Simulación del SCT en el Combined Testbeam 2004

Carlos Escobar Ibáñez*

16 de junio de 2004

Resumen

Actualmente, la simulación completa y detallada de un experimento es un tema relevante e incluso necesario en Física de Altas Energías. Con este objetivo, se presenta la necesidad de la simulación de los módulos del SCT de ATLAS en el Combined Testbeam 2004. Esta simulación es la que precederá a la simulación total de ATLAS. Para llevar a cabo esta tarea se hace uso del entorno de trabajo oficial de ATLAS, Athena, y de Geant4. Para más información: http://ific.uv.es/~cescobar/simulation/

*IFIC / Universidad de Valéncia

Índice

1. Introducción **3**

1.1. Testbeam . . . 3

1.2. Athena framework . . . 5

2. Simulación **5**

2.1. Descripción del Detector . . . 6

2.1.1. Definición de la Geometría. Base de datos NOVA . . . 6

2.1.2. Construyendo la geometría en GeoModel desde NOVA . . . 7

2.1.3. Conversión a la geometría de Geant4 . . . 9

2.1.4. Definición de *Envelopes* para el CTB . . . 9

2.2. Simulación . . . 10

2.2.1. Generador de Partículas . . . 11

2.2.2. Campos Magnéticos . . . 11

2.2.3. *G4Sim* y *G4Svc* . . . 12

2.2.4. Física relevante . . . 13

2.2.5. Definición de la parte sensible . . . 14

2.3. Digitalización y Reconstrucción . . . 14

2.3.1. Digitalización y clustering . . . 15

2.3.2. *Spacing* . . . 16

2.3.3. *Traking* . . . 16

2.4. Validación y Chequeos . . . 19

2.5. Resultados de la Simulación . . . 20

2.6. Simulación Completa y Física . . . 21

3. Bibliografía **21**

1. Introducción

1.1. Testbeam

Los TestBeams (TB) no son más que pruebas preliminares para testear, poner a punto y conocer los detectores que se utilizarán en un determinado experimento. Actualmente, los experimentos de HEP (High Energy Physics) son extremadamente complejos y poseen multitud de detectores, tanto en cantidad como en variedad. Es necesario pues, probar estos detectores bajo unas condiciones conocidas y favorables, es decir, se someten los detectores a pruebas, tests y chequeos con campos magnéticos controlados, haces de partículas de los cuales se conoce prácticamente todo, alimentaciones correctas, refrigeración suficiente, etc, y en un periodo de tiempo muy corto (1 mes aprox.). Obviamente, todas estas condiciones ideales no se darán siempre en el experimento. Como siempre, la manera de actuar de los físicos pasa por empezar con los casos más ideales posibles. Se podría decir que los TB son las situaciones más ideales que pueden tener hoy en día los físicos experimentales de HEP.

Experimentos como han sido LEP o como será el LCH son de una complejidad enorme. Cada detector de estos experimentos esta compuesto por multitud de grupos diferentes de subdetectores. No obstante a tal complejidad, todo es estudiado hasta el más mínimo detalle pero hay un factor determinante, que hace que aunque se puedan iniciar estos experimentos en condiciones casi ideales acaben produciendose problemas, el tiempo. Estos experimentos no son ya como los de la física nuclear que duran poco (días, semanas o meses) sino que estan diseñados para trabajar durante casi 10 años. Durante este periodo surgirán diversos problemas y complicaciones que se agraban considerando que una vez ensamblados todos los subdetectores no se volverán a tocar.

Es por todo esto por lo que es necesario conocer a fondo los detectores. Además los TB son un entrenamiento ideal para los físicos, ingenieros, etc, ya que para conseguir esas condiciones ideales (que no se consiguen siempre) surgen muchos problemas. Son la fase preliminar donde conseguir las condiciones de operación de ATLAS.

En concreto, los TB de ATLAS[1] se realizan en el CERN en el periodo de verano. Se utiliza el haz del SPS del CERN, quien suministra las partículas requeridas con propiedades (momento, dirección, energía, etc) conocidas. Esto sirve para calibrar muchos de los subdetectores. Tambíen se simulan los campos magnéticos que existirán en ATLAS con imanes superconductores fijos.

Figura 1: Área H8 del CERN donde se realizan los TB de ATLAS

[1] Detector del LCH

Son, en general, una fase preliminar donde conseguir las condiciones de operación de ATLAS. Estas son:

- Frecuencia de trabajo[2] de 40 MHz equivalente a paquetes (*bunches*) cada 25 ns.
- Temperatura bajo cero en refrigeración.
- Campos Magnéticos de 2.6 a 4.1 Tesla.
- Radiación 10 veces mas que en LEP

Los TB de ATLAS se vienen haciendo algunos años ya pero este año (2004) se va a dar un paso importante ya que se va a testear todos los subdetectores de ATLAS a la vez. En la fig. 1 puede verse las instalacines del TB del CERN.

El Combined TestBeam (CTB) de ATLAS se realizará por primera vez en verano de 2004. Se sube un par de escalones en complejidad. Lo que se pretende es probar a la vez todos y cada uno de los subdetectores que forman parte de ATLAS siguiendo en la medida de lo posible la misma geometría que tendrá éste. En concreto, lo que se hará es reconstruir una rodaja, o mejor dicho un cierto ángulo sólido, de una sección transversal del diseño final ATLAS. Las partículas que se producirán en la colisión prontón-prontón saldrán, en su mayoria, transversalmente a la línea del haz. Es pues esto lo que se simulará, construyendo una sección y enviando diferentes partículas (similares a las que se producirán).

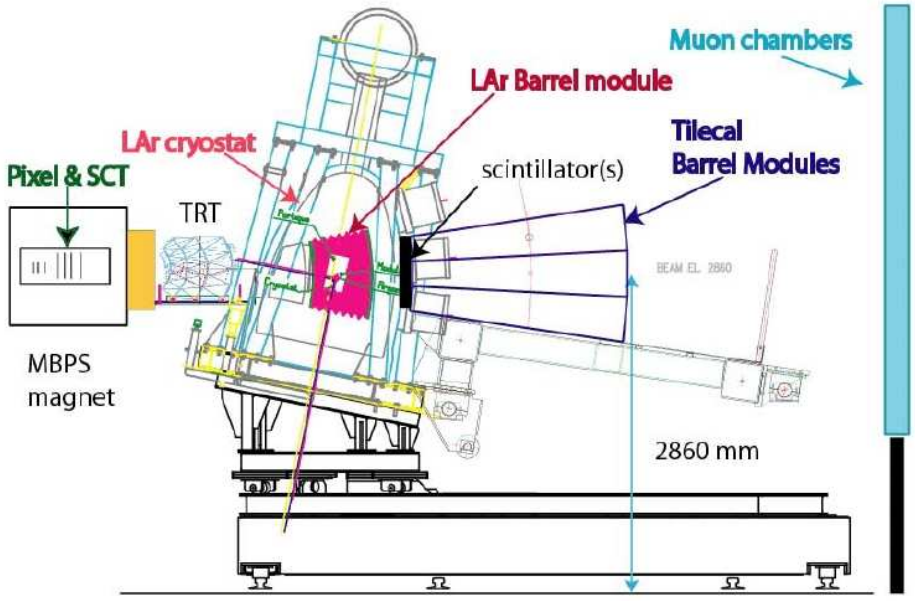

Figura 2: Diseño del área H8 del CERN para el CTB

[2] Esta frecuencia es 10 veces más rápida que cualquier otro experimento

1.2. Athena framework

Como se dijo anteriormente los experimentos son extremadamente complejos pero a su vez extremadamente caros. Esto hace necesario que no se puedan realizar todas las pruebas necesarias para conocer pequeñas dependencias que puedan existir. Tampoco se puede intentar reproducir todos los problemas que se sabe se pueden tener. Bien, pues con todo esto nace la necesidad de simular el TB. Hasta ahora se habían hecho simulaciones independientes. Es ahora cuando surge la necesidad de hacer algo de manera ordenada, conjunta y sólida. Surge la necesidad de simular el CTB.

Dicha simulación se esta realizando bajo Athena. Athena es el entorno de trabajo (framework) oficial de ATLAS. Está basado en la arquitectura GAUDI, desarrollado originalmente para el LCHb [3]. Athena es la suma de este *kernel* y las mejoras específicas para ATLAS.

2. Simulación

El proyecto de simulación del SCT tiene como objetivo implementar completamente toda la cadena necesaria. La implementación está basada en un paquete de software[4] que se está desarrollando. Usar Athena como entorno de trabajo es lo que se pretende, partiendo de cero. Primero fué necesario generar y llenar una base de datos, NOVA[5], que contiene todos los valores necesarios para reproducir la geometría. Luego, accediendo automáticamente a esta base de datos se construye tridimensionalmente la geometría utilizando las *kernel-classes* de GeoModel[6]. Después, dicha geometría se convierte a geometría en Geant4 (G4). Junto con esto se realiza la definición de los volúmenes sensibles (*sensitive detector*). Con estas dos cosas se comienza la simulación propiamente dicha. Después, con los datos de la simulación se pasa a la digitalización, el clustering, el spacing, el traking, a realizar histogramas de resultados y por último a analizar y comparar los resultados. En la fig. 3 se puede ver el proceso de simulación.

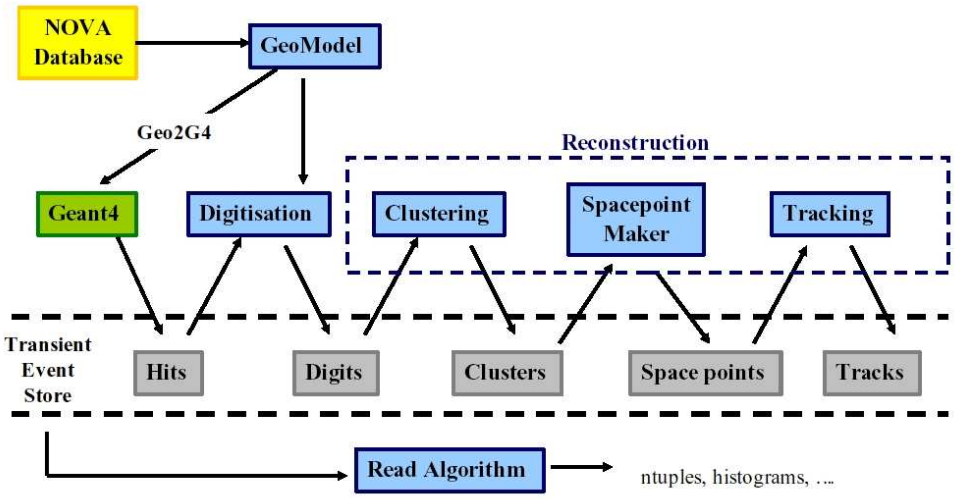

Figura 3: Esquema del proceso de simulación

[3] Otro de los cuatrodetectores del LHC junto con ATLAS
[4] Athena está estructurada en diferentes paquetes interconectados
[5] Ver http://web13.cern.ch/atlas-php/NOVA/ para más detalles
[6] Paquete creado por Grant Gorfine y mantenido por él y nuestro pequeño grupo. Más información en: http://atlas-sw.cern.ch/cgi-bin/viewcvs.cgi/offline/InnerDetector/InDetDetDescr/SCT_GeoModel/

2.1. Descripción del Detector

Lo primero es la descripción de la geometría y otras características. Primero, la defición de los parámetros necesarios para reconstruir geométricamente (en 3D) la configuración (los módulos y objetos adicionales relevantes). También se definen los materiales de cada objeto. Después se reconstruye la geometría. Esto último se realiza en tres pasos que se describirán a continuación.

2.1.1. Definición de la Geometría. Base de datos NOVA

Todos los parámetros necesarios para definir la geometría y reconstruir los objetos son implementados en una base de datos. Esta base de datos esta diseñada para albergar los números básicos para la descripcion del detector ATLAS. Esta base de datos oficial es conocida con el nombre de NOVA[7].

En esta base de datos se creó una nueva sección llamada *tb2004*, que contiene todos los parámetros iniciales y básicos para reconstruir todos y cada uno de los subdetectores así como su posición, materiales, etc. Como su nombre indica esta especialmente desarrollada para el CTB de 2004.

La subsección *zsctgeotb* fué desarrollada para la parte del SCT (únicamente los módulos tipo *EndCap*[8]). Aquí, se tiene toda la información necesaria para construir los detectores de silicio (los 4 tipos distintos), los bloques de soporte de la caja donde iran situados los módulos durante los tests así como las ventanas que tendrá la caja. Por supuesto, aquí estan todos los materiales usados asignados a los distintos dispositivos. También estan todas las distancias y medidas que se necesitan para simular la geometría real que se tendrá.

Las medidas de los componentes de los módulos de silicio se extrajerón de los planos de diseño y la disposición geomética de los módulos de documentos oficiales. Los materiales se definieron siguiendo datos de documentos para fabricación. Algunos de los diseños utilizados como referencia pueden verse en las fig. 4 y en la fig. 5.

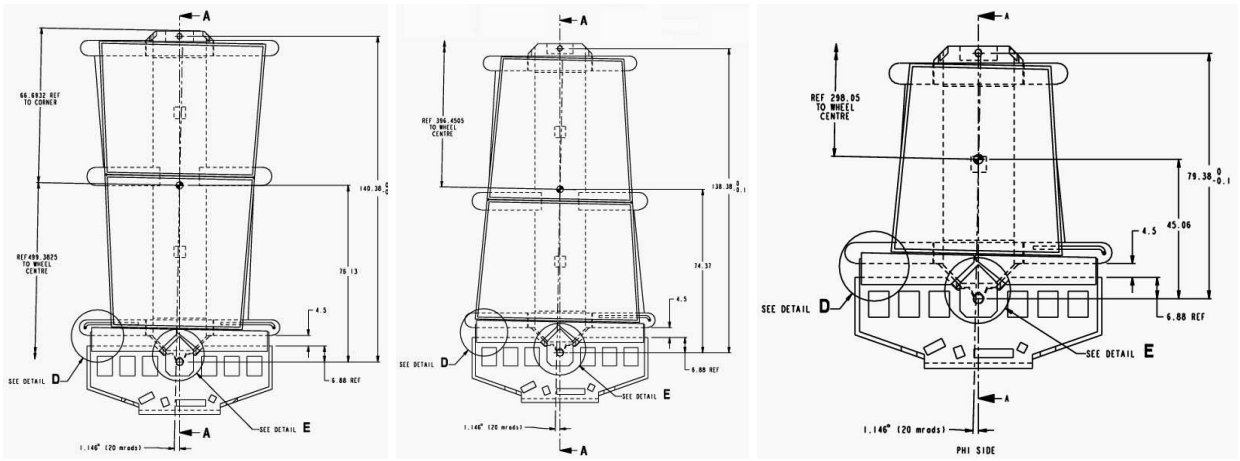

Figura 4: Estos son 3 de los diseños. De izquierda a derecha: *Outer*, *Middle* e *Inner*

[7] Ver http://web13.cern.ch/atlas-php/NOVA/ para más detalles

[8] Se trabaja únicamente con módulos *EndCap* porque a parte de que son los únicos usados en el CTB2004, en los tipo *Barrel* trabajan otros grupos

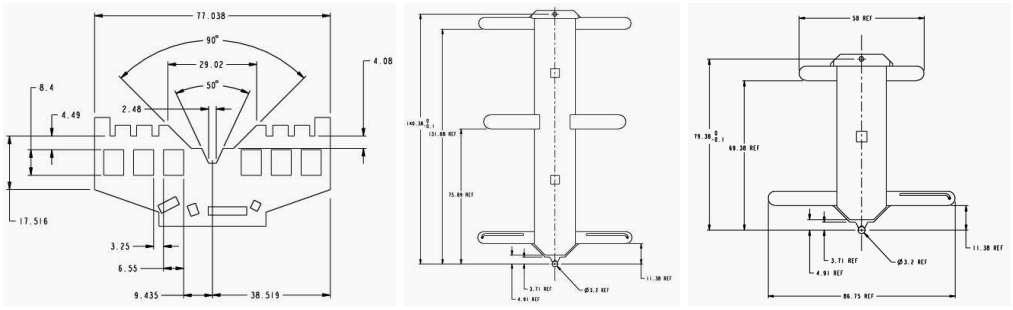

Figura 5: Aquí se ven el híbrido y dos tipos de espinas

Por último, los materiales utilizados se presentan en siguiente la tabla[9].

Objeto	Material definido	Comentarios
Ventana	AirexR82	Material de las 2 ventanas de la SCTBox
Soporte	CFiberFwdSupportFrame	Material de los 4 soportes del módulo dentro de la SCTBox
Sensores	Silicio	Material de los 4 sensores de cada módulo SCT
Híbrido	FwdHybrid	Material medio del híbrido
Espina	Beryllia	Material de la espina
Subespina	AluNit	Material de la 6 subespinas del módulo

2.1.2. Construyendo la geometría en GeoModel desde NOVA

Una vez se tienen los parámetros básicos en NOVA se puede empezar a reconstruir la geometría. Para ello, se empezó a trabajar sobre un par de paquetes (bastante antíguos) ya desarrollados en el entorno Athena, estos son, *SCT_ GeoModel* y *SCT_ TestBeamDetDescr*. En ambos paquetes (programados en C++) se ha trabajado para conseguir los resultados deseados.

Ahora mismo la reconstrucción de los módulos se realiza por parte del paquete *SCT_ GeoModel*. Éste accede on-line a NOVA, lee los parámetros y reconstruye los componentes (las obleas de silicio, la espina y subespinas y el híbrido), después construye el módulo (4 tipos diferentes). En la fig. 6 se pueden ver los módulos reconstruidos (todos de tipo *EndCap*).

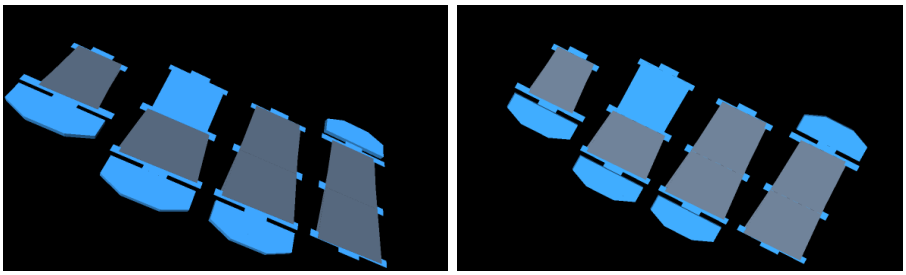

Figura 6: Aquí estan los 4 tipos reconstruidos. De izquierda a derecha: *Inner, Short Middle, Middle y Outer*

Después el paquete *SCT_ TestBeamDetDescr* se encarga entre otras muchas cosas de reconstruir la geometría que se tendrá en el CTB 2004 con los módulos que le proporciona *SCT_ GeoModel* y con datos de

[9] Para más detalles consultar NOVA

NOVA. También construye objetos adicionales como los bloques donde van sujetados los módulos en la caja (caja diseñada específicamente para el TB) y las dos ventanas que ésta posee.

En las fig. 7 y 8 pueden verse los resultados de esta reconstrucción en GeoModel.

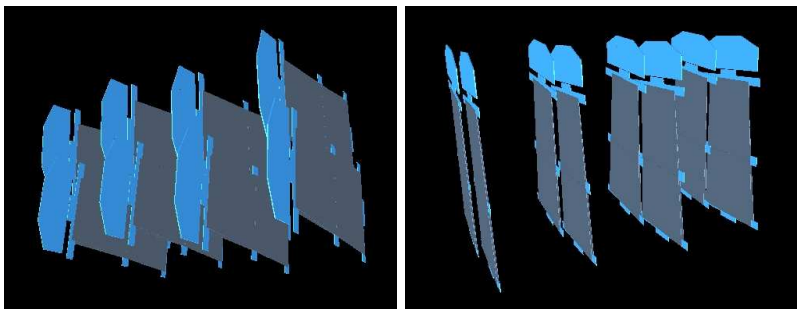

Figura 7: Reconstrucción de la configuración del CTB2004 (sólo módulos)

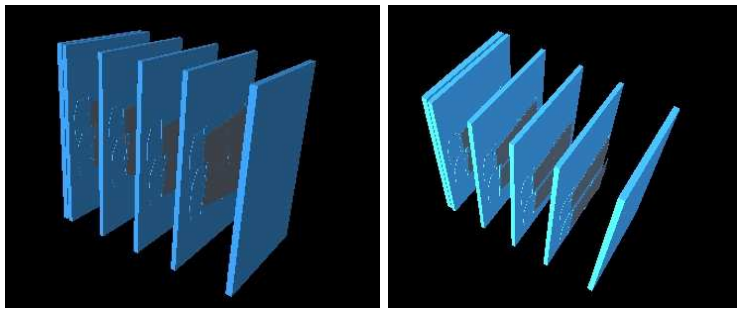

Figura 8: Reconstrucción de la configuración del CTB2004

Como cosa adicional, el código presenta la posibilidad de reproducir las configuraciones utilizadas en los TB de mayo y septiembre de 2003. Con esto se espera poder validar la simulación ya que ya se tienen los datos experimentales analizados y listo para comparar. Estas configuraciones pueden verse en las fig. 9 y 10.

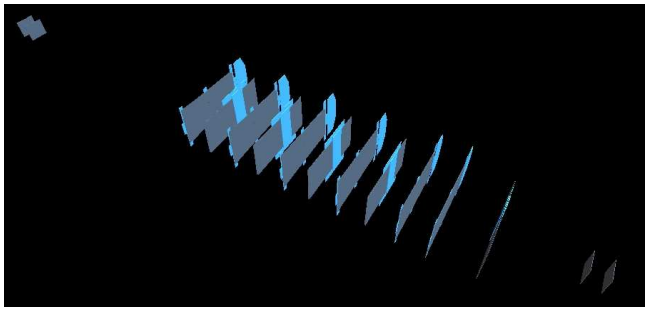

Figura 9: Reconstrucción de la configuración del TBMay2003

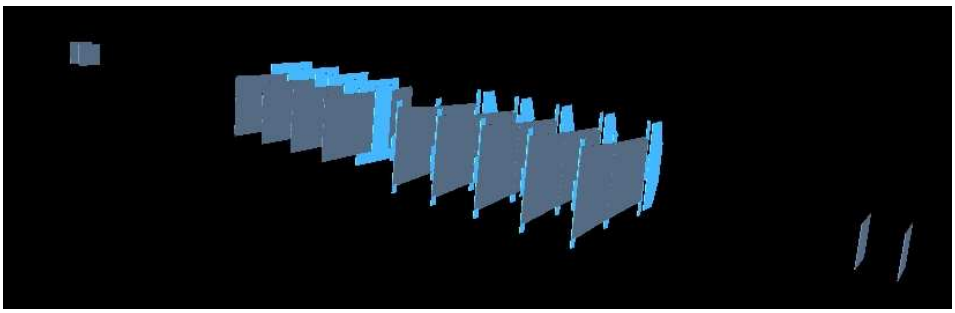

Figura 10: Reconstrucción de la configuración del TBSep2003

2.1.3. Conversión a la geometría de Geant4

Una vez se tiene la reconstrucción geométrica en GeoModel se tiene que hacer la conversión a geometría propia de Geant4. Esta conversión es muy simple y sistemática ya que nuestro sistema no posee ningún objeto no estándar. La conversión la realiza un paquete llamado *Geo2G4*, ya existente. Después de la conversión se realizó un chequeo de que la conversión fuera correcta. La verdad es que el proceso de conversión funcionó muy bien desde el principio.

En las fig. 11 y 12 se pueden ver algunos ejemplos de la conversión de la geometría a Geant4.

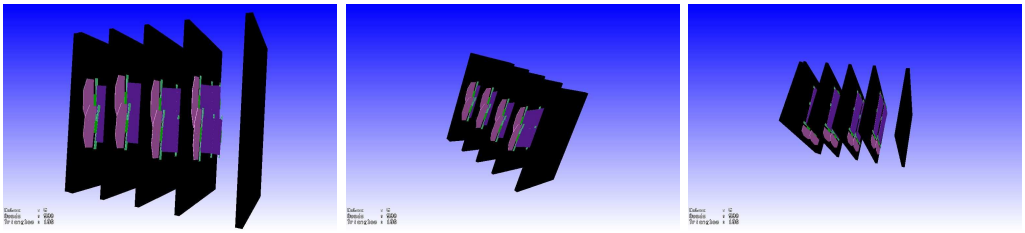

Figura 11: Resultado de la conversión de GeoModel a Geant4

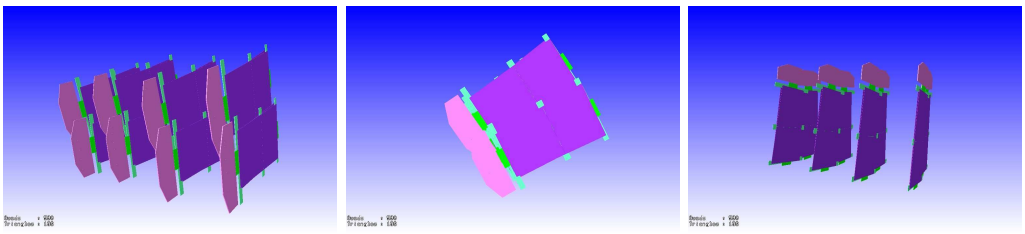

Figura 12: Resultado de la conversión de GeoModel a Geant4 (sólo módulos)

2.1.4. Definición de *Envelopes* para el CTB

Para finalizar con la implementación de la geometría falta tratar el tema de los *Envelopes*. Estos objetos sirven para albergar los sistemas donde irán los distintos subdetectores. Son tremendamente útiles para poder

corregir distancias y limitar los espacios. Estos *Envelopes* son creados en el paquete *CTB_ G4Sim*[10] con la ayuda de cada subgrupo de simulación (entre los que nos encontramos nosostros).

Estos *Envelopes* se definen en archivos *.mac* (archivos propios de Geant4) y reproducen el volumen total del entorno experimental del CERN para los TB, todas las medidas de las cajas de cada subdetector, los espacios de que se dispone y las medidas de los imames (y sus huecos) y en general, todos y cada uno de los volumenes de que se dispone en la realidad para albergar cada objeto. De modo que una vez definidos los distintos objetos (*SCTBox*, imanes, etc...) deben de estar dentro de su *envelope* para asegurarnos que no hay ningún problema. EL *envelope* del SCT viene definido por las dimensiones reales de la *SCTBox*.

En la fig. 14 pueden verse algunos ejemplos de los *Envelopes* creados.

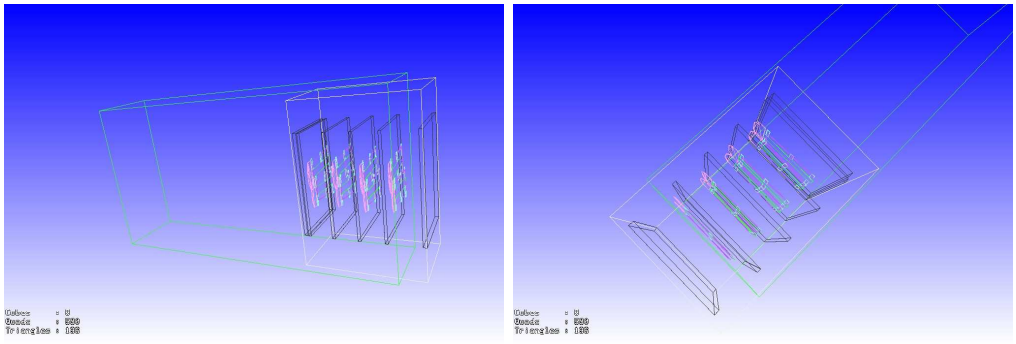

Figura 13: Se puede ver los *envelopes* del hueco del imán (verde) y de la *SCTBox* (blanco)

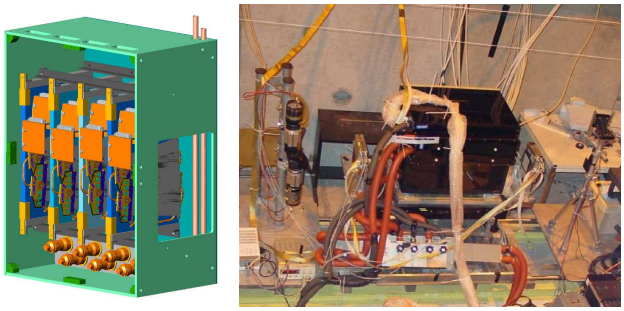

Figura 14: Diseño de la *SCTBox* y la *SCTBox* de Sep2003 en el TB

2.2. Simulación

Con la geometría bien definida y los materiales correctos se puede empezar la simulació de los procesos físicos en esa geometría. Para esto se utiliza Geant4. Pero bajo Athena lo que se utiliza son los llamados *Servicios*. Así, tenemos el paquete *G4Svc* que realiza de forma transparente las funciones de Geant4 para la simulación. También son necesarios otros paquetes para la simulación como son *ParticleGenerator* (generador de partículas), *G4Field* y *FadsField* (para generar y controlar los campos magnéticos) y *G4SimAlgs* (que contiene multitud de algoritmos de diversa utilidad).

[10] Paquete creado y mantenido por Manuel V. Gallas. Más información en: http://mgallas.home.cern.ch/mgallas/ctb_atlas/CTB.html

2.2.1. Generador de Partículas

Lo primero que se necesita es un generador de partículas. Al principio se utilizó el paquete *SingleParticleGenerator* pero hace 1 mes se pasó a utilizar el *ParticleGenerator* por recomendación de fuentes oficiales (Atlas MC Interfaces group).

El *ParticleGenerator* genera partículas simples controladas (una partícula primaria por evento), es decir, que el usuario puede controlar los parámetros de las partículas primarias. Así, tipo de partícula, el vértice inicial, la energía inicial, los momentos, los ángulos, posición del target, polarización a lo largo de los ejes, energía transversal, etc... pueden ser controlados por el usuario.

Además, no sólo permite definir los parámetros fijos sino que puede hacer *loops* sobre diversos valores para cada parámetro (en cada evento coge un valor distinto), o puede seguir distintas distribuciones como por ejemplo generar partículas siguendo una distribución gaussiana alrededor del vértice para acercarse más a la realidad.

Las unidades son las naturales de HEP. Las energías son definidas en MeV y las distancias en mm. Se pueden seleccionar fácilmente las partículas siguiendo el ID de las PDGTABLE.

También puede seguir distribuciones complejas e incluso sacadas de datos experimentales gracias a una nueva función que sigue la distribución almacenada en un histograma. Esto puede ser muy útil, por ejemplo, la generación de partículas alrededor del vértice puede ser un tema no conocido en algunos haces, pero sabiendo la distribución experimental, este generador puede emularla.

2.2.2. Campos Magnéticos

Los campos magnéticos son una parte muy importante en HEP. Nos permiten identificar y seleccionar partículas, así como guiarlas en anillos (por ejemplo, tipo SPS, LEP o LCH en el CERN). En ATLAS los campos magnéticos intensos estarán presentes y serán de una gran utilidad en la identificación de las partículas. En el CTB también habrán 3 imanes que simularán estos campos magnéticos.

Figura 15: Imán MBPS01 donde irá situada la *SCTBox*

El imán MBPS01 (ver fig. 15) es el que albergará la *SCTBox* y la *PIXELBox*. Generará un campo magnético de 1.4 Tesla. La utilidad de los campos magnéticos en física nuclear y de partículas reside en la expresión 1. Con ésta se pueden llegar a la expresión 2 siempre y cuando la dirección de la partícula y el campo magnético sean perpendiculares, cómo es el caso con el $\vec{B}$ generado por MBPS01. Ésta última se utiliza tanto para identificar partículas en distintos detectores como para seleccionar iones en los espectrómetros de masas. En el caso del SCT, que se dedicará pricipalmente a *tracking*, permitirá distinguir partículas.

$$\vec{F} = -q(\vec{E} + \vec{v} \times \vec{B}) \tag{1}$$

$$\vec{F} = -q(\vec{E} + \vec{v} \times \vec{B}) \longrightarrow p = qrB \tag{2}$$

En las fig. 16 y 17 se pueden ver un electrón (amarillo) y un pión (azul) bajo el efeto del campo magnético. La masa es la que hace que a un mismo $\vec{p}$, e^- y π^-, se puedan distinguir en el *tracking* bajo un mismo $\vec{B}$.

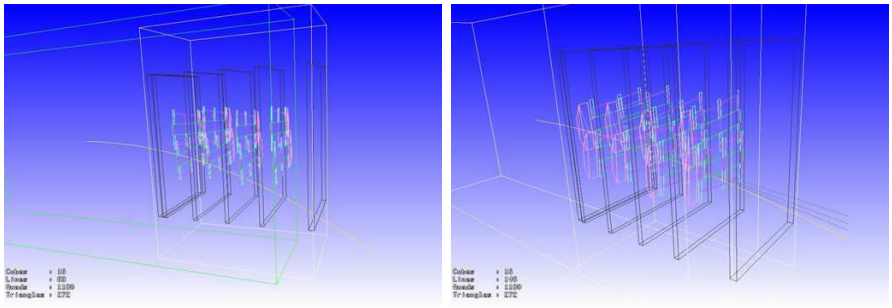

Figura 16: Se tiene un e^- como partícula inidente dentro de un $\vec{B}$ de 1.4 Tesla

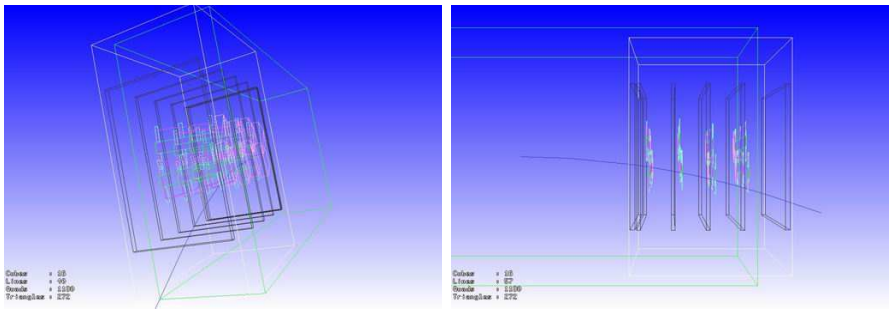

Figura 17: Aquí esta un π^- dentro del mismo $\vec{B}$

2.2.3. *G4Sim* y *G4Svc*

El paquete *G4Svc* es un prototipo de *interface* de Geant4 bajo Athena. Es lo que se conoce como *servicio* en el argot de Athena. Ahora mismo tiene algunas limitaciones y no es tan potente como puede ser el software Geant4 (esto contando únicamente *G4Svc*). La simulación de los subsistemas, incluido el SCT, con este servicio es lo que se llama una *testbed* (un banco de pruebas). El *G4Svc* y otros algoritmos forman parte de un paquete llamado *G4Sim*.

Aquí no vamos a entrar en cómo realiza su trabajo este servicio ya que no es el objetivo de este documento. No obstante, se comentarán las opciones que tiene y cómo se puede personalizar para que la simulación se realice de una manera controlada.

El servicio *G4Svc* es el que invoca la física que se quiere utilizar en dicha simulación. Esta parte se comentará más adelante. También, es el encargado de decidir cuál será la información que se mostrará en pantalla (el grado de detalle), de si se salvarán los datos en memoria o incluso si se quiere iniciar una sesión interactiva en G4. Pero lo más útil es que puede llamar a una serie de macros para realizar todo esto de una

forma estructurada. Esto permite tener un verdadero control y poder optimizar o variar nuestros sistemas de una manera sumamente sencilla y cómoda.

En nuestro caso el servicio puede llamar a dos archivos que hemos creado para nuestras distintas necesidades. El primero realiza toda la simulación considerando todos los volúmenes teóricos, *envelopes*, de los distintos subdetectores en el CTB. El segundo tiene en cuenta únicamente los volúmenes para el SCT y el imán dentro del cuál esta nuestro sistema. La estructura se muestra en la fig. 18.

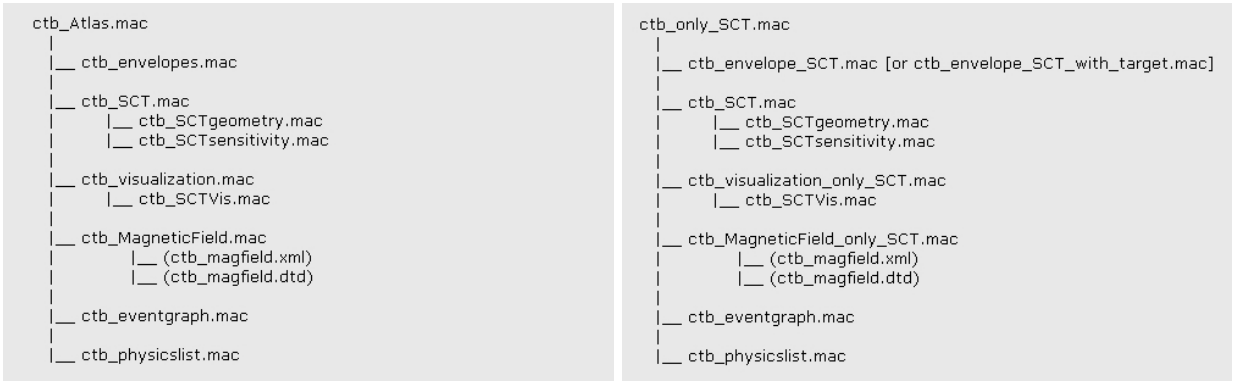

Figura 18: Estructura de los archivos de funciones

El paquete *G4SimAlg* contiene multitud de algoritmos de los cuales nosotros utilizamos el que integra una serie de utilizades para la física de ATLAS.

2.2.4. Física relevante

Los procesos físicos describen cómo las partículas interaccionan con los materiales. La física que se puede poner en juego en la simulación se puede personalizar, por así decirlo. Se puede seleccionar los fenómenos físicos que se pondrán en juego según nuestras necesidades. En algunos casos, evaluar todos los procesos no es relevante y puede consumir muchos recursos (tiempo de CPU y memoria) si la simulación es muy compleja. Esto, normalmente, se traduce en que se tardará más en procesar los datos.

En el paquete *G4Svc* gestiona todos los procesos que se tienen en G4: electromagnéticos, hadrónicos, desintegraciones, opticos, transporte, etc... Todos los procesos físicos son tratados desde el punto de vista del *tracking*. Los procesos (llamados *Physics list*) que se tiene ahora mismo en Athena son:

- *Atlas_Physics* (Procesos relevantes en ATLAS)
- *ExN01* (Transporte de la partícula prueba geantino)
- *ExN02* (Procesos electromagnéticos con geantinos, electrones, positrones y gammas)
- *ExN03* (Procesos electromagnéticos con geantinos, gammas, leptones, mesones ligeros, bariones e iones)
- *ExN04* (Todos los procesos disponibles)
- *Fast_Physics* (Parametrizaciones de modelos EM, etc...)
- *LHEP_GN* (Utiliza modelos de *scattering* inelásticos parametrizados en LEP y HEP)
- *LHEP_BERT* (El anterior es para energías menores a 3 GeV. Para más de 3 GeV aparecen las cascadas de Berini en reacciones inducidas de nucleones y piones)

- *QGSP_GN* (Usa teorías con modelos de reacciones con piones, kaones y nucleones energéticos)
- *QGSP_BERT* (Lo mismo que LHEP_BERT pero para QGSP)

No se tratarán aquí los temas relacionados con su implementación, creación ni bases teóricas. En la simulación del CTB se utiliza *Atlas_Physics*. Las partículas[11] que se usan para chequear y probar la simulación son básicamente electrones, muones, piones y protones a energías de aproximadamente 180 GeV. Por esto, hay que considerar procesos EM, hadrónicos, parametrizaciones, etc... Es en *Atlas_Physics* donde se estan reuniendo los más importantes para optimizar resultados/CPU.

2.2.5. Definición de la parte sensible

Por último, antes de pasar a la digitalización se deben de definir los volúmenes que son sensibles al paso de la radiación. En nuestro caso, estos volúmenes sensibles son las obleas de silicio (4 por módulo, menos en el *short middle* que son 2 por módulo). La definición de estas partes la realiza un paquete adicional llamado *SCT_G4_SD* [12].

Aquí es donde se obtiene la información relativa a la posición local en cada oblea (un punto en el espacio para la oblea de la cara *front* y otro para la oblea de la cara *back*). Esto es necesario para reconstruir el *hit* en la oblea. También, se obtiene información de la energía perdida y de la posición del módulo (posición relativa, es decir, el número de módulo, la cara, etc...). Toda esta información se guarda en memoria (Athena usa lo que se llama *Transient Event Store*[13] para guardar datos en memoria de los eventos. Para guardar cualquier cosa lo hace en *StoreGate* siempre de manera indexeada) para ser posteriormente procesada por otros paquetes.

2.3. Digitalización y Reconstrucción

Como se ha visto las partículas son generadas por *ParticleGenerator* y los efecos físicos que sufre dicha partícula y sus partículas secundarias son simuladas por Geant4. Las señales de cada *strip* individual son creadas por la deposición de energía mediante un algoritmo llamado *digitalización*. A partir de aqui la manipulación de la información se realiza como si de una señal real se tratara. La señal de ambas caras del módulo (1 de cada sensor) forma un *space point*, que no es más que un punto de una traza.

Con lo cual lo que falta por simular es el comportamiento de los módulos SCT. La *digitalización* es la conversión de los datos que nos da Geant4 a *bits*, llamados *digits*, como se ha dicho. Es decir, aquí se emula el comportamiento real del módulo a nivel de electrónica. El paquete que se encarga de realizar esto se llama *InDetDigitization*. Luego se utilizan otros subpaquetes incluidos en el paquete *InDetRecAlgs* (*SiClusterizationTool*, *InDetRIO_Maker*, *SiSpacePointTool* y *SiSpacePointFormation*), para reconstruir los *digits* y formar *clusters*, que no son más que agrupaciones que suelen darse porque el paso de una partícula da señal en más de una *strip* a la vez. Después se forma *space point* como se comentó que es la coordenada real que nos da un módulo SCT en sus coordenadas locales. Después de todo esto ya se puede realizar el *tracking*. De esto se encargan los subpaquetes incluidos en *TrkEvent*. Veámos un poco más a fondo cada paso.

[11] Las partículas utilizadas serán las mismas que las que se utilizaran en el CTB y a las mismas energías. Para más información ver el documento oficial *2004 ATLAS SPS Beam Request* en la web del Testbeam

[12] Paquete desarrollado y mantenido por Davide Costanzo, Grant Gorfine y Thijs Cornelissen. Más información en: http://atlas-sw.cern.ch/cgi-bin/viewcvs.cgi/offline/InnerDetector/InDetG4/SCT_G4/SCT_G4_SD

[13] También conocido por *Event Data Store*

2.3.1. Digitalización y clustering

Como se ha dicho, la *digitalización* es un algoritmo que simula la electrónica. Para conseguir esto se debe de conocer a fondo dicha electrónica y como responde esta a las señales que dan las *strips*. Así por ejemplo, se definen las 768 *strips* que componen cada sensor. También se deben de conocer los *thresholds* de los *strips*, de los *clusters*, etc... Resumiendo, hace falta un profundo conocimiento teórico de los módulos y también hace falta saber cómo responden realmente. Esto último se conoce gracias a todas las pruebas realizadas a los módulos, jugando un papel muy importante los pasados *testbeams*.

Como ejemplo se pone a continuación un trozo del código de salida del la simulación cuando esta procesando los *hits* y convirtiendolos en *digits*.

```
SCT_Digitization DEBUG SiDigitization::getNextEvent()
NovaCnvSvc DEBUG updateAddress failed for CLID 2102
PileUpMergeSvc DEBUG default PileUpEventInfo not found, trying with EventInfo
StoreGateSvc DEBUG retrieve(default)::Retrieved const handle to default object
of type EventInfo(CLID 2101)
SCT_Digitization DEBUG 1 SiHitCollections with key SCT_Hits found
SCT_Digitization DEBUG SiTrackerHitCollection found with 16 hits
SCT_FrontEnd::process starts
old pre at 22 Analogue=22 17546.089
Clustering with threshold 6242.2
strip >th at 22
cluster >th at 22 to 22
there are 1strips
digits >threshold 1 1
SCT_FrontEnd::process ends
SCT_Digitization DEBUG Hit collection ID=[2.2.0.0.0.0.0.0]
SCT_Digitization DEBUG SCTDigitization::createRDO() Collection[2.2.0.0.0.0.0.0]
StoreGateSvc DEBUG Recorded object 0
of type SCT_RDO_Collection(CLID 2535)
object modifiable when retrieved
SCT_Digitization DEBUG SCT RDOs '0' symlinked in StoreGate
SCT_FrontEnd::process starts
old pre at 21old pre at 22 Analogue=21 7575.8955
Analogue=22 23358.96
```

La salida de este paquete la guarda en memoria (*StoreGate*) como debe de hacerlo de manera oficial[14] y en un archivo de texto. Un ejemplo del archivo de texto guardado en un suceso puede verse a continuación.

```
Hit: moduleID=(0,0,0) energyLoss=0.076403 meanTime=0.096548 start=( -0.142500, ...
Hit: moduleID=(0,0,1) energyLoss=0.085674 meanTime=0.099817 start=( 0.142500, ...
Hit: moduleID=(0,1,0) energyLoss=0.073742 meanTime=0.146790 start=( -0.142500, ...
Hit: moduleID=(0,1,1) energyLoss=0.147363 meanTime=0.150216 start=( 0.142500, ...
```

[14] Para poder similar todos los subdetectores a la vez se han creado unas normas. Una de estas normas es trabajar desde memoria, *StoreGate*.

Digit: ID= 0 0 0 ndig=1 ldig=1 clusterLow=22 clusterHigh=22 treshold=6242.200000
Digit: ID= 0 1 0 ndig=1 ldig=2 clusterLow=21 clusterHigh=22 treshold=6242.200000
Digit: ID= 0 0 1 ndig=1 ldig=2 clusterLow=745 clusterHigh=746 treshold=6242.200000
Digit: ID= 0 1 1 ndig=1 ldig=1 clusterLow=746 clusterHigh=746 treshold=6242.200000

2.3.2. *Spacing*

Bien, pues siguiendo el orden de la simulación una vez tenemos los *digits* y/o los *clusters* ya podemos construir los *space points*. La manera de entender como se hace esto es en teoria simple. En un suceso dado se mira en un módulo donde esta el *digit* o *cluster*, es decir, la posición en cada oblea. Después, hay que transformar las coordenadas en las que estas esten en coordenadas locales del módulo. Esto es algo complicado y hay que llevar cuidado. A la fecha en la que se escribe esto tenemos algunos problemas en este apartado. El cambio o transformación de coordenadas se realiza varias veces, ya que se pasa de coordenadas globales (las que da Geant4) a strips (que tienen unas coordenadas dentro de la oblea de silicio) y luego a coordenadas locales del módulo.

Como se hizo anteriormente se presenta un trozo del código de salida en *run-time* de la simulación cuando se encuentra realizando esta tarea.

NewSiTrackerSpa... DEBUG SCT clusters found
NewSiTrackerSpa... DEBUG SiTrackerSpacePointFiner algorithm found no space points
point: (-119.92737,-0.013119721,0.031376184)
StoreGateSvc DEBUG Recorded object 1745
of type SpacePointCollection(CLID 1156991496)
object modifiable when retrieved
NewSiTrackerSpa... DEBUG SpacePoints '1745' recorded in StoreGate
NewSiTrackerSpa... DEBUG SiTrackerSpacePointFiner algorithm found no space points
point: (-104.85704,0.039029775,-1.5700097)
StoreGateSvc DEBUG Recorded object 1747
of type SpacePointCollection(CLID 1156991496)
object modifiable when retrieved

Y también se guarda en memoria los *SpacePoints* a la vez que en un fichero de texto. Un fragmento típico del contenido de este fichero es el siguiente.

SpacePoint: -119.927374 -0.013120 0.031376
SpacePoint: -104.857036 0.039030 -1.570010
SpacePoint: -46.460718 0.009370 2.066802
SpacePoint: -31.319140 0.020941 0.483205
SpacePoint: 27.072870 -0.013279 0.016926
SpacePoint: 42.249373 -0.075812 2.665705

2.3.3. *Traking*

Por último, esta el *traking* que se encarga de intentar combinar los *SpacePoints* anteriores para reconstruir las trazas de las partículas. Esto que a primera vista puede parecer muy fácil, teniendo los puntos, no es trivial. Por supuesto, si tenemos una única partícula y tenemos por ejemplo 4 puntos alineados, es fácil. Sin embargo, lo que se suelen tener es un montón de puntos debidos a las partículas secundarias. Entonces se

deben de realizar todas las combinaciones posibles para encontrar las trazas. Para ello se utilizan criterios de minimización del χ^2 en los ajustes.

Las complicaciones no acaban aquí ya que se deben de considerar otros factores que complican muchísimo la busqueda de las trazas. Por ejemplo, una partícula puede pasar únicamente por el borde de un módulo y hacer que hayan puntos extras que consumen recursos en el procesos de busqueda. También puede ocurrir que las partículas desaparezcan, es decir, que se desintegren o que se paren.

Si a todo esto le sumamos la existencia de campos magnéticos la cosa se vuelve verdaderamente difícil. Aquí hay que considerar curvaturas y por lo tanto realizar algoritmos complicados para intentar ajustar las trazas curvas.

De momento, sólo funciona para trazas rectas, sin campo magnético. No obstante la se esta trabajando muy duro para optimizar estos procesos y ampliarlos.

Se muestra una salida de la simulación cuando ha encontrado una traza.

```
SCT_TrackFitter INFO execute()
StoreGateSvc DEBUG Retrieved const handle to object SCT_SpacePoints of type ...
track: no. of point cols: 16
x: -119.92737 y: -0.013119721 z: 0.031376184
x: -104.85704 y: 0.039029775 z: -1.5700097
x: -46.460718 y: 0.0093699319 z: 2.0668021
x: -31.31914 y: 0.020940743 z: 0.48320514
x: 27.07287 y: -0.013279381 z: 0.016926134
x: 42.249373 y: -0.075811968 z: 2.6657047
x: 100.53841 y: 0.0099618546 z: 2.1203741
x: 115.749 y: -0.075636936 z: 2.6462669
finding tracks with 8 points
fitpoints points.size(): 8
TRACK FOUND: y=-0.012906712 gradient y=-0.00027768216
z=1.0848068 gradient z=0.012846561
```

Como siempre se guarda en memoria los resultados y en un fichero de texto. Aqui esta el contenido de un fichero cualquiera para este caso.

```
Track-point: x= -105.007135 y= 0.891547 z= 8.909213
Track-point: x= -46.748244 y= 1.005266 z= 9.862436
Track-point: x= -31.580262 y= 1.054535 z= 8.002233
Track-point: x= 26.664950 y= 1.146038 z= 14.164527
Track-point: x= 41.943568 y= 1.185044 z= 12.672264
Track-point: x= 100.152845 y= 1.298968 z= 13.140083
Track-point: x= 115.373864 y= 1.347646 z= 11.980317
TRACK_FOUND: y= 1.103693 gradient_y= 0.002016 z= 10.939496
gradient_z= 0.021375 chi2y= 0.270848 chi2z= 4.672927
```

Por último, para ilustrar esto un poco, si consideramos los *Hits, Digits, Clusters* y *SpacePoints* tenemos un montón de números de partículas que pasan por los módulos (8 en el mejor de los casos).Para entender mejor estos números se han realizado un ajuste para ver cómo quedan. Tenemos la posición global que nos da Geant4 mediante los *Hits* (en realidad tenemos la posición de entrada y la de salida en la oblea sensible) y los *SpacePoints* definidos gracias a *strips* con señal y algunos parámetros de las trazas. Después de una manipulación conveniente de las coordenadas de unos y otros tenemos lo siguiente.

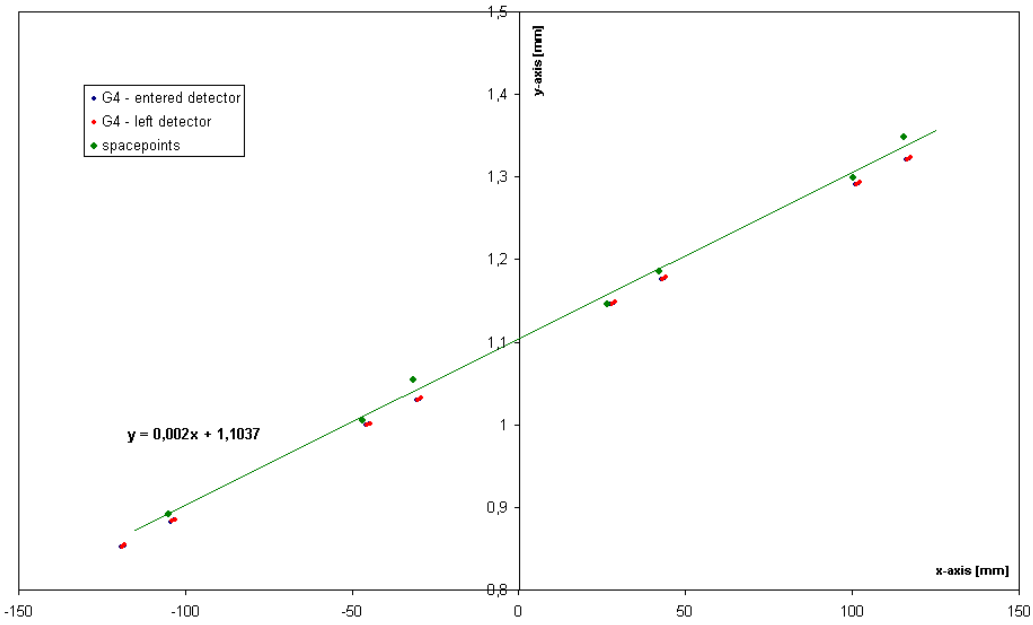

Figura 19: Vista X-Y

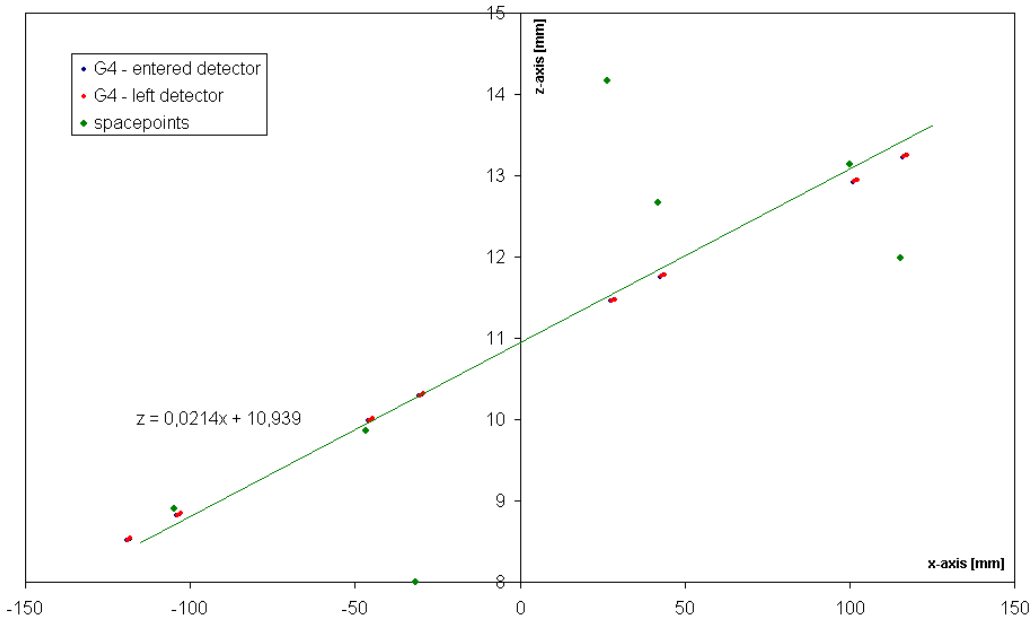

Figura 20: Vista X-Z

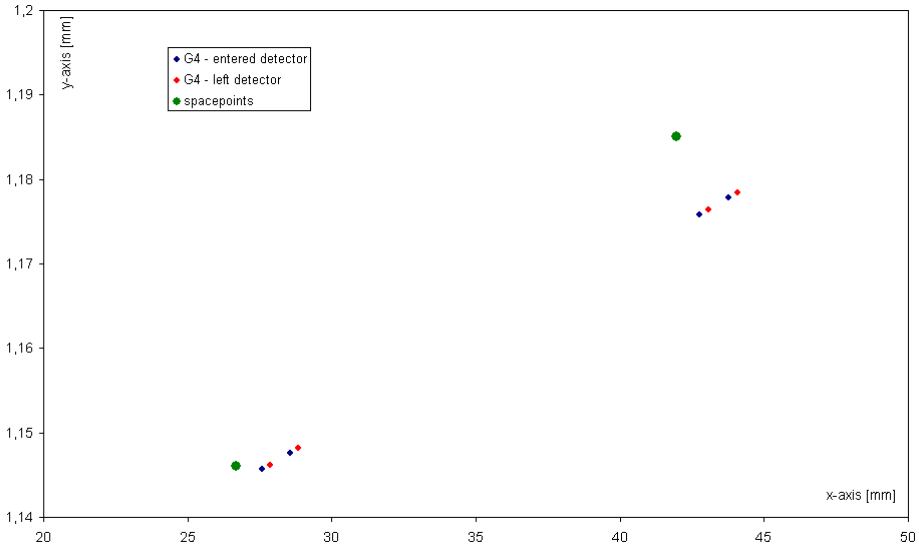

Figura 21: Detalle Vista X-Y

Como se ha podido ver antes los resultados que se obtienen cuando se encuentra un traza son: *y, gradient_y,z, gradient_z* y los χ^2 de y y de z. Este ajuste nos da las proyecciones de la traza en 3D sobre los planos XY y XZ. Recordar que el haz es paralelo a X. Las expresiones con las que se trabaja son:

$$Y = y + gradient\_y * X \tag{3}$$

$$Z = z + gradient\_z * X \tag{4}$$

2.4. Validación y Chequeos

El proceso de validación es uno de los pasos más laboriosos. Se trata de contrastar los datos y resultados obtenidos en la simulación con datos y resultados experimentales. El proceso pasa primero por ir puliendo posibles diferencias con la realidad. Después, hay un gran trabajo por parte de mucha gente para producir eventos y analizar los resultados.

Actualmente, se esta trabajando en esta fase en el simulación del SCT en el CTB. Para realizar esto se utiliza la configuración de mayo y septiembre de 2003. Durante abril y mayo de 2004 se espera tener realizada esta tarea. La simulación de todos los subdetectores a la vez está algo más retrasada (ya que es muy difícil coordinar todos los grupos), aún se esta comprobando la generación de los hits y el correcto almacenamiento en *POOL*[15].

También se esta utilizando la configuración del *CTB2004* y comparandola con los resultados de un *script* de simulación hecho en *ROOT* por Peter Kodys.

Lo único que se puede comentar son los resultados que se tienen de la simulación. En el apartado siguiente se exponen los resultados que se tienen hasta la fecha.

[15] *POOL* es el formato elegido para almecenar los datos de la simulación del CTB global

2.5. Resultados de la Simulación

Actualmente se han generado unos ficheros en $ROOT$[16] con unas características interesantes para el análisis de los datos. Esto es muy útil para la gente que se encarga del *tracking* y del alineamiento.

Los ficheros se diferencian en las configuraciones utilizadas para realizar la simulación. Además se han incluido algunas restricciones para hacer las cosas más sencillas:

- 256 *hits* de partículas secundarias por evento como máximo
- 512 *clusters* para la digitalización por evento como máximo
- Busqueda de 256 *spacepoints* para fittar trazas por evento como máximo
- Busqueda de 5 trazas por evento como máximo

En la figura 22 se puede ver la sección a considerar en los módulos.

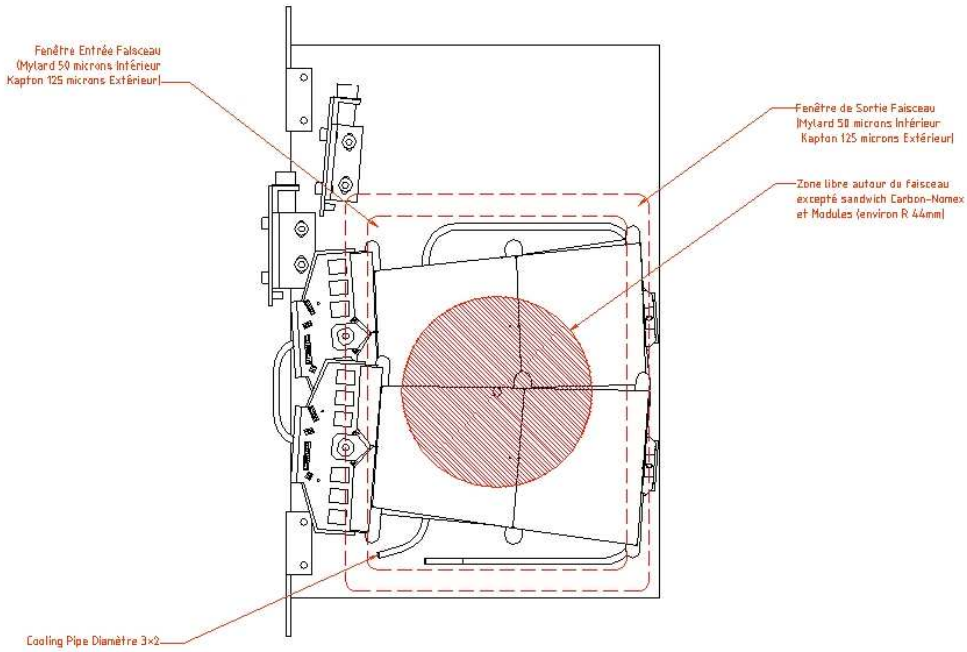

Figura 22: Sección transversal al haz a considerar

Para generar los ficheros se han utilizado combinaciones de las siguientes características:

- Posición de los módulos según la configuración de CTB2004
- Se utiliza un haz de Piones de 180 GeV paralelos al eje X. Se define dicha fuente con una sección simétrica 50x50 mm aleatoria alrededor del (0,0,0) en coordenadas de Geant4
- Campo magnético de 1.4 Tesla en dirección Z
- Un blanco de Tungsteno de 3mm de grosor y 10x10 mm en Y y Z paralello al haz
- 5000 o 10000 eventos

[16] Los datos de los archivos de texto se han convertido a datos estructurados en ficheros $ROOT$

2.6. Simulación Completa y Física

Lo que se espera en un futuro muy cercano es que se empiecen a medir las cosas que se miden en el *testbeam* realmente. Es decir, se espera empezar con cuestiones físicas tales como la eficiencia, la ocupancia de ruido, las trazas falsas (ruido), etc... porque lo que nunca hay que olvidar en una simulación es que se pretenden responder o ayudar a reponder preguntas que tengan utilidad y contribuyan a enriquecer la Física actual.

3. Bibliografía

La bibliografía es básicamente la primera referencia ya que hemos reunido todo en la web. La web ha sido estructurada siguiendo todo el proceso de simulación. El equipo lo forman: Peter Kodys, Zdenka Broklova, Thijs Cornelissen y Carlos Escobar

1. SCT Simulation - *http://ific.uv.es/~cescobar/simulation/*
2. Combined TestBeam Simulation - *http://mgallas.home.cern.ch/mgallas/ctb_atlas/CTB.html*
3. NOVA DATABASE - *http://web13.cern.ch/atlas-php/NOVA/*
4. ATLAS Barrel Combined Run in 2004 Test Beam Layout, *Beniamino Di Girolamo*, 2003
5. ATLAS Barrel Combined Run in 2004 Test Beam Layout - The Inner Detector, *Beniamino Di Girolamo*, 2003
6. Full description of the MBPS magnet with curves and a drawing - CTB2004
7. Combined Testbeam 2004 - SCT Box Design, 2004