UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

CARACTERIZACIÓN DE LA INTERACCIÓN DE RAYOS CÓSMICOS CON EL VOLCÁN PICO DE ORIZABA POR MEDIO DE UNA SIMULACIÓN MONTE CARLO CON GEANT4.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Físico

PRESENTA:

Luis Joel Hernández Martínez

TUTOR

Hermes León Vargas

Ciudad Universitaria, Cd. Mx. 2017

JURADO ASIGNADO

Presidente:	Dr. Ernesto José María de la Salete Belmont Moreno.
Vocal:	Dr. Eric Vázquez Jáuregui.
Secretario:	Dr. Hermes León Vargas.
1er Suplente:	Dr. Luis Gottdiener Gutmann.
2o Suplente:	Dr. Irving Omar Morales Agiss.

Sitio donde se realizó el tema:

Operaciones LH-HAWC
Edificio Colisur del Instituto de Física de la UNAM.

AGRADECIMIENTOS

Al Dr. Hermes León Vargas *por su apoyo ($) y permitirme participar en este proyecto.*

Un agradecimiento especial al Dr. Eric Vázquez Jáuregui *por sus consejos y valiosa asesoría.*

Gracias al Programa UNAM-DGAPA-PAPIIT IA102715, *al proyecto* CONACyT 254964 *y al* Fondo Sectorial de Investigación para la Educación *por apoyar este trabajo.*

Dedicado a...

Mis padres. Por el apoyo recibido, pero en especial a mi madre porque ella es la razón de haber llegado tan lejos.

Mis amigos. Su presencia y los buenos momentos me permitieron seguir adelante a lo largo de la licenciatura y durante el proceso de esta tesis.

RESUMEN

En esta tesis se presenta una introducción al uso del software Geant4 y se estudia la capacidad del volcán Pico de Orizaba para absorber muones de alta energía, entre 30 GeV y 10 TeV. El capítulo 1 es un manual que explica como usar Geant4 para construir un programa de cómputo capaz de simular la interacción entre partículas y materia. En el capítulo 2 se dan los motivos por los cuales se estudia la absorción de muones en el volcán y su relación con el observatorio de rayos gamma HAWC. El capítulo 3 muestra los resultados de ese estudio.

Este trabajo nació de la idea de utilizar al observatorio HAWC como un instrumento para la detección indirecta de neutrinos ultra energéticos. La detección es posible si los neutrinos tienen interacción con los nucleones del volcán Pico de Orizaba. Esta interacción produce, a través de una corriente cargada, leptones cargados del mismo sabor que el neutrino incidente. Sin embargo, la principal fuente de ruido para detectar estos leptones cargados son los muones que acompañan a cascadas atmosféricas muy inclinadas y que provienen de la dirección del volcán. Por eso la importancia de estimar la capacidad del Pico de Orizaba para absorber este ruido.

Se encontró que los resultados de las simulaciones hechas con Geant4 son compatibles con la ecuación para la pérdida continua de energía de muones en agua pura y roca estándar,

$$-\frac{d\mathrm{E}}{dX} = \mathrm{a}\,(\mathrm{E}) + \mathrm{b}\,(\mathrm{E})\,\mathrm{E}$$

Donde a(E) y b(E) representan las pérdidas de energía por ionización y radiación respectivamente. También se encontró que, de acuerdo a las simulaciones, el 95 % de los muones que penetran en el volcán y tienen una energía $\leq$ 1 TeV son absorbidos.

ÍNDICE

ÍNDICE DE FIGURAS

LISTA DE TABLAS

OBJETIVOS

Objetivo Principal: Implementar una simulación sencilla en GEANT4 con el fin de estudiar la capacidad del volcán Pico de Orizaba de absorber leptones cargados de alta energía y estimar el ruido esperado en el estudio de neutrinos ultra-energéticos usando la técnica Earth-Skimming en el observatorio de rayos gamma HAWC.

Objetivo secundario: Realizar una descripción detallada del software GEANT4, tanto su uso para simular fenómenos físicos como su capacidad para simular los medios de detección.

INTRODUCCIÓN

Este trabajo no es sobre neutrinos aunque está muy relacionado con ellos. Por algún tiempo los neutrinos eran partículas de interés exclusivamente teórico, que solo ayudaban a explicar el espectro de los electrones durante la desintegración β. El trabajo del físico italiano Bruno Pontecorvo [1] fue esencial en la física de neutrinos pues mostró que no son partículas pasivas, ya que pueden ser detectadas. En la década de 1950 se logró detectar por primera vez un neutrino mediante el decaimiento β^- [2]. Posteriormente, en 1968 se lograron detectar neutrinos procedentes del Sol [3]. Los neutrinos solares ayudaron a confirmar la teoría de John Bahcall sobre las reacciones nucleares que ocurren en el interior del Sol y también fue la primera vez que se usaron para estudiar un cuerpo astrofísico. En 1987 los neutrinos volvieron a ser noticia cuando los detectores Kamiokande de Japón e IMB de EUA detectaron neutrinos procedentes de una supernova ubicada en la Gran Nube de Magallanes. Cuando ocurrió la explosión, nombrada SN 1987A, se detectaron 11 señales por parte de Kamiokande [4] y 8 eventos por parte de IMB [5]. La energía de estos neutrinos dio información de las condiciones en las que fueron producidos y se confirmó la validez de las predicciones teóricas hechas por los astrofísicos.

En el siglo XXI los neutrinos se han convertido en una herramienta útil para estudiar objetos astrofísicos pues, a diferencias de los fotones, éstos no son atenuados por los fotones del fondo cósmico. La astrofísica de neutrinos promete mucho: estudiar directamente las regiones más lejanas del Universo. Para ello es necesario observatorios capaces de detectar neutrinos y ese es precisamente el problema, porque los neutrinos tienen baja interacción con la materia. La técnica Earth-skimming propone utilizar detectores de partículas y apuntarlos hacia la corteza terrestre con el fin de detectar la interacción entre los neutrinos y los nucleones. Una ventaja de esta técnica es que se pueden utilizar arreglos de detectores ubicados cerca de montañas o volcanes para que funcionen como un observatorio de neutrinos, independientemente de si estos arreglos fueron diseñados para este fin.

El observatorio de rayos gamma HAWC es un arreglo de fotomultiplicadores sumergidos en agua, distribuidos en una superficie mayor a 2 hectáreas que se ubica a casi 6 kilómetros del volcán Pico de Orizaba. Esto lo convierte en un candidato a observatorio de neutrinos. Sin embargo, es necesario de-

terminar si es capaz de discriminar la señal generada por un neutrino extragaláctico y el ruido generado por la enorme cantidad de partículas creadas en las cascadas atmosféricas. El objetivo de este trabajo es estimar la capacidad del volcán Pico de Orizaba para absorber los muones generados en las cascadas atmosféricas. Los muones son las partículas más abundantes a 4 100 metros de altura respecto del nivel del mar, la altitud a la que se encuentra HAWC. Si el volcán es capaz de absorber los muones horizontales que se generan en cascadas inclinadas, se eleva la probabilidad de que los leptones cargados que lleguen a HAWC desde la dirección del volcán tengan como origen la interacción de un neutrino con un nucleón. Con este fin se utiliza el software GEANT4, para realizar simulaciones Monte Carlo de la interacción de los muones con el volcán.

Además de determinar la absorción de muones por parte del volcán, en este trabajo se incluyó un manual introductorio a Geant4. Este manual integra todo el capítulo 1 y está dirigido a personas que nunca han usado Geant4 pero están interesadas en escribir código y simular procesos físicos con él. Es en el capítulo 2 donde se dan los detalles sobre la técnica Earth-skimming, sobre las características del observatorio HAWC y sobre la simulación que estima la capacidad del volcán Pico de Orizaba para absorber leptones cargados. Los resultados de la simulación se encuentran en el capítulo 3.

CAPÍTULO 1

SIMULACIONES CON GEANT4

Este capítulo se divide en tres secciones. En la sección 1.1 se ofrece un panorama general de Geant4. Se explica a grandes rasgos lo que ofrece el software y el tipo de materiales, partículas y sólidos que se pueden simular. Con esto las personas interesadas en trabajar con Geant4 podrán saber si es el tipo de herramienta que necesitan.

En la segunda sección, 1.2, se explica cómo implementar una simulación con Geant4. No es una explicación completa ya que no se tocan varios temas, por ejemplo: el funcionamiento de los modelos y procesos que utiliza Geant4 para simular las interacciones partícula-materia, tampoco se mencionan todas las herramientas que ofrece Geant4 para visualizar las simulaciones. En realidad es un complemento a los manuales de Geant4, [6] y [7], los cuales tienen información técnica útil pero carecen de ejemplos claros que ayuden a aquellos que empiezan a trabajar con esta herramienta.

Finalmente en la sección 1.3 se muestra una simulación con Geant4. A esas alturas del trabajo se presupone que el lector conoce cómo implementar una simulación, por eso solo se explican las particularidades de las simulación y sus resultados. La sección 1.3 tiene como fin completar el objetivo del presente capítulo: explicar que es Geant4 y ayudar a otras personas que desean construir una simulación con esta herramienta.

1.1. Introducción a Geant4

En esta sección se ofrece una visión general sobre Geant4. Primero se da el contexto bajo el cual nació y cuáles eran sus objetivos (sección 1.1.1). Después, se habla a grandes rasgos de todos los paquetes que integran a Geant4 (sección 1.1.2). Finalmente, en las sub-secciones 1.1.3 y 1.1.4 se muestran algunos sólidos, partículas y materiales que se pueden simular con Geant4.

1.1.1. Los inicios de Geant4.

Geant4 (GEometry ANd Tracking) es un proyecto de colaboración que inició en 1993 con la participación de científicos e ingenieros de Canadá, EUA, Europa, Japón y Rusia [8]. El objetivo era crear un software capaz de simular el funcionamiento de los detectores utilizados en experimentos de física nuclear y de partículas. En esos años, gracias a la capacidad de los sistemas de cómputo para realizar simulaciones cada vez más complejas, se abrió la posibilidad de simular los detectores utilizados en experimentos de física nuclear y de partículas. El gran tamaño de esos detectores, su sensibilidad y complejidad, sumado al relativamente bajo costo de los sistemas de cómputo, impulsó el desarrollo de herramientas para simular a los detectores y las interacciones que ocurrían en ellos. Herramientas como ésta se requerían en otras disciplinas tales como: física de radiaciones, medicina nuclear, y de hecho, en toda área donde la interacción de partículas con materia desempeña un papel central [8]. De esta forma, Geant4 surgió como un ambicioso proyecto para diseñar una herramienta de cómputo con la capacidad de simular cualquier interacción partícula-materia.

Un sistema de cómputo (software) con aplicación en diferentes áreas debe ser flexible a futuras modificaciones, es decir, que al cambiar partes del software el resto siga trabajando sin problemas. Por esa razón, durante el primer año de desarrollo (1995), el equipo de Geant4 estudio diferentes metodologías de ingeniería de software buscando aquella que cubriera los siguientes requisitos [8]:

- Un proceso de desarrollo flexible.
- Un camino para fragmentar el software en paquetes independientes, y así poder hacer cambios en cada paquete sin afectar el funcionamiento general del software y del resto de paquetes.
- Proporcionar las herramientas, los modelos y la notación necesaria para permitir a cualquier persona modificar el software sin importan su ubicación geográfica.

Siguiendo esas directrices se eligió la metodología Booch, que incluye una descripción clara y detallada del análisis y diseño que se debe llevar a cabo durante el desarrollo de un software. Un punto importante es que la metodología Booch opera con el paradigma de *programación orientada a objetos* (POO) [8], esa es la razón por la que Geant4 utiliza un lenguaje de programación orientado a objetos, el lenguaje C++. Una descripción detallada de la metodología Booch se puede consultar en el libro de Grady Booch [9].

1.1.2. Qué ofrece Geant4.

Geant4 ofrece un conjunto de módulos o paquetes que facilitan la construcción de una simulación. Cada paquete realiza una tarea, siendo el usuario quien decide que funciones agregar a su proyecto. Estos paquetes se pueden dividir en las siguientes categorías.

Geometría del detector. Permiten al usuario construir tanto el detector como otros cuerpos sólidos que se necesiten en la simulación. Incluye figuras básicas como: esferas, cilindros y paralelepípedos.

Incluso geometrías más complejas que pueden ser o no ser simétricas, por ejemplo: pirámides, conos o volúmenes irregulares.

Simulación de partículas y materiales. Simulan las principales propiedades físicas de las partículas conocidas (carga eléctrica, masa, energía, entre otras) y diferentes materiales, como son: los elementos químicos presentes en la tabla periódica hasta el elemento 98; moléculas orgánicas; tejido y hueso humano [6].

Procesos físicos. Simulan los diferentes procesos que pueden darse en una interacción partícula-materia. Es el usuario quien decide que procesos físicos agregar a su simulación.

Adquisición de datos. Toman y guardan información que se genera durante la simulación, por ejemplo: la energía perdida por las partículas en su viaje por el detector; el vector de momento de las partículas; la desintegraciones ocurridas durante la simulación; la carga eléctrica y la masa de las partículas.

Visualización. Permiten representar en el monitor tanto la geometría del detector como algunas interacciones y propiedades físicas de las partículas.

1.1.3. Sólidos presentes en Geant4.

Son varios archivos los encargados de construir las diferentes formas geométricas disponibles en Geant4. Es importante recordar que las longitudes se registran escribiendo la mitad de su valor real, a excepción de longitudes radiales. En Geant4 el centro del sólido es el punto de referencia, a partir de él se construyen dos segmentos, tanto en el sentido positivo del eje coordenado como en el sentido negativo. Ambos segmentos tienen una longitud igual al valor indicado, por eso, si se pide un cilindro de 5 cm de altura se obtendrá un cilindro con 5 cm de altura por arriba de su centro y 5 cm por debajo del mismo.

En otras ocasiones será necesario definir dos ángulos: el ángulo inicial y el ángulo final de la figura. Ambos ángulos toman valores entre 0 y 2π pero no representan lo mismo. El ángulo inicial indica en que valor angular comenzará a dibujarse el sólido, mientras el ángulo final expresa qué tantos grados cubrirá el sólido a partir del ángulo inicial. Si el ángulo inicial es 270° y el ángulo final es 30°, la figura dibujada será una pequeña cuña que comienza en $\theta_i = 270^\circ$ y termina en $\theta_f = 300^\circ$.

Todo esto se puede entender mejor al observar los códigos de la siguiente lista de sólidos que se pueden declarar en Geant4. Cada sólido está acompañado de ejemplos e imágenes. Los fragmentos de código muestran cómo declarar cada sólido y obtener cada ejemplo.

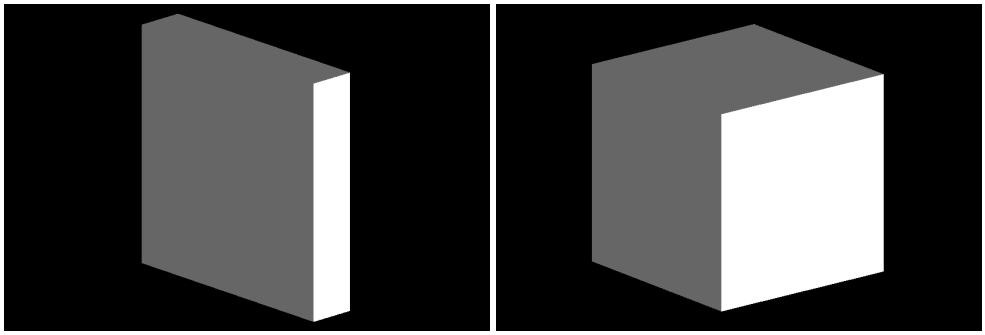
Paralelepípedos: Son objetos de la clase `G4Box` que se define en el archivo `G4Box.hh`. Su declaración se hace en coordenadas cartesianas y son necesarios cuatro datos: el nombre de la figura, su longitud en el eje X, su longitud en el eje Y y su longitud en el eje Z. Crear un prisma rectangular es muy sencillo, como se muestra en el siguiente código. En la figura 1.1 (a) se puede observar el resultado.

```
//===| Metodo Construct() del archivo |===
//===|      DetectorConstruction.cc    |===

#include "DetectorConstruction.hh"
#include "G4Box.hh"
...
...
G4VPhysicalVolume* DetectorConstruction::Construct()
{
...
...
//===| Paralelepipedo |===
 G4double XLong = 50*cm, YLong = 50*cm, ZLong = 10*cm;

 G4Box* GranBox = new G4Box(
 "Box_Gran",   //--| Nombre
 0.5*XLong,    //--| Mitad longitud X
 0.5*YLong,    //--| Mitad longitud Y
 0.5*ZLong);   //--| Mitad longitud Z
...
...
}
```

Un caso particular de la clase `G4Box` es un cubo. Para declarar un cubo solo es necesario que el tamaño del paralelepípedo en los tres ejes coordenados sea el mismo, como muestra el siguiente código. En la figura 1.1 (b) se puede observar el resultado.

```
//===| Metodo Construct() del archivo |===
//===|      DetectorConstruction.cc    |===

#include "DetectorConstruction.hh"
#include "G4Box.hh"
...
...
G4VPhysicalVolume* DetectorConstruction::Construct()
{
...
...
//===| Cubo |===
 G4double XLong = YLong =  ZLong = 50*cm;

 G4Box* CuboBox = new G4Box(
 "Box_Cubo",   //--| Nombre
 0.5*XLong,    //--| Mitad longitud X
 0.5*YLong,    //--| Mitad longitud Y
 0.5*ZLong);   //--| Mitad longitud Z
...
...
```

```
}
```



(a) Paralelepípedo. (b) Cubo.

Figura 1.1: Sólidos que se pueden construir en Geant4: paralelepípedos y cubos.

Conos: Para declarar un cono se necesitan 8 datos: el nombre del cono, el radio interno y externo de su cima, el radio interno y externo de su base, su altura, el ángulo inicial y el ángulo final del cono. Estas variables permiten declarar diferentes clases de conos: conos truncados, conos completos y conos abiertos. Un cono truncado es aquel que no termina en punta, por tanto, su cima y su base son superficies planas. El siguiente código muestra cómo declarar uno, el cual se puede observar en la figura 1.2 (a).

```
//===| Metodo Construct() del archivo |===
//===|      DetectorConstruction.cc     |===

#include "DetectorConstruction.hh"
#include "G4Cons.hh"
...
...
G4VPhysicalVolume* DetectorConstruction::Construct()
{
...
...
//===| Cono Truncado |===
 G4double RadioA_min = 0.*cm,    RadioA_max = 2.*cm;
 G4double RadioB_min = 0.*cm,    RadioB_max = 1.*cm;
 G4double Altura     = 1.*cm;
 G4double Angulo_min = 0.,       Angulo_max = twopi;

 G4Cons* TrunCono = new G4Cons(
 "Cono_Sol",     //--| Nombre Vol. Solido
 RadioA_min,     //--| Radio min. extremo A
 RadioA_max,     //--| Radio max. extremo A
 RadioB_min,     //--| Radio min. extremo B
 RadioB_max,     //--| Radio max. extremo B
```

```
 Altura,          //--| Altura
 Angulo_min,      //--| Angulo Inicial
 Angulo_max);     //--| Angulo Final
...
...
}
```

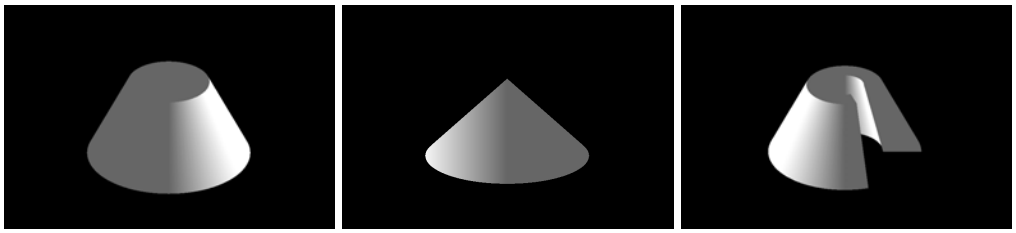
En cambio, un cono completo tiene uno de sus extremos en forma de punta. Para declararlo basta con que el radio interno y externo de ese extremo sean nulos. El siguiente código es un ejemplo de un cono completo el cual se muestra en la figura 1.2 (b).

```
//===| Metodo Construct() del archivo |===
//===|      DetectorConstruction.cc     |===

#include "DetectorConstruction.hh"
#include "G4Cons.hh"
...
...
G4VPhysicalVolume* DetectorConstruction::Construct()
{
...
...
//===| Cono Completo |===
 G4double RadioA_min = 0.*cm,    RadioA_max = 2.*cm;
 G4double RadioB_min = 0.*cm,    RadioB_max = 0.*cm;
 G4double Altura     = 1.*cm;
 G4double Angulo_min = 0.,       Angulo_max = twopi;

 G4Cons* completoCono = new G4Cons(
 "Cono_Compl",    //--| Nombre Vol. Solido
 RadioA_min,      //--| Radio min. extremo A
 RadioA_max,      //--| Radio max. extremo A
 RadioB_min,      //--| Radio min. extremo B
 RadioB_max,      //--| Radio max. extremo B
 Altura,          //--| Altura
 Angulo_min,      //--| Angulo Inicial
 Angulo_max);     //--| Angulo Final
...
...
}
```

Por último, un cono abierto tiene un corte que lo atraviesa de extremo a extremo y penetra hasta su centro. Para obtenerlo solo es necesario modificar los valores inicial y final del ángulo. El código siguiente es un ejemplo y su correspondiente imagen se muestra en la figura 1.2 (c).

```
//===| Metodo Construct() del archivo |===
//===|      DetectorConstruction.cc     |===
```

```
#include "DetectorConstruction.hh"
#include "G4Cons.hh"
...
...
G4VPhysicalVolume* DetectorConstruction::Construct()
{
...
...
//===| Cono Abierto |===
 G4double RadioA_min = 1.*cm,    RadioA_max = 2.*cm;
 G4double RadioB_min = 0.5*cm,   RadioB_max = 1.*cm;
 G4double Altura     = 1.*cm;
 G4double Angulo_min = 0.,       Angulo_max = 0.8*twopi;

 G4Cons* AbiertoCono = new G4Cons(
 "Cono_Abie",    //--| Nombre Vol. Solido
 RadioA_min,     //--| Radio min. extremo A
 RadioA_max,     //--| Radio max. extremo A
 RadioB_min,     //--| Radio min. extremo B
 RadioB_max,     //--| Radio max. extremo B
 Altura,         //--| Altura
 Angulo_min,     //--| Angulo Inicial
 Angulo_max);    //--| Angulo Final
...
...
}
```



(a) Cono truncado. (b) Cono Completo. (c) Cono Abierto.

Figura 1.2: Sólidos que se pueden construir en Geant4: conos.

Poliedros: Son implementados por `G4Polyhedra.hh`. Un poliedro es una serie de planos que se unen para formar un sólido. Todos los planos tienen un número fijo de caras, un radio interno y un radio externo. Por ejemplo, un plano con 3 caras es un triángulo, pero uno con 5 es un pentágono. Aquí se habla de caras y no de lados porque en un poliedro todos los planos tienen el mismo número de lados y, cuando se unen, esos lados se convierten en las caras del poliedro. Para obtener un poliedro cuadrado todos los planos deben tener cuatro caras, en cambio, si los planos tienen diez lados entonces el poliedro tendrá diez caras.

Para declarar un poliedro se necesitan 5 datos: nombre del sólido, ángulo inicial y final, número de caras, número de planos y 3 arrays. Los *arrays* [1] son un tipo de dato derivado o estructurado que agrupa de forma ordenada otros tipos de datos, como son: `int`, `float`, `double`, `char` u objetos de una clase ya declarada [10, secciones 2.2.5 y 7.3]. Los arrays que se necesitan para declarar un poliedro manejan información sobre sus planos: un array para la posición de cada plano, un segundo array para el tamaño del radio interno de cada plano y un tercer array para el tamaño del radio externo. El tamaño de los tres arrays es igual al número de planos que forman al poliedro. Es importante tener orden al escribir los arrays porque ese orden influye en la construcción del poliedro. El primer elemento de cada array corresponde a los datos del primer plano, el segundo elemento a los valores del segundo plano, y así hasta el último elemento o plano.

Los ejemplos que se muestran en la figura 1.3 muestran los diferentes tipos de poliedros que se pueden declarar. Por ejemplo, la figura 1.3 (a) muestra un poliedro de 5 planos y 10 caras con un hueco a lo largo del eje Z. El tamaño del hueco no es constante porque en algunos puntos mide 0.5 cm y en otros 2.5 cm. El código para ese poliedro se muestra a continuación.

```
//===| Metodo Construct() del archivo |===
//===|      DetectorConstruction.cc     |===

#include "DetectorConstruction.hh"
#include "G4Polyhedra.hh"
...
...
G4VPhysicalVolume* DetectorConstruction::Construct()
{
...
...
//===| Poliedro Hueco |===
 G4double Angulo_min = 0;
 G4double Angulo_max = twopi;
 G4int Caras         = 10;   //--| No. de Caras
 G4int Planos        = 5;    //--| No. de Planos

//==| Arrays con Radio interno, radio externo |==
//==|  y posicion de cada plano del poliedro  |==
 const G4double Posicion[Planos] =
                      {4.*cm, 2*cm, 0.*cm, -2.*cm, -4.*cm };
 const G4double Radio_min[Planos] =
                      {1.*cm, 0.5*cm, 0.4*cm, 2.*cm, 2.5*cm};
 const G4double Radio_max[Planos] =
                      {2*cm, 1.5*cm, 1.5*cm, 2*cm, 3*cm};

 G4Polyhedra* PolyHueco = new G4Polyhedra(
```

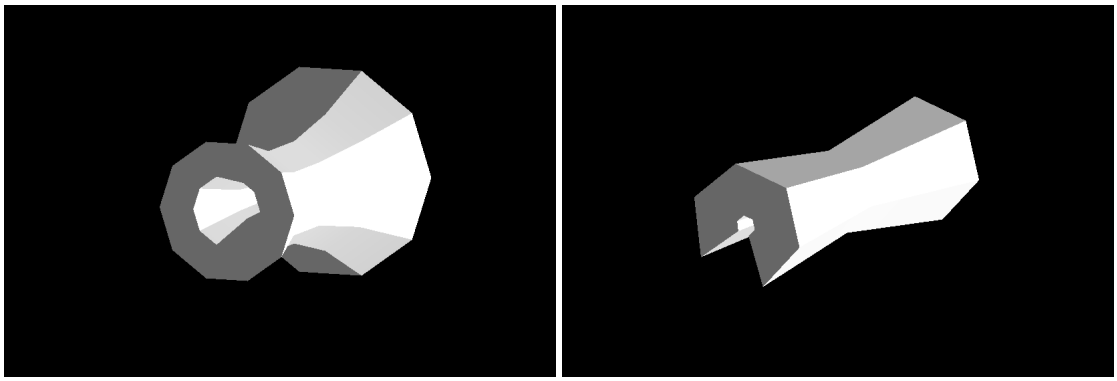
[1] La traducción de array en español es *colección*, en decir, una lista de objetos. Se mantiene el nombre en inglés porque el término es de uso común en español, al menos en textos de lenguaje C. Cambiar la palabra *array* por *colección* podría crear confusión en caso de que el lector busque información.

```
 "Poly_H",    //--| Nombre Vol. Solido
 Angulo_min,  //--| Angulo Inicial
 Angulo_max,  //--| Angulo Final
 Caras,       //--| No. Caras
 Planos,      //--| No. de Planos
 Posicion,    //--| Posicion Planos
 Radio_min,   //--| Radios Internos
 Radio_max);  //--| Radios Externos
...
...
}
```

El siguiente ejemplo es un poliedro hueco y abierto con 3 planos y 5 caras. Es abierto porque no se cierra sobre si mismo, como se muestra en la figura 1.3 (b).

```
//===| Metodo Construct() del archivo |===
//===|      DetectorConstruction.cc    |===

#include "DetectorConstruction.hh"
#include "G4Polyhedra.hh"
...
...
G4VPhysicalVolume* DetectorConstruction::Construct()
{
...
...
//===| Poliedro Abierto |===
 G4double Angulo_min = -0.1*twopi;
 G4double Angulo_max = 0.8*twopi;
 G4int Caras         = 5;  //--| No. de Caras
 G4int Planos        = 3;  //--| No. de Planos

//==| Arrays con Radio interno, radio externo |==
//==|  y posicion de cada plano del poliedro  |==
 const G4double Posicion[Planos] = {10*cm, 0.*cm, -10.*cm };
 const G4double Radio_min[Planos] = {0.5*cm, 0.2*cm, 0.5*cm};
 const G4double Radio_max[Planos] = {3*cm, 2*cm, 3*cm};

 G4Polyhedra* AbiertoPoly = new G4Polyhedra(
 "Poly_A",    //--| Nombre Vol. Solido
 Angulo_min,  //--| Angulo Inicial
 Angulo_max,  //--| Angulo Final
 Caras,       //--| No. Caras
 Planos,      //--| No. de Planos
 Posicion,    //--| Posicion Planos
 Radio_min,   //--| Radios Internos
 Radio_max);  //--| Radios Externos
...
...
```

```
}
```



(a) Poliedro Hueco. (b) Poliedro Abierto.

Figura 1.3: Sólidos que se pueden construir en Geant4: poliedros.

En el manual de Geant4, [6, capítulo 4], se ofrece más información sobre el resto de geometrías disponibles y su implementación.

Además de sólidos, Geant4 ofrece la opción de usar modelos CAD (Computer-Aided Design). Los programas de diseño CAD permiten hacer representaciones detalladas de cuerpos geométricos en 2-D y 3-D, algo difícil de lograr usando solo Geant4. Estas herramientas son útiles cuando los detectores tienen varias piezas o cuando es necesario representar estructuras naturales. Para utilizar modelos CAD es necesario incorporar a Geant4 una clase llamada `CADMesh` la cual trabaja con la librería `VCGLIB`. El código fuente de la clase `CADMesh` se puede descargar en la página `https://code.google.com/p/cadmesh/`. En el siguiente ejemplo se agrega un sólido CAD desde el archivo `SolidoCAD.stl`.

```
//===| Metodo Construct() de la clase |===
//===|       DetectorConstruction      |===

#include "DetectorConstruction.hh"
#include "CADMesh.hh"
...
...
G4VPhysicalVolume* DetectorConstruction::Construct()
{
...
...

//===| Se agrega el solido CAD |===
 CADMesh* mesh = new CADMesh("SolidoCAD.stl");

//===| Escala del solido |===
```

```
 mesh->SetScale(10*cm);
 mesh->SetReverse(false);
 mesh->SetReverse(false);

//===| El solido CAD se convierte en un |==
//===| solido identificable por Geant4  |=
 G4VSolid* CAD = mesh->TessellatedMesh();

 G4LogicalVolume* CADLV= new G4LogicalVolume(
 CAD,                //--| Vol. Solido
 CAD_Material,       //--| Material del CAD
 "CAD_LV",           //--| Nombre Vol. Logico
 0 ,0 ,0 );

 new G4PVPlacement(
 0,                  //--| Ninguna Rotation
 G4ThreeVector(),    //--| Centrado en (0,0,0)
 CADLV,              //--| Vol. Logico
 "CAD_PV",           //--| Nombre Vol. Fisico
 World,              //--| Volumen Madre
 false,              //--| Sin Operaciones Booleanas
 0,                  //--| No. de copias
 true);              //--| Verificar Sobreposicion con otros Solidos
...
...
}
```

Para mayor información sobre `CADMesh` se puede consultar [11].

1.1.4. Partículas y Materiales.

Partículas.

En Geant4 las partículas se dividen en grupos: bariones, iones, leptones, mesones y quarks. Además, hay que sumar otras cuatro partículas que no pertenecen a ninguna de las categorías anteriores: gamma, geantino, opticalphoton y chargedgeantino. No es posible poner en este trabajo una tabla con todas las partículas que permite declarar Geant4, la cantidad es enorme, pero sí es posible mostrar algunas y mencionar dónde encontrar listas completas para cada clasificación.

Bariones.

Los bariones pertenecen a la familia de los hadrones, eso significa que participan en procesos de interacción fuerte. Tienen estructura interna porque están formados por tres quarks, además satisfacen la estadística de Fermi-Dirac [12]. La lista de bariones que ofrece Geant4 se puede consultar en su página electrónica (https://geant4.web.cern.ch). En la tabla 1.1 se muestran algunas partículas de esta clasificación.

Tabla 1.1: *Algunos bariones presentes en Geant4 con sus respectivas antipartículas.*

Nombre	Masa $\left(\mathrm{GeV/c^2}\right)$	Carga	Antipartícula
proton	$9.3827x10^{-1}$	1.0	anti_proton
neutron	$9.3956x10^{-1}$	0.0	anti_neutron
delta+	1.2320	1.0	anti_delta+
omega−	1.6724	−1.0	anti_omega−
sigma0	1.1926	0.0	anti_sigma0
sigma+	1.1893	1.0	anti_sigma+

Nota: La carga eléctrica de las partículas está en múltiplos de la carga fundamental e, y c es la velocidad de la luz. Las columnas *Nombre* y *Antipartícula* muestran el nombre asignado a la partícula en Geant4. Tabla adaptada de la página https://geant4.web.cern.ch.

Leptones.
Es un pequeño pero importante grupo de partículas. Solo se conocen seis leptones con sus correspondientes antipartículas y ninguna es afectada por la fuerza fuerte, pero sí participan en procesos de interacción débil y electromagnética [12]. Junto a los quarks son las únicas partículas conocidas que no tienen estructura interna. La lista de leptones que ofrece Geant4 se puede consultar en la página electrónica https://geant4.web.cern.ch.

Tabla 1.2: *Leptones presentes en Geant4 con sus respectivas antipartículas.*

Nombre	Masa $\left(\mathrm{GeV/c^2}\right)$	Carga	Antipartícula
e−	$5.1099x10^{-4}$	−1.0	e+
mu−	$1.0565x10^{-1}$	−1.0	mu+
tau−	1.7768	−1.0	tau+
nu_e	0.0*	0.0	anti_nu_e
nu_mu	0.0*	0.0	anti_nu_mu
nu_tau	0.0*	0.0	anti_nu_tau

Nota: La carga eléctrica de las partículas está en múltiplos de la carga fundamental e, y c es la velocidad de la luz. Las columnas *Nombre* y *Antipartícula* muestran el nombre asignado a la partícula dentro de Geant4. Tabla adaptada de la página https://geant4.web.cern.ch.

* Geant4 considera a los neutrinos como partículas sin masa, sin embargo, hoy se sabe que los neutrinos sí tienen masa. Los experimentos no han logrado determinar una masa exacta pero se tienen límites, por ejemplo, se sabe que la masa del antineutrino del electrón ($\overline{\nu}_e$) debe ser menor a 10 $\mathrm{eV/c^2}$ [13].

Mesones.
Al igual que los bariones, los mesones son hadrones. Participan en procesos de interacción fuerte y están conformados por un quark y un antiquark. A diferencia de los bariones, los mesones satisfacen la estadística de Bose-Einstein [12]. La lista correspondiente de esta categoría se ubica en la dirección electrónica de Geant4 (https://geant4.web.cern.ch)

Tabla 1.3: *Algunos mesones presentes en Geant4 y sus antipartículas.*

Nombre	Masa $\left(\mathrm{GeV/c^2}\right)$	Carga	Antipartícula
pi0	$1.3497x10^{-1}$	0.0	-
pi+	$1.3957x10^{-1}$	1.0	pi−
kaon+	$4.9367x10^{-1}$	1.0	kaon−
kaon0	$4.9761x10^{-1}$	0.0	anti_kaon0
kaon0S	$4.9761x10^{-1}$	0.0	-
eta	$5.4786x10^{-1}$	0.0	-

Nota: La carga eléctrica de las partículas está en múltiplos de la carga fundamental e, mientras c es la velocidad de la luz. Las columnas *Nombre* y *Antipartícula* muestran el nombre asignado a la partícula dentro de Geant4. Tabla adaptada de la página https://geant4.web.cern.ch.

Quarks.
Los quarks integran a casi todas las partículas: por un lado los mesones tienen un quark y un antiquark, y por el otro los bariones tienen tres quarks. Junto a los leptones son consideradas partículas elementales por carecer de estructura interna. Los nombres que asigna Geant4 a cada quark y sus propiedades se puede consultar en la página electrónica de Geant4 (https://geant4.web.cern.ch). En la tabla 1.4 se muestra una lista incompleta de los quarks que se pueden simular.

Tabla 1.4: *Algunos quarks presentes en Geant4 con sus respectivas antipartículas.*

Nombre	Masa $\left(\mathrm{GeV/c^2}\right)$	Carga	Antipartícula
b_quark	4.1800	−0.33	anti_b_quark
t_quark	$1.7321x10^{2}$	0.66	anti_t_quark
d_quark	$4.8000x10^{-3}$	−0.33	anti_d_quark
gluon	0.0	0.0	-
u_quark	$2.3000x10^{-3}$	0.66	anti_u_quark
ud0_diquark	$7.1000x10^{-3}$	0.33	anti_ud0_diquark
ud1_diquark	$7.0000x10^{-3}$	0.33	anti_ud1_diquark

Nota: La carga eléctrica de las partículas está en múltiplos de la carga fundamental e, mientras c es la velocidad de la luz. Las columnas *Nombre* y *Antipartícula* muestran el nombre asignado a la partícula en Geant4. Tabla adaptada de la página https://geant4.web.cern.ch.

Otras.
Cuatro partículas que no pertenecen a las otras categorías han sido colocadas en la tabla 1.5. Dos de estas partículas son el Geantino y el Geantino Cargado, las cuales son partículas de prueba que no existen en la naturaleza. Ambas tienen masa cero y no decaen, no presentan ningún tipo de interacción a excepción del Geantino Cargado que tiene una carga eléctrica igual a la carga del electrón. La lista correspondiente de esta clasificación se puede encontrar en la página https://geant4.web.cern.ch.

Tabla 1.5: *Otras partículas presentes en Geant4.*

Partículas	Nombre en Geant4	Masa $\left(\mathrm{GeV/c^2}\right)$	Carga	Antipartículas
Geantino	geantino	0.0	0.0	-
Rayo Gamma	gamma	0.0	0.0	-
Fotón	opticalphoton	0.0	0.0	-
Geantino Cargado	chargedgeantino	0.0	1.0	-

Nota: La carga eléctrica de las partículas está en múltiplos de la carga fundamental e, con c es la velocidad de la luz. Las columnas *Nombre* y *Antipartícula* muestran el nombre asignado a la partícula en Geant4. Tabla adaptada de la página https://geant4.web.cern.ch.

Materiales.

La materia no solo es forma, también tiene estructura. Cuando se habló de los sólidos, en la sección 1.1.3, solo se mencionó su forma y nada se dijo sobre su estructura. La estructura de la materia viene representada en Geant4 por los materiales, siendo la lista bastante amplia. En caso de que el material no esté declarado, se puede definir y agregar para usarlo en Geant4. La lista completa de materiales se puede consultar en [6, apéndice 6], aquí solo se muestra una lista resumida.

Elementos químicos.

Hasta la versión 10.1 de Geant4 solo se incluyen los primeros 98 elementos químicos. La sintaxis para declarar un elemento en Geant4 es simple: `G4_[Símbolo Elemento químico]`. La tabla 1.6 muestra algunos elementos químicos que se pueden encontrar en la tabla del apéndice 6.1 de [6].

Tabla 1.6: *Algunos elementos químicos presentes en Geant4.*

Número atómico	Nombre en Geant4	Densidad $\left(\mathrm{g/cm^3}\right)$
1	G4_H	$8.3748x10^{-5}$
2	G4_He	$1.6632x10^{-4}$
3	G4_Li	0.534
4	G4_Be	1.848
5	G4_B	2.37
26	G4_Fe	7.874
79	G4_Au	19.32
92	G4_U	18.95
98	G4_Cf	10

Nota: Adaptado de la tabla del apéndice 6.1 (p. 362) ubicada en [6].

Tabla 1.7: *Algunos materiales bioquímicos que se pueden declarar en Geant4.*

Nombre en Geant4	Densidad $\left(\mathrm{g/cm^3}\right)$
G4_CYTOSINE	1.55
G4_THYMINE	1.23
G4_URACIL	1.32
G4_DNA_CYTOSINE	1
G4_DNA_THYMINE	1
G4_DNA_MONOPHOSPHATE	1

Nota: Adaptado de la tabla del apéndice 6.5 (p. 374) ubicada en [6].

Materiales de NIST.
NIST significa *National Institute of Standards and Technology*. Fue creado en 1901 y en la actualidad es parte del Departamento de Comercio de EUA, su objetivo es dar servicios de calibración a la industria y desarrollar normas, o como ellos afirman en su sitio electrónico (www.nist.gov): "Fomentar la innovación y la competitividad industrial [...] a través de avances en metrología, normas y tecnología con el fin de mejorar la seguridad económica y la calidad de vida."

Geant4 posee una lista de materiales extraída de la base de datos de NIST. Esta lista se divide en cuatro categorías: materiales bioquímicos, materiales compuestos, materiales espaciales y materiales nucleares.

Los **materiales bioquímicos** se ubican en la tabla del apéndice 6.5 de [6] e incluye cerca de veinte materiales. En la tabla 1.7 se muestran algunos.

Los **materiales compuestos** se ubican en la tabla del apéndice 6.2 de [6] la cual agrupa más de cien materiales. La lista incluye materiales comunes como el aire, pero también materiales enfocados a la esfera médica como tejido y hueso, o incluso líquidos de uso corriente en laboratorio como la acetona. La tabla 1.8 ofrece una ventana a este gran catálogo.

Tabla 1.8: *Algunos materiales compuestos que se pueden declarar en Geant4.*

Nombre en Geant4	Densidad $(\mathrm{g/cm^3})$
G4_ACETONE	0.7899
G4_ADIPOSE_TISSUE_ICRP	0.95
G4_AIR	0.00120479
G4_ALUMINUM_OXIDE	3.97
G4_BONE_CORTICAL_ICRP	1.92
G4_BRAIN_ICRP	1.04
G4_CONCRETE	2.3
G4_FERROUS_OXIDE	5.7
G4_GRAPHITE	2.21
G4_GLYCEROL	1.2613
G4_WATER_VAPOR	0.000756182

Nota: Adaptado de la tabla del apéndice 6.2 (p. 363) ubicada en [6].

Tabla 1.9: *Algunos materiales nucleares y espaciales que se pueden declarar en Geant4.*

Nombre en Geant4	Densidad $(\mathrm{g/cm^3})$
G4_lH2	0.0708
G4_lAr	1.396
G4_Galactic	10^{-25}
G4_BRONZE	8.82
G4_STAINLESS-STEEL	8
G4_CR39	1.32
G4_OCTADECANOL	0.812
G4_KEVLAR	1.44
G4_DACRON	1.4
G4_NEOPRENE	1.23

Nota: Adaptado de la tabla del apéndice 6.3 (p. 373) y de la tabla del apéndice 6.4 (p. 374) ubicada en [6].

En lo que se refiere a **materiales espaciales** y **materiales nucleares**, la lista es pequeña. La lista de materiales espaciales se ubica en el apéndice 6.4 de [6] y solo tiene 3 materiales, mientras la lista de materiales nucleares se ubica en el apéndice 6.3 y ofrece 16 materiales.

Declaración de materiales.
Si el material que nos interesa no lo ofrece Geant4, se tiene la opción de definirlo. Se pueden declarar elementos químicos que, si bien Geant4 los tiene definidos, tal vez el que nos interesa difiere en la densidad. Por ejemplo, el argón definido en Geant4 es un gas con densidad igual a 0.001662 g/cm^3, si nos interesa tener argón liquido con una densidad de 1.390 g/cm^3 entonces hay que decirle a Geant4 que lo defina para usarlo posteriormente. En el siguiente código se muestra cómo declarar nuevos elementos químicos o un material conformado por dos o más elementos. En la primer parte del código se declaran dos elementos químicos, sodio y oxígeno. Después, éstos se usan para definir el oxido de sodio, Na_2O.

```
//===| Metodo DefineMaterials() de la clase |===
//===|          DetectorConstruction         |===

void  DetectorConstruction::DefineMaterials()
{
 G4double a;              //--| Masa Molar
 G4double z;              //--| Numero de Protones
 G4double densidad;
 G4double Composicion;    //--| Composicion Porcentual
 G4int natomos;           //--| Numero de Atomos
 G4int ncomponentes;      //--| Numero de Elementos

//===| Nuevos elementos quimicos |===
 a= 22.989*g/mole;        //--| Masa Molar

 G4Element* g4Na = new G4Element(
 "Sodio",     //--| Nombre
 "Na",        //--| Simbolo
 z=11.,       //--| Protones
 a);          //--| Masa Molar

 a = 15.999*g/mole;       //--| Masa Molar

 G4Element* g4O = new G4Element(
 "Oxigeno",   //--| Nombre
 "O",         //--| Simbolo
 z=8.,        //--| Protones
 a);          //--| Masa Molar

//===| Nuevo compuesto |===
 densidad = 2.27*g/cm3;

 G4Material* g4Na2O = new G4Material(
 "Oxido-Sodio",     //--| Nombre
 densidad,          //--| Densidad
 ncomponentes=2);   //--| No. de Elementos

//===| Se asignan elementos a g4Na2O |===
 g4Na2O->AddElement(
```

```
  g4Na,          //--| Elementos
  natomos=2);    //--| No. de atomos

  g4Na2O->AddElement(
  g4O,           //--| Elementos
  natomos=1);    //--| No. de atomos
}
```

1.2. Construyendo una simulación en Geant4.

En esta segunda sección se presentan ejemplos y código. A diferencia de la sección 1.1 donde se habló de forma general sobre Geant4, en ésta se pretende explicar al lector cómo escribir el código de una simulación. Se vuelven a tocar los temas de los cuerpos sólidos (sección 1.2.4) y las partículas (sección 1.2.5) que simula Geant4, con el objetivo de mostrar los pasos que se deben seguir para su implementación. Por otro lado, en las secciones 1.2.1 y 1.2.2 se tocan dos temas básicos pero bastante útiles. En el primero se habla sobre los tipos de datos que usa Geant4, es decir, las variables que guardan números enteros, números decimales o cadenas de caracteres. El segundo tema es sobre las constantes que posee Geant4 y cómo puede el usuario hacer uso de ellas. Las constantes son muy útiles porque definen valores de uso corriente en físicas y matemáticas, ejemplos de estas constantes son: la velocidad de la luz, la constante de Planck y π.

En 1.2.3 se habla sobre cómo se debe escribir una simulación sencilla o básica. Se muestran los diferentes archivos que integran a una simulación con el fin de que el lector conozca su propósito. Dentro de estos archivos destacan dos por su importancia y por eso tienen su propia sección, el primero es `PhysicsList` (sección 1.2.6) y el segundo es `ActionInitialization` (sección 1.2.10). En `PhysicsList` se define la física de la simulación, es decir, que interacciones tomará en cuenta Geant4 a lo largo de ésta. Por su parte, `ActionInitialization` maneja la implementación de casi todas las clases y es la responsable de controlar la simulación multi-hilos. En la sección 1.2.8 se explica con más detalle este tema.

En las secciones 1.2.7 y 1.2.9 se explica cómo Geant4 guarda información durante la simulación y cómo ésta puede ser almacenada en archivos, a fin de usarla posteriormente en programas de análisis de datos. Por último, en 1.2.11 se detalla sobre la creación del archivo ejecutable.

1.2.1. Tipos de datos.

C++ cuenta con sus propios tipos de datos, como son: `char`, `int`, `double`, `float`, entre otros. Sin embargo, aunque C++ asigna a estas variables un tamaño de memoria fijo, éste puede variar de computadora a computadora dependiendo de su arquitectura y del compilador [6, 10]. Por esa razón Geant4 cuenta con sus propios tipos de datos facilitando la portabilidad del código al estandarizar el tamaño de memoria [6]. En la tabla 1.10 se muestran estos tipos de datos.

Tabla 1.10: Tipos de datos en Geant4.

Tipo de Dato	Nombre en Geant4	Tipo de Dato	Nombre en Geant4
Entero	G4int	Booleano	G4bool
Flotante	G4float	Complejo	G4complex
Double	G4double	Cadena Caracteres	G4String

Nota: La columna *Tipo de Dato* indica el nombre común por el que se conoce a ese tipo de dato, mientras *Nombre en Geant4* es la etiqueta que se debe escribir en el código para declarar ese tipo de dato en Geant4.

Para trabajar con ellos se debe incluir el archivo `globals.hh`, escribiendo el siguiente código:

```
#include "globals.hh"
```

Geant4 permite declarar tipos de datos de la librería CLHEP (Computing Library for High Energy Physics), el cual es un proyecto que provee herramientas de cómputo para física de altas energías. Los tipos de datos que se utilizan en este trabajo y provienen de la librería CLHEP son:

`G4ThreeVector:` Vector con tres componentes (x, y, z).

`G4RotationMatrix :` Matriz $(3x3)$ de Rotación.

Para usarlos es necesario incluir el archivo cabecera correspondiente. Por ejemplo, para trabajar con un objeto de la clase `G4ThreeVector` se debe escribir el siguiente código:

```
#include "G4ThreeVector.hh"
```

Mientras para la clase `G4RotationMatrix` el código es:

```
#include "G4RotationMatrix.hh"
```

1.2.2. Constantes y unidades.

Geant4 permite manejar constantes y unidades de uso común en física y matemáticas: el valor de π, unidades de distancia, la velocidad de la luz o unidades de energía. Las constantes y unidades están definidas en dos archivos:

`G4SystemOfUnits.hh:` Define constantes matemáticas y unidades físicas de área, distancia, energía, tiempo, etc.

`G4PhysicalConstants.hh:` Define constantes de uso común en física, como la velocidad de la luz o la constante de Planck.

Ambos archivos se ubican en el directorio:

`[DirectorioGeant4]/source/externals/clhep/include/CLHEP/Units.`

La expresión `[DirectorioGeant4]` hace referencia al directorio donde se descomprimió Geant4, que no es igual al directorio donde se instaló. Para poder usar las constantes y unidades es importante incluir ambos archivos escribiendo el código;

```
#include "G4SystemOfUnits.hh"
#include "G4PhysicalConstants.hh"
```

Las constantes matemáticas que define Geant4 son pocas y todas están relacionadas con π, tal como se puede observar en la tabla 1.11.

Tabla 1.11: Constantes matemáticas definidas en Geant4.

Nombre en Geant4	Valor	Nombre en Geant4	valor
halfpi	$\pi/2$	twopi	2π
pi	π	pi2	π^2

En cuanto a las constantes físicas, el archivo `G4PhysicalConstants.hh` define cerca de treinta constantes, algunas de ellas se muestran en la tabla 1.12.

Tabla 1.12: Algunas constantes físicas definidas en Geant4.

Nombre en Geant4	Constante	Valor
Avogadro	N_A	$6.02214179x10^{23}$ mol^{-1}
c_light	c	$2.99792458x10^{8}$ m/s
electron_mass_c2	masa electrón	0.510998910 MeV
h_Planck	$\hbar$	$6.62606896x10^{-34}$ joule $\cdot$ s
proton_mass_c2	masa protón	938.272013 MeV
STP_Temperature	$T_{estándar}$	273.15° kelvin
universe_mean_density	-	10^{-25} g/cm^3

Nota: La columna *Valor* muestra el valor que tiene la constante en el archivo `G4PhysicalConstants.hh`.

El archivo `G4SystemOfUnits.hh` define poco más de veinte unidades, por mencionar algunas: de longitud, tiempo, carga eléctrica, energía, masa, fuerza, presión, corriente eléctrica, resistencia eléctrica y temperatura.

1.2.3. Estructura de una simulación.

Una simulación sencilla en Geant4 se compone de 6 clases, clases del paradigma de programación orientado a objetos. Por cada clase se necesitan dos archivos: un archivo con la extensión `.hh` para declarar los métodos de la clase, y otro con la extensión `.cc` para implementar los métodos. El nombre de ambos archivos debe coincidir con el nombre de la clase que definen. Por ejemplo, si se define la clase `G4Solid` se debe tener un archivo llamado `G4Solid.hh` y un archivo `G4Solid.cc`.

A continuación se muestran las 6 clases necesarias para un programa sencillo en Geant4. Los nombres son a modo de ejemplo y el usuario puede cambiarlos por otros mas apropiados.

DetectorConstruction: Define todos los sólidos y, de éstos, cuales van a operar como detectores (sección 1.2.4).

PrimaryGeneratorAction: Define todas las partículas a simular (sección 1.2.5).

PhysicsList: Se declaran los modelos y procesos físicos a tomar en cuenta durante la interacción entre partículas y materia (sección 1.2.6).

RunLocal: Se encarga de guardar los datos relevantes generados durante la simulación.

RunAction: Al concluir la simulación esta clase agrupa toda la información guardada y la muestra al usuario.

ActionInitialization: Implementa todas las clases excepto `DetectorConstruction` y `PhysicsList`. Esta clase es relevante cuando se utiliza multi-hilos (sección 1.2.10).

Además de los 12 archivos anteriores se necesitan dos más;

Ejecutable.cc: El archivo principal encargado de implementar a todas las clases (sección 1.2.11).

CMakeLists.txt: Aunque todos los archivos se escriben en lenguaje C++ es recomendable compilar con `CMAKE` (sección 1.2.11). Este archivo le indica a `CMAKE` como ligar todas las clases y generar el archivo ejecutable.

La simulación de un detector real seguramente tendrá más clases y por lo tanto más archivos, tantos como el usuario necesite. Ejemplos de simulaciones sencillas se pueden encontrar en la carpeta de ejemplos básicos, ubicada en,

`[CarpetaInstacionGeant4]/share/Geant4-10.1.2/examples/basic`

La expresión `[CarpetaInstacionGeant4]` hace referencia a la carpeta donde se instalo Geant4. En este trabajo se está usando la versión 10.1.2.

1.2.4. Declaración de Sólidos.

Anteriormente se tocó el tema de los sólidos, en la sección 1.1.3, y se mencionaron algunas figuras geométricas que se pueden declarar en Geant4. Aquí se vuelve a tocar el tema pero desde una perspectiva diferente, pues el objetivo es explicar cómo implementarlos.

Volúmenes físicos, lógicos y sólidos.

Geant4 separa los atributos de los sólidos (composición, forma y posición) en tres volúmenes diferentes: el primer volumen define la forma del sólido; el segundo volumen indica su composición y el tercero se encarga de la posición. Estos volúmenes se conocen respectivamente como volumen sólido, volumen lógico y volumen físico [6]. Así, el volumen sólido es la forma que tendrá el cuerpo: una esfera, un cubo, una pirámide o un volumen irregular. El volumen lógico es el volumen sólido más el material del que está hecho: hierro, oro, hueso, entre otros. En cambio, el volumen físico es el volumen lógico más la posición del sólido.

El sólido World.

Un sólido destaca del resto porqué su implementación es obligatoria, se trata del sólido `World`. El sólido `World` es el más grande de la simulación y dentro de él existen todos los demás, por eso es el primer sólido que debe declararse. Al ser `World` un sólido obligatorio que debe existir en toda simulación, se recomienda que éste sea un sólido sencillo, un sólido de la clase `G4Box` por ejemplo.

El Volumen Madre y la posición del sólido.

En Geant4 un volumen madre es un volumen que encapsula a otro. Encapsular no quiere decir exclusivamente que el volumen madre sea una envoltura que encierra a otro volumen, significa que el volumen madre es el universo del sólido que encierra: un sólido solo existe dentro de su volumen madre y no fuera de él. Cualquier sólido puede ser el volumen madre de otro, pero `World` siempre será el volumen madre de todos: `World` es el universo de toda la simulación.

Para aclarar conceptos, un ejemplo. Imagine que se tienen tres sólidos: una caja, un cono y una esfera. La esfera está encerrada en el cono, de tal forma que no puede salir de él. El cono, a su vez, está contenido en la caja y tampoco puede salir de ésta. Esta misma configuración en Geant4 sería así: el volumen madre de la esfera es el cono; el volumen madre del cono es la caja; la caja es el volumen `World`. Observe como hay más de un volumen madre, y sin embargo `World` contiene a todos.

Otro aspecto importante en el que está involucrado el concepto de volumen madre es la posición del sólido. Se dijo que el volumen físico es el responsable de la posición del sólido y, como bien se sabe, toda posición necesita un punto de referencia. El punto de referencia del volumen físico siempre es el centro del volumen madre, es a partir de éste que el volumen físico asigna un lugar al sólido.

Para registrar la posición del sólido en necesario definir un vector de la clase `G4ThreeVector`. La clase `G4ThreeVector` tiene constructores para coordenadas cartesianas, cilíndricas y polares. Una vez que se tiene un vector de posición éste se debe agregar al constructor del volumen físico. Con el fin de comprender mejor todos estos conceptos, el siguiente código muestra su aplicación a través de un ejemplo.

Se declara un sólido `World` en forma de caja con medidas: 10 cm $\times$ 10 cm $\times$ 15 cm. El material de

`World` es *vacío* para evitar que las partículas modifiquen sus propiedades debido a interacciones que puedan ocurrir cuando viajen por su interior. En el interior de `World` se coloca un sólido de la clase `G4Polyhedra` que sirve como detector. Un detector debe registrar ciertas propiedades de las partículas, sin modificarlas, en el momento en que éstas lo cruzan. Por esa razón, el detector tipo `G4Polyhedra` también está hecho de vacío. El volumen madre del detector es `World`.

```
//==| Archivo DetectorConstruction.cc |==
#include "DetectorConstruction.hh"

#include "G4Box.hh"              //--| Vol. Solido (Paralelepipedo)
#include "G4Polyhedra.hh"        //--| Volumen Solido (Poliedro)
#include "G4LogicalVolume.hh"    //--| Volumen Logico
#include "G4PVPlacement.hh"      //--| Volumen Fisico

#include "G4NistManager.hh"      //--| Materiales

#include "G4SystemOfUnits.hh"
#include "G4PhysicalConstants.hh"
#include "G4Colour.hh"           //--| Color para solidos
#include "G4VisAttributes.hh"
#include "globals.hh"            //--| Tipos Datos Geant4

//==| Metodo Construct() |==
G4VPhysicalVolume* DetectorConstruction::Construct() {

//===| Materiales |===
G4NistManager* find=G4NistManager::Instance();
//===| Material: Vacio |===
G4Material* vacuum = find->FindOrBuildMaterial("G4_Galactic");

//===| Dimensiones World |===
G4double WorldSizeXY = 10.*cm;
G4double WorldSizeZ  = 15.*cm;

//===| Volumen Solido World |===
G4Box* SolidWorld = new G4Box(
"World",                 //--| Nombre Vol. Solido
0.5*WorldSizeXY,         //--| Tamano en X
0.5*WorldSizeXY,         //--| Tamano en Y
0.5*WorldSizeZ);         //--| Tamano en Z

//===| Volumen Logico World |===
G4LogicalVolume* LogicWorld = new G4LogicalVolume(
SolidWorld,              //--| Volumen Solido
vacuum,                  //--| Material de World
"World");                //--| Nombre Vol. Logico

//===| Volumen Fisico World |===
G4VPhysicalVolume* PhysWorld = new G4PVPlacement(
```

```
0,                  //--| Ninguna Rotacion
G4ThreeVector(),    //--| Centrado en su Vol. Madre
LogicWorld,         //--| Volumen Logico
"World",            //--| Nombre Vol. Fisico
0,                  //--| Volumen Madre
false,              //--| Ninguna Operacion Booleana
0,                  //--| No. Copias
true);              //--| Verificar Sobreposicion con otros Solidos

//===| Parametros Detector |===
G4double rInnerGap = 0.5*cm;
G4double Gap       = 3.0*cm;
G4double Phybegin  = 0;
G4double PhyFinish = twopi;
G4int    Sides     = 4;
G4int    ZPlanes   = 5;

G4double Zplane[ZPlanes] = {
2.*cm,              //--| Posicion Plano 1
1.5*cm,             //--| Posicion Plano 2
1.4*cm,             //--| Posicion Plano 3
-1.9*cm,            //--| Posicion Plano 4
-2.*cm };           //--| Posicion Plano 5

G4double rinner[ZPlanes] = {
rInnerGap,          //--| Radio Interior Plano 1
rInnerGap,          //--| Radio Interior Plano 2
Gap,                //--| Radio Interior Plano 3
Gap,                //--| Radio Interior Plano 4
0.};                //--| Radio Interior Plano 5

G4double router[ZPlanes] = {
1.05*rInnerGap,     //--| Radio Exterior Plano 1
1.05*rInnerGap,     //--| Radio Exterior Plano 2
1.05*Gap,           //--| Radio Exterior Plano 3
1.05*Gap,           //--| Radio Exterior Plano 4
1.05*Gap};          //--| Radio Exterior Plano 5

//===| Volumen Solido Detector |===
G4Polyhedra* Cover = new G4Polyhedra(
"Cover",    //--| Nombre Vol. Solido
Phybegin,   //--| Angulo Inicial
PhyFinish,  //--| Angulo Final
Sides,      //--| No. Caras
ZPlanes,    //--| No. Vertices en Eje Z
Zplane,     //--| Ubicacion Vertices
rinner,     //--| Radio Interior
router);    //--| Radio Exterior

//===| Volumen Logico Detector |===
```

```
G4LogicalVolume* fCoverLV = new G4LogicalVolume(
Cover,     //--| Volumen Solido
vacuum,    //--| Material Detector
"Cover"); //--| Nombre Vol. Logico

//===| Volumen Fisico Detector |===
new G4PVPlacement(
0,                 //--| Ninguna Rotacion
G4ThreeVector(), //--| Centrado en su Vol. Madre
fCoverLV,          //--| Volumen Logico
"Cover",           //--| Nombre Vol. Fisico
LogicWorld,        //--| Volumen Madre
false,             //--| Ninguna Operacion Booleana
0,                 //--| No. Copias
true);             //--| Verificar Sobreposicion con otros Solidos

//===| Color Detector |===
visAttributes = new G4VisAttributes(
G4Colour(0,1,0));                              //--| Verde
fCoverLV->SetVisAttributes(visAttributes); //--| Asigna Color

//===| Devuelve el Volumen Fisico World |===
  return PhysWorld;
 }
```

1.2.5. Declaración de partículas.

En la sección 1.1.4 se mencionó el tipo de partículas que se pueden simular en Geant4. Ahora el objetivo es declarar esa partícula que nos interesa y simularla.

En Geant4, una partícula es un objeto de la clase `G4VPrimaryGenerator`. En cada evento o simulación se puede generar una o varias partículas. Como todo objeto, una partícula posee atributos pero solo algunos de ellos se pueden modificar, tales como: la carga eléctrica, la energía cinética y los vectores de momento, de polarización y de posición. Para cambiar los atributos se debe usar alguno de los siguientes generadores de partículas:

`G4ParticleGun`**:** El generador mas sencillo de usar ya que sólo necesita de unos cuantos datos. Por lo general, es el generador que se usa en la mayoría de los ejemplos que ofrece Geant4 y tal vez sea el único que necesite saber usar. Su característica más destacable es la libertad que ofrece para implementar nuevos métodos.

`G4GeneralParticleSource`**:** Muy fácil de usar. Se maneja a través de archivos llamados macros que tienen la extensión `.mac`. Es un generador que ofrece varias opciones para simular partículas, ideal cuándo no se desea escribir código.

G4ParticleGun.

Desde `G4ParticleGun` se pueden modificar los siguientes atributos de la partícula, todo gracias a los métodos `Set` que se listan:

`SetParticleCharge`**:** Carga eléctrica.

`SetParticleEnergy`**:** Energía cinética.

`SetParticleMomentumDirection`**:** Vector de momento.

`SetParticlePolarization`**:** Vector de polarización.

`SetParticlePosition`**:** Vector de posición.

Aunque parecen pocos, en realidad son los únicos que realmente se necesitan en la mayoría de los casos. Nuevos métodos pueden ser implementados con este generador siempre y cuando se esté dispuesto a escribir el código. Por ejemplo, métodos que permitan generar partículas en ciertas regiones del volumen `World` o generar partículas que sigan una distribución de energía determinada.

En el siguiente ejemplo se muestra como usar los métodos de `G4ParticleGun`.

```
//=========================================
//===| Archivo PrimaryGeneratorAction.hh |===
//=========================================

#ifndef PrimaryGeneratorAction_h
#define PrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"
#include "G4ParticleGun.hh"
#include "globals.hh"

class G4ParticleGun;
class G4Event;

class PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{

 public:
 PrimaryGeneratorAction();
 virtual ~PrimaryGeneratorAction();

 virtual void GeneratePrimaries(G4Event*);

 private:

 G4ParticleGun* fParticleGun;
```

```
};
#endif
//...oooOO0OOooo...oooOO0OOooo...

//=========================================
//===| Archivo PrimaryGeneratorAction.cc |===
//=========================================

#include "PrimaryGeneratorAction.hh"

#include "G4Event.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"       //--| Particulas
#include "G4ParticleDefinition.hh" //--| Particulas

#include "G4ThreeVector.hh"         //--| Vectores
#include "G4SystemOfUnits.hh"
#include "G4PhysicalConstants.hh"

//===| Metodo Constructor |===
PrimaryGeneratorAction::PrimaryGeneratorAction():
G4VUserPrimaryGeneratorAction()
{
 G4int n_particle = 1;  //--| Numero Particulas por evento

//===| Generador de Particulas |===
 fParticleGun = new G4ParticleGun(n_particle);

//===| Particula a generar: Alfa |===
 G4ParticleDefinition* particleDefinition = G4ParticleTable::
            GetParticleTable()->FindParticle("alpha");

//===| Se agrega Particula alfa al generador |===
 fParticleGun->SetParticleDefinition(particleDefinition);

//===| Energia cinetica Particula alfa |===
 fParticleGun->SetParticleEnergy(5.0*MeV);

//===| Posicion Generacion Particula alfa |===
fParticleGun->SetParticlePosition(G4ThreeVector(0.,0.,0.5*cm));

//===| Vector de Momento Particula alfa |===
 fParticleGun->
    SetParticleMomentumDirection(G4ThreeVector(0.,0.,-1.));
}
//...oooOO0OOooo...oooOO0OOooo...

//===| Metodo Destructor |===
PrimaryGeneratorAction::~PrimaryGeneratorAction()
{
```

```
//===| Se elimina el puntero del generador de particulas |===
 delete fParticleGun;
}
//....oooOOOOOooo.....oooOOOOOooo.....

//===| Metodo GeneratePrimaries |===
void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
//===| Se genera un evento |===
 fParticleGun->GeneratePrimaryVertex(anEvent);
}
```

Es importante que al momento de escribir la clase `PrimaryGeneratorAction` se elimine el puntero al objeto `G4ParticleGun` en el destructor de la clase.

G4GeneralParticleSource.

El generador `G4GeneralParticleSource` crea flujos de partículas. Su gran ventaja es que el código del archivo `PrimaryGeneratorAction.cc` se reduce en extensión porque sólo se necesita declarar el puntero a la clase `G4GeneralParticleSource`, como se muestra en el siguiente ejemplo

```
//=============================================
//===|  Archivo PrimaryGeneratorAction.hh |===
//=============================================

#ifndef PrimaryGeneratorAction_h
#define PrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"
#include "globals.hh"

class G4Event;
class G4GeneralParticleSource;

class PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
 public:

 PrimaryGeneratorAction();
 ~PrimaryGeneratorAction();

 void GeneratePrimaries(G4Event*);

 private:
 G4GeneralParticleSource* fParticleSource;
};
#endif
//....oooOOOOOooo.....oooOOOOOooo.....
```

```
//=============================================
//===|  Archivo PrimaryGeneratorAction.cc |===
//=============================================

#include "PrimaryGeneratorAction.hh"

#include "G4Event.hh"
#include "G4GeneralParticleSource.hh"

//===| Metodo Constructor |===
PrimaryGeneratorAction::PrimaryGeneratorAction():
G4VUserPrimaryGeneratorAction()
{
//===| Se declara un objeto G4GeneralParticleSource |===
 fParticleSource = new G4GeneralParticleSource();
}
//....oooOO0OOooo.....ooooOO0OOooo.....

//===| Metodo Destructor |===
PrimaryGeneratorAction::~PrimaryGeneratorAction()
{
//===| Se elimina el puntero del generador de particulas |===
 delete fParticleSource;
}
//....oooOO0OOooo.....ooooOO0OOooo.....

//===| Metodo GeneratePrimaries |===
void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
//===| Se genera un evento |===
 fParticleSource->GeneratePrimaryVertex(anEvent);
}
```

Los datos de la partícula se agregan en un macro (extensión .mac) a través de comandos. La lista completa de comandos se encuentra en [6, sección 2.7] y permiten definir: el tipo de partícula; su polarización; la distribución de energía (mono-energética, lineal, Gaussiana, cuerpo negro, entre otras); la distribución de momento (isotrópica, unidireccional, ley-coseno, haz); y la distribución de las partículas, es decir, si se generan en un plano (círculo, cuadrado, elipse, ...) o en un volumen (cilindro, elipsoide o esfera). El siguiente ejemplo muestra un macro sencillo para este tipo generador.

```
#==|      Archivo Generador.mac. Ejemplo     |==
#==| de macro para  G4GeneralParticleSource |==

/run/initialize                #--| Inicia Geant4

#===| Particulas y energia del flujo |===
/gps/particle proton           #--| Genera protones
/gps/ene/type Mono             #--| Distribucion de energia
/gps/ene/mono 0.5 MeV          #--| Valor de la energia
```

```
#===| Posicion del Flujo |===
/gps/pos/type Volume          #--| Tipo de distribucion
/gps/pos/shape Sphere         #--| Forma de la distribucion
/gps/pos/radius 1 km          #--| Radio distribucion
/gps/pos/centre 0 0 10 km     #--| Centro de la distribucion
#===| Momento del flujo |===
/gps/ang/type iso             #--| Distribucion isotropica

/run/beamOn 500               #--| Simular 500 particulas
```

La lectura del macro se realiza al ejecutar la simulación y escribir el comando

`/control/execute [NombreArchivoMacro].mac`

1.2.6. La clase PhysicsList.

La interacción entre partículas y materia se gestiona a través de *procesos* y *modelos*. Los procesos agrupan modelos y cada proceso está relacionado con una interacción partícula-materia [7]. Estos procesos se dividen en siete categorías: procesos de decaimiento, procesos electromagnéticos, procesos hadrónicos, procesos ópticos, procesos para interacción entre leptones y rayos gamma con núcleos, parametrización y transporte [6].

La clase `PhysicsList` implementa los procesos y modelos físicos de la simulación. No es recomendable aventurarse a escribir el código de esta clase porque la información que existe en los manuales de Geant4 es insuficiente para este fin. Lo recomendable es buscar en los ejemplos que vienen con Geant4 algún archivo `PhysicsList` que tenga los modelos y procesos adecuados para su aplicación. Los ejemplos se encuentran en la carpeta:

`[CarpetaInstacionGeant4]/share/Geant4-10.1.2/examples/`

La carpeta `[CarpetaInstacionGeant4]` es la misma carpeta que usa `CMAKE` en el comando `-DCMAKE_INSTALL_PREFIX` al momento de instalar Geant4. Una vez se tenga un archivo `PhysicsList` se deben copiar los archivos `PhysicsList.hh` y `PhysicsList.cc` a su aplicación. Las carpetas `advanced` (ejemplos avanzados) y `extended` (ejemplos extendidos o ampliados) cuentan con ejemplos muy útiles.

Los ejemplos de la carpeta `extended` tienen una función de aprendizaje y enseñan diferentes opciones que ofrece Geant4, como son: generar partículas (carpeta `eventgenerator`), colocar un sólido usando rotaciones y traslaciones (carpeta `geometry`), simular interacciones electromagnéticas (carpeta `electromagnetic`) o cómo implementar un campo magnético en la simulación (carpeta `field`) entre otras.

Los ejemplos de la carpeta `advanced` muestran aplicaciones reales. Por ejemplo, la simulación `air_shower` simula un detector de rayos cósmicos inspirado en el experimento ULTRA [14], en cambio,

gammaray_telescope simula un telescopio de rayos gamma. También se incluyen ejemplos aplicados al área médica, como hadrontherapy que muestra una aplicación en el campo de terapia con iones y protones. Otra aplicación interesante es underground_physics que simula un detector subterráneo para búsqueda de materia oscura.

Otro camino es utilizar alguno de los paquetes PhysicsList que tiene Geant4. Estos se ubican en la carpeta:

[DirectorioGeant4]/source/physics_lists/constructors/

La carpeta [DirectorioGeant4] es aquella en la que se descomprimió Geant4. Para usar estos paquetes basta con declararlos en el archivo PhysicsList. El ejemplo básico B3 de Geant4 muestra como hacerlo. En ese ejemplo se utilizan los paquetes: G4DecayPhysics, G4RadioactiveDecayPhysics y G4EmStandardPhysics. Estos nombres se deben escribir en las primeras líneas del archivo PhysicsList.cc, usando la palabra reservada #include. El siguiente fragmento de código, extraído del archivo B3PhysicsList.cc del ejemplo B3, muestra como hacerlo.

```
//===| Archivo  B3PhysicsList.cc |===

#include "B3PhysicsList.hh"

#include "G4DecayPhysics.hh"
#include "G4RadioactiveDecayPhysics.hh"
#include "G4EmStandardPhysics.hh"

// ...oooOOOOOooo..... ..oooOOOOOooo..... ..oooOOOOOooo.

B3PhysicsList::B3PhysicsList() : G4VModularPhysicsList()
{
//--| Se registran los paquetes PhysicsList

  RegisterPhysics(new G4DecayPhysics());
  RegisterPhysics(new G4RadioactiveDecayPhysics());
  RegisterPhysics(new G4EmStandardPhysics());
}
```

1.2.7. Captura de datos.

Geant4 tiene la capacidad de almacenar la información que se genera en una simulación. Para ello es necesario contar con uno o más detectores. Técnicamente, en Geant4 un detector es un objeto de la clase G4VSensitiveDetector asignado a un sólido, éste construye objetos llamados *hit* que toman información de las partículas que viajan por el sólido/detector [6, 8]. La asignación de un sólido al objeto G4VSensitiveDetector se realiza dentro del método ConstructSDandField() de la clase DetectorConstruction. Se pueden tomar tres caminos para implementar un detector.

1) Generar un detector mediante comandos

Para usar esta opción se debe crear un puntero a la clase `G4ScoringManager` dentro del archivo principal de la aplicación (sección 1.2.11). El siguiente código es un ejemplo:

```
//===| Archivo principal de la |===
//===| simulacion: Aplicacion.cc |===
...
...
#include "G4ScoringManager.hh"
...
...
int main(int argc,char** argv)
{
G4RunManager* runManager = new G4MTRunManager;
G4ScoringManager* scManager = G4ScoringManager::GetScoringManager();
...
...
```

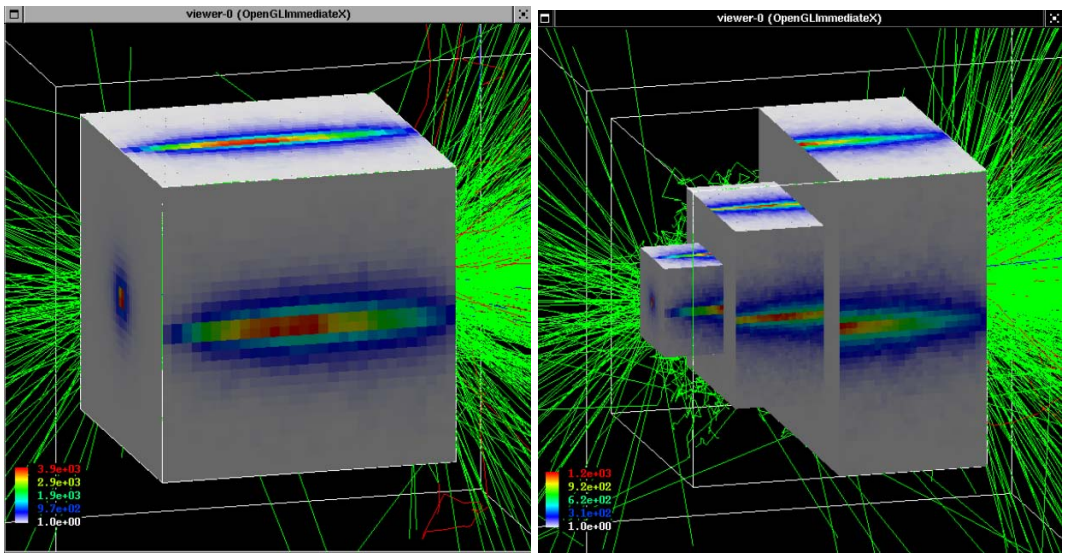
Un detector de este tipo se conoce, al menos en la documentación que hay sobre Geant4, como un *scoring mesh*. Actualmente solo scoring mesh en forma de paralelepípedo (cajas) y cilíndricos pueden ser declarados. Deben tener tamaño, siguiendo los argumentos de la clase `G4Box` y la clase `G4Tubs` (sección 1.1.3); número de bins en cada eje y posición respecto al centro del sólido `World` (sección 1.2.4). Los bins dividen cada eje del scoring mesh en regiones. Por ejemplo, suponga un scoring mesh de la clase `G4Box` con el siguiente número de bins: 30 para X, 15 para Y y 5 para Z. Este scoring mesh tendrá en el eje X 30 divisiones, 15 divisiones en el eje Y y 5 en el Z. Cada región registra la información que se pidió de forma independiente, de tal forma que al finalizar la simulación, los datos serán desplegados por regiones. El siguiente código es un ejemplo tomado del archivo `run1.mac` ubicado en la carpeta:

`[CarpetaInstacionGeant4]/share/Geant4-10.1.2/examples/extended/runAndEvent/RE03/`

```
#===| Ejemplo de un scoring mesh |===
/score/create/boxMesh boxMesh_1 #--| Tipo (boxMesh) y Nombre

/score/mesh/boxSize 100. 100. 100. cm #--| Dimensiones: X Y Z
/score/mesh/nBin 30 30 30 #--| Bins por Eje: X Y Z
/score/mesh/translate/xyz 0. 0. 100. cm #--| Posicion

#---| Informacion a guardar
/score/quantity/energyDeposit eDep #--| Energia Depositada(MeV)
/score/quantity/nOfStep nOfStepGamma
/score/filter/particle gammaFilter gamma #--| Solo Rayos Gamma
/score/quantity/nOfStep nOfStepEMinus
/score/filter/particle eMinusFilter e-
/score/quantity/nOfStep nOfStepEPlus
/score/filter/particle ePlusFilter e+
```

```
/score/close                          #---| Se cierra
...
...
```

Se pueden definir uno o más scoring mesh durante una simulación, como muestra la figura 1.4.



(a) Un único scoring mesh tipo `G4Box`. (b) Tres scoring mesh tipo `G4Box`.

Figura 1.4: Es posible declarar uno o más scoring mesh. Imágenes adaptadas y obtenidas de las presentaciones de Makoto Asai, disponibles en la pagina de SLAC (geant4.slac.stanford.edu)

2) Generar un detector mediante primitive scorers

`G4VPrimitiveScorer` es una clase que permite guardar información de las partículas que pasan por un sólido. La ventaja de esta técnica es la rapidez porque solo es necesario declarar los primitive scorers en el método `ConstructSDandField()` de la clase `DetectorConstruction`. Cada primitive scorer que se declara debe tener asignado al menos un sólido, tal como se muestra en el siguiente ejemplo.

```
//===|  Metodo ConstructSDandField()   |===
//===| Archivo DetectorConstruction.cc |===

void DetectorConstruction::ConstructSDandField()
{
   G4SDManager* SDPointerManager = G4SDManager::GetSDMpointer();
   G4String DetectorName;
   G4MultiFunctionalDetector* SDetector = new
       G4MultiFunctionalDetector("DetectorName = SDetector");
//---| Se Registra el Detector Sensible |---
   SDPointerManager->AddNewDetector(SDetector);
```

```
//=================
//===| Scorers |===
//=================
   G4String PrimitiveScoreName;
//---| Registra la Energia depositada en el Detector |---
   G4VPrimitiveScorer* scorer0 = new
      G4PSEnergyDeposit(PrimitiveScoreName = "TotalEnergy");
//---| Registra la Distancia que viaja la Particula |---
//---|          dentro del Detector                 |---
   G4VPrimitiveScorer* scorer1 = new
      G4PSTrackLength(PrimitiveScoreName = "Lenght");
//---| Se Agregan los Scorers al Detector |---
   SDetector->RegisterPrimitive(scorer0);
   SDetector->RegisterPrimitive(scorer1);
//---| Se asigna el volumen logico "fCoverLV" |---
//---|        al Detector Sensible            |---
   fCoverLV->SetSensitiveDetector(SDetector);
}
```

En el manual de Geant4 [6, sección 4.4.5] se muestra qué tipo de primitive scorers se pueden declarar. También en la pagina de SLAC (geant4.slac.stanford.edu) se encuentran las presentaciones de Makoto Asai que ofrecen más información sobre este tema.

Cada primitive scorer genera un objeto de la clase `G4THitsMap`. Estos objetos son muy similares a los objetos `map` de C++ porque son matrices que guardan un valor tipo `G4double` (la variable física que almacena el primitive scorer) y un valor tipo `G4int` asociado al sólido/detector. El manual de Geant4 [6, sección 4.4.1] no dice mucho sobre estos objetos pero en la carpeta:

`[CarpetaInstacionGeant4]/share/Geant4-10.1.2/examples/extended/runAndEvent/`

hay ejemplos de cómo usarlos para extraer la información guardada por los primitive scorers. El siguiente código muestra como hacer esto utilizando los primitive scorers (scorer0 y scorer1) declarados en el código anterior. De especial relevancia en el método `GetDoubleValue` escrito en el archivo `RunLocal.hh` porque él suma y extrae el valor de la variable física que registra cada `G4THitsMap` generado.

```
//===| Archivo RunLocal.hh |===

#ifndef RunLocal_h
#define RunLocal_h 1

#include "G4Run.hh"
#include "G4Event.hh"
#include "G4THitsMap.hh"
#include "G4ThreeVector.hh"
#include "globals.hh"
```

```
class RunLocal: public G4Run
{
public:
 RunLocal();              //--| Constructor
 virtual ~RunLocal();     //--| Destructor
//--| Metodo donde se toman los G4THitsMap
 virtual void RecordEvent(const G4Event*);
//--| Este hilo se une al hilo principal (Multi-hilos)
 virtual void Merge(const G4Run*);

private:
G4int fTotalEnergyID, fLenghtID;

public:
//======================
//===| Get Methods |===
//======================
//---|  Este metodo recibe un G4THitsMap y extrae el  |---
//---| valor total de la variable fisica que almacena |---
G4double GetDoubleValue(const G4THitsMap<G4double> &MAP) const
{
 G4double value;     value = 0.;
 std::map<G4int,G4double*>::iterator itr = MAP.GetMap()->begin();
 for ( ; itr != (MAP.GetMap()->end()); itr++)
   { value += *(itr->second); }
 return value;
}

};
#endif

//============================
//===| Archivo RunLocal.cc |===
//============================
#include "RunLocal.hh"

#include "G4SDManager.hh"
#include "G4HCofThisEvent.hh"
#include "G4RunManager.hh"
#include "G4PhysicalConstants.hh"

//....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo

RunLocal::RunLocal() : G4Run()
{
 G4SDManager* SDManagerPointer = G4SDManager::GetSDMpointer();
//---| Se toman los identificadores para cada G4THitsMap |---
//---|   El nombre para extraer el identificador sigue   |---
```

```
//---| la estructura: DetectorName/PrimitiveScoreName |---
 fTotalEnergyID = SDManagerPointer->GetCollectionID("SDetector/TotalEnergy");
 fLenghtID = SDManagerPointer->GetCollectionID("SDetector/Lenght");
}
//....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo

RunLocal::~RunLocal() {}
//....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo

void RunLocal::RecordEvent(const G4Event* event)
{
 G4HCofThisEvent* HCE = event->GetHCofThisEvent();

 if(!HCE) return;
//---| Contador
 numberOfEvent++;
//-----| Se Toman los G4THitsMap |---
 G4THitsMap<G4double>* TEmap = (G4THitsMap<G4double>*)(HCE->GetHC(fTotalEnergyID));
 G4THitsMap<G4double>* Lmap  = (G4THitsMap<G4double>*)(HCE->GetHC(fLenghtID));

//---| Se imprime el valor de cada variable fisica |---
 G4cout <<
 "EnergiaTotal: " << GetDoubleValue(*TEmap)  <<
 " Longitud: "    << GetDoubleValue(*Lmap)/m <<
 G4endl;
 G4Run::RecordEvent(event);
}
//....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo

void RunLocal::Merge(const G4Run* run)
{
  G4Run::Merge(run);
}
//....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo
```

3) Generar un detector mediante la clase `DetectorSD`

Este método es muy parecido al anterior pero más largo y con mayor libertad. La idea es escribir dos clases por detector: una clase dedicada a tomar la información que va registrando el sólido/detector (la clase `DetectorSD`) y otra clase que se encarga de guardar esa información (la clase `DetectorHit`).

Como la clase `DetectorHit` solo almacena información, se puede decir que es el análogo a los objetos `G4THitsMap` que generan los primitive scorers, ya que sus métodos son principalmente para guardar y copiar información.

La clase `DetectorSD` es más interesante. En ella se define qué datos guardará el sólido/detector cuando una partícula viaje por él. La información es tomada de la clase `G4Step`, pero también es posible obtener

información de la clase `G4Track` gracias al método `G4Step::GetTrack()`. Los métodos para obtener información de la partícula se puede consultar en los archivos `G4Step.hh` y `G4Track.hh` ubicados en la carpeta:

```
[CarpetaInstacionGeant4]/include/Geant4/
```

Se pueden encontrar ejemplos para conocer la declaración de estas clases en los ejemplos básicos proporcionados por Geant4. La forma como se declara el detector es similar al empleado con los primitive scorers. La asignación se realiza en el método `ConstructSDandField()` de la clase `DetectorConstruction`, tal como se muestra en el siguiente código.

```
//===| Metodo ConstructSDandField() del |===
//===| archivo DetectorConstruction.cc  |===

void DetectorConstruction::ConstructSDandField()
{
//--| Se declara un detector |--
 DetectorSD* SDet = new DetectorSD("/Detector_SD");
//--| Se Registra el detector en Geant4 |--
 G4SDManager* SDMan= G4SDManager::GetSDMpointer();
 SDMan->AddNewDetector(SDet);
//--| Se asigna el solido fCoverLV al detector |--
 if(fCoverLV)
    SetSensitiveDetector(fCoverLV, Sdet);
}
```

1.2.8. Simulación Multi-hilos.

Multi-hilos (*Multi-Thread* en inglés) permite crear aplicaciones capaces de correr en plataformas con varios núcleos (cores). Un hilo es una copia de la aplicación que se puede ejecutar junto a otros hilos al mismo tiempo, de esta forma se optimiza el uso del hardware pues cada núcleo del procesador ejecuta un hilo. En una computadora con k núcleos, sin multi-hilos, el equipo trabajará con k proceso independientes, pero con multi-hilos se obtendrá un único proceso con k hilos [15].

Una aplicación que ejecuta varios hilos está conformada por un hilo principal y varios hilos secundarios. El hilo principal es al que tiene acceso el usuario y se encarga de administrar al resto de hilos, en el caso de Geant4, se trata de la terminal o ambiente gráfico donde se ingresan comandos. Los hilos secundarios se conocen en inglés como *background*, estos hilos se encargan de procesar la información y hacer los cálculos. Así, en una aplicación multi-hilo el hilo principal está disponible para el usuario, mientras los cálculos y tareas son ejecutadas en el background, fuera de la vista. La simulación de las interacciones físicas, el transporte de la partícula y las modificaciones a sus atributos son ejemplos de tareas que se ejecutan en el background del multi-hilos de Geant4.

El uso de multi-hilos se habilita en el archivo principal de la aplicación (sección 1.2.11) escribiendo el siguiente código.

```
//===| Archivo Aplicacion.cc |===

#include "G4MTRunManager.hh" //--| Clase para Multi-hilos
...
...
int main(int argc,char** argv)
{
//==| Puntero a RunManager |==
 G4MTRunManager* runManager = new G4MTRunManager;

 runManager->SetNumberOfThreads(
//==| Se generan tantos hilos como numero |==
//==|         de cores tiene el equipo     |==
 G4Threading::G4GetNumberOfCores());
...
...
}
```

1.2.9. Histogramas.

Los histogramas son herramientas gráficas que muestran la frecuencia de aparición de una variable discreta. Por ejemplo, en un histograma de calificaciones se muestra cuántos alumnos obtuvieron cierta calificación: cuántos obtuvieron un seis, cuántos un ocho y cuántos un diez. En este caso, la calificación es la variable discreta y el número de alumnos indica la frecuencia de aparición.

En Geant4 es la clase `G4AnalysisManager` la encargada de crear histogramas. Se pueden crear histogramas de una dimensión (1-D), bidimensionales (2-D), y tridimensionales (3-D) [6]. También es posible crear tablas llamadas *ntuple*. Se pueden encontrar ejemplos para aprender a implementar histogramas o tablas en la carpeta:

`[CarpetaInstacionGeant4]/share/Geant4-10.1.2/examples/analysis/`

Se encuentran disponibles cuatro formatos para guardar tanto histogramas como tablas. Los dos primeros formatos responden a software creado especialmente para análisis estadístico en el área de física de altas energías.

ROOT: Software mantenido por el CERN, escrito principalmente en lenguaje C++. Se puede descargar en la pagina root.cern.ch.

AIDA: Diseñado para la creación y manipulación de histogramas, tablas y otros objetos que tengan como fin analizar datos. Existen dos versiones de `AIDA`: una para `C++` y otra para `Java`. Puede encontrar una descripción más completa en el manual de `AIDA` que se encuentra en la pagina `http://aida.freehep.org`

XML: Es un lenguaje de marcas. A diferencia de los lenguajes de programación, los lenguajes de marcas o marcado (*markup languaje* en inglés) utilizan una notación especial que señala las diferentes partes de un documento o archivo. Estos lenguajes no tienen como función realizar operaciones aritméticas, se utilizan exclusivamente para organizar información.

CSV: La información se guarda en tablas donde el delimitador es una coma. Una línea de texto en el fichero equivale a una fila en la tabla de datos, los valores de cada columna están separados por una coma.

Para guardar información en alguno de estos formatos se recomienda crear una clase destinada a ese fin, la clase `HistogramManager`, donde se crearan los histogramas y tablas que necesite la aplicación.

En el siguiente ejemplo se crean 4 histogramas 1-D que registran la siguiente información: ángulos de dispersión en coordenadas esféricas (ϕ y θ), energía depositada y distancia recorrida dentro del detector.

```
//===| Archivo HistogramManager.cc |===

#include "HistogramManager.hh"
#include "G4UnitsTable.hh"

HistogramManager::HistogramManager
{
//===| Se llama al metodo que crea los histogramas |===
 BookStore();
}
//...oooOOOOOooo...oooOOOOOooo...oooOOOOOooo

HistogramManager::~HistogramManager()
{
//===| Se borra puntero a G4AnalysisManager |===
 delete G4AnalysisManager::Instance();
}
//...oooOOOOOooo...oooOOOOOooo...oooOOOOOooo

void HistogramManager::BookStore()
{
//===| Puntero a G4AnalysisManager |===
 G4AnalysisManager* analysisManager =
              G4AnalysisManager::Instance();

//--| Nombre del archivo .root |--
 analysisManager->SetFileName("Aplicacion");

//---| Identificadores para los Histogramas 1-D |--
 const G4String id[] = {"0","1","2","3","4"};

//---| Titulos de los Histogramas 1-D |--
 const G4String title[] = {
```

```
  "Dummy.",           //--| 0
  "Total energy.",    //--| 1
  "Distancia.",       //--| 2
  "Angulo Phi",       //--| 3
  "Angulo Theta"};    //--| 4

  G4int nbins;              //--| No. de intevarlo en el histograma
  G4int j;                  //--| Identificador del histograma
  G4double Minbin,Maxbin; //--| valores Max. y Min. histograma

//---| Histogramas 1-D |---
  j = 1; nbin = 300;  Minbin = 0 ; Maxbin = 299;
  G4int H1 = analysisManager->
             CreateH1(id[j], title[j], nbins,Minbin,Maxbin);
  j = 2; nbin = 100;  Minbin = 0 ; Maxbin = 99;
  G4int H2 = analysisManager->
             CreateH1(id[j], title[j], nbins,Minbin,Maxbin);
  j = 3; nbin = 360;  Minbin = 0 ; Maxbin = 359;
  G4int H3 = analysisManager->
             CreateH1(id[j], title[j], nbins,Minbin,Maxbin);
  j = 4; nbin = 180;  Minbin = 0 ; Maxbin = 179;
  G4int H4 = analysisManager->
             CreateH1(id[j], title[j], nbins,Minbin,Maxbin);
}
```

Posteriormente, en la clase `RunAction`, se llamará a la clase `HistogramManager` para crear los histogramas, al iniciar de la simulación, y cerrar el archivo `.root` en el cual se almacenaron los datos generados en la simulación.

```
//===| Archivo RunAction.cc |==

#include    RunAction    .hh"
#include "HistogramManager.hh"

#include "G4RunManager.hh"
#include "G4Run.hh"
#include "G4PhysicalConstants.hh"
#include "G4SystemOfUnits.hh"
#include "G4UnitsTable.hh"

RunAction::RunAction()  :  G4UserRunAction()
{
//--| En el archivo RunAction.hh se declara   |--
//--| el puntero "HistogramManager* fManager" |--
//--|      y se crean los histogramas         |--

   fManager = new HistogramManager();
}
//....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo
```

```
RunAction::~RunAction()
{
//--| Se borra puntero a HistogramManager() |--
 delete fManager;
}
//....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo

void RunAction::BeginOfRunAction(const G4Run*)
{
 G4AnalysisManager* analysisManager = G4AnalysisManager::Instance();

//===| Abre el archivo Aplicacion.root |===
 analysisManager->OpenFile();
}
//....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo

void RunAction::EndOfRunAction(const G4Run* )
{
 G4AnalysisManager* analysisManager = G4AnalysisManager::Instance();
...
...
//--| Se llama a los metodos |---
//--| que guardan los datos  |---
//--|  en los histogramas    |---
...
...
//===| Guarda los datos en Aplicacion.root |===
 analysisManager->Write();

//===| Cierra el archivo Aplicacion.root |===
 analysisManager->CloseFile();
}
```

1.2.10. La clase ActionInitialization.

La clase `ActionInitialization` implementa casi todas las clases de las simulación, excepto `DetectorConstruction` y `PhysicsList`. Esta clase también gestiona los hilos cuando la simulación es Multi-hilos. El hilo principal es implementado en el método `BuildForMaster()` mientras los hilos secundarios son implementados por el método `Build()`. Si la simulación no es multi-hilos entonces `Build()` ejecutará el único proceso. El siguiente código es un ejemplo de la clase.

```
//==| Archivo ActionInitialization.hh  |==

#ifndef ActionInitialization_h
#define ActionInitialization_h 1
```

```
#include "G4VUserActionInitialization.hh"

class ActionInitialization : public G4VUserActionInitialization
{
 public:
  ActionInitialization();
  virtual ~ActionInitialization();

  virtual void BuildForMaster() const;
  virtual void Build() const;
};

#endif

//=====================================
//.....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo
//=====================================

//==| Archivo ActionInitialization.cc |==

#include "ActionInitialization.hh"
#include "PrimaryGeneratorAction.hh"
#include "RunAction.hh"
#include "EventAction.hh"

//==| Constructor |==
ActionInitialization::ActionInitialization() :
G4VUserActionInitialization()
{;}
//.....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo

//==| Destructor |==
ActionInitialization::~ActionInitialization()
{;}
//.....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo

//==| Clases que operan en el hilo principal |==
void ActionInitialization::BuildForMaster() const
{
 SetUserAction(new RunAction);
}
//.....oooOOOOOooo.....oooOOOOOooo.....oooOOOOOooo

//==| Clases que operan en hilos secundarios |==
void ActionInitialization::Build() const
{
 SetUserAction(new PrimaryGeneratorAction);
 SetUserAction(new RunAction);
 SetUserAction(new EventAction);
}
```

```
// ...oooOOOOOooo...oooOOOOOooo...oooOOOOOooo
```

1.2.11. El archivo ejecutable.

Si todas las clases de nuestra aplicación han sido escritas, es momento de escribir el archivo `Aplicacion.cc`. Este archivo implementa tres clases: `ActionInitialization`, `DetectorConstruction` y `PhysicsList`. Escrito `Aplicacion.cc` la simulación está lista para compilarse. La compilación se hace con `CMAKE` a través del archivo `CMakeLists.txt`, el cual le indica como compilar el código.

El archivo Aplicacion.cc

Lo primero que debe tener este archivo es un puntero a un objeto de la clase `G4MTRunManager` y otro puntero a un objeto de la clase `G4RunManager`. La primer clase se utiliza en simulaciones multi-hilos, por el contrario, la segunda se invoca en simulaciones sin multi-hilos. En seguida, se invocan las clases `ActionInitialization`, `DetectorConstruction` y `PhysicsList`. Seguramente se recuerda que la clase `DetectorConstruction` implementa la geometría de la simulación (sección 1.2.4), la clase `PhysicsList` implementa los modelos y procesos físicos (sección 1.2.6) mientras la clase `ActionInitialization` invoca las demás clases de la simulación.

Por último, a través de la clase `G4VisManager` se inicia una sesión gráfica siguiendo las instrucciones del archivo `grafico.mac` o, en su lugar, se ejecutan las instrucciones de un macro diferente. Ejemplos de este tipo de archivos se pueden encontrar en la mayoría de los ejemplos que vienen con Geant4 bajo el nombre de `vis.mac`. El siguiente código es un ejemplo del archivo `Aplicacion.cc`. El siguiente código es un ejemplo del archivo `Aplicacion.cc`.

```
//===| Archivo Aplicacion.cc |===

#ifdef G4MULTITHREADED
#include "G4MTRunManager.hh" //--| Opcion Multi-hilos
#else
#include "G4RunManager.hh"   //--| Opcion Mono-hilo
#endif

#include "DetectorConstruction.hh"
#include "ActionInitialization.hh"
#include "PhysicsList.hh"

#include "G4UImanager.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"

int main(int argc,char** argv)
{
```

```
//===| Si no hay argumentos de entrada se genera  |===
//===|         una sesion interactiva             |===
 G4UIExecutive* ui = 0;
 if (argc == 1) {ui = new G4UIExecutive(argc, argv);}

//---| Se genera una simulacion multi-hilos |---
 #ifdef G4MULTITHREADED
 G4MTRunManager* runManager = new G4MTRunManager;
//---| Numero de hilos a generar |---
 runManager->
        SetNumberOfThreads(G4Threading::G4GetNumberOfCores());

//---| Se genera una simulacion de un solo hilo |---
 #else
 G4RunManager* runManager = new G4RunManager;
 #endif

//---| Se invocan las clases de la simulacion |---
 runManager->SetUserInitialization(new DetectorConstruction());
 runManager->SetUserInitialization(new PhysicsList());
 runManager->SetUserInitialization(new ActionInitialization());

//===|  G4VisManager maneja la parte  |===
//===|    grafica de la simulacion    |===
 G4VisManager* visManager = new G4VisExecutive;
 visManager->Initialize();
 G4UImanager* UImanager = G4UImanager::GetUIpointer();

//---| Si se recibe un macro con instrucciones se lee |---
 if ( ! ui )
 {
  G4String command = "/control/execute ";
  G4String fileName = argv[1];
  UImanager->ApplyCommand(command+fileName);
 }

//===|  De lo contrario, se ejecutan las  |===
//===| instrucciones del macro grafico.mac |===
 else
 {
  UImanager->ApplyCommand("/control/execute grafico.mac");
  ui->SessionStart();
  delete ui;
 }

//===| Al terminar la simulacion se libera memoria |===
 delete visManager;
 delete runManager;
}
```

CMAKE.

CMAKE es una herramienta que ayuda a la construcción de software manipulando el proceso de compilación, gracias a archivos de configuración llamados CMakeLists.txt que le indican a CMAKE qué archivos considerar y cómo llevar a cabo la compilación [16]. Una vez que CMAKE termina, entrega al usuario un archivo ejecutable.

El siguiente código es un ejemplo de un archivo CMakeLists.txt. El primer comando se asegura de que CMAKE esté instalado en el equipo y que la versión sea igual o superior a la versión mínima, de no ser así se mandará un mensaje de error y CMAKE no creará el ejecutable. Después se escribe el nombre del proyecto, que también será el nombre del archivo ejecutable. En seguida, CMAKE busca la carpeta donde se instaló Geant4 e incluye las clases que necesita el proyecto. Finalmente CMAKE incluye las clases del proyecto y crea el archivo ejecutable. Si todo sale bien, al concluir el proceso se tendrá un archivo ejecutable.

```
#---| Ejemplo de un archivo CMakeLists.txt |---

#--| Version Minima necesaria de CMAKE |---
cmake_minimum_required(VERSION 2.8 FATAL_ERROR)

#---| Nombre del proyecto |---
project(Aplicacion)

#---| Busca Geant4 y los paquetes graficos, si |---
#---|   es que se utiliza el ambiente grafico  |---
option(WITH_GEANT4_UIVIS "Build Aplicacion with Geant4 UI and Vis drivers" ON)

if(WITH_GEANT4_UIVIS)
  find_package(Geant4 REQUIRED ui_all vis_all)
else()
  find_package(Geant4 REQUIRED)
endif()

#---| Agrega las clases de Geant4 |---
include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/include)

#---| Agrega las clases del proyecto |---
file(GLOB sources ${PROJECT_SOURCE_DIR}/src/*.cc)
file(GLOB headers ${PROJECT_SOURCE_DIR}/include/*.hh)

#---| Genera el ejecutable uniendo  |---
#---| todas las clases del proyecto |---
add_executable(Aplicacion Aplicacion.cc ${sources} ${headers})
target_link_libraries(Aplicacion ${Geant4_LIBRARIES})

#---| Agrega el archivo grafico.mac al proyecto |---
set(APLICACION_SCRIPTS grafico.mac )
```

```
foreach(_script ${APLICACION_SCRIPTS})
configure_file(
    ${PROJECT_SOURCE_DIR}/${_script}
    ${PROJECT_BINARY_DIR}/${_script}
    COPYONLY
    )
endforeach()

install(TARGETS Aplicacion DESTINATION bin )
```

Antes de usar `CMAKE` se recomienda crear una carpeta vacía llamada `build`, en el interior de esta carpeta estarán todos los archivos generados por `CMAKE` en el momento de compilar el proyecto.

1.3. Simulación de la teoría de dispersión de Rutherford.

Esta sección pretende mostrar una simulación sencilla usando Geant4. Se eligió la teoría de dispersión de Rutherford porque es una teoría conocida y de fácil simulación ya que solo necesita de una placa delgada y rectangular de oro más un detector sensible a las partículas alfa dispersadas por la placa.

1.3.1. La teoría de Rutherford.

En la primer década del siglo XX surgió un gran interés por usar la radiación como una herramienta para conocer la estructura interna del átomo. Para 1900 se sabía que el átomo no era indivisible pues se conocía una partícula subatómica: el electrón [17]. Éste fue descubierto en 1897 por el físico británico Joseph John Thomson al proponer que los rayos catódicos observados en tubos de descarga eran electrones [18]. La segunda partícula fundamental en descubrirse fue el protón, pero hasta 1918, cuando Rutherford logró aislarla y observarla después de hacer incidir partículas alfa en una placa de Nitrógeno.

La pregunta que se intentaba responder era ¿Cuál es la estructura del átomo? J. J. Thomson dio como respuesta que el átomo poseía N corpúsculos de carga negativa (electrones) rodeados de una carga eléctrica positiva, de valor Ne, uniformemente distribuida en una esfera [18, 19]. En cambio, Rutherford propuso que esa carga positiva tenía que estar en el centro de la esfera con los electrones distribuidos uniformemente alrededor de ésta [19]. Así, Rutherford propuso un modelo donde la dispersión de partículas debido a una placa delgada ocurre principalmente por la fuerza eléctrica que sienten al acercarse a un núcleo atómico.

Las ecuaciones 1.1, sobre el parámetro de impacto b, y 1.2, sobre la sección eficaz diferencial de interacción, son las fórmulas más mencionadas en los libros cuando se habla de la teoría de dispersión de Rutherford. En esas fórmulas Z_1 y Z_2 representan, respectivamente, el número atómico de las partículas incidentes y el número atómico de los núcleos en la placa objetivo, E es la energía cinética de las partículas, y e la carga eléctrica fundamental.

$$b = \frac{Z_1 Z_2 e^2}{2E} \cot\frac{\theta}{2} \tag{1.1}$$

$$\frac{d\sigma(\theta)}{d\Omega} = \left(\frac{Z_1 Z_2 e^2}{4E}\right)^2 \frac{1}{\sin^4\frac{\theta}{2}} \tag{1.2}$$

Pero en este trabajo se utiliza una ecuación diferente obtenida de [20, p. 55].

$$\Delta N = N\, n\, t\, \pi\, b^2 \tag{1.3}$$

En la ecuación 1.3 N representa el número de partículas incidentes en la placa, ΔN es el número de partículas dispersadas un ángulo igual o mayor a θ, n $\left[\text{átomos}/\text{cm}^3\right]$ es la densidad atómica de la placa, t el grueso de la placa y b el parámetro de impacto.

1.3.2. Geant4: dispersión de partículas.

Geant4 ofrece procesos y modelos de dispersión múltiple y dispersión simple para electrones, muones y hadrones [7, 21]. Los modelos de dispersión múltiple se clasifican en *algoritmos condensados* y *algoritmos detallados*. Los algoritmos detallados simulan todas la interacciones y colisiones experimentadas, sin embargo, solo son adecuados cuando el número de colisiones es pequeño. En cambio, los algoritmos condensados no simulan todas las interacciones, en su lugar, cada vez que se desplaza la partícula se simulan de forma global los efectos de todas las interacciones y se modifican los atributos de la partícula [22–24].

Los modelos de dispersión en Geant4 se basan en algoritmos condensados, excepto por el modelo de dispersión múltiple `G4WentzelVIModel` y el modelo de dispersión simple `G4CoulombScattering`, ambos modelos basados en la teoría de Wentzel [25]. El modelo `G4UrbanMscModel` esta basado en la teoría de Lewis [23], es el modelo de dispersión múltiple por defecto en Geant4 [21] y es válido para todas las partículas en el rango de energía de eV a TeV [7, 21]. En cambio, el modelo `G4GoudsmitSaundersonMscModel` utiliza la teoría de Goudsmit-Saunderson [24], se puede utilizar con todas las partículas pero fue creado para aumentar la precisión en la simulación de electrones y positrones [7, 21]. Por su parte, el modelo `G4WentzelVIModel` ofrece mayor precisión cuando se simulan muones y hadrones [7, 21].

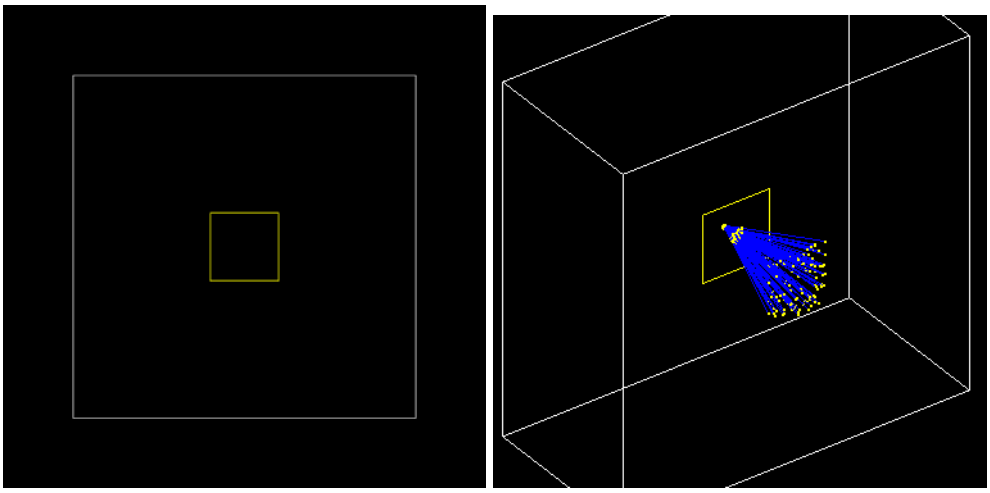
1.3.3. Simulación con Geant4.

En la simulación se construyen dos sólidos: el volumen World y una placa delgada de oro de 5 cm de largo por 5 cm de alto y grosor variable, como se puede observar en la figura 1.5. Se utiliza el generador `G4ParticleGun` para lanzar partículas α de 5 MeV de energía cinética hacia la placa de oro. Con el fin de obtener un haz no puntual de partículas, éstas se generan en el interior de un círculo de 0.25 cm de radio. Su vector de momento es anti-paralelo al eje Z, es decir, $\hat{\mathbf{p}} = (0, 0, -1)$.

No se definió un sólido/detector porqué solo se necesita guardar el ángulo de dispersión de cada partícula α en un histograma 1-D. El ángulo de dispersión se obtiene con el vector de momento de la partícula, el cual se obtiene con el método `G4Track::GetMomentumDirection()` de la clase `G4Track`. El siguiente

código muestra el proceso.

```
//===| Metodo PostUserTrackingAction |===
//===| del archivo TrackingAction.cc |===

void DisperseTracking::
      PostUserTrackingAction(const G4Track* aTrack)
{
//--| Se obtiene el momento final e inicial de la particula |--
 G4ThreeVector MomentoInicial = aTrack->GetVertexMomentumDirection();
 G4ThreeVector MomentoFinal   = aTrack->GetMomentumDirection();

//--| Devuelve la funcion coseno del angulo formado |--
//--|            por lo vectores                    |--
//--|     MomentoInicial y MomentoFinal             |--
 G4double cosTheta = MomentoInicial.cosTheta(MomentoFinal)

 //---| Angulo de dispersion |---
 G4double theta = std::acos(cosTheta);
}
```



(a) Frente. (b) Perfil.

Figura 1.5: Sólidos de la simulación. El sólido de contorno blanco representa el volumen `World` mientras el de contorno amarillo representa a la placa de oro. Las líneas azules representan a las partículas alfa.

El grosor de la placa fue diferente en cada simulación, comenzando con 0.01 µm de grosor, seguido de 0.1 µm, 1.0 µm, 2.0 µm, 3.0 µm, 4.0 µm, 5.0 µm, 6.0 µm y 7.0 µm de grosor. Mientras el número de partículas incidentes se mantuvo constante, un millón de partículas en cada simulación.

1.3.4. Rutherford: datos teóricos para la simulación.

Después de realizar la simulación con Geant4 era necesario comparar los resultados con la teoría de dispersión de Rutherford. Para este fin se escribió un programa de cómputo con C++ donde se calcula la probabilidad de que una partícula tenga un ángulo de dispersión entre θ y $\theta+1$. Se utiliza la relación 1.4 que indica cuántas partículas, $N(\theta)$, tienen un ángulo de dispersión igual o mayor que θ [20].

$$\frac{N(\theta)}{N_i} = \frac{\pi\, n\, L\, K^2\, Z^2 z^2\, e^4}{4\, T_\alpha^2} \cot^2\left(\frac{\theta}{2}\right),\ K = \frac{1}{4\pi\,\varepsilon_0} \tag{1.4}$$

En la relación 1.4 N_i es el número de partículas α incidentes en la placa de oro, ze es la carga eléctrica de las partículas α y T_α su energía cinética. En cambio, n indica la densidad atómica del oro $\left(5.9\times10^{22}\ \frac{\text{átomos}}{\text{cm}^3}\right)$, Ze la carga eléctrica del núcleo atómico de oro y L el grosor de la placa. Varios de estos datos son conocidos, cuando se sustituye el valor de todas estas constantes por:

$$a = \frac{\pi\, nK^2\, Z^2\, z^2\, e^4}{4} = 23.86\ \frac{\text{MeV}^2}{\text{cm}}$$

se obtiene la relación 1.5;

$$\frac{N(\theta)}{N_i} = a\frac{L}{T_\alpha^2}\cot^2\left(\frac{\theta}{2}\right) \tag{1.5}$$

La ecuación 1.5 indica cuantas partículas tendrán un ángulo de dispersión igual o mayor que θ. El siguiente paso es obtener el número de partículas con un ángulo de dispersión igual o mayor a $\theta+1$;

$$\frac{N(\theta+1)}{N_i} = a\frac{L}{T_\alpha^2}\cot^2\left(\frac{\theta+1}{2}\right) \tag{1.6}$$

La diferencia entre la ecuación 1.5 y 1.6 nos dice cuantas partículas tienen un ángulo de dispersión entre θ y $\theta+1$;

$$\frac{N(\theta)-N(\theta+1)}{N_i} = a\frac{L}{T_\alpha^2}\left[\cot^2\left(\frac{\theta}{2}\right)-\cot^2\left(\frac{\theta+1}{2}\right)\right] \tag{1.7}$$

La ecuación 1.7 se utiliza en el programa escrito en C++, el programa se muestra en el siguiente código. Los datos generados con este programa se observan en la curva de puntos negros de las figuras 1.6 y 1.7.

```
//===| Programa Rutherford.cc Teoria |===

#include <iostream>
#include <fstream>
#include <cmath>
#include <cstdlib>

using namespace std;

/* Programa que calcula el porcentaje de particulas
   dispesadas usando potencial de Coulomb y una placa
```

```
   de oro, con la formula;
(T^2_alpha/t) * (Delta N / N_total) = 23.8609 [MeV^2*atomo/cm] * cot^2(theta/2)
*/

main ()
{
 double resultado = 0, cot = 0;
 double actual_probabilidad = 0;
 double anterior_probabilidad = 0;
 double Energy = 5.0;        //===| (MeV) Fija
 double thick = 0.0001;      //===| (cm) Variable

 double PI = 3.14159265358979323846;
 double a  = 23.86127;       //===|  MeV^{2} / cm

 //==| Archivo donde se guarda el resultados |==
 ofstream particulas;
 particulas.open("Teorico5MevNormalizado10.txt");

//===| El primer valor, con theta = 1 Grado |===
 anterior_probabilidad = a*(thick/(Energy*Energy))* std::cos(PI/360)*
    std::cos(PI/360) / (std::sin(PI/360)*std::sin(PI/360));

 for (int theta=2; theta<=91 ; theta++)
 {
//==|     Conversion de theta en grados a radianes      |==
//==| Se ha convetido (theta/2)*(pi/180) a theta*pi/360 |==
  cot= std::cos(theta*PI/360)/std::sin(theta*PI/360);
  actual_probabilidad = a*cot*cot*thick/(Energy*Energy);
  resultado = anterior_probabilidad - actual_probabilidad;
  anterior_probabilidad=actual_probabilidad;
//==| Se guarda resultado en el archivo |==
  particulas << theta-1 << " " << resultado << endl;   } }
```

1.3.5. Resultados.

Los resultados se muestran gráficamente en las figuras 1.6 y 1.7. Como se mencionó en la sección 1.3.3, las partículas α tienen una energía cinética constante de 5 MeV mientras el grosor de la placa de oro tomó valores entre 0.1 µm y 7.0 µm. Se usaron tres modelos de dispersión por simulación: `G4UrbanMscModel`, `G4GoudsmitSaundersonMscModel` y `G4WentzelVIModel`.

Los resultados se pueden separar en dos grupos: simulaciones donde el número de partículas dispersadas aumenta conforme el ángulo de dispersión disminuye, sin un máximo definido, y simulaciones donde se observa un máximo en el número de partículas dispersadas. En la primer categoría se tienen la simulaciones de 0.01 µm, 0.1 µm y 1.0 µm. En la segunda las simulaciones de 2.0 µm a 7.0 µm. Ambas simulaciones se comportan como lo indica la teoría de Rutherford excepto que algunas muestran un máximo en el número de partículas dispersadas para un ángulo menor a 10°.

El comportamiento más interesante es que los tres modelos usados en la simulación dan el mismo resultado. Eso indica que para este tipos de aplicaciones el modelo de dispersión de Geant4 que se utilice no es importante. También se puede observar en las imágenes de las figuras 1.6 y 1.7 que para un cierto ángulo, siempre mayor al ángulo que corresponde al máximo, el número de partículas dispersadas por parte de Geant4 es mayor al número de partículas dispersadas por la teoría de Rutherford. Además, entre más gruesa es la placa de oro mayor es el número de partículas dispersadas por parte de Geant4 para ese mismo ángulo. Este comportamiento es resultado de la forma como operan los modelos de dispersión de Geant4. En la sección 1.3.2 se mencionó que estos modelos simulan las interacciones de la partícula en ciertos puntos de su trayectoria, si la placa es más gruesa entonces se simularan más interacciones y el ángulo de dispersión de la partícula será mayor.

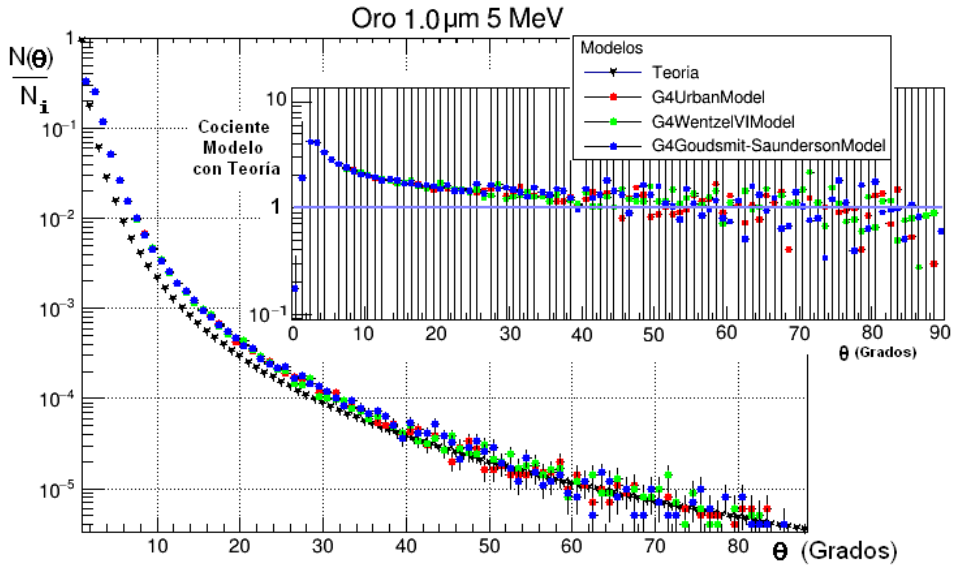
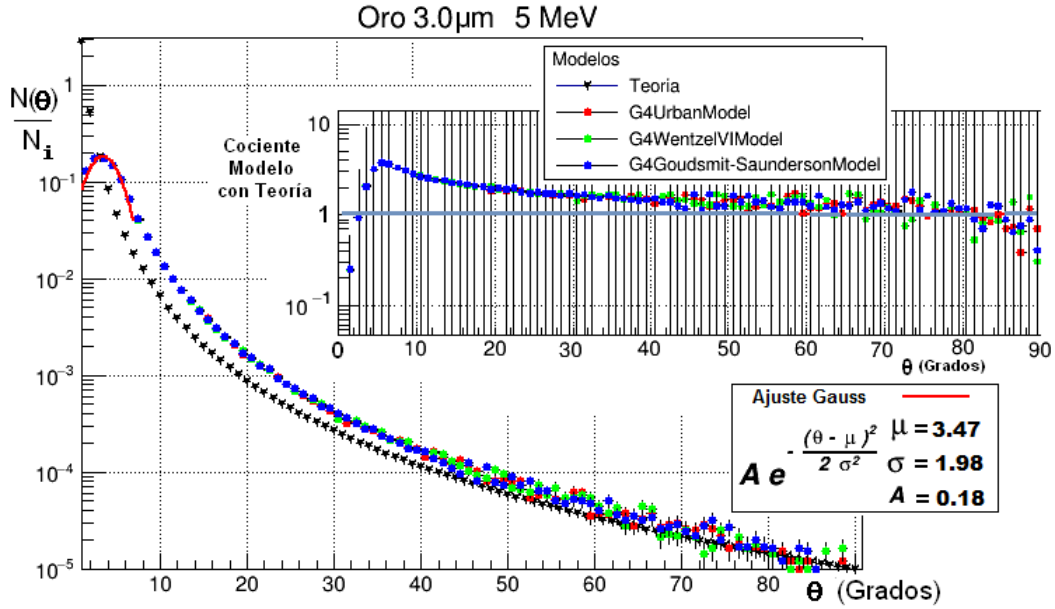
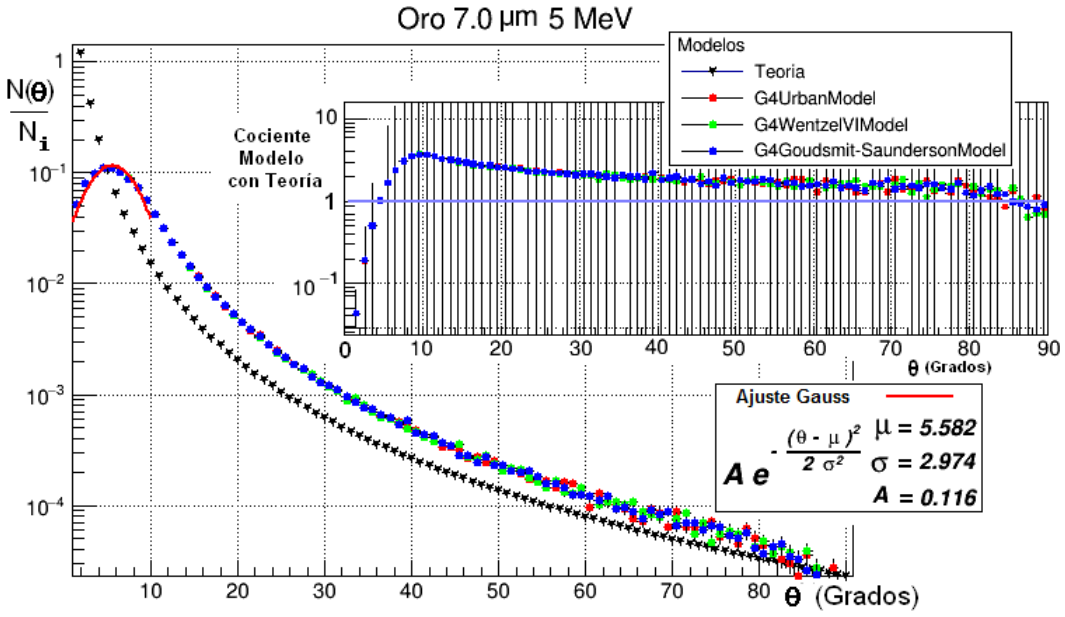


Figura 1.6: Placa de oro de 1 μm de grosor. La gráfica se encuentra normalizada, siendo $N(\theta)$ el número de partículas dispersadas y $N_i = 10^6$ el número de partículas usadas en la simulación. La gráfica más pequeña es el resultado de dividir los resultados obtenidos con Geant4 entre los resultados de la teoría de Rutherford.



(a) Placa de oro de 3 μm de grosor



(b) Placa de oro de 7 μm de grosor

Figura 1.7: Placas de oro de 3 μm y 7 μm de grosor. Las gráficas se encuentran normalizadas, siendo $N(\theta)$ el número de partículas dispersadas y $N_i = 10^6$ el número de partículas usadas en cada simulación. La gráfica más pequeña es el resultado de dividir los resultados obtenidos con Geant4 entre los resultados de la teoría de Rutherford.

CAPÍTULO 2

SIMULACIÓN MONTE CARLO DEL VOLCÁN PICO DE ORIZABA

En este capítulo se estudia por medio de simulaciones la capacidad del volcán Pico de Orizaba para absorber leptones cargados. En la sección 2.1 se explica, de forma breve, que es un rayo cósmico y que son las cascadas atmosféricas. Por su parte, la sección 2.2 está dedicada al observatorio de rayos gamma HAWC: su operación y objetivos científicos. En 2.3 se describe la técnica Earth-Skimming, que propone estudiar neutrinos con energía del orden de 10^{19} eV, así como su relación con este trabajo y el observatorio HAWC. Por último, en la sección 2.5 se dan los detalles concernientes a las simulaciones hechas con Geant4 para conocer la capacidad del volcán Pico de Orizaba de absorber leptones cargados.

2.1. Rayos cósmicos y cascadas atmosféricas.

Los rayos cósmicos fueron descubiertos por primera vez a principios del siglo XX por Victor Hess cuando realizaba ascensiones en globo [26]. En esa época se sabía que ciertas rocas producían ionización, la cuál era detectada con ayuda de un electroscopio. Por esa razón, se creía que la corteza terrestre producía una cierta cantidad de radiación ionizante, pero investigaciones posteriores con electroscopios mostraron resultados contradictorios: algunos resultados indicaban un aumento de la radiación ionizante con la altura; pero otros, una disminución [27]. Eso fue lo que motivo a Victor Hess a realizar su propia investigación llevándolo a descubrir los rayos cósmicos. La confirmación de la existencia de la antimateria, al detectarse el positrón en 1932, así como el descubrimiento de muones, piones y otras partículas ha sido posible gracias al estudio de los rayos cósmicos.

Los rayos cósmicos se clasifican de acuerdo a su origen en primarios o secundarios. Los primarios son partículas aceleradas en fuentes astrofísicas, mientras los rayos cósmicos secundarios son partículas que resultan de la interacción entre los rayos cósmicos primarios y el medio interestelar. Los electrones, los protones y los núcleos atómicos sintetizados en las estrellas son primarios. Núcleos que no son

abundantes en la nucleosíntesis estelar -como litio, berilio y boro- son secundarios [28].

Estos rayos cósmicos pueden llegar a la Tierra e interaccionar con su atmósfera, cuando esto ocurre se genera una cantidad importante de partículas. Aquí es necesario hacer una pausa y explicar que las partículas que se generan en la atmósfera por un rayo cósmico se clasifican en partículas primarias y secundarias. Las partículas primarias son las que vienen desde el exterior de la Tierra e interaccionan con la atmósfera, en cambio, las partículas secundarias son las que se generan por la interacción de las partículas primarias con la atmósfera terrestre. El proceso comienza con la colisión de un rayo cósmico con un núcleo de la atmósfera, esta primera colisión produce más de 50 partículas secundarias, la mayoría de ellas piones [29]. Los piones pueden ser de tres tipos: piones con carga eléctrica positiva (π^+), carga eléctrica negativa (π^-), y carga eléctrica neutra (π^0).

Tabla 2.1: Vida media de los mesones π.

Partícula	Vida media s
$\pi^{\pm}$	(2.6033 ± 0.0005) x 10^{-8}
π^0	(8.52 ± 0.18) x 10^{-17}

Nota: Datos extraídos de [30, p. 25].

La diferencia entre la vida media de los piones cargados y la vida media de los piones neutros (tabla 2.1) aumenta la probabilidad de que los piones cargados colisionen con otras moléculas de la atmósfera antes de decaer. El decaimiento más común para estas partículas es:

$$\pi^+ \rightarrow \mu^+ + \nu_\mu$$

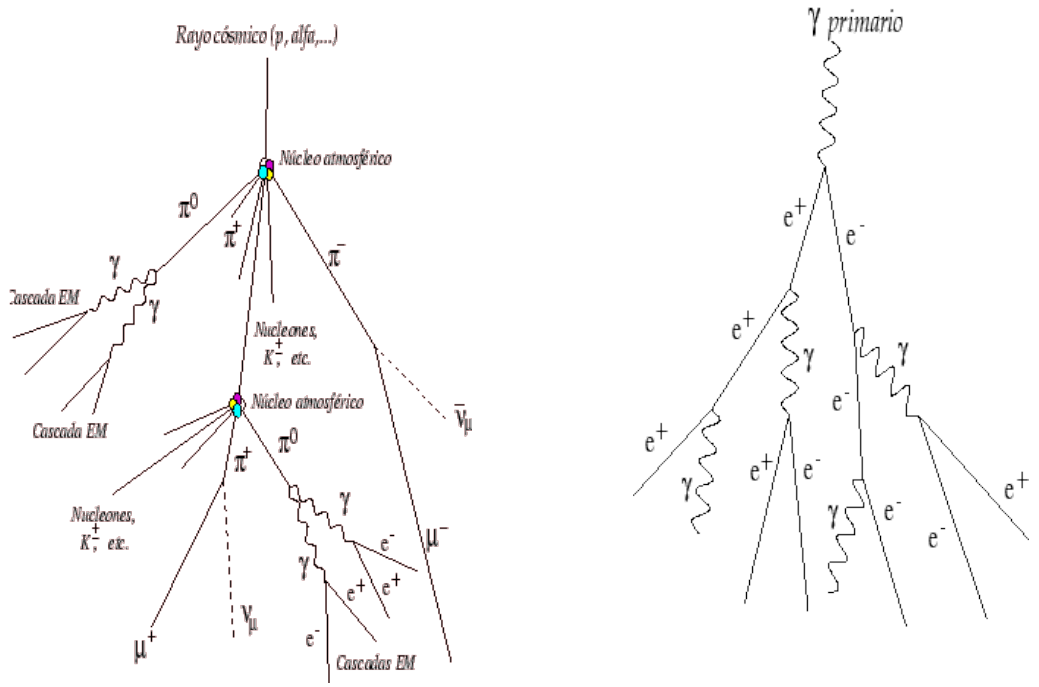
$$\pi^- \rightarrow \mu^- + \overline{\nu}_\mu$$

$$\pi^0 \rightarrow 2\gamma$$

La colisión de piones cargados con núcleos de la atmósfera crea partículas, de forma similar a la interacción del rayo cósmico con la atmósfera. Así, en cada colisión se generan más y más partículas. Esta multiplicación de partículas crea una estructura que se conoce como *cascada atmosférica hadrónica*, representada de forma esquemática en la figura 2.1 (a).

Las cascadas no se generan exclusivamente por un rayo cósmico, también pueden ser creadas por un rayo gamma. El rayo gamma puede ser de una fuente astrofísica o creado en una cascada atmosférica, en ambos casos, el gamma crea un par electrón-positrón después de interaccionar con algún núcleo de la atmósfera. Tanto los electrones como los positrones se desplazan por la atmósfera emitiendo fotones a través del proceso *bremsstrahlung*, estos fotones vuelven a crear un par electrón-positrón y el proceso se repite. Esta segunda cascada se conoce como *cascada atmosférica electromagnética* y también se

representa de forma esquemática en la figura 2.1 (b). La energía mínima (teórica) que necesita un fotón para crear un par es aproximadamente 1.02 MeV, el doble de la energía en reposo del electrón. Pero si el rayo gamma tiene energía suficiente se pueden crear otros pares partícula-antipartícula, por ejemplo pares protón-antiprotón.



(a) cascada hadrónica. (b) cascada electromagnética.

Figura 2.1: Tipos de cascadas atmosféricas. Cuando la partículas generadora es un hadrón se le llama cascada hadrónica, en cambio, si la partícula generadora es un rayo gamma se le conoce como cascada electromagnética. Imágenes tomadas de `http://www.gae.ucm.es/`.

Conforme la cascada evoluciona, ya sea ésta electromagnética o hadrónica, llegará un momento en que su energía promedio es insuficiente para producir más partículas, a este estado de la cascada se le conoce como **máximo de la cascada** (*shower maximum* en inglés). Una cantidad útil para describir la cascada atmosférica es la cantidad de materia penetrada por ésta hasta cierto lugar en la atmósfera. La penetración atmosférica se representa con el símbolo X y se calcula integrando la densidad del aire a lo largo de la trayectoria de la cascada, desde el punto de entrada (en lo alto de la atmósfera) hasta el punto de interés. La densidad del aire tiene unidades de g/cm^3, el desplazamiento por la atmósfera tiene unidades de longitud (cm), así que las unidades de X son densidad por longitud (g/cm^2).

Una trayectoria vertical en descenso que atraviesa aproximadamente 1 000 g/cm^2 es capaz de alcanzar el nivel del mar. Este valor, 1 000 g/cm^2, se puede interpretar como una atmósfera de presión [29]. En la figura 2.2 se muestra el flujo de diferentes partículas a lo largo de la atmósfera. Se puede observar

que por arriba de los 5 km de altura (menos de 500 g/cm^2) los protones, neutrones y muones son las partículas más abundantes. Pero conforme disminuye la altitud, o aumenta X, los muones se convierten en la partícula dominante.

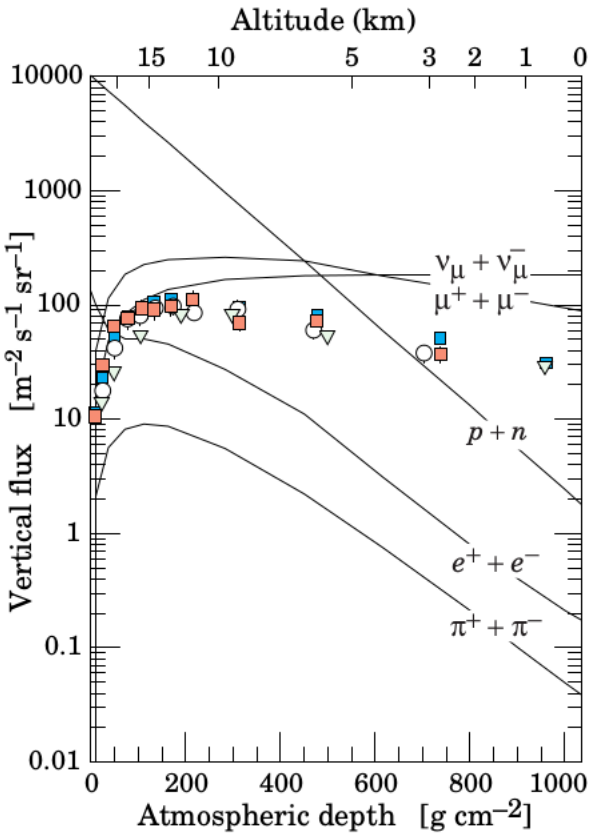


Figura 2.2: Flujo vertical de partículas en una cascadas atmosféricas. Figura adaptada de [28].

Además de partículas secundarias, las cascadas atmosféricas generan radiación Cherenkov. La luz Cherenkov es emitida por partículas que viajan a una velocidad mayor que la velocidad de la luz en el medio [31]. Existe una energía mínima, o umbral, a partir de la cual se producirá el efecto Cherenkov, dada por,

$$E_{Cherenkov} = mc^2 \frac{n}{\sqrt{n^2-1}}$$

Donde m es la masa de la partícula y n el índice de refracción del medio. Así, la energía cinética mínima que debe tener la partícula es:

$$E_K = mc^2 \left(\frac{n}{\sqrt{n^2-1}} - 1 \right)$$

Partículas con diferente masa, o en diferentes medios, tendrán diferente energía cinética umbral, como muestra la tabla 2.2. La luz Cherenkov se utiliza en algunos observatorios, ubicados en la superficie terrestre, para estudiar las cascadas atmosféricas.

Tabla 2.2: *Energía cinética mínima que deben tener diferentes partículas para emitir luz Cherenkov en aire o en agua.*

Partícula	Masa MeV	E_K (MeV) Aire (n = 1.0003)	E_K (MeV) Agua (n = 1.33)
e^+ / e^-	0.51	20.35	0.26
μ^+ / μ^-	105.66	4 208.80	54.60
π^+ / π^-	139.57	5 559.64	72.12
π^0	134.98	5 376.66	69.75
p	938.27	37 375.14	484.86

Nota: La masa de las partículas fue tomada de [30].

2.2. El observatorio HAWC.

La detección de rayos cósmicos y rayos gamma de alta energía se puede realizar tanto con detectores instalados sobre la superficie terrestre o con detectores instalados en satélites. Entre los detectores de radiación gamma instalados en la superficie terrestre una técnica destaca: la técnica *Imaging Atmosphere Cherenkov Telescope* (IACT).

En la técnica IACT se utilizan instrumentos que funcionan de forma similar a los telescopios ópticos: por medio de un espejo primario de alta reflectividad, concentran la luz Cherenkov generada durante una cascada atmosférica y la transmiten a una matriz de fotomultiplicadores ultrasensibles y ultrarápidos. La señal generada por los fotomultiplicadores es procesada en una computadora, el resultado es una imagen de la propagación de la luz Cherenkov durante la cascada de partículas [32]. Ejemplos de estos observatorios son FACT [33], en Roque de los Muchachos (Islas Canarias), que opera con un telescopio del experimento HEGRA; H.E.S.S. [34], en Namibia, integrado por cuatro telescopios de 12 metros de diámetro y uno de 28 metros; MAGIC [35], en Roque de los Muchachos, formado por dos telescopios de 17 metros; y VERITAS [36], en Arizona, con cuatro instrumentos de 12 metros. La técnica IACT tiene algunas desventajas, por ejemplo, necesita de un cielo claro y noches oscuras (sin influencia de la luz lunar) y solo pueden observar una pequeña fracción del cielo (del orden de 4×10^{-3} sr[1]) [37].

Otra técnica para detectar rayos cósmicos y rayos gamma es colocar a nivel de superficie un arreglo de detectores de partículas, conocido en inglés como EAS array (*Extensive Air Showers* array). Un arreglo EAS puede operar las 24 horas del día y observar todo el cielo por encima de él [37].

[1] Símbolo de estereorradián.

HAWC es un observatorio EAS de segunda generación y utiliza fotomultiplicadores sumergidos en agua. Cuando una partícula secundaria de la cascada atmosférica entra en contacto con el agua se genera luz Cherenkov. HAWC se ubica en el volcán Sierra Negra, Estado de Puebla, a una altura aproximada de 4 100 m sobre el nivel del mar, cerca del volcán Pico de Orizaba. Es un proyecto binacional EUA-México integrado por 29 instituciones de ambos países, una institución de Alemania y otra más de Polonia.

El observatorio opera con 300 tanques cilíndricos idénticos de 7.3 m de diámetro por 4.5 m de altura que cubren un área aproximada de 22 000 m^2 (2.2 hectáreas) [38]. Cada tanque tiene dentro de sí una bolsa con cubierta interior negra, diseñada para evitar reflexiones de la luz. A su vez, cada bolsa almacena aproximadamente 200 000 litros de agua purificada con longitudes de atenuación de 10 m o más. El agua purificada se obtiene a través de un proceso de filtrado y esterilización, la esterilización se realiza con radiación ultravioleta. La filtración del agua busca remover partículas que puedan absorber o dispersar la luz, en cambio, la esterilización busca evitar que el agua se contamine con el paso del tiempo.

Sumergidos en el agua purificada, cada tanque está equipado con un arreglo de 4 fotomultiplicadores ubicados en el fondo del mismo. Un fotomultiplicador Hamamatsu R7081-HQE de 10 pulgadas, con alta eficiencia cuántica, está colocado en el centro del tanque rodeado por tres fotomultiplicadores Hamamatsu R5912 de 8 pulgadas [39].

Entre los objetivos científicos que se pretenden estudiar con HAWC están:

- Descubrir nuevas fuentes de emisión de rayos gamma.
- Búsqueda de emisión de rayos gamma con energía de TeV en destellos de rayos gamma.
- Caracterizar la emisión de rayos gamma con energía de TeV de fuentes galácticas y extra galácticas.
- Estudios de la emisión difusa de nuestra galaxia y de fuentes extendidas.
- Dar seguimiento a regiones con emisión de neutrinos ultra energéticos y ondas gravitacionales en busca de una contra parte electromagnética de alta energía.
- Estudios de anisotropías en la dirección de llegada de rayos cósmicos.
- Estudios de física solar.
- Búsqueda de señales consistentes con la aniquilación de materia oscura.

En el Instituto de Física, se está estudiando la posibilidad de utilizar a HAWC también como un instrumento para realizar la detección indirecta de neutrinos con energía de 10^{15} eV (PeV) utilizando la técnica Earth-skimming.

2.3. La técnica Earth-Skimming.

La técnica "Earth-Skimming" fue originalmente propuesta para ser utilizada en observatorios IACT (sección 2.2), apuntando su campo de visión hacia un volumen con una gran cantidad de masa. En esos cuerpos es más factible que ocurra una interacción entre un neutrino y un nucleón a través de una corriente cargada. En estas interacciones el neutrino desaparece y se produce un leptón del mismo sabor que el neutrino incidente, es decir, un neutrino del muón daría lugar a un muón mientras un neutrino del electrón a un electrón. El nucleón que participa en la interacción pierde carga eléctrica, la cual es portada por un bosón $W^{\pm}$ que decae en un leptón cargado.

$$\overline{\nu}_e + p \rightarrow e^+ + n$$

$$\nu_\mu + n \rightarrow \mu^- + p$$

$$\overline{\nu}_\tau + p \rightarrow \tau^+ + n$$

Los lugares sugeridos para apuntar los instrumentos IACT son: cadenas de montañas, como ocurrió en el experimento Ashra [40], en Hawái; en cascadas atmosféricas iniciadas por neutrinos cerca de detectores a nivel de superficie, como en Pierre Auger [41]; o apuntar hacia el mar por debajo del horizonte, como lo está haciendo MAGIC [42].

La propuesta que se está haciendo en el Instituto de Física de la UNAM, es utilizar al volcán Pico de Orizaba como medio para convertir un neutrino en un leptón cargado y utilizar al observatorio HAWC como un detector de rastreo para estos leptones ultra-energéticos. Así, la propuesta es detectar directamente a los leptones ultra-energéticos mientras éstos atraviesan los detectores de HAWC, y no la cascada atmosférica que se produce cuando los leptones decaen. En consecuencia, se podrían detectar neutrinos cuya energía es del orden de decenas o centenas de PeVs, un rango inexplorado hasta ahora.

Un aspecto interesante de la técnica Earth-skimming es que al utilizar como medio de conversión a una montaña o volcán, este inmenso volumen de materia ayuda a filtrar las partículas cargadas provenientes de la dirección del volcán y que son generadas por cascadas atmosféricas muy inclinadas.

El objetivo principal de esta tesis es realizar simulaciones, con Geant4, para estudiar la capacidad del volcán Pico de Orizaba de absorber partículas cargadas que representan ruido en la detección indirecta de neutrinos.

2.4. Simulación del Pico de Orizaba.

Representar una estructura natural en una computadora tiene sus complicaciones. El principal problema es la memoria. Entre más detallada sea la representación es necesario guardar la posición de más puntos

para obtener más polígonos. Por eso, la representación de un volcán como el Pico de Orizaba se traduce en un problema de recursos de cómputo y tiempo. Además, una vez obtenida la versión digital del volcán es necesario simular el paso de partículas a través de él y sus interacciones. Es más razonable y sencillo buscar algún sólido que no necesite muchos puntos para ser creado y a su vez sea una buena aproximación al volcán.

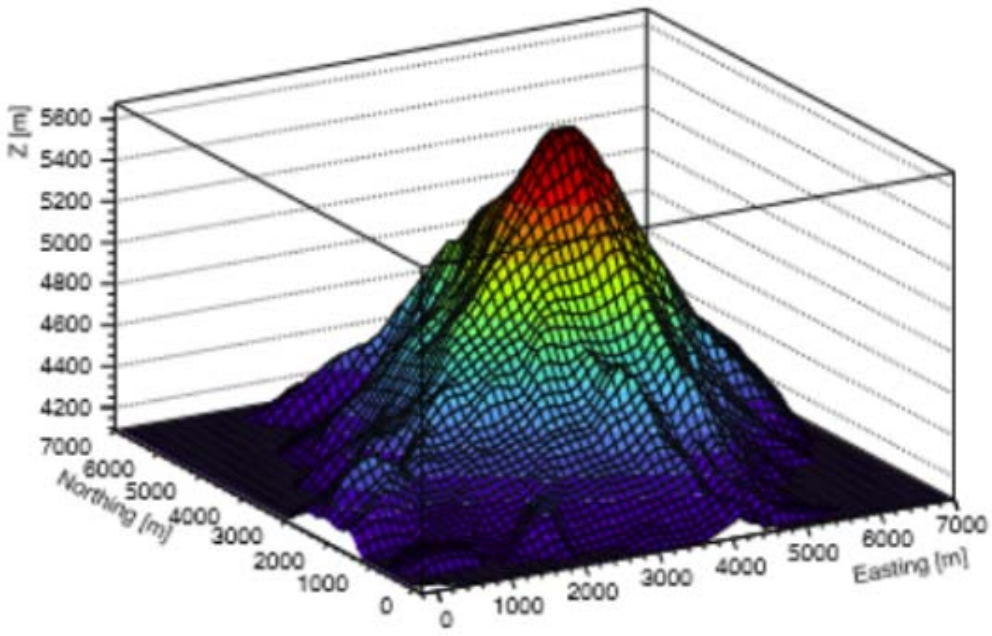
En la figura 2.3 (a) se muestra el Pico de Orizaba visto desde la altura y posición del observatorio HAWC. Cuando se proyecta ese perfil sobre el eje que va de Este-Oeste, figura 2.3 (b), y sobre el eje que va de Norte-Sur, figura 2.3 (c), se observa que el perfil del volcán se puede aproximar por dos líneas rectas que se cruzan formando un triángulo. Eso significa que a partir de la altitud de HAWC, aproximadamente 4 100 metros de altura respecto del nivel del mar, el volcán Pico de Orizaba se puede aproximar por un cono sólido y cerrado de 6 kilómetros de diámetro en su base y 1.5 kilómetros de altura. Aunque sí se usó un cono para representar al volcán en las simulaciones, sus medidas al final fueron mayores: 12 kilómetros de diámetro en su base y 3 kilómetros de altura. Los detalles sobre esta decisión se encuentran en la sección 2.5.5.

Tabla 2.3: Composición química de las rocas representativas del volcán Pico de Orizaba.

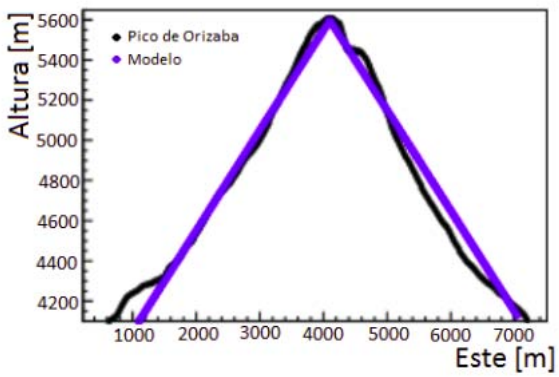
Molécula	Nombre	Porcentaje mínimo-máximo
SiO_2	Sílice	53.18 - 71.94
Al_2O_3	Alúmina	14.94 - 18.99
Na_2O	Óxido de sodio	3.48 - 5.43
Fe_2O_3	Óxido de hierro (III)	1.68 - 8.34
CaO	Cal viva	1.40 - 8.55
K_2O	Óxido de potasio	1.57 - 3.55
TiO_2	Dióxido de titanio	0.24 - 1.39
MnO	Óxido de manganeso (II)	0.06 - 0.13
MgO	Magnesia	0.40 - 7.02
P_2O_5	Óxido de fósforo (III)	0 - 0.40

Nota: Datos tomados de [46, tabla 4]. La columna *Porcentaje mínimo-máximo* muestra la abundancia mínima y máxima, en porcentaje, de las diferentes moléculas que integran a las rocas representativas del volcán Pico de Orizaba.

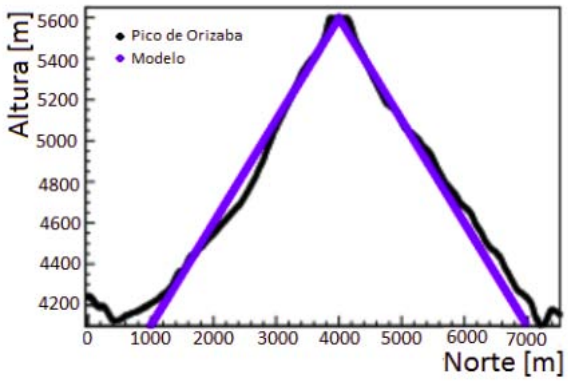
Esta representación es simplemente una aproximación, para nada el volcán es un cono. Empezando por las irregularidades que tiene el Pico de Orizaba en su superficie y las cuales el cono no posee pues su superficie es lisa. Otra diferencia es que el cono termina en un punto, mientras el volcán tiene una superficie irregular de 450 metros de diámetro [46]. Y para concluir, el cono es simétrico mientras el volcán no. Este último aspecto está documentado por Robin y Cantagrel en [47] al señalar que el volcán tiene una elevación de 3 000 metros visto desde el Oeste, y una elevación de 4 000 a 4 500 metros visto desde el Este.



(a) Pico de Orizaba visto desde HAWC



(b) Proyección sobre el eje Este-Oeste



(c) Proyección sobre el eje Norte-Sur

Figura 2.3: Perfil del volcán Pico de Orizaba con proyecciones. El origen se ubica en las coordenadas geográficas y a la altura del observatorio HAWC. Imágenes adaptadas de [45]. En (a) y (b) la línea negra es la proyección del perfil del volcán.

Por otro lado, el volcán no está conformado por un solo material. Carrasco-Núñez [46] estudió la composición química de las rocas representativas del volcán. Aunque cada roca es diferente todas tienen presencia de al menos diez moléculas, como se muestra en la tabla 2.3. El dióxido de silicio (SiO_2) es la molécula más abundante.

En la simulación, el volcán está hecho cien por ciento de SiO_2 con una densidad de $\rho = 2.65$ g/cm^3, que es la densidad estándar de una roca [45].

2.5. Simulaciones.

Como se realizaron tres simulaciones, se describe cada una por separado. Las primeras dos son simulaciones de muones penetrando en agua, ambas se describen a detalle en las secciones 2.5.3 y 2.5.4. La tercer simulación es sobre muones penetrando en el volcán Pico de Orizaba, su correspondiente descripción se ubica en la sección 2.5.5.

El límite superior de energía para las partículas es 10 TeV. Este límite no tiene que ver con la física sino con Geant4 mismo. De acuerdo a [7, sección 7.1.1] y [48, apéndice B.2], en Geant4 10 TeV es la máxima energía tanto en las tablas de sección transversal (cross section) como en los cálculos de pérdida de energía (dE/dx). Para muones y partículas pesadas los procesos de pérdida de energía se pueden extender a través de modelos. En el caso de los muones el límite superior se puede extender hasta 1000 PeV [7]. El límite inferior de energía en todas las simulaciones es 30 GeV. Esta energía se eligió de forma arbitraria a partir del espectro a 75° respecto del zenit que se muestra en la figura 2.4.

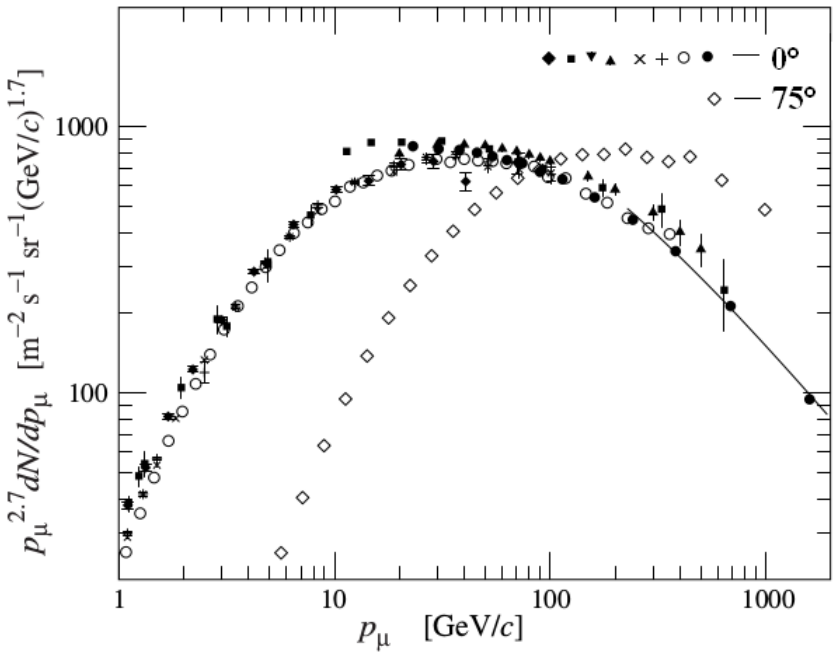


Figura 2.4: Espectro de muones para dos ángulos, 0° y 75° respecto del zenit. Imagen adaptada de [28]. Datos: ♦ [49], ■ [50], ▼ [51], ▲ [52], × y + [53], ○ [54], ● [55], y ◇ [56].

2.5.1. Principales procesos físicos.

En este trabajo se utilizan muones de carga eléctrica negativa. Así que los procesos físicos relevantes son aquellos en los que éstas y las partículas secundarias que se generan participan. El rango de energía que utilizamos es de 30GeV hasta 10 TeV.

Muones: Cuando los muones penetran en algún material, la pérdida de energía por ionización es el proceso dominante antes de alcanzar la energía critica. Ésta es definida como la energía a la cuál las pérdidas por ionización son iguales a la suma de pérdidas de energía por los otros procesos radiactivos (Bremsstrahlung, creación de pares e interacción nuclear) [43]. Para muones esto ocurre cerca de los 700 GeV en roca estándar [44, tabla IV-6]. Si la energía es superior, los proceso de Bremsstrahlung y creación de pares e^-/e^+ son más importantes. El procesos de creación de pares es el dominante cuando la energía del muón es mayor a 1 TeV. La interacción entre muones y núcleos empieza a ser relevante cuando la energía supera los 20 TeV, una energía que supera el rango considerado en este trabajo.

Electrones: La pérdidas por ionización son importantes si la energía es menor a 1 GeV. Para energías más altas el proceso de pérdida de energía por Bremsstrahlung es el más relevante. Así, los electrones irán emitiendo radiación hasta que el proceso de ionización sea dominante y finalmente sean capturados por un átomo.

Rayos gamma: El efecto fotoeléctrico y la dispersión Rayleigh no son dominantes en la gran mayoría de los materiales cuando la energía es superior a 1 GeV. En cambio, la dispersión Compton y la producción de pares electrón-positrón son los procesos relevantes en el rango de energía de este trabajo.

2.5.2. Simulación de los procesos físicos.

El código que simula los procesos físicos se tomo del ejemplo avanzado de Geant4 `gammaray_telescope`. Este ejemplo simula un telescopio espacial que analiza rayos gamma. El código se divide en varios archivos: un archivo para los procesos de electrones, positrones y gammas; un archivo para habilitar procesos de decaimiento; un archivo para los procesos de muones y tauones; uno más para piones, kaones, protones, antiprotones, neutrones y antineutrones; y un archivo para los procesos de iones: alpha, He-3, tritio y deuterio.

Estamos interesados particularmente en los procesos para electrones, muones, positrones y radiación gamma. Debido a que son las principales partículas que se generan en la interacción de los μ^- con el volcán.

Los procesos para electrones son: dispersión múltiple, ionización y Bremsstrahlung.

Los procesos para positrones son: dispersión múltiple, ionización, Bremsstrahlung y aniquilación. Este último simula la aniquilación de un positrón con un electrón atómico, es decir, con un electrón del volcán todavía unido a un átomo.

Los procesos para radiación gamma son: efecto fotoeléctrico, dispersión Compton y creación de pares electrón-positrón.

Los procesos para μ^- son: ionización, Bremsstrahlung, creación de pares electrón-positrón, dispersión múltiple y captura de muones por núcleos.

Los procesos para μ^+ son: ionización, Bremsstrahlung, creación de pares electrón-positrón y dispersión múltiple.

Observe que los procesos para electrones, μ^- y gammas son precisamente los que necesitamos.

2.5.3. Paso de muones por una caja de agua.

El objetivo de esta simulación es conocer cuánta energía cinética pierden muones de diferente energía tras cruzar un kilómetro de agua. Se definieron dos sólidos: el sólido `World` y una caja llena de agua con 100 metros de largo, 100 metros de ancho y un kilómetro de altura. El material del sólido `World` es `G4_Galactic` (vacío), mientras el material de cilindro es `G4_WATER` (agua pura).

La simulación se repitió diez veces y en cada ocasión la energía de los muones fue diferente. La primera vez se simularon muones de 30 GeV, la segunda vez muones de 50 GeV, seguido de muones con 100 GeV, 300 GeV, 500 GeV, 800 GeV, 1 TeV, 3 TeV, 5 TeV y, por último, muones de 10 TeV.

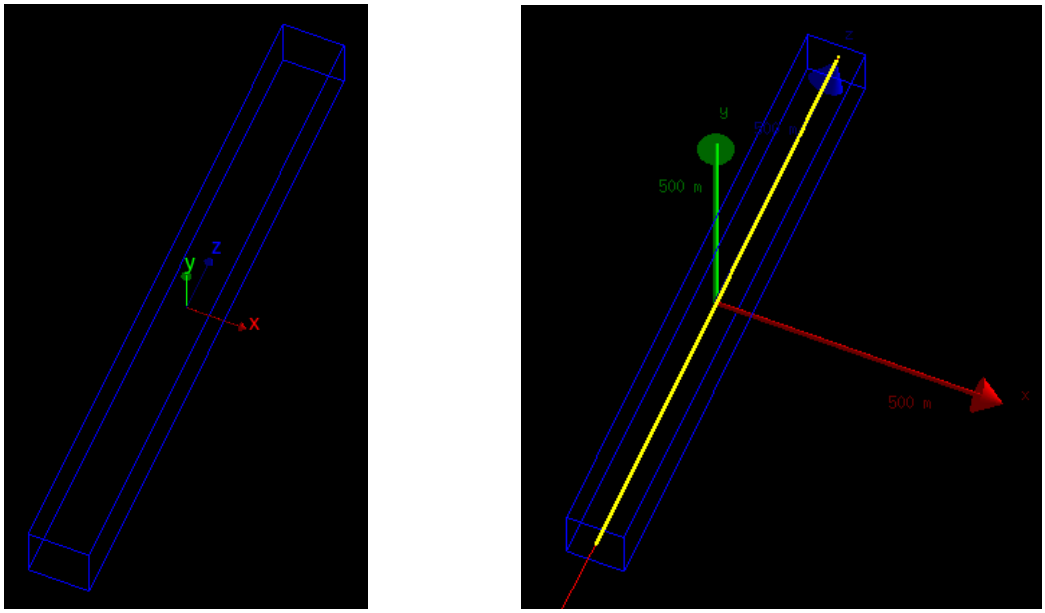
Cerca de cien mil muones (10^5) fueron lanzados en cada simulación. Se lanzaron desde el exterior de la caja hacía ésta, desde una distancia de 10 metros uno detrás de otro, es decir, cada muón era disparado cuando la simulación del muón precedente había concluido. En todas las simulaciones se guardó la distancia recorrida por el muón dentro de la caja y su energía cinética perdida.

2.5.4. Paso de muones por un tanque de HAWC.

El objetivo de la simulación es conocer cuánta energía cinética pierden muones de diferente energía después de atravesar un cilindro lleno de agua con las mismas dimensiones que un tanque de HAWC. Dos sólidos se declararon para este fin: el sólido `World` hecho del material `G4_Galactic` (vacío) y un cilindro de 7.3 metros de diámetro y 4.5 metros de altura hecho del material `G4_WATER` (agua pura). El cilindro se muestra en la figura 2.6.

Del mismo modo que en la simulación de muones atravesando una caja llena de agua (sección 2.5.3), se hicieron diez simulaciones con muones de diferente energía. La primer simulación se realizó con muones de 30 GeV, la segunda con muones de 50 GeV, después se simularon muones de 100 GeV, 300 GeV, 500 GeV, 800 GeV, 1 TeV, 3 TeV, 5 TeV y, por último, muones de 10 TeV.

En cada simulación se lanzaron cerca de cien mil muones (10^5) desde el exterior del tanque a una distancia de 5 metros respecto del centro del mismo. La información almacenada durante la simulación fue la distancia recorrida por el muón dentro del cilindro y su energía cinética perdida.



(a) Caja llena de agua (b) Muón cruzando la caja

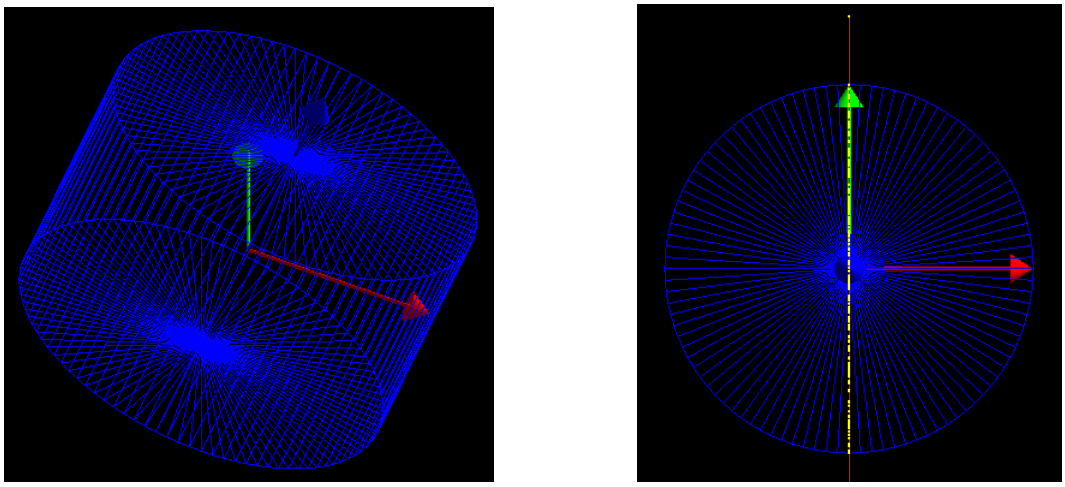
Figura 2.5: En (a) se muestra la caja llena de agua usada en esta simulación. La caja está rodeada de vacío y tiene un kilómetro de altura. En (b) un muón de 300 GeV viaja por el interior de la caja y la cruza totalmente. Los punto amarillos son lugares donde Geant4 simula las interacciones del muón. Las flechas de colores son los eje coordenados: eje X en rojo, eje Y en verde, y eje Z en azul.

2.5.5. Muones atravesando el volcán Pico de Orizaba.

Esta simulación es diferente a las demás ya que en las dos simulaciones anteriores se quería conocer cuanta energía pierden los muones al pasar por un cuerpo lleno de agua, pero en ésta se quiere conocer cuantos muones son capaces de penetrar en roca sin ser absorbidos. La simulación consiste en lanzar muones de diferente energía, entre 30 GeV y 10 TeV, y dirigirlos hacia un cono sólido hecho de sílice (SiO_2) con el objetivo es conocer cuántos de ellos son capaces de cruzarlo sin ser absorbidos. Aquellos muones que logran atravesar el volcán siguen su viaje hacia un cuerpo de agua que representa a todo el observatorio HAWC.

Tal como se mencionó en la sección 2.4, el cono sólido representa al volcán Pico de Orizaba. La idea original era que el cono fuera de 6 kilómetros de diámetro en su base y 1.5 kilómetros de altura, pero se decidió hacerlo con otras dimensiones. Por otro lado, HAWC es representado por una caja llena de agua con un tamaño igual al ocupado por los 300 tanques del observatorio HAWC.

Para explicar los detalles de la simulación se dividió esta sección en tres partes: en la primer parte se describen algunos problemas que se tuvieron a lo largo de la construcción y prueba del código; la segunda tiene información sobre la forma, las dimensiones y la posición de los sólidos dentro de la simulación; en la tercer parte se dan los detalles de la simulación.



(a) Tanque lleno de agua. (b) Muón viajando por el tanque.

Figura 2.6: En (a) se muestra el tanque lleno de agua creado para esta simulación. El tanque esta rodeado de vacío. En (b) un muón de 300 GeV cruza el cilindro, los punto amarillos son los lugares donde Geant4 actualiza el estado del muón, por ejemplo: calcula la energía perdida o las desviaciones en el trayecto del muón (dispersión). Las flechas de colores son los eje coordenados: eje X en rojo, eje Y en verde, y eje Z en azul.

Errores durante la simulación.

El primer problema que se tuvo en la simulación fue al crear los muones. Se observó que muones muy horizontales, con un ángulo entre 0° y 5° respecto del horizonte, provocan un error durante la simulación cuando alcanzan la superficie del cono sólido, lo que lleva a Geant4 a realizar un proceso infinito. Lo normal es que estos muones alcancen la superficie del cono sólido y entonces el proceso `G4Transportation` calcule un desplazamiento que les permita entrar en el cono y continuar con la simulación. Cuando ocurre este error, el proceso `G4Transportation` hace que el muón ingrese al cono, después lo regresa al sólido `World` y posteriormente lo vuelve a introducir en el volcán. Este proceso se repite infinidad de veces y es necesario detener la simulación.

La primer medida que se tomó para resolver este problema fue cambiar la semilla de la simulación con el fin de entender que estaba pasando. Se probaron cerca de 100 semillas y se encontró que para la mayoría este comportamiento aparece después de simular poco más de 800 partículas, pero para otras después de simular unas 20 000 partículas. Conforme se cambiaban estas semillas se notó que los muones que caen en este comportamiento se generan cerca del horizonte. Esto nos hizo creer que el problema eran los bordes del cono por lo que se repitió la simulación evitando generar muones muy horizontales. Como resultado, disminuyó la frecuencia con que aparecía este error, incluso se llego a eliminar cuando los muones se generaron por arriba de 5° sobre el horizonte. De esta forma, se tenían dos opciones para resolver este problema: restringir la simulación y no crear muones muy horizontales o extender las dimensiones del cono y simular muones horizontales. Al final la solución que se eligió fue una combinación de ambas: se duplicaron las dimensiones originales del cono pero como el problema

siguió apareciendo se restringió la generación de muones horizontales haciendo que éstos siempre se crearan por arriba de 1° sobre el horizonte. Con esto se logró que en la mayoría de las simulaciones se alcanzara a simular más de 9 000 partículas sin interrupción.

El segundo problema es interno de Geant4 y no se buscó una solución para éste. En las simulaciones donde los muones tienen una energía menor a 110 GeV, Geant4 aborta con frecuencia la simulación después de crear algunas partículas, mostrando el siguiente mensaje de error:

```
++ G4CrossSectionDataStore::GetCrossSection ERROR: no isotope cross section found
++ In G4CrossSectionDataStore.cc, line 198: no applicable data set found for the
isotope
++ Unrecoverable error in the method GetMeanFreePath of hadElastic
++ TrackID = 2830 ParentID = 1 N16
++ Ekin(GeV) = 0.000291746; direction = (-0.0561644,0.868175,0.49307)
++ Position (mm) = (-17173.2,-31617.5,-1367.28); material G4_WATER
++ PhysicalVolume <HAWC_PV>
++ Cross section is not available
```

Después de este mensaje la simulación termina y para continuar es necesario comenzar la simulación con una nueva semilla.

El tercer problema ocurrió al guardar los datos de la simulación en archivos. La información que se guarda durante la simulación es: la distancia que recorrió el muón en el cono, la distancia que recorrió en HAWC, la energía que perdió en el cono, la energía que perdió en HAWC y las coordenadas angulares del vector de momento al ingresar en HAWC. Cada vez que un muón es simulado su información es almacenada en archivos, un archivo por variable. Se supone que los datos almacenados en estos archivos deben ser correctos, pero en varias ocasiones la información guardada nos hizo dudar. Lo mejor será mencionar un ejemplo. En la simulación de muones con 500 GeV de energía inicial, el archivo donde se guarda la energía que pierden al ir penetrando en el cono tiene los siguientes datos: muón 1 perdió 500 GeV, muón 2 perdió 500 GeV, muón 5 perdió ¿5.82 10^{-13} GeV?. Es una cantidad ridículamente pequeña, pero es posible que ese muón recorriera una distancia igualmente pequeña dentro del cono. Y resulta que sí, porque todos lo muones que pierden una energía del orden de 10^{-13} GeV dentro del cono, recorren distancias del orden de 10^{-12} metros en ese sólido.

Ante este escenario hay dos caminos: eliminar esos datos o considerarlos. Se eligió eliminarlos, es decir, considerar que eso muones nunca penetraron en el cono aunque sí fueron simulados. En la tabla 2.4 se muestran dos columnas, en una se indica el número de muones simulados y en la otra el número de muones que entraron en la montaña pero se eliminaron. En esa misma columna se muestra el número de muones eliminados como un porcentaje respecto al número de muones simulados.

También se encontró que algunos muones al entrar en HAWC guardaban ceros en los archivos, es decir, que en los archivos donde se almacenó la distancia recorrida y la energía perdida al penetrar en HAWC,

en vez de una distancia o una energía muy pequeña se guardaron ceros. Esto no tiene un significado único: por un lado puede significar que son muones que salen del cono con suficiente energía como para seguir existiendo en la simulación pero a la vez una energía tan pequeña que son absorbidos inmediatamente por el cuerpo de agua o es posible que sea basura, es decir, que ha pasado algo durante la simulación de esos muones en particular y Geant4 guarda una energía y una distancia que nada tiene que ver con lo que realmente pasó. Independientemente de la razón, al no disponer de suficiente información para elegir algún escenario, no es adecuado considerar estos datos y es mejor desecharlos, es decir, pensar que esos muones nunca penetraron en el cuerpo de agua. En la tablas 2.5 se muestra cuántos datos fueron eliminados en cada simulación por este motivo, tanto en número como en porcentaje respecto al número de muones simulados.

Tabla 2.4: *Número de muones simulados y número de muones que penetraron en el cono pero la energía perdida y la distancia recorrida fue muy pequeña y se eliminaron.*

Energía	Muones Simulados	Muones que entraronel en el cono pero se eliminaron	
30 GeV	57 564	10 502	18.24 %
70 GeV	354 906	65 252	18.39 %
110 GeV	398 308	73 488	18.45 %
500 GeV	233 427	43 163	18.49 %
800 GeV	273 900	50 795	18.54 %
1 TeV	363 058	67 209	18.51 %
3 TeV	178 339	33 015	18.51 %
6 TeV	153 992	28 483	18.50 %
8 TeV	154 149	28 497	18.49 %
10 TeV	321 589	59 591	18.53 %

Tabla 2.5: *Número de muones simulados por energía y número de muones que penetraron en el cuerpo de agua pero fueron eliminados porque su energía perdida y su distancia recorrida fue igual a cero.*

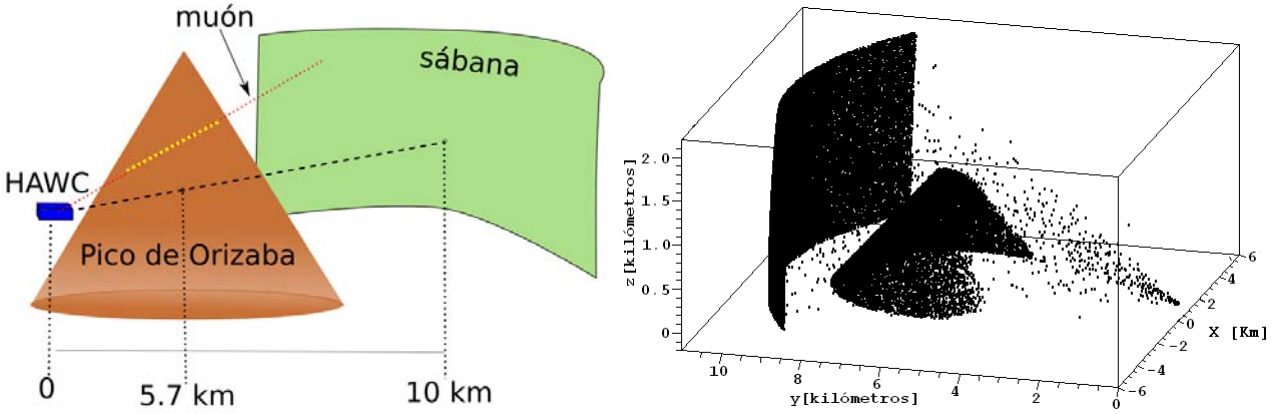
Energía	Muones Simulados	Muones que entran en HAWC pero se eliminaron	
30 GeV	57 564	14	0.02 %
70 GeV	354 906	146	0.04 %
110 GeV	398 308	296	0.07 %
500 GeV	233 427	1 700	0.73 %
800 GeV	273 900	3 663	1.38 %
1 TeV	363 058	6 142	1.69 %
3 TeV	178 339	8 667	4.86 %
6 TeV	153 992	10 573	6.87 %
8 TeV	154 149	11 388	7.39 %
10 TeV	321 589	24 726	7.69 %

Cuerpos simulados.

Los únicos sólidos que hay en la simulación son el sólido `World`, el cono sólido y el prisma rectangular lleno de agua. El prisma rectangular representa a todo el observatorio HAWC y se encuentra en el centro del sólido `World`. Aunque HAWC es un arreglo de 300 tanques cilíndricos llenos de agua, como se explicó en la sección 2.2, el observatorio se puede aproximar por un prisma rectangular de 140 metros de largo, 140 metros de ancho y 4.5 metros de altura [45]. Para llenarlo de agua se definió el material del prisma rectangular como `G4_WATER` (agua).

El cono representa a la parte del volcán Pico de Orizaba que se encuentra a una altitud igual o mayor a la altitud del observatorio HAWC, que es aproximadamente 4 100 metros sobre el nivel del mar. Las

dimensiones finales del cono son 12 kilómetros de diámetro y 3 kilómetros de altura. Se colocó a 5.7 kilómetros de distancia del centro de HAWC, esta distancia fue tomada de [45] que a su vez la obtuvo de datos del INEGI. El material del cono sólido es sílice (SiO_2) con una densidad de 2.65 g/cm^3, que es la densidad estándar de una roca. Se eligió este material porque es el más abundante dentro de las diferentes rocas que se pueden encontrar en las cercanías del volcán, como se muestra en la tabla 2.3. En la figura 2.7 se muestra una representación de los sólidos presentes en la simulación.



(a) Esquema con la ubicación de todos los sólidos.

(b) Simulación de muones de 30 GeV.

Figura 2.7: En (a) se muestra un esquema con la ubicación de todos los sólidos presentes en la simulación del Pico de Orizaba. El observatorio de rayos gama HAWC es representado por un prisma rectangular y el volcán por un cono de 3 km de altura y 12 km de diámetro en su base. La sábana es la región donde se generan los muones y se ubica a 10 km del centro de HAWC. Todos los muones generados se desplazan hacia el observatorio y cuando uno de ellos entra en el volcán, Geant4 marca con un punto amarillo los lugares donde simula sus interacciones. En (b) se muestra la posición de varios muones en una simulación de 30 GeV. Cada punto negro representa la posición de alguno de los muones simulados a lo largo de su trayectoria. La silueta del cono se puede distinguir fácilmente, detrás de él se encuentra la cortina donde se generan los muones. HAWC se encuentra frente al cono en la región donde se concentran los puntos.

En la sección 2.4 se mencionó que, visto desde la posición del observatorio de rayos de gamma HAWC, el Pico de Orizaba se puede aproximar por un cono sólido de 6 kilómetros de diámetro y 1.5 kilómetro de altura. También se ha dicho que las dimensiones finales del cono fueron el doble de eso: 12 kilómetros de diámetro y 3 kilómetros de altura. Ambas afirmaciones se cumplieron en el código de la simulación porque la posición del cono se eligió de tal forma que su altura fuera de 1.5 kilómetros visto desde el horizonte de HAWC. Para lograrlo solo se tuvo que bajar el centro del cono. En la figura 2.8 se muestra en color rojo las dimensiones originales del cono y en color verde las dimensiones finales. Como se ve en esa figura, es posible sobreponer un cono sobre el otro simplemente eligiendo de forma adecuada la posición de sus respectivos centros.

Además de estos sólidos se definió una región llamada sábana, en donde se generan los muones. La sábana es una pequeña sección de esfera con 10 kilómetros de radio cuyo centro se ubica en el mismo centro

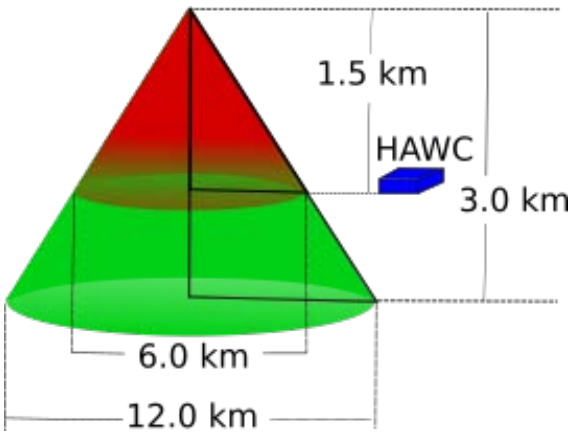


Figura 2.8: Se muestra en color rojo las medidas originales que iba a tener el volcán Pico de Orizaba en la simulación, 1.5 km de altura y 6 km de diámetro en su base. Debido a problemas durante la simulación se aumentaron las dimensiones del volcán, el tamaño final se muestra en color verde: 3 km de altura y 12 km de diámetro.

de HAWC. Las dimensiones de la sábana están en coordenadas esféricas: su radio es de 10 kilómetros; ϕ toma valores entre 60° y 120°; θ abarca desde 78° hasta 89°. Las partículas que se generan en la sábana se desplazan siempre hacia el centro de HAWC. Estas dimensiones, excepto por el radio, no son arbitrarias, se eligieron para garantizar que la mayoría de los muones generados entraran en el cono.

Simulaciones realizadas.

Se comentó al principio de esta sección un problema que se tuvo durante la simulación, en el que Geant4 realiza un proceso infinito cuando los muones son muy horizontales, entre 0° y 5° sobre el horizonte. Este proceso impide crear una cantidad arbitraria de muones en una misma simulación, por lo que es necesario repetirla cambiando la semilla hasta obtener la cantidad de muones deseada. Por este motivo se decidió no simular la misma cantidad de muones por energía inicial, sino repetir cada simulación 40 veces. Para algunas energías se logró este objetivo y para otras no. En un principio se tenían planes para simular únicamente muones con energía inicial de 30 GeV, 110 GeV, 1 TeV y 10 TeV. Cada una de estas simulaciones se repitió 40 veces con semillas diferentes y se logró simular la cantidad de muones que indica la tabla 2.6.

Tabla 2.6: *Número de muones simulados con energía inicial de 30 GeV, 110 GeV, 1 TeV y 10 TeV.*

Energía	Muones simulados
30 GeV	57 564
110 GeV	398 308
1 TeV	363 058
10 TeV	321 589

Tabla 2.7: *Número de muones simulados con energía inicial de 70 GeV a 8 TeV.*

Energía	Muones simulados
70 GeV	354 906
500 GeV	233 427
800 GeV	273 900
3 TeV	178 339
6 TeV	153 992
8 TeV	154 149

Estas energías se seleccionaron por diversas razones: 30 GeV simplemente para tener un mínimo de energía, que resultó coincidir con la energía del máximo en el flujo vertical de muones (el espectro de 0° en la figura 2.4); 110 GeV es una energía cercana al máximo en el flujo horizontal de muones (el espectro de 75° en la figura 2.4); 1 TeV para tener una energía intermedia y 10 TeV porque es el límite superior en las tablas de pérdida de energía y de sección transversal (cross section) que maneja Geant4 [7, 48].

Después se tuvo la necesidad de simular otras energías que ayudaran a llenar huecos de información. En ese momento hacía falta conocer que pasa cuando los muones tienen una energía entre 100 GeV y 1 TeV, pero también cómo es la absorción cuando su energía está entre 1 TeV y 10 TeV. Así que se realizaron simulaciones con otras 6 energías: 70 GeV, 500 GeV, 800 GeV, 3 TeV, 6 TeV y 8 TeV. En la tabla 2.7 se muestra cuántos muones se simularon en cada una de estas seis simulaciones. Solo la simulación de 70 GeV se repitió 40 veces, la simulación de 500 GeV se repitió 29 veces, la de 800 GeV 34 veces, la simulación de 3 TeV 23 veces, y las de 6 TeV y 8 TeV 19 veces. La principal razón fue el tiempo, entre más alta es la energía de los muones más tiempo se necesita para que concluya.

CAPÍTULO 3

RESULTADOS Y DISCUSIÓN.

Los resultados de este capítulo se muestran por simulación, es decir, por cada simulación hay una sección con sus respectivos resultados. Son tres las simulaciones que se ejecutaron para este capítulo del trabajo.

La primer simulación tiene como objetivo conocer cuánta energía cinética pierden muones de diferente energía tras cruzar un kilómetro de agua, para ello se hicieron pasar miles de muones por una caja llena de agua y se obtuvo información sobre la energía y la distancia que recorrieron dentro de la caja. Más detalles sobre la simulación se pueden encontrar en la sección 2.5.3. Los resultados se muestran en la sección 3.1.

El objetivo de la segunda simulación es conocer cuánta energía cinética pierden muones de diferente energía después de atravesar un cilindro lleno de agua que tiene las mismas dimensiones que un tanque de HAWC. La simulación consiste de miles de muones atravesando un cilindro de 7.3 metros de diámetro y 4.5 metros de altura lleno de agua. Los detalles de la simulación se ubican en la sección 2.5.4 y los resultados en la sección 3.2.

La tercer y última simulación tiene como fin conocer cuántos muones son absorbidos por un cono de sílice (SiO_2), que representa al volcán Pico de Orizaba, cuando éstos son lanzados a una caja llena de agua que representa al observatorio de rayos gamma HAWC. Una descripción más detallada de esta simulación está disponible en la sección 2.5.5, los resultados se muestran en la sección 3.4.

3.1. Resultados de muones atravesando una caja de agua.

Se realizaron diez simulaciones de tal forma que la energía cinética inicial de los muones en cada una se fue incrementando. Se simularon en primer lugar muones de 30 GeV, seguido de muones de 50 GeV, 100 GeV, 300 GeV, 500 GeV, 800 GeV, 1 TeV, 3 TeV, 5 TeV y por último de 10 TeV. En cada simulación

los muones viajaron por el interior de una caja llena de agua hasta recorrer una distancia máxima de un kilómetro.

Analizando de forma individual cada simulación, se observa que hay espectros tanto en la energía que pierden los muones como en la distancia que recorren. Estos espectros se muestran en el apéndice de la tesis y corresponden a las figuras 9-12. Solo en las simulaciones con muones de 30 GeV, 50 GeV y 100 GeV el cien por ciento de los muones pierde toda su energía inicial antes de salir de la caja. Es a partir de 300 GeV que una cierta cantidad de muones no pierde toda su energía y logran cruzar la caja de agua.

El hecho de que haya espectros en la energía y en la distancia implica que no es posible decir con exactitud la distancia va a recorrer un muón que penetra en la caja o la energía va a perder. Sin embargo es posible dar un número que represente de forma adecuada todos esos posibles valores, ese número es la media[1]. Así, en este trabajo se considera a la media aritmética como la cifra más adecuada para representar los resultados de cada simulación.

Existe un modelo que nos permite verificar si los resultados de la simulación son consistentes con ésta. De acuerdo a Nakamura [28], Klimushin [57] y Groom [44], la energía que pierden los muones al penetrar en la materia se puede aproximar por la siguiente relación.

$$-\frac{d\mathrm{E}}{dX} = \mathrm{a}\,(\mathrm{E}) + \mathrm{b}\,(\mathrm{E})\,\mathrm{E} \tag{3.1}$$

Donde E es la energía del muón, X representa la cantidad de materia penetrada[2] y tiene unidades de g/cm^2, a(E) representa las pérdidas de energía por ionización y b(E) representa las pérdidas de energía por radiación: creación de pares e^+/e^-, emisión por bremsstrahlung e interacción fotonuclear. El valor de a(E) y b(E) depende de la energía que tenga el muón pero también de las características de la materia que penetra.

Para el caso de agua pura [57] indica que,

[1] La media aritmética es el número x que hace mínima la suma de cuadrados,

$$\min_x\{(x_1-x)^2+\cdots+(x_N-x)^2\}$$

Donde las x_i son mediciones. La derivada respecto a x de esa suma, cuando se hace igual a cero, nos dice que la media es,

$$x=\frac{x_1+\cdots+x_N}{N}$$

En un histograma, la media se obtiene sumando los productos de la frecuencia relativa de cada intervalo o *bin* (f_k) con el valor que representa a ese intervalo (x_k).

$$x=f_1x_1+\cdots+f_kx_k$$

[2] Esta cantidad se mencionó por primera vez en la sección 2.1 como la penetración atmosférica de una cascada. Aquí tiene el mismo significado solo que aplicado a agua.

$$\left.\begin{array}{l} \mathrm{a\,(E)} = \alpha = 2.67 \text{ MeV cm}^2 \text{ g}^{-1} \\ \mathrm{b\,(E)} = \beta = 3.40\ 10^{-6} \text{ cm}^2 \text{ g}^{-1} \end{array}\right\} \text{ si } 30 \text{ GeV} < \mathrm{E} \leq 35.3 \text{ TeV}$$

Mientras [44] da una extensa variedad de valores para a(E) y b(E) dependiendo de la energía.

Tabla 3.1: *Valores de a(E) y b(E) tomados del trabajo de Groom en el rango de 30 GeV a 10 TeV.*

Energía GeV	a (E) MeV cm^2 g^{-1}	b (E) 10^{-6} cm^2 g^{-1}
30	2.64	1.83
50	2.68	1.97
100	2.79	2.27
300	2.92	2.64
500	2.95	2.72
800	3.03	2.90
1 000	3.05	2.96
3 000	3.19	3.16
5 000	3.22	3.21
10 000	3.33	3.32

Nota: Los valores de a(E) y b(E) fueron tomados de la tabla II-28 de [44]. El rango de energía de esa tabla es 10 MeV - 100 TeV pero aquí se muestran solo los valores de a(E) y b(E) relevantes para el trabajo.

Si a(E) y b(E) son constantes, α y β respectivamente, se puede integrar la ecuación 3.1 de la siguiente manera,

$$\int_{\mathrm{E}_0}^{\mathrm{E}_\mu} \frac{d\mathrm{E}}{\alpha + \beta \mathrm{E}} = -\int_0^X dX'$$

Donde E_0 es la energía inicial del muón y E_μ es la energía del muón después de penetrar X cantidad de materia. Al integrar se obtienen dos ecuaciones:

- La energía del muón después de atravesar X cantidad de materia.

$$\mathrm{E}_\mu = \left(\frac{\alpha}{\beta} + \mathrm{E}_0\right) e^{-\beta X} - \frac{\alpha}{\beta} \tag{3.2}$$

- La máxima distancia que puede recorrer un muón con energía inicial E_0

$$r = \frac{1}{\rho\beta} \ln\left(1 + \frac{\beta}{\alpha}\mathrm{E}_0\right) \tag{3.3}$$

Donde ρ es la densidad del medio, en este caso agua pura, y aparece en la ecuación 3.3 porque $X = r\rho$. La ecuación 3.3 se obtiene de 3.2 al hacer $E_\mu = 0$.

Tabla 3.2: *Distancia que recorren los muones y energía que pierden al penetrar en la caja llena de agua. También se muestra la energía que pierden de acuerdo a la ecuación 3.2 y la distancia máxima que pueden penetrar en agua de acuerdo a la formula 3.3.*

Energía	Distancia recorrida			Energía perdida		
	Distancia Máxima		Simulación Geant4	Groom	Klimushin	Simulación Geant4
	Groom	Klimushin				
GeV	m			GeV		
30	112.26	110.27	118.79 ± 9.49	30.0	30.0	30.0 ± 0
50	183.08	181.55	191.31 ± 17.48	50.0	50.0	50.0 ± 0
100	344.32	352.53	363.53 ± 38.52	100.0	100.0	100.0 ± 0
300	909.87	951.61	939.34 ± 114.92	300.0	300.0	293.10 ± 9.07
500*	1 393.87	1 449.07	987.93 ± 69.48	377.36	370.46	359.34 ± 55.47
800*	1 960.46	2 066.08	994.35 ± 49.20	464.41	456.93	436.77 ± 110.82
1 000*	2 288.77	2 415.53	995.85 ± 41.97	520.65	514.57	487.41 ± 146.16
3 000*	4 370.11	4 625.94	998.78 ± 23.24	1 085.17	1 091.03	993.02 ± 510.23
5 000*	5 576.64	5 873.58	999.12 ± 19.92	1 647.0	1 667.49	$1\,501.11 \pm 885.62$
10 000*	7 215.63	7 705.53	999.60 ± 13.43	3 106.10	3 108.64	$2\,772.39 \pm 1\,809.70$

Notas: *Estas energías no se pueden comparar porque dentro de la simulación la distancia máxima que pueden recorrer los muones en agua es 1 km. Los valores de las constantes α y β para la columna *Klimushin* son $\alpha = 2.67$ MeV cm^2 g^{-1} y $\beta = 3.40\ 10^{-6}$ cm^2 g^{-1}, para la columna *Groom* se muestran en la tabla 3.1. En las columnas "simulación Geant4" los datos corresponden a la media de los histogramas que se encuentran en el apéndice, la incertidumbre es el valor de la desviación estándar (RMS).

Se escribió con C++ un programa que calcula la distancia máxima que puede penetrar un muón en agua usando la ecuación 3.3 y la energía que pierde al cruzar un kilómetro de agua usando la ecuación 3.2. En ambos casos se tomaron las mismas energías que se usaron en las simulaciones. La tabla 3.2 muestra los resultados arrojados por este programa, además de la distancia y la energía media que pierden los muones en la simulación. Para el caso de distancia recorrida, la ecuación 3.3 y lo obtenido con Geant4 es comparable solo hasta los 300 GeV. Para 500 GeV o energías mayores el máximo obtenido con la ecuación 3.3 supera el kilómetro de longitud y como la caja mide exactamente eso la comparación carece de sentido.

Una representación gráfica de la tabla 3.2 se puede ver en las figuras 3.1 y 3.3, donde se grafica la energía perdida y la distancia recorrida por los muones respectivamente. En esas imágenes se puede ver que si bien los resultados obtenidos con Geant4 para esta simulación son diferentes a lo que predice la teoría, estas diferencias no son lo suficientemente grandes como para desconfiar por completo de las simulaciones. Es importante notar que conforme aumenta la energía de los muones las diferencias son cada vez

mayores. La figura 3.2 (a) muestra que tan diferentes son los resultados de Geant4 y los resultados de energía perdida por parte de la teoría. Esta figura indica que ambos métodos difieren menos de 7 % si la energía de los muones es $\leq$ 1 TeV y la diferencia aumenta a 11 % si la energía es de 10 TeV. En la figura 3.3 (b) se hace la misma comparación pero para la distancia recorrida. Para 30 GeV la diferencia es del 6 % cuando se usan las constantes α y β de Groom y 8 % cuando se usan las constantes de Klimushin. Conforme la energía aumenta la diferencia disminuye, para 300 GeV la diferencia entre Geant4 y Klimushin es del 1 % y para Groom es de alrededor del 4 %. Para energías superiores la comparación no tiene sentido porque la ecuación 3.3 predice que los muones pueden recorrer más de un kilómetro en agua.

3.2. Resultados de muones atravesando un tanque de HAWC.

Se realizaron diez simulaciones con muones de diferente energía. Al igual que en la simulación de una caja llena de agua (sección 3.1), primero se realizó una simulación con muones de 30 GeV, otra con muones de 50 GeV, seguido por muones de 100 GeV, 300 GeV, 500 GeV, 800 GeV, 1 TeV, 3 TeV, 5 TeV y una décima simulación con muones de 10 TeV.

En estas simulaciones los muones atraviesan un tanque cilíndrico lleno de agua con las mismas dimensiones que un tanque de HAWC: 7.3 metros de diámetro y 4.5 metros de altura. Los muones cruzan el cilindro de forma perpendicular a su eje de simetría, es decir, lo cruzan por una trayectoria radial recorriendo como máximo 7.3 metros en agua.

A diferencia de la simulación de muones atravesando una caja de agua, en esta simulación todos los muones cruzan el tanque sin ser absorbidos. Esto concuerda con la máxima distancia que pueden recorrer en agua y que se muestra en la tabla 3.2. En dicha tabla se puede ver que para la energía más baja, 30 GeV, la máxima distancia que pueden recorrer los muones en agua está entre 110 metros y 112 metros, distancia 15 veces superior a los 7.3 metros del tanque.

La energía que pierden los muones por cruzar el tanque se puede observar en la tabla 3.3 o gráficamente en la figura 3.1. La figura muestra la energía media que pierden en cada simulación, mientras la figura 3.2 (b) muestra qué tan diferentes son los resultados obtenidos con Geant4 y con la ecuación 3.2. Si la energía inicial de los muones es $\leq$ 1 TeV la diferencia es inferior al 10 %, alcanzando el 16 % cuando la energía es igual a 10 TeV.

Tabla 3.3: *Energía media que pierden los muones al cruzar un cilindro lleno de agua con las mismas dimensiones que un tanque de HAWC, energía que pierden de acuerdo a las ecuación 3.2 y distancia media que recorren en el cilindro.*

Energía	Energía perdida			Distancia recorrida
	Groom	Klimushin	Geant4	Geant4
GeV				m
30	1.97	2.02	1.96 ± 0.94	$7.3 \pm 2.2x10^{-4}$
50	2.03	2.07	2.04 ± 1.43	$7.3 \pm 1.7x10^{-5}$
100	2.20	2.19	2.18 ± 2.42	$7.3 \pm 6.7x10^{-6}$
300	2.70	2.69	2.64 ± 6.33	$7.3 \pm 8.5x10^{-6}$
500	3.14	3.19	3.10 ± 10.79	$7.3 \pm 1.6x10^{-3}$
800	3.90	3.93	3.67 ± 15.64	$7.3 \pm 8.5x10^{-6}$
1 000	4.39	4.43	4.03 ± 18.10	$7.3 \pm 8.5x10^{-6}$
3 000	9.23	9.38	8.41 ± 60.92	$7.3 \pm 9.7x10^{-6}$
5 000	14.04	14.34	12.31 ± 96.08	$7.3 \pm 5.6x10^{-6}$
10 000	26.61	26.74	22.55 ± 187.89	$7.3 \pm 4.0x10^{-6}$

Notas: Los datos de las columnas *Geant4* son los valores medios de los histogramas que aparecen en el apéndice de la tesis, la incertidumbre corresponde al valor de la desviación estándar (RMS). El valor de las constantes α y β para la columna *Klimushin* son α = 2.67 MeV cm^2 g^{-1} y β = 3.40 10^{-6} cm^2 g^{-1}, para la columna *Groom* se muestran en la tabla 3.1.

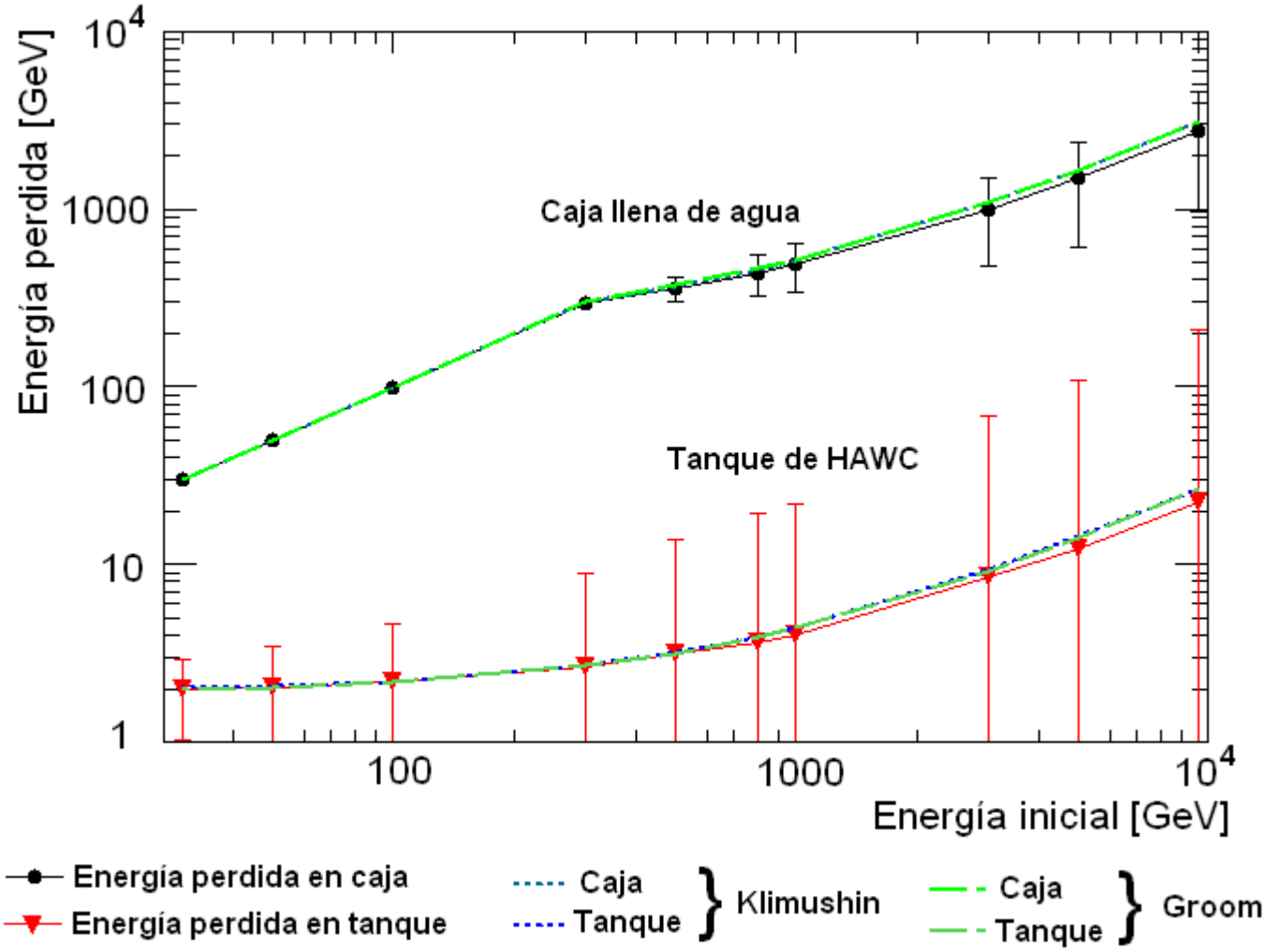
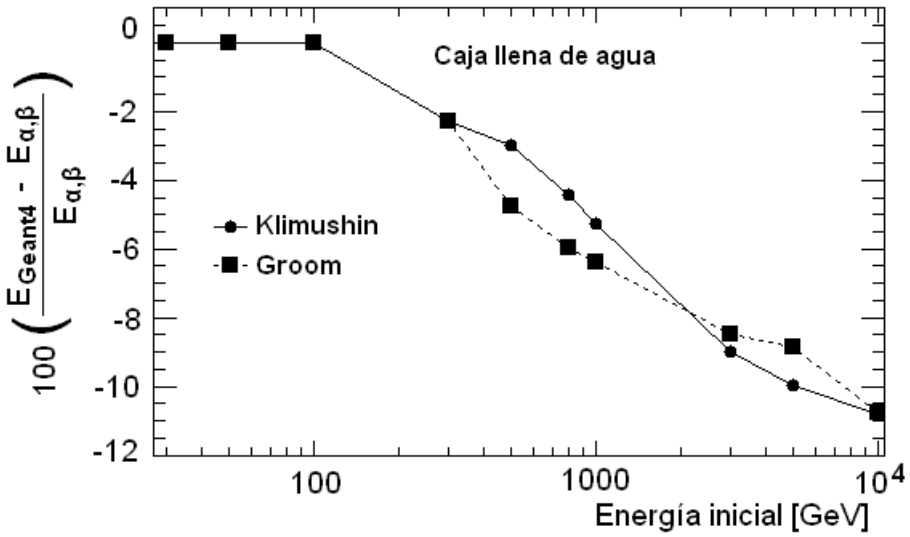
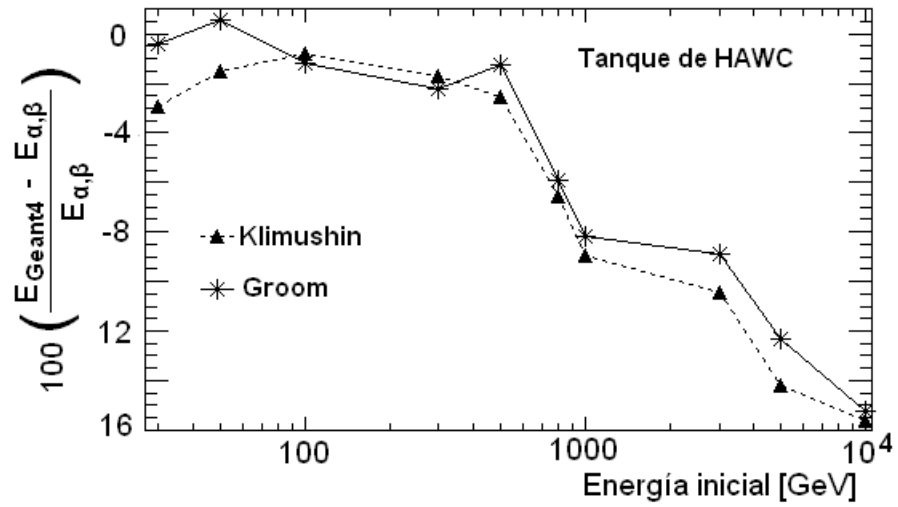


Figura 3.1: Energía media que pierden los muones en la simulación de la caja llena de agua (línea continua superior) y en la simulación del tanque lleno de agua (línea continua inferior). Se muestra con líneas punteadas la energía que pierden de acuerdo a la ecuación 3.2. Los valores de las constantes α y β son, $\alpha = 2.67$ MeV cm^2 g^{-1} y $\beta = 3.40\ 10^{-6}$ cm^2 g^{-1}, para Klimushin [57]. mientras los valores valores de α y β para Groom [44] se muestran en la tabla 3.1.



(a) Muones penetrando en una caja llena de agua.



(b) Muones penetrando en un tanque de HAWC.

Figura 3.2: Comparación entre la energía que pierden los muones en las simulaciones (E_{Geant4}) y la energía que pierden de acuerdo a la ecuación 3.2 ($E_{\alpha,\beta}$). Los valores de las constantes α y β son, α = 2.67 MeV cm^2 g^{-1} y β = 3.40 10^{-6} cm^2 g^{-1}, para Klimushin [57] y para Groom [44] se muestran en la tabla 3.1.

3.3. Conclusiones concernientes a las simulaciones de muones atravesando una caja y un tanque llenos de agua.

Tras la discusión que se llevó a cabo en la sección 3.1 sobre los resultados obtenidos con la simulación de muones viajando por una caja llena de agua, y la discusión en la sección 3.2 de los resultados obtenidos en la simulación de muones viajando por un tanque de HAWC lleno de agua, se concluye que:

(a) El valor medio que se obtiene en los histogramas de energía perdida y distancia recorrida es adecuado para representar de forma global los resultados de la correspondiente simulación.

(b) Suponiendo que la ecuación 3.1 es una adecuada aproximación de la energía que pierde un muón al penetrar en un medio, los resultados de energía perdida y distancia recorrida que se obtienen con la simulación de Geant4 son consistentes.

3.4. Resultados de muones atravesando el volcán Pico de Orizaba.

En las simulaciones anteriores, muones penetraban en un cuerpo lleno de agua y se guardaba la siguiente información: cantidad de energía perdida y la cantidad de metros recorridos antes de salir del cuerpo o antes de perder toda su energía y ser absorbidos. Se comentó que la energía perdida también se puede obtener a partir de la ecuación 3.1 y se comprobó que ambos métodos ofrecen resultados muy similares.

En esta simulación también se tiene el interés de conocer la penetración de los muones pero esta vez en roca y no en agua. La simulación crea muones que se dirigen hacia un cono sólido, que es la representación del volcán Pico de Orizaba. Si los muones logran cruzarlo entonces siguen viajando hacia un cuerpo de agua que representa a todo el observatorio HAWC. Entre el cono y el cuerpo de agua solo hay vacío. El objetivo es contar la cantidad de muones que penetran en el cono sólido y además logran atravesarlo sin ser absorbidos. Este conteo se realiza en dos pasos: primero hay que conocer el porcentaje de muones simulados que penetran en el cono y después conocer cuántos de esos muones llegan a HAWC. El primer conteo se realiza en el tabla 3.4 donde se muestra la cantidad de muones generados por simulación, el número de muones que penetraron en el cono y el porcentaje que éstos representan respecto al número de muones simulados.

El siguiente paso es averiguar cuántos muones de los que penetraron en el cono llegan a HAWC. En la tabla 3.5 se muestra la cantidad de muones que penetraron en el volcán por simulación, cuántos de ellos llegaron a HAWC y el porcentaje que éstos representan respecto al número de muones que penetraron en el cono. Este porcentaje se representa gráficamente en la figura 3.4.

Tabla 3.4: *Número de muones simulados y porcentaje que penetra en el volcán.*

Energía	Muones Simulados	Muones que penetran en el volcán	
30 GeV	57 564	24 536	42.62 %
70 GeV	354 906	151 942	42.81 %
110 GeV	398 308	170 716	42.86 %
500 GeV	233 427	100 641	43.11 %
800 GeV	273 900	117 959	43.07 %
1 TeV	363 058	156 368	43.07 %
3 TeV	178 339	76 852	43.09 %
6 TeV	153 992	66 486	43.17 %
8 TeV	154 149	66 598	43.20 %
10 TeV	321 589	138 855	43.18 %

Tabla 3.5: *Número de muones que penetran en el volcán y porcentaje que llega a HAWC.*

Energía	Muones que entran al volcán	Muones que llegan a HAWC	
30 GeV	24 536	0	0.0 %
70 GeV	151 942	6	0.004 %
110 GeV	170 716	32	0.02 %
500 GeV	100 641	647	0.64 %
800 GeV	117 959	1 775	1.50 %
1 TeV	156 368	3 448	2.21 %
3 TeV	76 852	8 552	11.12 %
6 TeV	66 486	14 967	22.51 %
8 TeV	66 598	19 046	28.60 %
10 TeV	138 855	46 817	33.72 %

Estos datos indican que si la energía de los muones es $\leq$ 1 TeV, poco más del 95 % de los muones es absorbido por el volcán. Pero al superar esta energía el número de muones que cruza el volcán aumenta de forma significativa, para 3 TeV menos del 90 % de los muones es absorbido y para 10 TeV solo el 66 %. Es importante considerar que el flujo de muones a 75° respecto del zenit, que se muestra en la figura 2.4, tiene su máximo por 200 GeV y empieza a disminuir por 700 GeV. Esto quiere decir que la mayoría de los muones que se generan en las cascadas atmosféricas y que vienen de la dirección del volcán son absorbidos.

Una forma de comprobar que estos resultados son correctos es comparar la energía que pierden los muones al penetrar en el cono con la energía que pierden en roca estándar utilizando el modelo descrito en la sección 3.1. La comparación solo es posible si se conoce la distancia que viajan los muones en rocas estándar, de los contrario no se puede usar la formula 3.2. Para este fin, se utiliza la información de la tabla 3.6, que muestra la distancia media que recorren los muones en la simulación (r_{Geant4}). Con r_{Geant4} y la formula 3.2 se puede obtener la energía que pierden los muones al penetrar en roca estándar ($E_{\alpha,\beta}$) y esa energía se puede comparar con la energía media que pierden al penetrar en el cono durante la simulación (E_{Geant4}). Ambas energías se muestran en la tabla 3.6 y en la figura 3.5.

Como se puede ver en la figura 3.5 (a) los resultados obtenidos con Geant4 y con la ecuación -dE/dX = $\alpha + \beta$E no son tan diferentes. La diferencia entre ambos métodos se puede ver en la figura 3.5 (b) donde se calcula la discrepancia entre ambas técnicas. La diferencia es en realidad muy pequeña, menor al 5 %.

Tabla 3.6: *Energía media que pierden los muones al penetrar en el volcán durante la simulación (E_{Geant4}) y energía que pierden al penetrar en roca estándar utilizando un cálculo analítico ($E_{\alpha,\beta}$).*

Energía inicial	α MeV cm^2 g^{-1}	β 10^{-6} cm^2 g^{-1}	r_{Geant4} m	$E_{\alpha,\beta}$ GeV	E_{Geant4} GeV
30 GeV	2.31	2.43	51.64 ± 4.28	30	30.0 ± 0.15
70 GeV	2.34	2.57	112.51 ± 11.81	70	69.99 ± 0.70
110 GeV	2.44	3.03	169.09 ± 19.85	110	109.96 ± 1.48
500 GeV	2.58	3.62	587.92 ± 108.81	500	497.85 ± 22.35
800 GeV	2.65	3.86	811.96 ± 172.49	800	793.71 ± 47.72
1 TeV	2.68	3.93	931.71 ± 210.79	1 000	989.74 ± 67.44
3 TeV	2.79	4.17	$1\,584.36 \pm 461.06$	3 000	$2\,911.73 \pm 314.02$
6 TeV	2.83	4.23	$1\,983.39 \pm 639.52$	5 944.58	$5\,732.38 \pm 728.46$
8 TeV	2.91	4.33	$2\,128.53 \pm 712.75$	7 920.25	$7\,598.04 \pm 1\,007.47$
10 TeV	2.93	4.36	$2\,239.56 \pm 766.36$	9 872.18	$9\,464.09 \pm 1\,281.15$

Nota: Los valores de α y β fueron tomados de la tabla IV-6 de [44]. La distancia recorrida y la energía perdida durante la simulación corresponden a la media de los histogramas que se muestran en el apéndice de la tesis. La incertidumbre corresponde a la desviación estándar (RMS) de esos histogramas.

Una pregunta que no se ha contestado es por cuál región del cono penetran los muones sin ser absorbidos. Para contestar a esta pregunta se tienen las figuras 3.6 a 3.8. Éstas muestran los ángulos de llegada que registra HAWC cuando un muón penetra en él. Para obtener esas figuras se realizaron nuevas simulaciones. Éstas se hicieron con el fin de garantizar que el número de muones fuera el mismo para todas las energías: 48 934 muones simulados.

Las figuras 3.6 a 3.8 son histogramas 2D, donde el eje horizontal representa el ángulo φ, el mismo que se suele usar en coordenadas esféricas; el eje vertical representa la altitud en grados, donde el horizonte tiene un valor de 0° y el zenit un valor de 90°; el color indica la frecuencia. Además de esto, también se muestra la silueta del volcán. Esta silueta tienen una altitud de 14.7° y una anchura en su base de casi 56° que abarca desde $\varphi_{min} = 62°$ hasta $\varphi_{max} = 118°$.

Tal como se ve en los histogramas 2D, el perfil del cono y el perfil de los muones que no son absorbidos no encajan exactamente. Algunos muones parecen venir de una región exterior al volcán y sin embargo dibujan muy bien una silueta que se puede interpretar como la sombra del cono. Una explicación puede ser la dispersión debido a las múltiples interacciones que tienen en el interior del cono y que provocan que éste se vea más ancho de lo que realmente es.

Si la energía es menor a 1 TeV los muones solo pueden pasar por los lados del cono, que es la región más delgada del mismo. Cuando la energía es mayor a 1 TeV los muones empiezan a cruzar por una región

que es cercana al centro y que es más gruesa comparada con su periferia.

La cima del cono no tiene muones en los histogramas 2D, aunque esta zona es tan gruesa como la periferia. Cuando se estaba probando la simulación se restringió la creación de muones a una sección de esfera con el fin de garantizar que la mayoría de éstos penetrara en el cono. Esto provocó que su parte más alta no recibiera ningún muón durante la simulación, no tanto por un error en la construcción sino por eficiencia. En la simulación, la cima del cono es una región muy pequeña comparada con la sección de esfera donde se generan los muones. Los muones se crean de tal forma que todos se dirigen al centro de HAWC; si se intenta que estos muones pasen por la cima del cono solo una fracción de ellos va a penetrar en él. Generar muchos muones y que solo una fracción penetre en el cono no ayuda porque nos interesa saber que pasa cuando estos penetran. Por esa razón se decidió que la cima del cono no recibiera muones, para aumentar la cantidad que sí penetra en el volcán.

El resumen, cuando la energía inicial de los muones en menor o igual a 1 TeV estos pueden penetrar y salir del volcán sin ser absorbidos únicamente por su periferia. Si penetran por el centro, su energía debe ser mayor a 1 TeV para que no puedan ser absorbidos.

3.5. Conclusiones concernientes a la simulación de muones atravesando el volcán Pico de Orizaba.

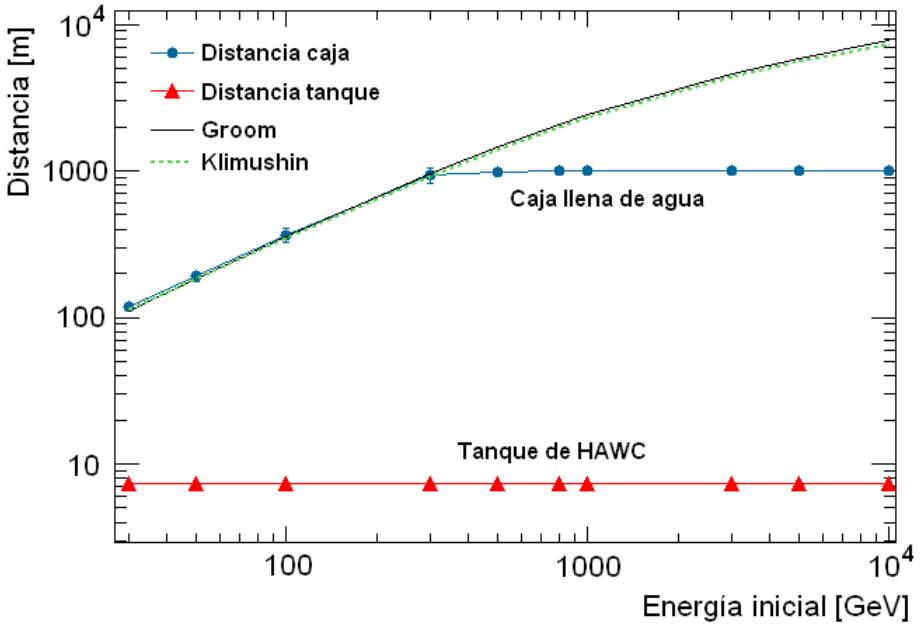
De los resultados mostrados y discutidos en la sección 3.4 se ha llegado a las siguientes conclusiones.

(a) La energía media que pierden los muones en el volcán como resultado de las simulaciones es consistente con la pérdida de energía en roca estándar expresada por la relación:

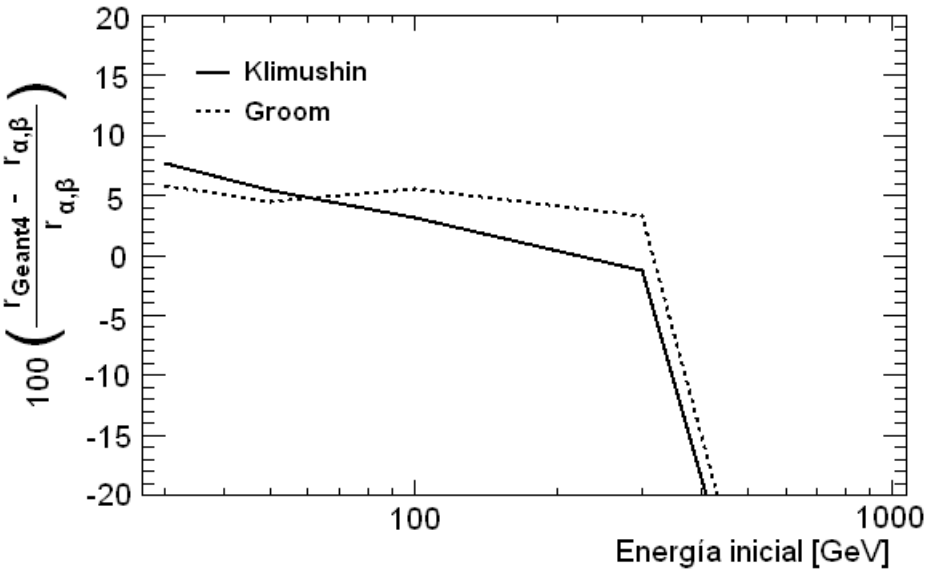
$$-\frac{d\mathrm{E}}{dX} = \alpha + \beta\mathrm{E}$$

(b) Si los muones que penetran en el volcán tienen una energía $\leq$ 1 TeV, se espera que al menos el 95 % del flujo sea absorbido, como se muestra en la figura 3.4.

(c) En la simulación, es por la orilla del cono por donde los muones con energía $\leq$ 1 TeV logran penetrar y cruzar sin ser absorbidos. El cono es, en la simulación, la representación del volcán Pico de Orizaba.



(a) Distancia que recorren en agua los muones



(b) Diferencia entre la distancia obtenida en la simulación y la teoría.

Figura 3.3: En (a) se muestra la distancia que recorren los muones en la simulación de la caja llena de agua (línea continua superior), la distancia que recorren en la simulación del tanque lleno de agua (línea continua inferior) y la distancia máxima que pueden recorrer en agua de acuerdo a la ecuación 3.3. En (b) se muestra la diferencia entre la distancia obtenida con Geant4 (r_{Geant4}) y la calculada con la ecuación 3.3 ($r_{\alpha,\beta}$). Las constantes usadas para calcular la máxima distancia son, $\alpha = 2.67$ MeV cm^2 g^{-1} y $\beta = 3.4\ 10^{-6}$ cm^2 g^{-1}, para Klimushin [57] y para Groom [44] los valores de α y β se muestran en la tabla 3.1.

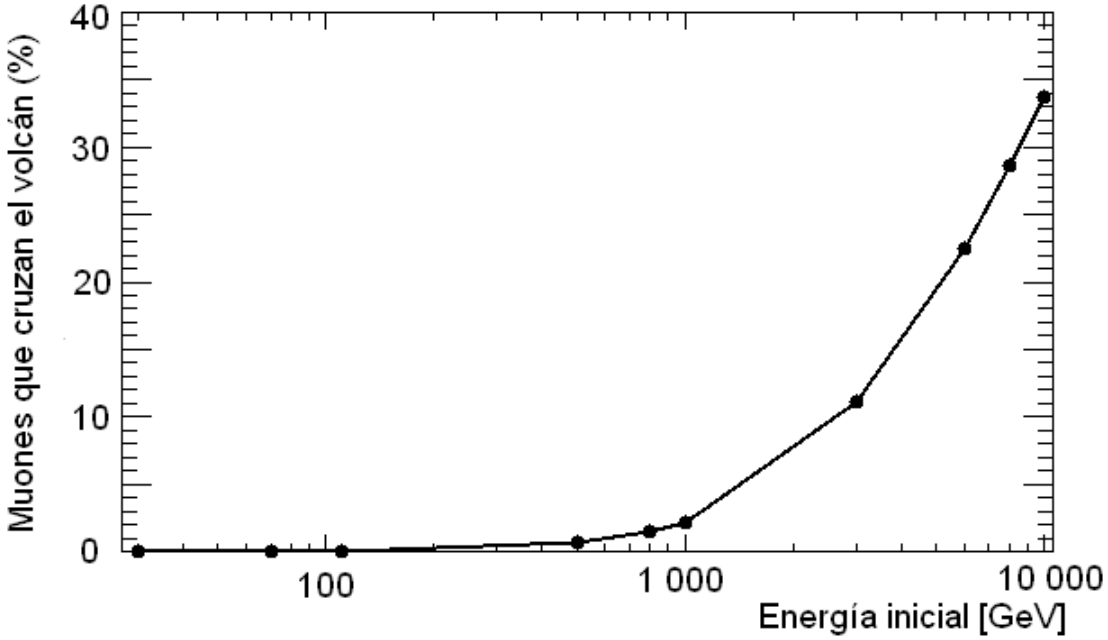
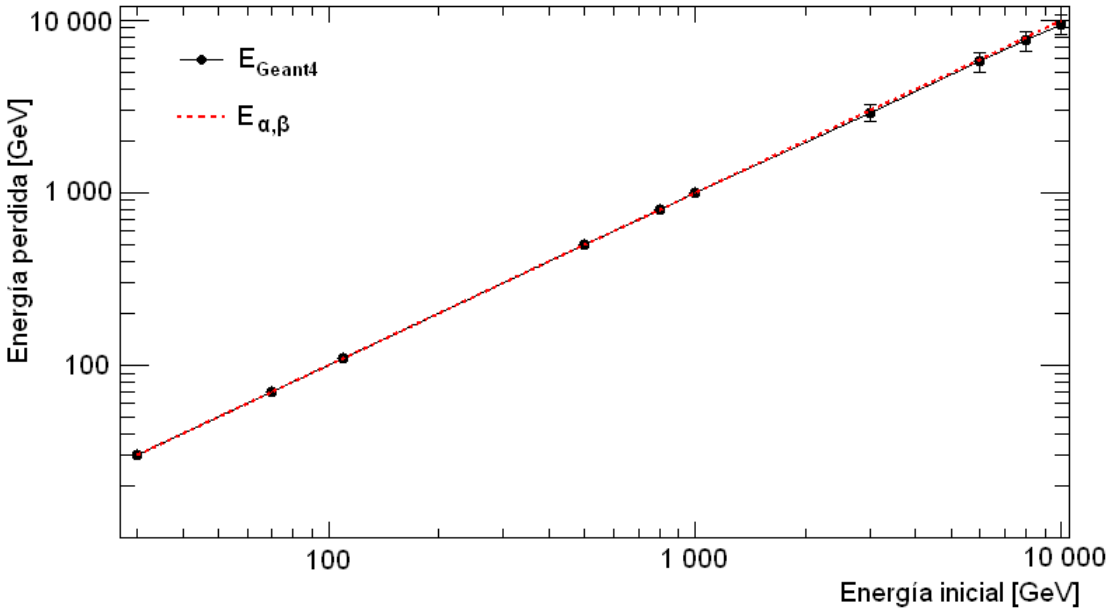
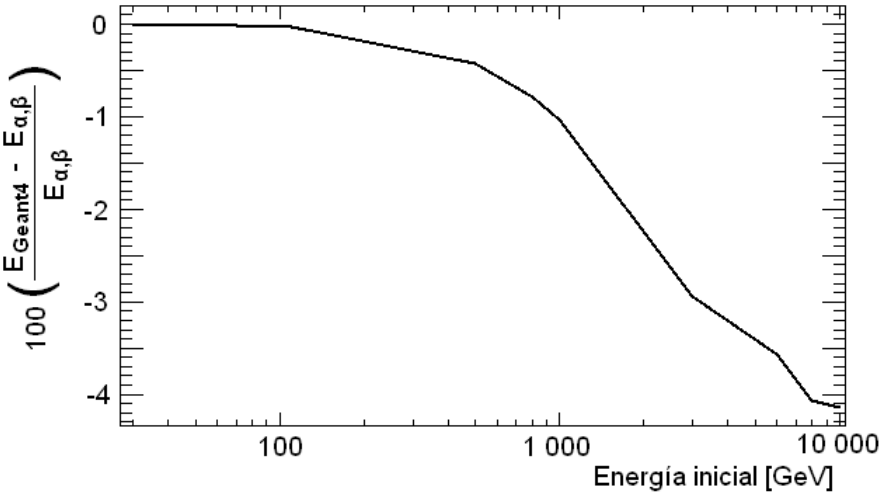


Figura 3.4: Porcentaje de muones simulados que cruzan el volcán Pico de Orizaba sin ser absorbidos y que penetran en HAWC. Datos tomados de la tabla 3.5.



(a) Energía perdida por los muones en la simulación y al penetrar en roca estándar.



(b) Diferencia entre la teoría y la energía perdida durante la simulación.

Figura 3.5: Energía que pierden los muones al penetrar en el volcán. En (a) los puntos representan la energía media que pierden los muones en cada simulación (E_{Geant4}) y la curva punteada es la energía que pierden al penetrar en roca estándar ($E_{\alpha,\beta}$) de acuerdo a la ecuación 3.2. Los datos para esta gráfica se ubican en la tabla 3.6. En (b) se comparan ambas pérdidas de energía.

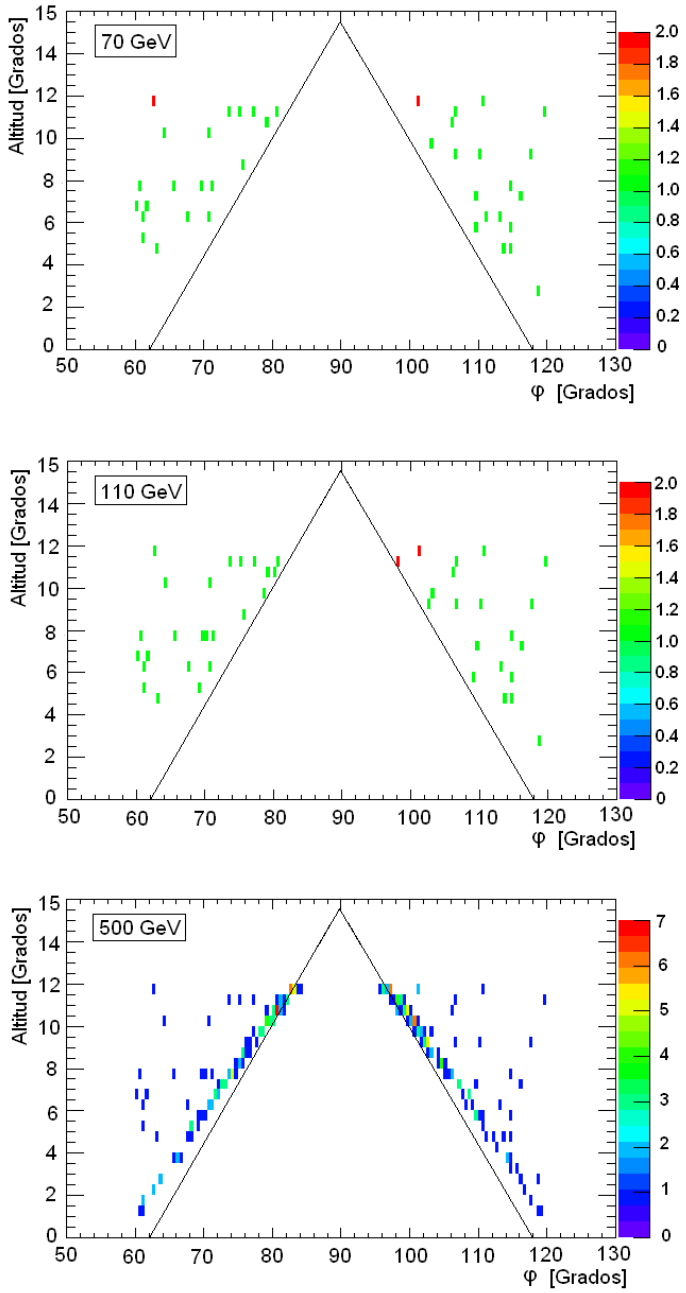


Figura 3.6: Posición angular de los muones que llegan a HAWC en las simulaciones de 70 GeV a 500 GeV. Cada histograma 2D muestra los ángulos de llegada que registra el sólido HAWC cuando un muón penetra en él. En estas imágenes se muestran todos los muones simulados, incluso los que no penetran en el volcán. El eje horizontal representa el ángulo φ que se suele usar en coordenadas esféricas, mientras el eje vertical representa la altitud en grados, donde el horizonte tiene un valor de 0° y el zenit un valor de 90°. El color indica la frecuencia y se interpreta como el número de veces que un muón cae en cada celda. El tamaño de las celdas es 0.5° de altura y 0.5° sobre φ.

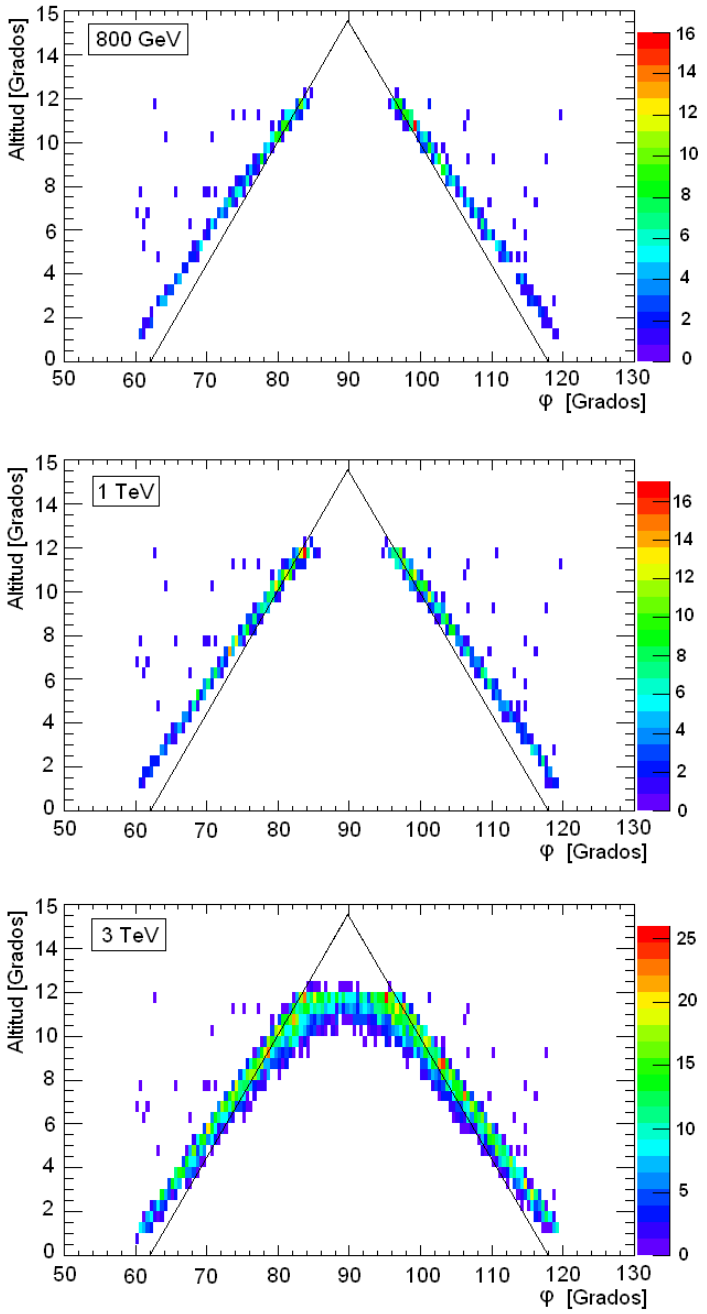


Figura 3.7: Posición angular de los muones que llegan a HAWC en las simulaciones de 800 GeV a 3 TeV. Cada histograma 2D muestra los ángulos de llegada que registra el sólido HAWC cuando un muón penetra en él. En estas imágenes se muestran todos los muones simulados, incluso los que no penetran en el volcán. El eje horizontal representa el ángulo φ que se suele usar en coordenadas esféricas, mientras el eje vertical representa la altitud en grados, donde el horizonte tiene un valor de 0° y el zenit un valor de 90°. El color indica la frecuencia y se interpreta como el número de veces que un muón cae en cada celda. El tamaño de cada celda es 0.5° de altura y 0.5° sobre φ.

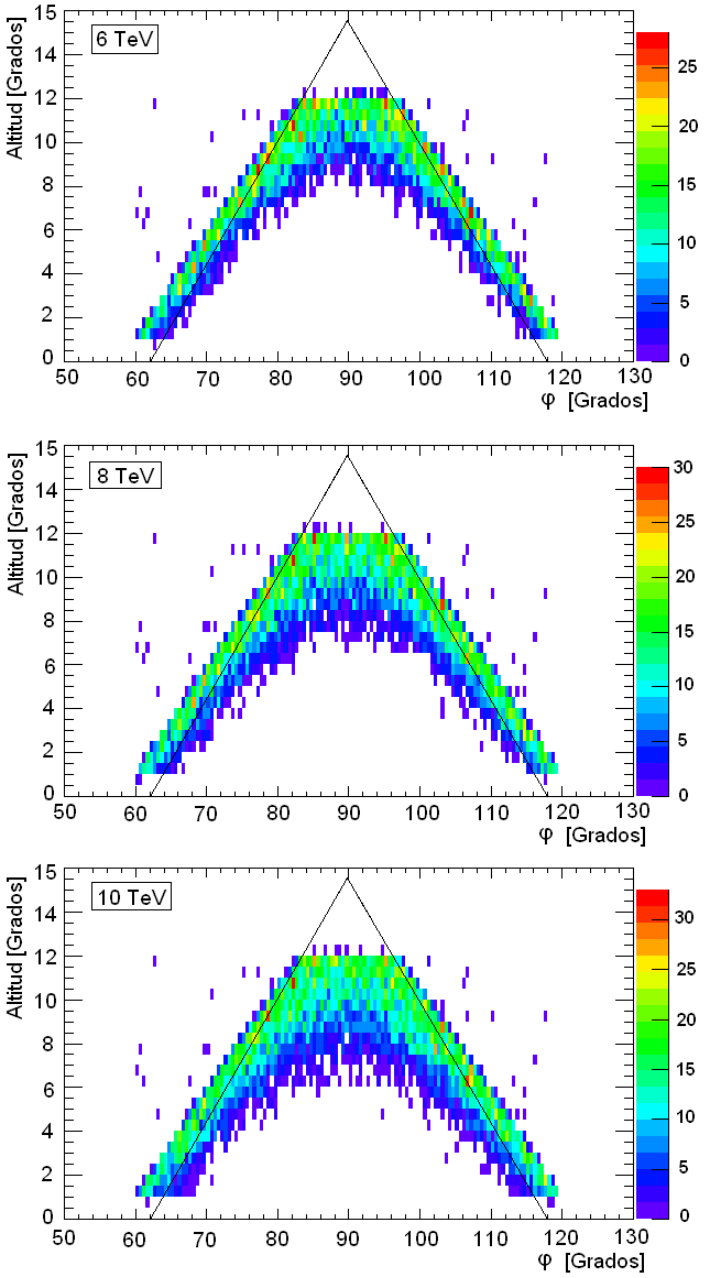


Figura 3.8: Posición angular de los muones que llegan a HAWC en las simulaciones de 6 TeV a 10 TeV. Cada histograma 2D muestra los ángulos de llegada que registra el sólido HAWC cuando un muón penetra en él. En estas imágenes se muestran todos los muones simulados, incluso los que no penetran en el volcán. El eje horizontal representa el ángulo φ que se suele usar en coordenadas esféricas, mientras el eje vertical representa la altitud en grados, donde el horizonte tiene un valor de 0° y el zenit un valor de 90°. El color indica la frecuencia y se interpreta como el número de veces que un muón cae en cada celda. El tamaño de las celdas es 0.5° de altura y 0.5° sobre φ.

CONCLUSIONES.

Se completó el manual de Geant4 que incluye temas básicos como son: las partículas y materiales que se pueden simular, los sólidos que se pueden construir y algunas técnicas para guardar los datos que se generan durante la simulación. Se incluyeron ejemplos en forma de código y algunos consejos para escoger los modelos y procesos que simulan las interacciones físicas de interés.

Los manuales existentes de Geant4 se enfocan en aspectos técnicos del software: explican la jerarquía que existe entre las clases que integran a Geant4; la clasificación de los modelos que simulan las interacciones físicas; explican qué es un evento; qué función tiene un detector; qué es un hit y muestran algún código a modo de ejemplo que, en el mejor de los casos, está comentado pero nunca se explica a detalle porqué se debe implementar cierta clase o cierto método. Tampoco explican como se comunican estas clases para que la simulación trabaje correctamente. Estas fueron las razones que motivaron a escribir un manual, esperando que pueda ayudar y motivar a otras personas a construir sus propias simulaciones.

En lo que concierne a la capacidad del Pico de Orizaba para absorber leptones cargados, la figura 3.4 muestra que los muones de energía igual o menor a 1 TeV son absorbidos casi en su totalidad ($\approx 98\,\%$), pero para energías mayores la cantidad de muones absorbidos disminuye de forma importante, para 10 TeV cerca del 65 % de los muones es absorbido. Considerando que el espectro de muones generados por cascadas atmosféricas es relevante en el intervalo de 10 GeV a 1 TeV se puede decir que el volcán absorbe casi en su totalidad el flujo de muones que lo cruza. Aunque un pequeño porcentaje, como del 2 %, lograría atravesarlo sin ser absorbido. Estos resultados abren la posibilidad de utilizar al observatorio de rayos gamma HAWC como un detector de neutrinos ultra energéticos de origen extragaláctico mediante la técnica Earth-skimming, porque los muones generados en las cascadas atmosféricas serian absorbidos por el volcán, eliminando la principal fuente de ruido.

BIBLIOGRAFÍA

[1] B Pontecorvo. Report PD-205, Chalk River Laboratory, 1946. B. Pontecorvo "Selected scientific works" p. 21. *Societa Italiana di Fisica, Bologna, Italia*, 1946.

[2] Frederick Reines Facts. Nobel Media AB 2014. `http://www.nobelprize.org/nobel_prizes/physics/lau` 26 de junio de 2017.

[3] Raymond Davis, Don S. Harmer, and Kenneth C. Hoffman. Search for neutrinos from the Sun. *Phys. Rev. Lett.*, 20:1205–1209, 1968.

[4] K. Hirata et al. Observation of a neutrino burst from the supernova SN1987A. *Phys. Rev. Lett.*, 58:1490–1493, Apr 1987.

[5] R. M. Bionta et al. Observation of a neutrino burst in coincidence with supernova 1987A in the large Magellanic Cloud. *Phys. Rev. Lett.*, 58:1494–1496, 1987.

[6] The Geant4 Collaboration. *Geant4 User's Guide for Application Developers*. 2015.

[7] The Geant4 Collaboration. *Physics Reference Manual*. 2015.

[8] S. Agostinelli et al. Geant4 - a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250 – 303, 2003.

[9] Grady Booch. *Object oriented analysis & design with application*. Pearson Education India, 2006.

[10] Bjarne Stroustrup. *The C++ programming language*. Adisson-Wesley, 4 edition, 2013.

[11] C. M. Poole, I. Cornelius, J. V. Trapp, and C. M. Langton. A CAD interface for Geant4. *Australasian Physical & Engineering Sciences in Medicine*, 35(3):329–334, 2012.

[12] Arthur Beiser. *Concepts of Modern Physics*. McGraw-Hill, 6 edition, 2003.

[13] Wick C. Haxton and Barry R. Holstein. Neutrino physics. *American Journal of Physics*, (68), 2000.

[14] G. Agnetta et al. Extensive air showers and diffused Cherenkov light detection: The ULTRA experiment. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 570(1):22 – 35, 2007.

[15] Xin Dong, Gene Cooperman, and John Apostolakis. *Multithreaded Geant4: Semi-automatic Transformation into Scalable Thread-Parallel Software*, pages 287 – 303. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[16] Ken Martin and Bill Hoffman. *Mastering CMake: updated for CMake version 2.2;[a cross-platform build system; covers installing and running CMake; details converting existing build processes to CMake; create powerful cross-platform build scripts]*. Kitware, 2006.

[17] Isaac Asimov, Jorge de Orus, and Manuel Vázquez. *Introducción a la ciencia*. Orbis, 1985.

[18] Diana Cruz-Garritz, José A Chamizo, and Andoni Garritz. *Estructura atómica: un enfoque químico*. Fondo Educativo Interamericano, 1986.

[19] Ernest Rutherford. LXXIX. The scattering of α and β particles by matter and the structure of the atom. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 21(125):669–688, 1911.

[20] Fujia Yang and Joseph H Hamilton. *Modern atomic and nuclear physics*. World Scientific Publishing Co Inc, 2010.

[21] V N Ivanchenko, O Kadri, M Maire, and L Urban. Geant4 models for simulation of multiple scattering. *Journal of Physics: Conference Series*, 219(3):032045, 2010.

[22] László Urbán. A model for multiple scattering in Geant4. Technical Report CERN-OPEN-2006-077, CERN, Geneva, 2006.

[23] H. W. Lewis. Multiple scattering in an infinite medium. *Phys. Rev.*, 78:526–529, Jun 1950.

[24] S. Goudsmit and J. L. Saunderson. Multiple scattering of electrons. *Phys. Rev.*, 57:24–29, Jan 1940.

[25] Wentzel G. Zwei Bemerkungen über die Zerstreuung korpuskularer Strahlen als Beugungserscheinung. *Z. Phys*, 40:590, 1926.

[26] A. A. Watson. Extensive Air Showers and Ultra High Energy Cosmic Rays. *Summer School in Mexico*, pages 1–50, 2002.

[27] Michael Friedlander. Un siglo de rayos cósmicos. *Investigación y Ciencia*, (433):46–47, 2012.

[28] K. Nakamura et al. *24- COSMIC RAYS*. PDG, 2012.

[29] Sonali Bhatnagar. Extensive Air Shower High Energy Cosmic Rays (II). 2009.

[30] K. A. Olive et al. *Particle Physics Booklet. Chin. Phys. C.* 2014.

[31] P. A. Cerenkov. Visible Radiation Produced by Electrons Moving in a Medium with Velocities Exceeding that of Light. *Phys. Rev.*, 52:378–379, Aug 1937.

[32] Juan Cortina and Manel Martínez. La Red de Telescopios Cherenkov. *Investigación y Ciencia*, (476):44–53, 2016.

[33] H. Anderhub et al. FACT – The first Cherenkov telescope using a G-APD camera for TeV gamma-ray astronomy. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 639:58–61, 2011. Proceedings of the Seventh International Workshop on Ring Imaging Cherenkov Detectors.

[34] F. Aharonian et al. Calibration of cameras of the H.E.S.S. detector. *Astroparticle Physics*, 22:109–125, 2004.

[35] J. Aleksic et al. The major upgrade of the MAGIC telescopes, Part I: The hardware improvements and the commissioning of the system. *Astroparticle Physics*, 72:61–75, 2016.

[36] J. Holder et al. The first VERITAS telescope. *Astroparticle Physics*, 25(6):391 – 401, 2006.

[37] R. Atkins et al. Observation of TeV Gamma Rays from the Crab Nebula with Milagro using a new background rejection technique. *The Astrophysical Journal*, 595(2):803, 2003.

[38] AU Abeysekara, R Alfaro, C Alvarez, JD Álvarez, R Arceo, JC Arteaga-Velázquez, HA Ayala Solares, AS Barber, BM Baughman, N Bautista-Elivar, et al. Observation of small-scale anisotropy in the arrival direction distribution of TeV cosmic rays with HAWC. *The Astrophysical Journal*, 796(2):108, 2014.

[39] Andrew J. Smith. HAWC: Design, Operation, Reconstruction and Analysis. 2015.

[40] Yoichi Asaoka and Makoto Sasaki. Cherenkov τ Shower Earth-Skimming Method for PeV-EeV ν_τ Observations with Ashra. *Astroparticle Physics*, 41:7–16, 2013.

[41] J. Abraham et al. Upper Limit on the Diffuse Flux of Ultrahigh Energy Tau Neutrinos from the Pierre Auger Observatory. *Phys. Rev. Lett.*, 100, May 2008.

[42] M. Gaug, C. Hsu, J. K. Becker, A. Biland, M. Mariotti, W. Rhode, and Teshima. T. Tau neutrino search with the MAGIC telescope. *30th International Cosmic Ray Conference*, 2007.

[43] J. Beringer et al. *31- Passage of particles through matter.* PDG, 2013.

[44] Donald E. Groom, Nikolai V. Mokhov, and Sergei I. Striganov. Muon stopping power and range tables 10 MeV–100 TeV. *Atomic Data and Nuclear Data Tables*, 78(2):183 – 356, 2001.

[45] León V Hermes et al. Feasibility study for the indirect detection of ultra-high energy neutrinos with the HAWC gamma-ray observatory. *Advances in Astronomy*, 2017.

[46] Gerardo Carrasco Núñez. Structure and proximal stratigraphy of Citlaltepetl volcano (Pico de Orizaba), Mexico. *Cenozoic tectonics and volcanism of Mexico*, 334:247, 2000.

[47] C. Robin and J. M. Cantagrel. Le Pico de Orizaba (Mexique): Structure et evolution d'un grand volcan andesitique complexe. *Bulletin Volcanologique*, 45(4):299–315, 1982.

[48] Leif Rädel and Christopher Wiebusch. Calculation of the Cherenkov light yield from low energetic secondary particles accompanying high-energy muons in ice and water with Geant4 simulations. *Astroparticle Physics*, 38:53 – 67, 2012.

[49] M. P. De Pascale et al. Absolute spectrum and charge ratio of cosmic ray muons in the energy region from 0.2 GeV to 100 GeV at 600 m above sea level. *Journal of Geophysical Research: Space Physics*, 98(A3):3501–3507, 1993.

[50] O.C. Allkofer, K. Carstensen, and W.D. Dau. The absolute cosmic ray muon spectrum at sea level. *Physics Letters B*, 36(4):425 – 427, 1971.

[51] B C Rastin. An accurate measurement of the sea-level muon spectrum within the range 4 to 3000 GeV/c. *Journal of Physics G: Nuclear Physics*, 10(11):1609, 1984.

[52] C A Ayre, J M Baxendale, C J Hume, B C Nandi, M G Thompson, and M R Whalley. Precise measurement of the vertical muon spectrum in the range 20-500 GeV/c. *Journal of Physics G: Nuclear Physics*, 1(5):584, 1975.

[53] J. Kremer et al. Measurements of Ground-Level Muons at Two Geomagnetic Locations. *Phys. Rev. Lett.*, 83:4241–4244, Nov 1999.

[54] S. Haino et al. Measurements of primary and atmospheric cosmic-ray spectra with the BESS-TeV spectrometer. *Physics Letters B*, 594(1–2):35 – 46, 2004.

[55] P. Archard and others (L3+C Collaboration). *Physics Letters B*, 598, 2004.

[56] H. Jokisch, K. Carstensen, W. D. Dau, H. J. Meyer, and O. C. Allkofer. Cosmic-ray muon spectrum up to 1 TeV at 75° zenith angle. *Phys. Rev. D*, 19:1368–1372, Mar 1979.

[57] S. I. Klimushin, E. V. Bugaev, and Igor A. Sokalski. Precise parametrizations of muon energy losses in water. In *27th International Cosmic Ray Conference (ICRC 2001) Hamburg, Germany, August 7-15, 2001*.

APÉNDICE

En este apéndice encontrará los histogramas de energía perdida y distancia recorrida de cada una de las simulaciones que se mencionan en el capítulo 3. Se presentan en el mismo orden que el mostrado en ese capítulo: primero se muestran los histogramas de muones penetrando en una caja llena de agua, después los de muones penetrando en un cilindro de HAWC y por último los histogramas de muones penetrando en el volcán Pico de Orizaba.

Histogramas correspondientes a la simulación de muones penetrando en una caja llena de agua.

En las figuras 9 y 10 se muestran los histogramas de energía perdida. En las figuras 11 y 12 se muestran los histogramas de distancia recorrida.

Cada simulación se puede analizar observando sus correspondientes histogramas. Por ejemplo, para muones con energía inicial de 100 GeV, su histograma de energía perdida indica que el 100 % de los muones simulados pierde toda su energía al cruzar la caja llena de agua, mientras el histograma de distancia recorrida indica que los muones se desplazan una distancia menor a 500 metros dentro de la caja. Esto significa que los muones no logran cruzar completamente la caja, ya que esta mide un kilómetro, porque pierden toda su energía y son absorbidos. En cambio, en la simulación de 5 TeV solo una fracción de los muones, 0.3 %, pierde toda su energía mientras el resto pierde menos de 5 TeV. Su correspondiente histograma de distancia muestra que la gran mayoría, casi el 100 %, cruzan toda la caja. Así, para esta energía la mayoría de los muones salen de la caja antes de ser absorbidos.

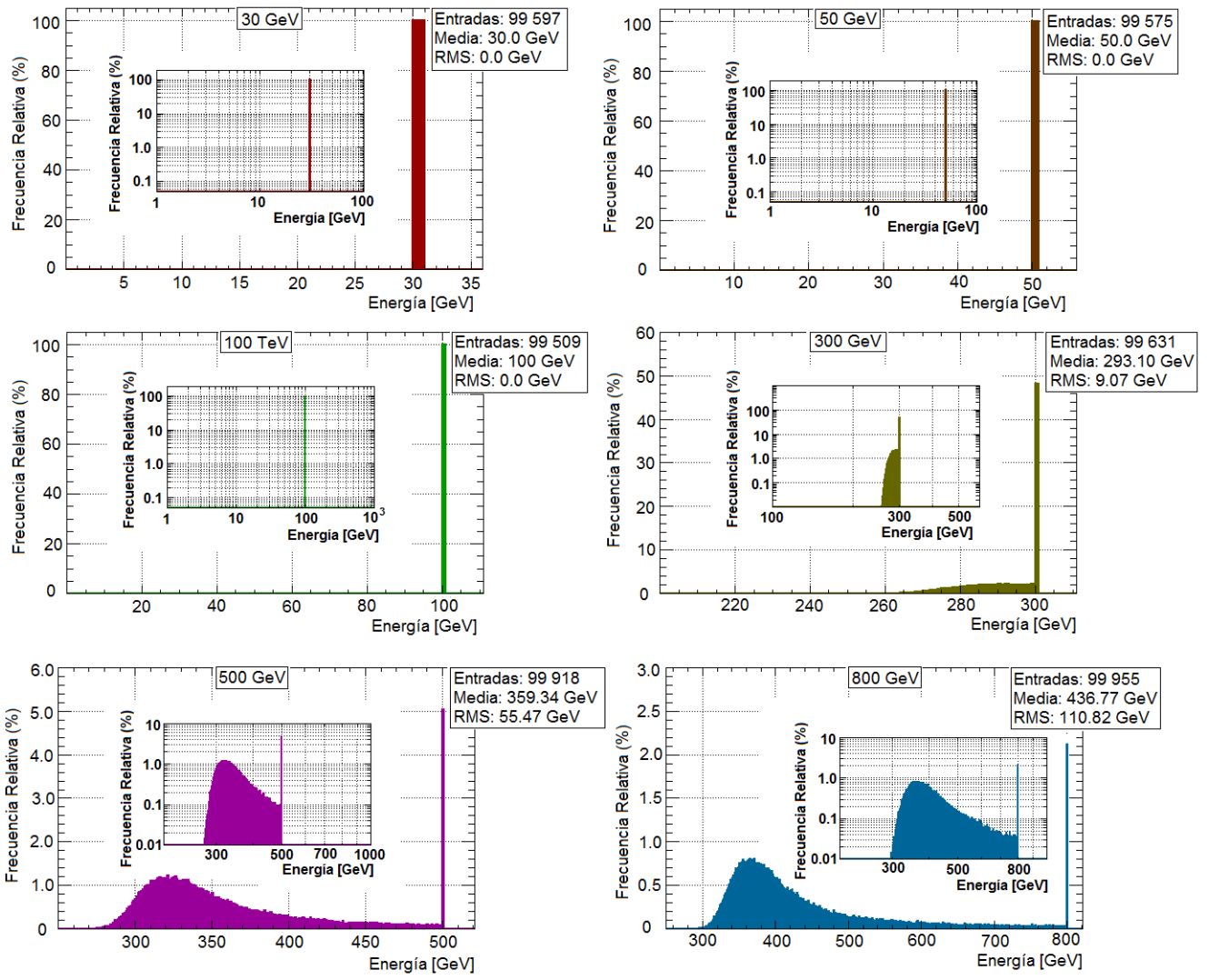


Figura 9: Energía perdida por muones con energía inicial entre 30 GeV y 800 GeV tras cruzar una caja llena de agua con una longitud de un kilómetro. A partir de 300 GeV los histogramas muestran solo una parte del espectro, por eso cada uno tiene dentro de sí un histograma más pequeño que muestra el espectro completo. El eje horizontal y vertical de estos pequeños histogramas está en escala logarítmica. Los histogramas, su valor medio y su desviación estándar (RMS) se obtuvieron con el software ROOT.

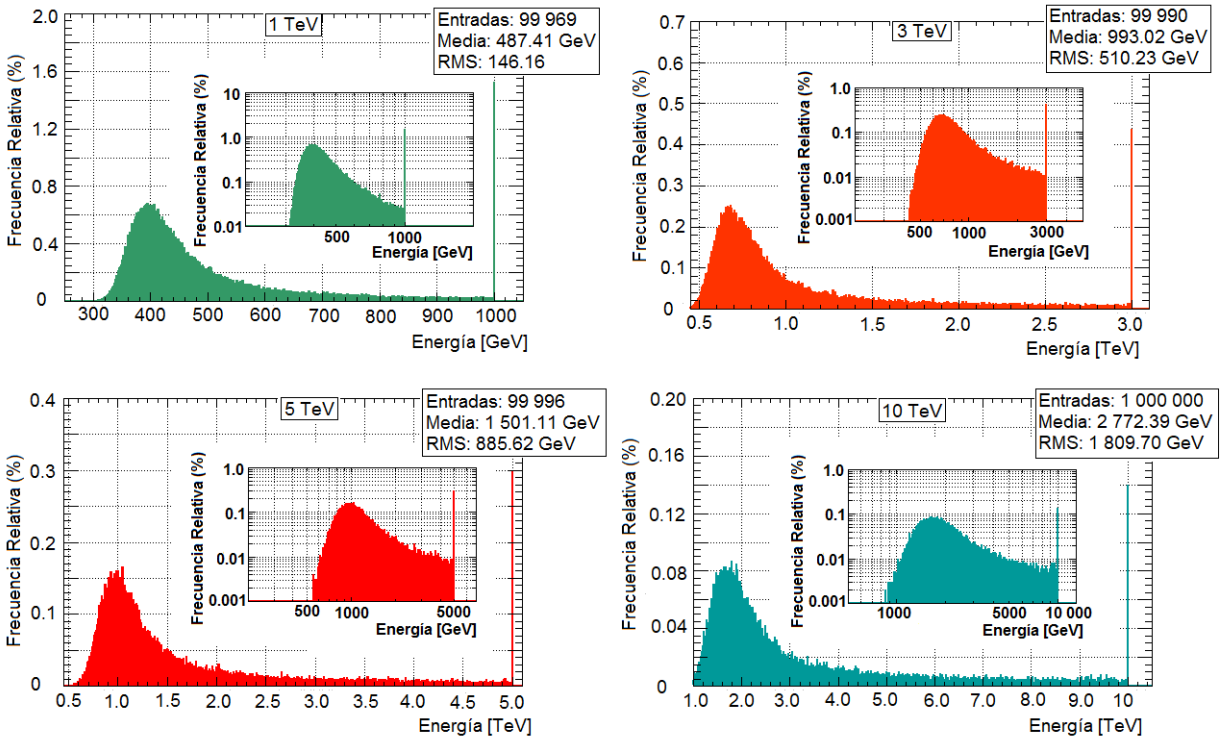


Figura 10: Energía que pierden los muones con energía inicial entre 1 TeV y 10 TeV tras penetrar en una caja llena de agua con una longitud de un kilómetro. Los histogramas muestran solo una parte del espectro de energía perdida, por eso tienen dentro de sí un pequeño histograma que muestra el espectro completo, cuyos ejes horizontal y vertical están en escala logarítmica. Los histogramas, su valor medio y su desviación estándar (RMS) se obtuvieron con el software ROOT.

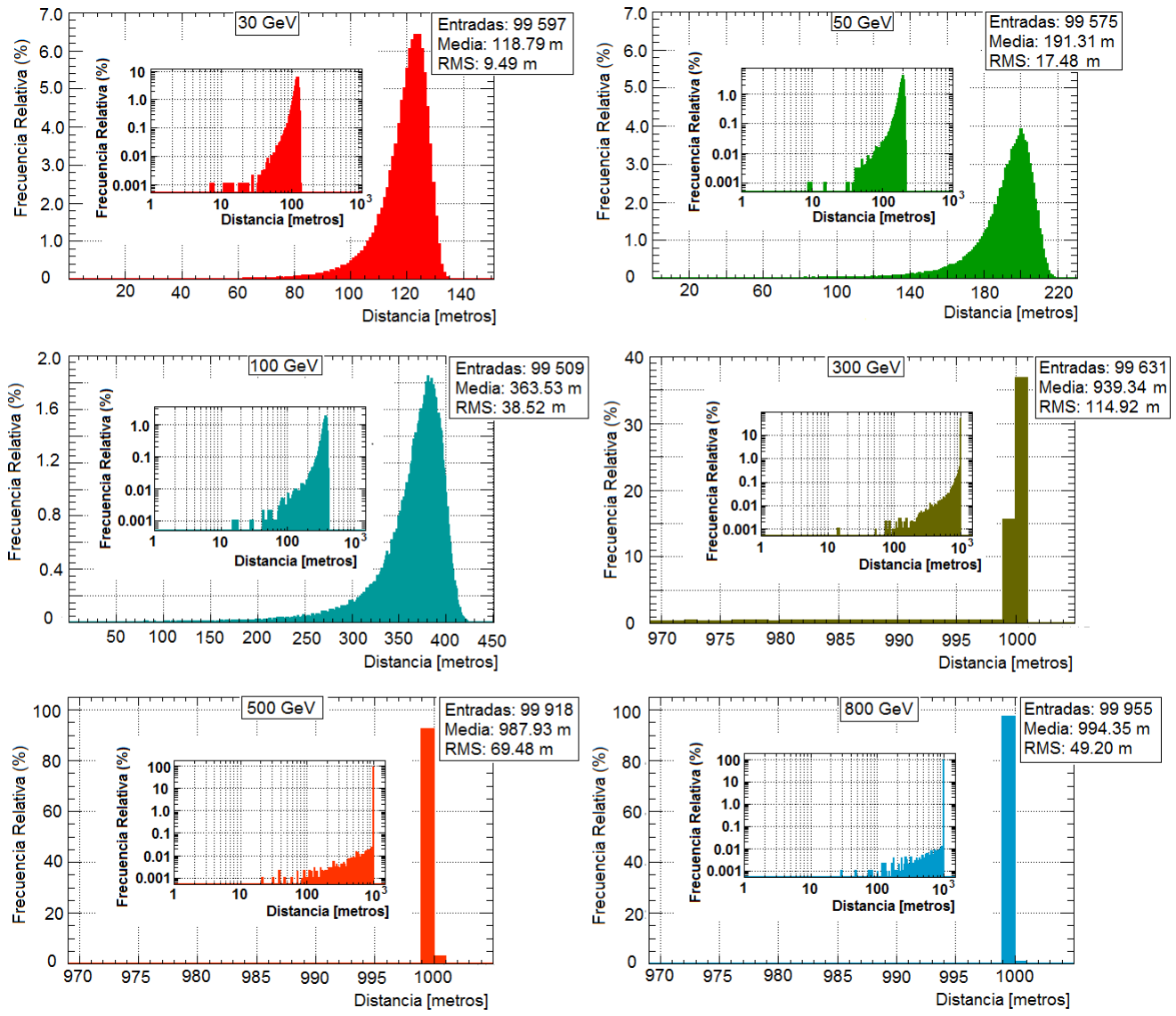


Figura 11: Distancia recorrida por muones con energía inicial de 30 GeV a 800 GeV tras cruzar una caja llena de agua de un kilómetro de longitud. Cada histograma muestra solamente una porción de todo el espectro, así que van acompañados de un histograma más pequeño que muestra el espectro completo. El eje horizontal y vertical de estos pequeños histogramas está en escala logarítmica. Los histogramas, su valor medio y el RMS se obtuvieron con el software ROOT.

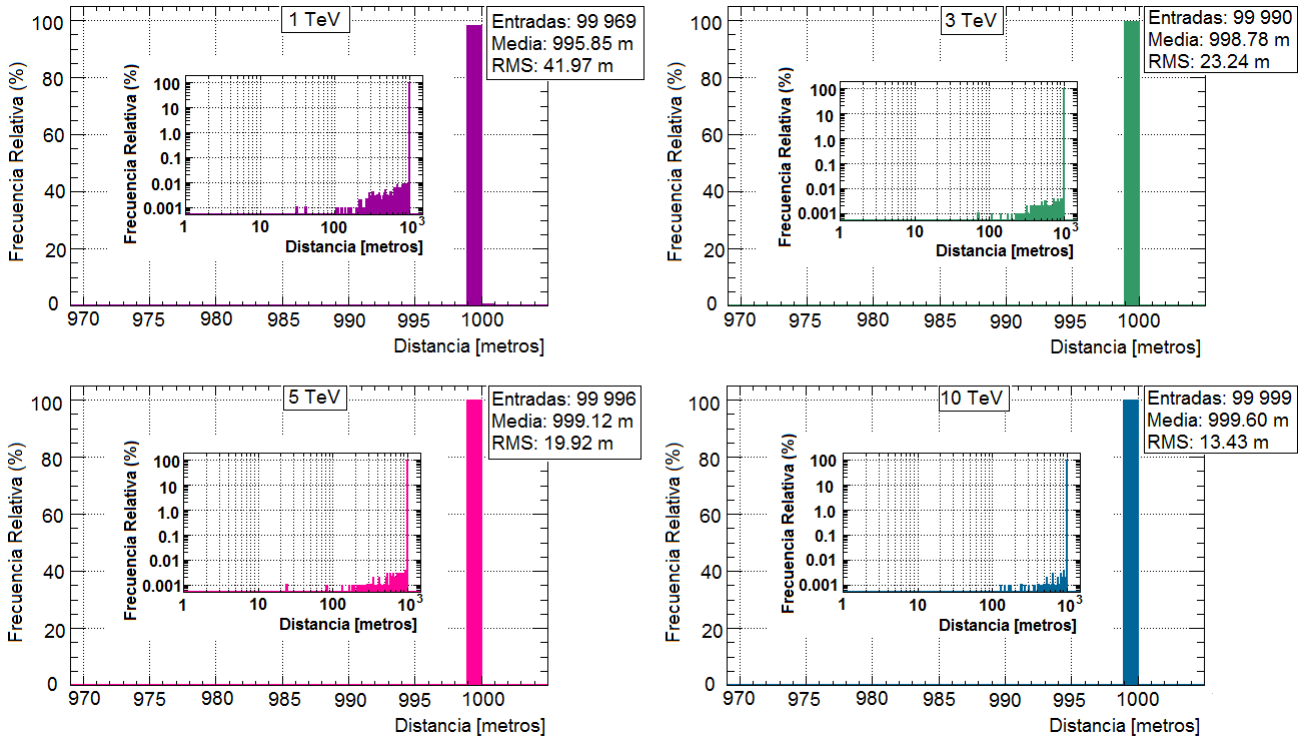


Figura 12: Distancia recorrida por muones cuya energía inicial está entre 1 TeV y 10 TeV, después de cruzar una caja llena de agua con una longitud de un kilómetro. Los histogramas muestran solo una parte del espectro pero van acompañados de uno más pequeño que muestra el espectro completo, cuyos ejes horizontal y vertical están en escala logarítmica. Los histogramas, su valor medio y el RMS se obtuvieron con el software ROOT.

Histogramas correspondientes a la simulación de muones penetrando en un tanque lleno de agua.

En las figuras 13 y 14 se muestran los histogramas de distancia recorrida, mientras las figuras 15 y 16 contienen los histogramas de energía perdida. Con estos histogramas se puede analizar cada simulación.

Para todas las energías simuladas, el 100% de los muones recorre 7.3 metros en agua, la máxima distancia que pueden recorrer dentro del cilindro. En cambio, la energía que pierden la mayoría de los muones es diferente para cada simulación pero siempre menor a su energía inicial. Esto significa que ningún muón es absorbido por el agua y que a mayor energía inicial mayor es la pérdida de energía por parte de las partículas.

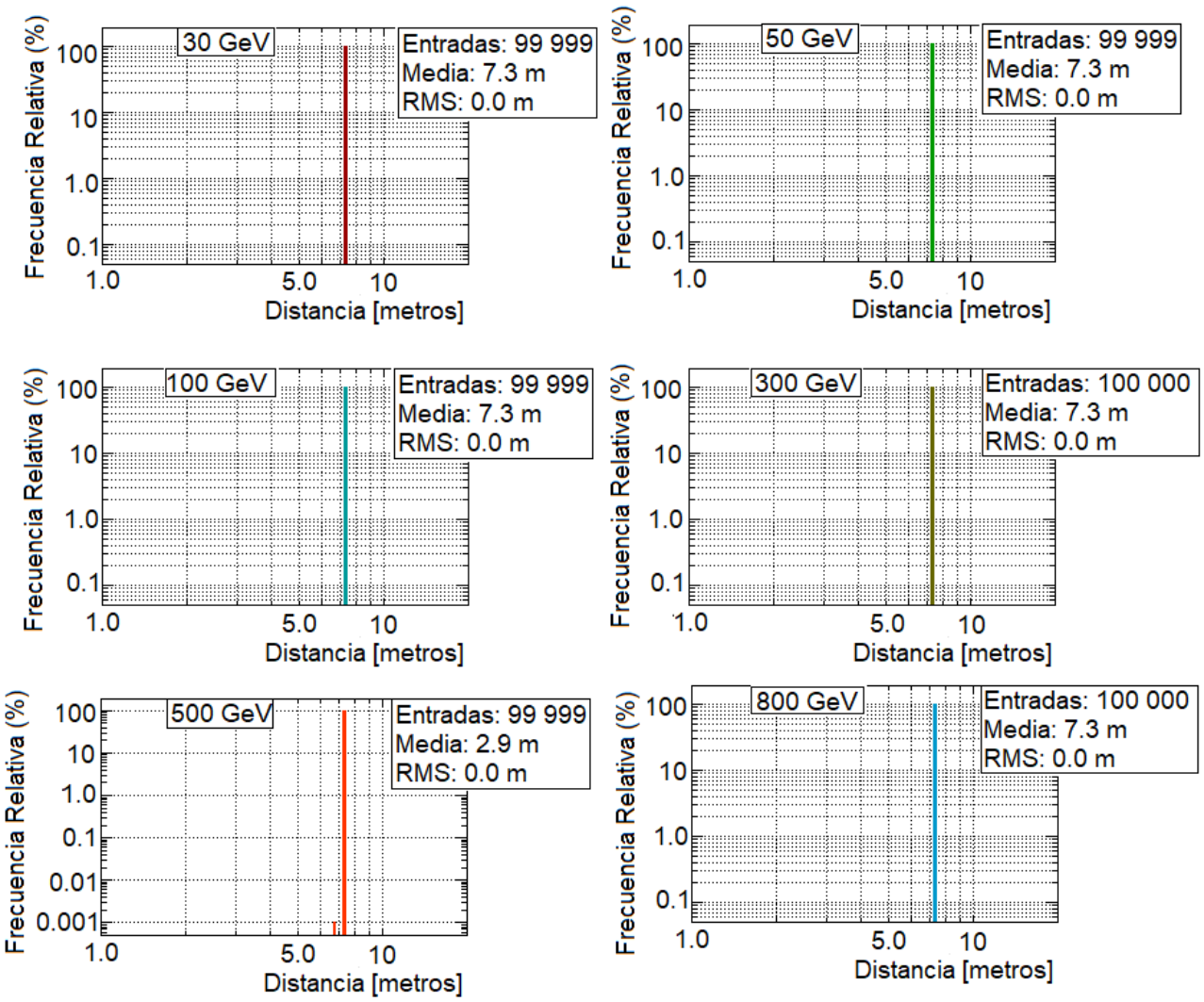


Figura 13: Distancia que recorren los muones dentro de un tanque lleno de agua con las mismas dimensiones que un tanque de HAWC y con energía inicial entre 30 GeV y 800 GeV. Los ejes horizontal y vertical de los histogramas están en escala logarítmica con el fin de mostrar todo el espectro. Los histogramas, su valor medio y RMS se obtuvieron con el software ROOT.

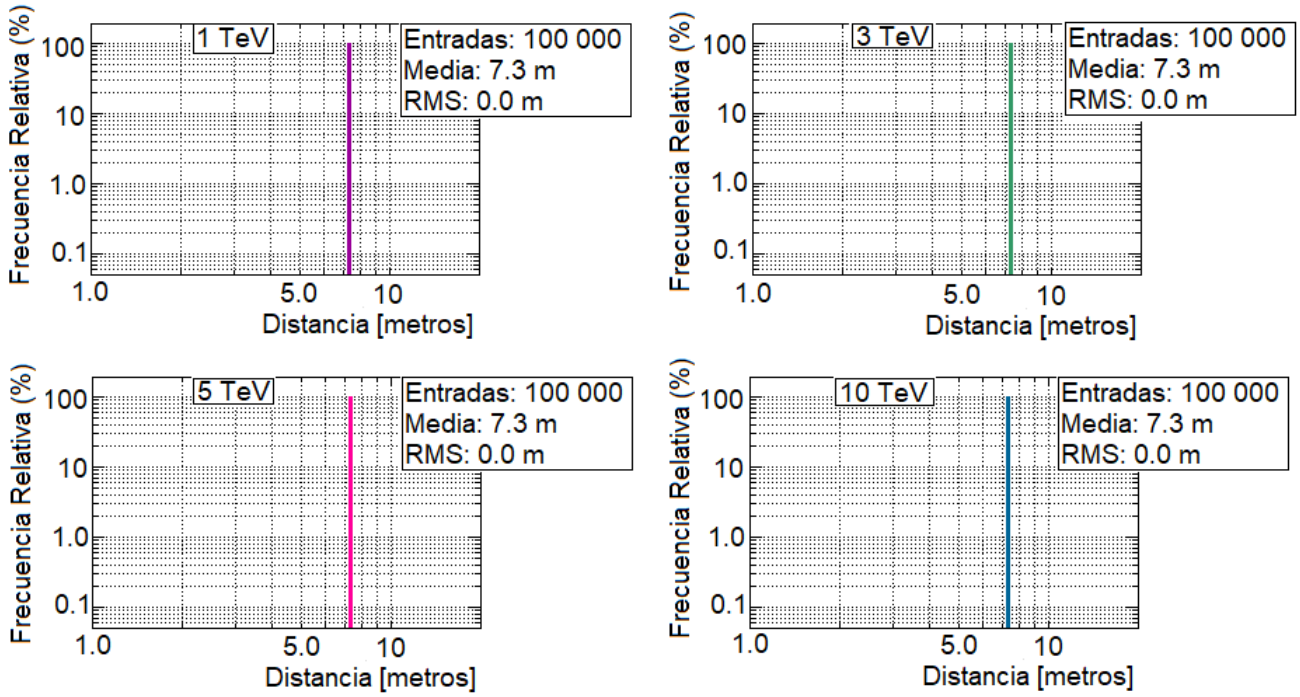


Figura 14: Distancia recorrida por muones con una energía inicial entre 1 TeV y 10 TeV después de atravesar un tanque lleno de agua con las mismas dimensiones que un tanque de HAWC. Los ejes horizontal y vertical de los histogramas están en escala logarítmica con el fin de mostrar todo el espectro. Los histogramas, su valor medio y RMS se obtuvieron con el software ROOT.

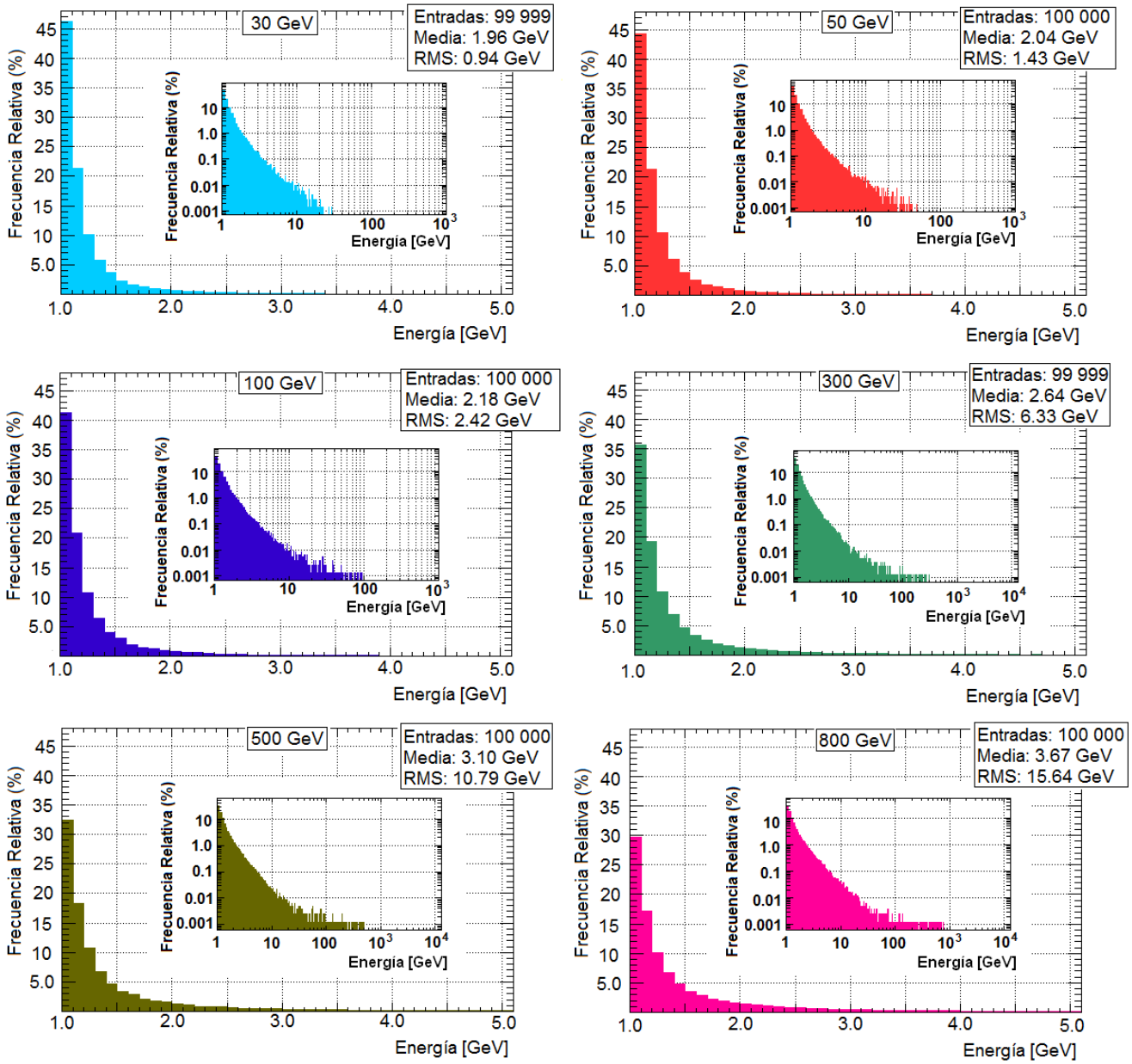


Figura 15: Energía perdida por muones con energía inicial entre 30 GeV y 800 GeV después de penetrar en un tanque lleno de agua con las mismas dimensiones que un tanque de HAWC. Por cada simulación se presentan dos histogramas, el histograma más grande muestra solo una parte del espectro mientras el histograma más pequeño muestra el espectro completo. El eje horizontal y vertical de los histogramas pequeños está en escala logarítmica. Los histogramas, su valor medio y RMS se obtuvieron con el software ROOT.

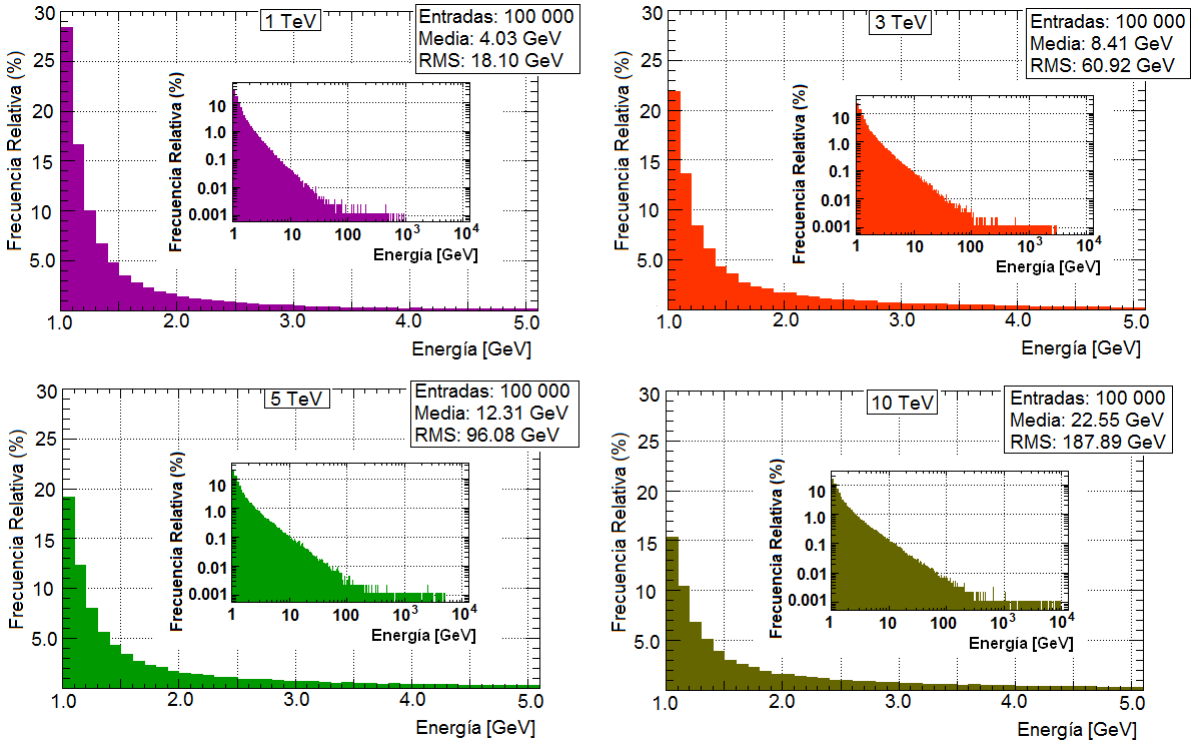


Figura 16: Energía que pierden muones con energía inicial entre 1 TeV y 10 TeV tras cruzar un tanque lleno de agua con las mismas dimensiones que un tanque de HAWC. Cada simulación tiene dos histogramas, el histograma más grande muestra solo una parte del espectro mientras el histograma más pequeño muestra el espectro completo. Tanto el eje horizontal como el eje vertical de los histogramas pequeños está en escala logarítmica. Los histogramas, su valor medio y RMS se obtuvieron con el software ROOT.

Histogramas correspondientes a la simulación de muones atravesando el volcán Pico de Orizaba.

En esta simulación se tienen dos sólidos, el cuerpo de agua que representa a HAWC y el cono que representa al volcán Pico de Orizaba. Ambos sólidos reciben muones y éstos a su vez pierden energía, pero como la simulación se enfocó en los muones que penetran en el volcán, solo se muestran los histogramas de distancia recorrida y energía perdida en el volcán.

Si la energía inicial de los muones es $\leq$ 1 TeV casi todos pierden su energía cuando van cruzando el volcán, pero si su energía inicial es mayor el número de muones que no pierden toda su energía y cruzan el volcán aumenta. La razón es que al tener poca energía, los muones solo pueden cruzar el volcán por regiones delgadas. Pero cuando la energía inicial es mayor a 1 TeV una cantidad importante de muones recorre distancias del orden de kilómetros, disminuyendo el número de muones absorbidos.

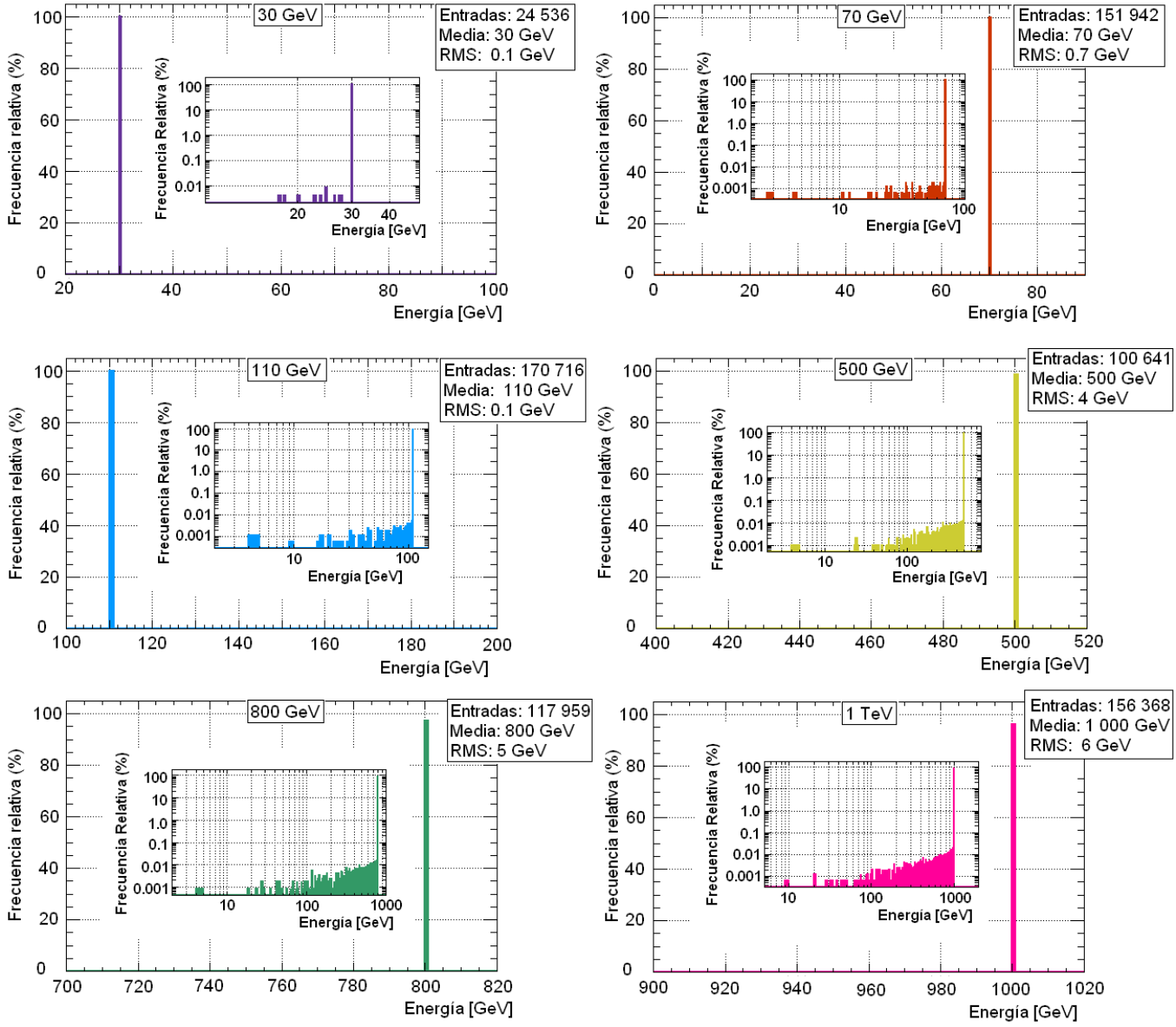


Figura 17: Energía que pierden los muones en las simulaciones de 30 GeV a 1 TeV después de penetrar en el volcán. Se muestran dos histogramas por simulación, el histograma grande muestra una porción del espectro mientras el histograma pequeño muestra el espectro completo, estando sus ejes horizontal y vertical en escala logarítmica. Los histogramas, su valor medio y su desviación estándar (RMS) se obtuvieron con el software ROOT.

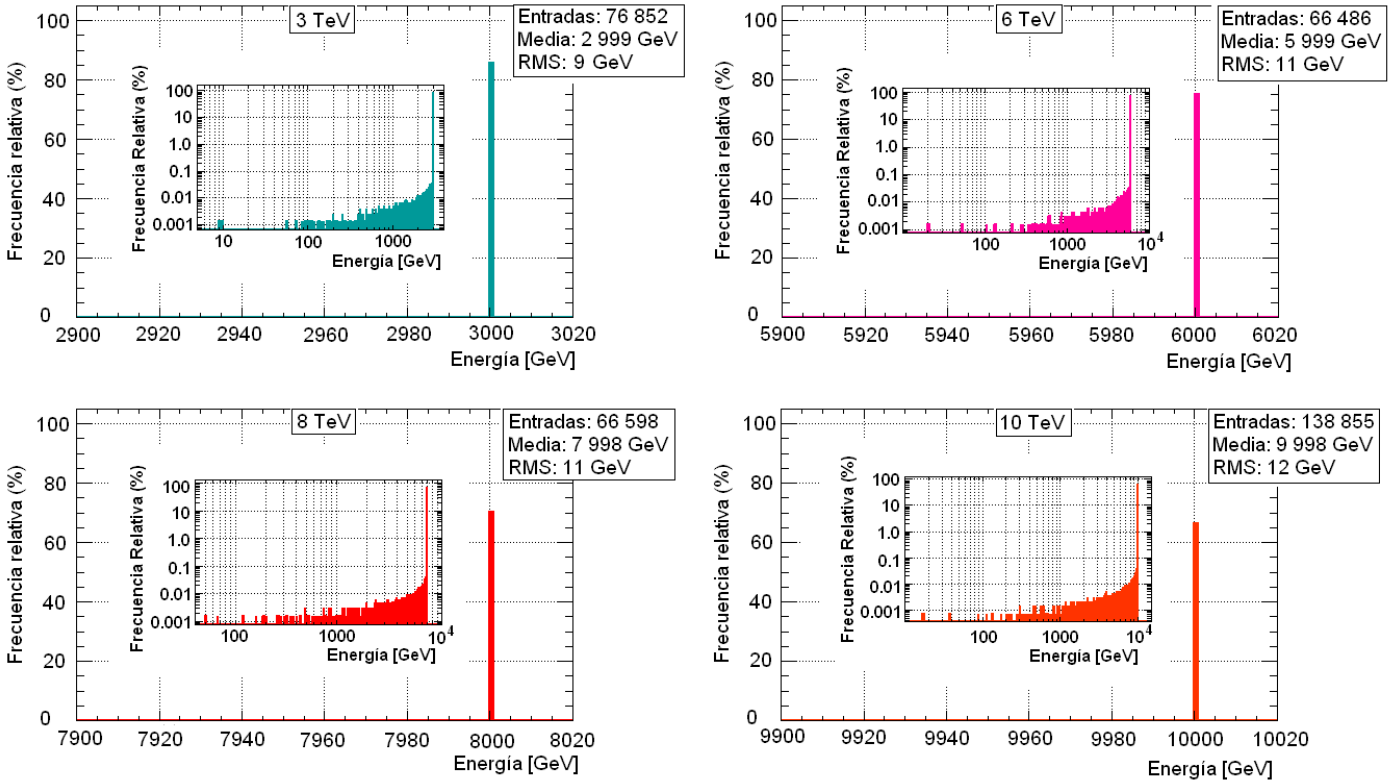


Figura 18: Energía perdida por muones en las simulaciones de 3 TeV a 10 TeV tras penetrar en el volcán. Se muestran dos histogramas por simulación, el histograma grande presenta una porción del espectro mientras el histograma pequeño muestra el espectro completo, estando sus ejes horizontal y vertical en escala logarítmica. Los histogramas, su valor medio y su desviación estándar (RMS) se obtuvieron con el software ROOT.

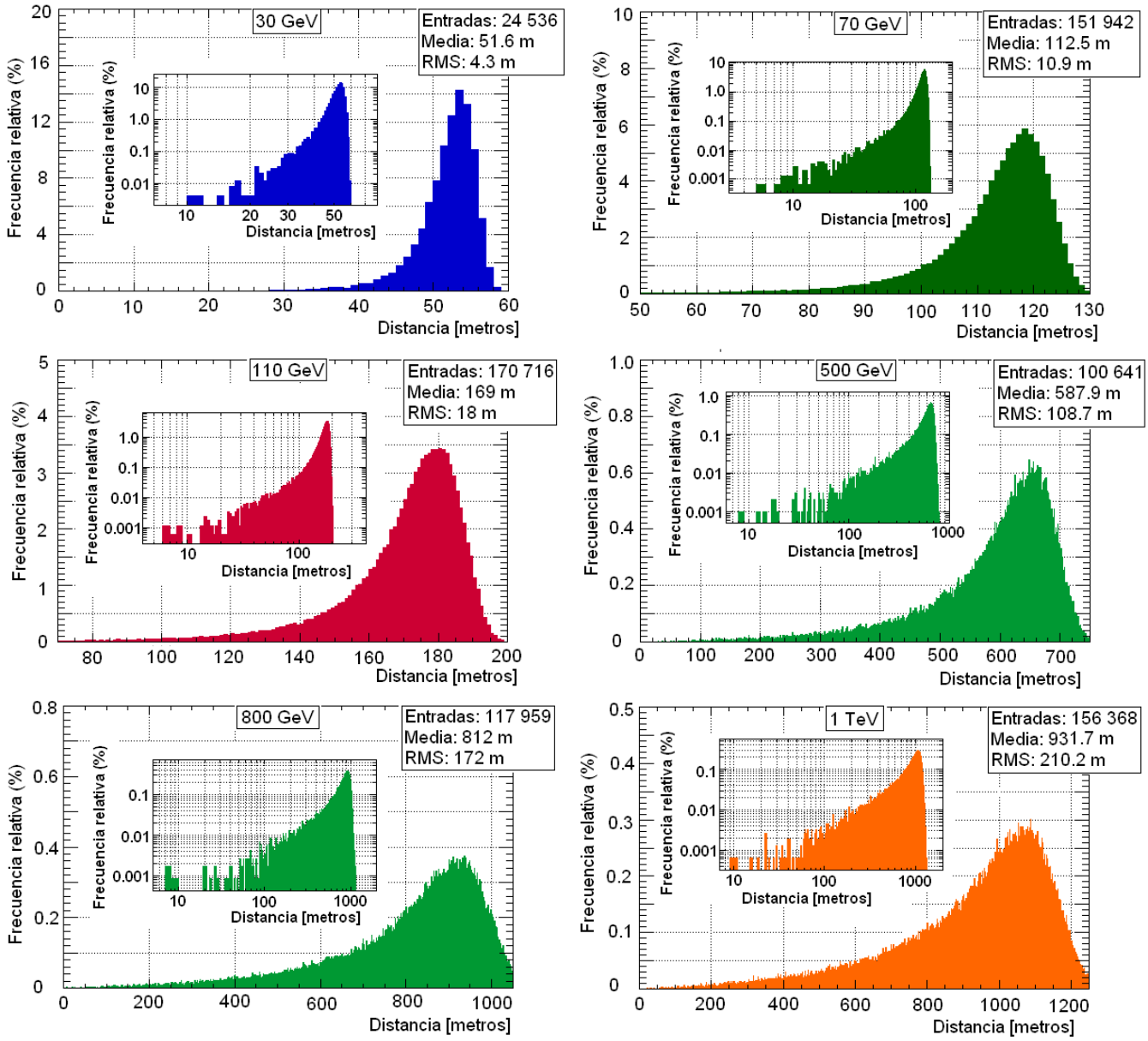


Figura 19: Distancia que recorren los muones dentro del volcán en las simulaciones de 30 GeV a 1 TeV. Se muestran dos histogramas por energía simulada, el histograma pequeño abarca todo el espectro y sus ejes horizontal y vertical están en escala logarítmica. El histograma grande contiene solo una porción del espectro. Los histogramas, su valor medio y su desviación estándar (RMS) se obtuvieron con el software ROOT.

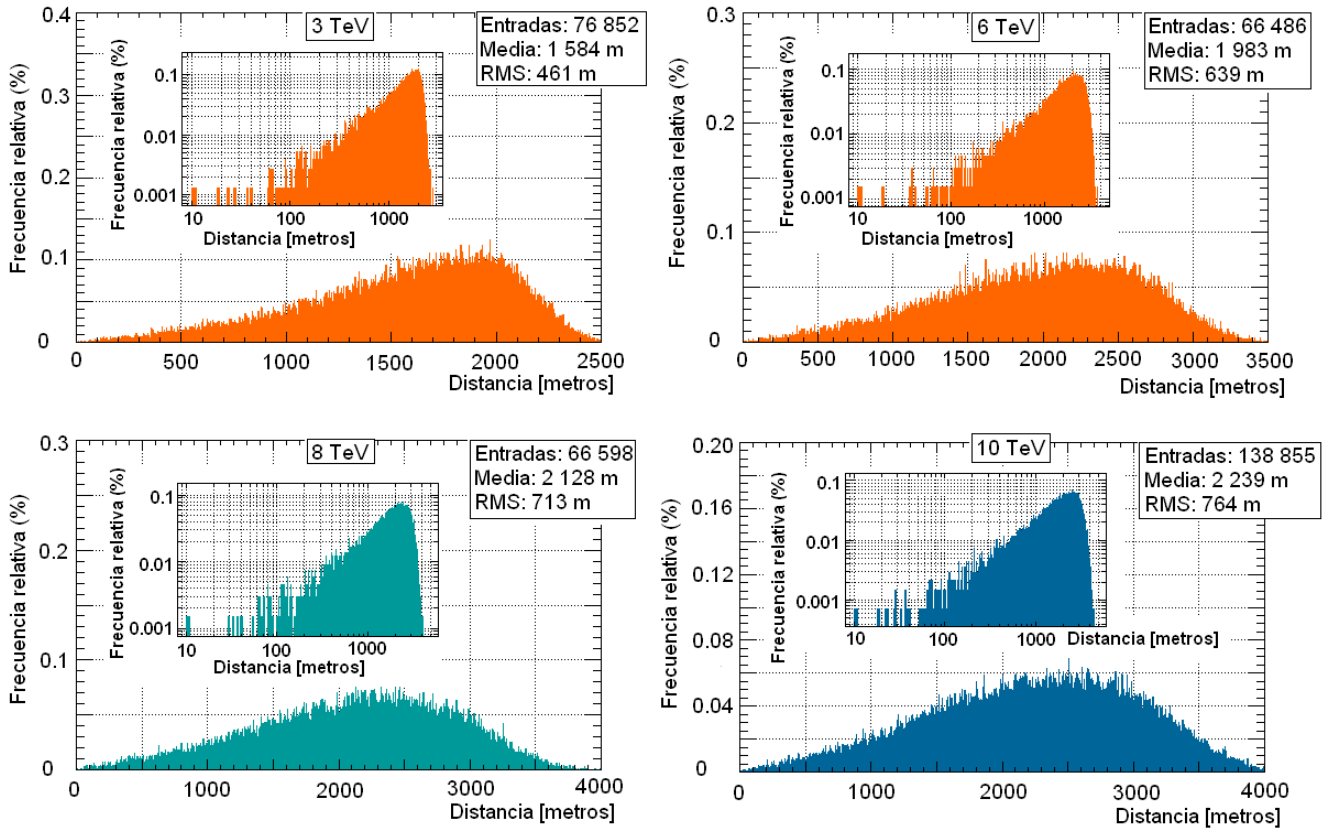


Figura 20: Distancia que recorren los muones dentro del volcán en las simulaciones de 3 TeV a 10 TeV. Se muestran dos histogramas por energía simulada, el histograma pequeño abarca todo el espectro y sus ejes horizontal y vertical están en escala logarítmica mientras el histograma grande contiene solo una porción del espectro. Los histogramas, su valor medio y su desviación estándar (RMS) se obtuvieron con el software ROOT.