0.97,0.97,
603   0.97,0.97,0.97,0.97};
604   //Tyvek
605   //
606     G4MaterialPropertiesTable* optyvek = new G4MaterialPropertiesTable();
607     optyvek->AddProperty("REFLECTIVITY", Ephoton, TRef,num);
608     G4OpticalSurface* OpTyvekSurface =
609     new G4OpticalSurface("TyvekSurface",unified,polished,dielectric_metal)
          ;
610     OpTyvekSurface->SetMaterialPropertiesTable(optyvek);
611     //**Create logical skin surfaces
```

```
612     new G4LogicalSkinSurface("tyvekw_surface",tyvek_log,OpTyvekSurface);
613     new G4LogicalSkinSurface("tyvekb_surface",tyvekbottom_log,
            OpTyvekSurface);
614     new G4LogicalSkinSurface("tyvekt_surface",tyvektop_log,OpTyvekSurface)
            ;
615     new G4LogicalSkinSurface("tyvekwlx_surface",tyvekwallshortx1_log,
            OpTyvekSurface);
616     new G4LogicalSkinSurface("tyvekwlx_surface",tyvekwallshorty1_log,
            OpTyvekSurface);
617   //**Photocathode surface properties
618     G4double photocath_EFF[num]={0.15,0.18}; //Enables 'detection' of
            photons
619     G4double photocath_REFL[num]={0.,0.};
620     G4MaterialPropertiesTable* photocath_mt = new
            G4MaterialPropertiesTable();
621     photocath_mt->AddProperty("EFFICIENCY",Ephoton,photocath_EFF,num);
622     photocath_mt->AddProperty("REFLECTIVITY",Ephoton,photocath_REFL,num);
623     G4OpticalSurface* photocath_opsurf=
624       new G4OpticalSurface("photocath_opsurf",glisur,polished,
625                            dielectric_metal);
626     photocath_opsurf->SetMaterialPropertiesTable(photocath_mt);
627
628     new G4LogicalSkinSurface("photocath_surf",photocath_logL,
            photocath_opsurf);
629
630   //
631   // Water Tank
632   //
633     G4OpticalSurface* opWaterSurface = new G4OpticalSurface("WaterSurface"
            );
634     opWaterSurface->SetType(dielectric_dielectric);
635     opWaterSurface->SetFinish(ground);
636     opWaterSurface->SetModel(unified);
637
638     new G4LogicalBorderSurface("WaterSurface",
639                                waterTank_phys,expHall_phys,
                                       opWaterSurface);
640     G4OpticalSurface* opAirSurface = new G4OpticalSurface("AirSurface");
641     opAirSurface->SetType(dielectric_dielectric);
642     opAirSurface->SetFinish(polished);
643     opAirSurface->SetModel(glisur);
644     G4LogicalSkinSurface* airSurface = new G4LogicalSkinSurface("
            AirSurface", bubbleAir_log, opAirSurface);
645     G4OpticalSurface* opticalSurface = dynamic_cast <G4OpticalSurface*>
646           (airSurface->GetSurface(bubbleAir_log)->GetSurfaceProperty());
647     if (opticalSurface) opticalSurface->DumpInfo();
```

```
// Generate & Add Material Properties Table attached to the optical surfaces
  G4double ephoton[num] = {2.034*eV, 4.136*eV};
  //OpticalWaterSurface
  G4double refractiveIndex[num] = {1.35, 1.40};
  G4double specularLobe[num]    = {0.3, 0.3};
  G4double specularSpike[num]   = {0.2, 0.2};
  G4double backScatter[num]     = {0.2, 0.2};
  G4MaterialPropertiesTable* myST1 = new G4MaterialPropertiesTable();
  myST1->AddProperty("RINDEX",                ephoton, refractiveIndex, num);
  myST1->AddProperty("SPECULARLOBECONSTANT",  ephoton, specularLobe, num);
  myST1->AddProperty("SPECULARSPIKECONSTANT", ephoton, specularSpike, num);
  myST1->AddProperty("BACKSCATTERCONSTANT",   ephoton, backScatter, num);
  G4cout << "Water Surface G4MaterialPropertiesTable" << G4endl;
  myST1->DumpTable();
  opWaterSurface->SetMaterialPropertiesTable(myST1);
  //OpticalAirSurface
  G4double reflectivity[num] = {0.3, 0.5};
  G4double efficiency[num]   = {0.8, 1.0};
  G4MaterialPropertiesTable *myST2 = new G4MaterialPropertiesTable();
  myST2->AddProperty("REFLECTIVITY", ephoton, reflectivity, num);
  myST2->AddProperty("EFFICIENCY",   ephoton, efficiency,   num);
  G4cout << "Air Surface G4MaterialPropertiesTable" << G4endl;
  myST2->DumpTable();
  opAirSurface->SetMaterialPropertiesTable(myST2);
G4VisAttributes* water_va= new G4VisAttributes(G4Colour(0.0, 1.0, 1.0)) ;  // cyan
waterTank_log -> SetVisAttributes(water_va);
water_va -> SetForceAuxEdgeVisible (true);//visualizacion del cilindro
water_va -> SetForceWireframe(true);
water_va -> SetForceSolid(false);
water_va -> SetVisibility(true);
//central
G4VisAttributes* water_va_c= new G4VisAttributes(G4Colour(0.0, 1.0, 1.0) ) ;  // cyan
waterTank_log_c -> SetVisAttributes(water_va_c);
water_va_c -> SetForceAuxEdgeVisible (true);//visualizacion del cilindro
water_va_c -> SetForceWireframe(true);
water_va_c -> SetForceSolid(false);
water_va_c -> SetVisibility(true);
G4VisAttributes* photocath_vaL= new G4VisAttributes(G4Colour::Yellow()) ;  // cyan
```

```
686  photocath_logL -> SetVisAttributes(photocath_vaL);
687  photocath_vaL-> SetForceSolid(true);
688  photocath_vaL-> SetVisibility(true);
689  G4VisAttributes* tyvek_blue= new G4VisAttributes(G4Colour(0.0, 1.0, 1.0)
         ) ;  // cyan
690  tyvek_log -> SetVisAttributes(tyvek_blue);
691  tyvek_blue-> SetForceSolid(true);
692  tyvek_blue-> SetVisibility(true);
693  G4VisAttributes* tyvek_shortx1_gray= new G4VisAttributes(G4Colour::White
         ()) ;  // cyan
694  tyvekwallshortx1_log -> SetVisAttributes(tyvek_shortx1_gray);
695  tyvek_shortx1_gray-> SetForceAuxEdgeVisible (true);//visualizacion de
         pared
696  tyvek_shortx1_gray-> SetForceWireframe(true);
697  tyvek_shortx1_gray-> SetForceSolid(true);
698  tyvek_shortx1_gray-> SetVisibility(true);
699  G4VisAttributes* tyvek_shorty1_gray= new G4VisAttributes(G4Colour::White
         ()) ;  // cyan
700  tyvekwallshorty1_log -> SetVisAttributes(tyvek_shorty1_gray);
701  tyvek_shorty1_gray-> SetForceAuxEdgeVisible (true);//visualizacion de
         pared
702  tyvek_shorty1_gray-> SetForceWireframe(true);
703  tyvek_shorty1_gray-> SetForceSolid(true);
704  tyvek_shorty1_gray-> SetVisibility(true);
705    return expHall_phys;
706  }
```

A.22. Archivo OpNovicePrimaryGeneratorAction.cc

```
1  #include "OpNovicePrimaryGeneratorAction.hh"
2  #include "OpNovicePrimaryGeneratorMessenger.hh"
3
4  #include "Randomize.hh"
5
6  #include "G4Event.hh"
7  #include "G4ParticleGun.hh"
8  #include "G4ParticleTable.hh"
9  #include "G4ParticleDefinition.hh"
10 #include "G4SystemOfUnits.hh"
11
12 #include "globals.hh"
13 #include <iostream>
14 #include <fstream>
15 #include <iomanip>
```

```
16 #include <stdlib.h>
17 #include <string>
18 using namespace std;
19 //....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........
       oooOOOOOooo......
20
21 OpNovicePrimaryGeneratorAction::OpNovicePrimaryGeneratorAction()
22  : G4VUserPrimaryGeneratorAction(),
23    fParticleGun(0)
24 {
25   G4int n_particle = 1;
26   fParticleGun = new G4ParticleGun(n_particle);
27 }
28
29 OpNovicePrimaryGeneratorAction::~OpNovicePrimaryGeneratorAction()
30 {
31   delete fParticleGun;
32   delete fGunMessenger;
33 }
34
35 void OpNovicePrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
36 {
37     G4double fTEnergy;
38     G4int nl=0;
39     G4int nl2=0;
40     ifstream inc("ya.dat",ios::in);
41     ofstream dat("dato.dat",ios::app);
42     if ( !inc ){
43     G4cout<<"Error al abrir archivo de datos"<<G4endl;
44     G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable()
          ;
45     G4String particleName;
46     fParticleGun->SetParticleDefinition(particleTable->
          FindParticle(particleName="mu-"));
47     fParticleGun->SetParticleEnergy(10000*MeV);
48     fParticleGun->SetParticlePosition(G4ThreeVector(0.0*m,0.0*m,2.6*m))
          ;;// vector anterior:(1748.0*cm , 1115.0*cm, 2.51*m)
49     fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,-1.))
          ;
50     fParticleGun->GeneratePrimaryVertex(anEvent);
51     }
52     else {  G4cout<<"Archivo de datos abierto exitosamente"<<G4endl;
53       G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable
            ();
54          G4String particleName;
55
```

```
56            while (inc>>fPartId>>fUx>>fUy>>fUz>>fX>>fY>>fTimeDelay>>
                  fTEnergy)
57           {   if ((nl2==0) && (fPartId == 999233900)){
58               fPrimEnergy = fUx;
59               fangle = fUy;
60               fPhi = fUz;
61               fxCore = fX;
62               fyCore = fY;
63               dat<<"999233900"<<" "<<fPrimEnergy<<" "<<fangle<<" "<<fPhi<<
                     " "<<fxCore<<" "<<fyCore<<" 0 0 0 0"<<endl;}
64
65             nl2++;
66               nl++;
67             if ( fPartId == 1)
68               {
69                 particleName = "gamma";
70               }
71             else if ( fPartId == 2 )
72               {                        //e+
73                 particleName = "e+";
74               }
75             else if ( fPartId == 3 )
76               {                        //e-
77                 particleName = "e-";
78               }
79             else if ( fPartId == 5 )
80               {                        //mu+
81                 particleName = "mu+";
82               }
83             else if ( fPartId == 6 )
84               {                        //mu-
85                 particleName = "mu-";
86               }
87             else//do not simulate other particles
88               continue;
89             fZ=450;//posicion inicial de la particula en z
90             G4double ttime;
91             ttime = fTimeDelay;
92             //    G4cout<<"Delay  "<<fTimeDelay<<"  ttime  "<<ttime<<
                     G4endl;
93             G4ThreeVector
94             direction ( fUx*GeV, fUy*GeV, -fUz*GeV );
95             G4ThreeVector
96               position ( fX*cm, fY*cm, fZ*cm );
97             //    G4cout<<"PartiName= "<<particleName<<"   "<<fUx<<G4endl;
98
```

```
99            fParticleGun->SetParticleDefinition ( particleTable->
                 FindParticle ( particleName ) );
100           fParticleGun->SetParticleMomentumDirection ( direction );
101           fParticleGun->SetParticlePosition ( position );
102           fParticleGun->SetParticleEnergy ( fTEnergy*GeV );
103           fParticleGun->SetParticleTime ( ttime*ns );
104           fParticleGun->GeneratePrimaryVertex(anEvent);
105         }  //while
106          G4cout<<"Total de eventos "<<nl2<<G4endl;
107          G4cout<<"Eventos dentro del Hall "<<nl<<G4endl;
108     }
109
110 dat.close();
111
112 }
113 void OpNovicePrimaryGeneratorAction::SetOptPhotonPolar()
114 {
115  G4double angle = G4UniformRand() * 360.0*deg;
116  SetOptPhotonPolar(angle);
117 }
118 void OpNovicePrimaryGeneratorAction::SetOptPhotonPolar(G4double angle)
119 {
120  if (fParticleGun->GetParticleDefinition()->GetParticleName()!="
        opticalphoton")
121    {
122      G4cout << "--> warning from PrimaryGeneratorAction::
           SetOptPhotonPolar() :"
123             "the fParticleGun is not an opticalphoton" << G4endl;
124      return;
125    }
126  G4ThreeVector normal (1., 0., 0.);
127  G4ThreeVector kphoton = fParticleGun->GetParticleMomentumDirection();
128  G4ThreeVector product = normal.cross(kphoton);
129  G4double modul2       = product*product;
130
131  G4ThreeVector e_perpend (0., 0., 1.);
132  if (modul2 > 0.) e_perpend = (1./std::sqrt(modul2))*product;
133  G4ThreeVector e_paralle    = e_perpend.cross(kphoton);
134
135  G4ThreeVector polar = std::cos(angle)*e_paralle + std::sin(angle)*
        e_perpend;
136  fParticleGun->SetParticlePolarization(polar);
137 }
```