Simulaciones de centelladores PET con LITRANI

Adriana Martín de Aguilera Moreno
Trabajo Académicamente Dirigido por José Manuel Udías Moinelo
y Samuel España Palomares

25 de septiembre de 2008

Índice

Introducción a la Tomografía por Emisión de Positrones

En este trabajo se va a llevar a cabo el estudio y simulación de uno de los elementos que conforman un aparato de PET común. La tomografía por emisión de positrones es una técnica de imagen médica que permite obtener imágenes *in vivo* del organismo de un animal o de un ser humano. Es por ello que vamos a empezar por realizar una revisión de los conceptos básicos que entran en juego en esta técnica, con especial atención a aquellos que se van a convertir en el objeto de nuestro estudio: los cristales centelladores que se utilizan como detectores.

1.1. Desintegraciones nucleares

Puede encontrarse un tratamiento más detallado sobre los procesos de desintegración nuclear en cualquier libro de Física Nuclear como, por ejemplo, [1]. En Física Nuclear, cuando un núcleo atómico es inestable, es decir, posee una cantidad de neutrones y protones que no le permite mantener su integridad indefinidamente, entra en un proceso de desintegración o decaimiento que culmina con la emisión de algún tipo de partícula o radiación a fin de modificar el propio número atómico para alcanzar una mayor estabilidad. Como es bien sabido, existen tres canales básicos a través de los que se produce esta desintegración:

1. **Emisión α:** $(Z, A) \rightarrow (Z-2, A-4) + {}^4_2He$.
2. **Emisión β^+:** $(Z, A) \rightarrow (Z-1, A) + e^+ + \nu_e$ o β^-: $(Z, A) \rightarrow (Z+1, A) + e^- + \overline{\nu}_e$.
3. **Emisión γ:** en este caso no se produce desintegración, sino decaimiento, puesto que el núcleo que la sufre se limita a pasar de un estado excitado a uno de menor energía mediante la emisión de un fotón gamma sin que esto afecte a su número atómico.

El ritmo al que se producen las desintegraciones nucleares es un dato de gran importancia, pues el tiempo en el que una muestra de cierto material puede pasar a convertirse en otro va desde los nanosegundos hasta tiempos superiores a la edad del Universo. La *ecuación de supervivencia* o *ley de desintegración* describe cómo evoluciona en el tiempo una muestra inicial de $N(0)$ núleos susceptibles de sufrir una cierta desintegración:

$$N(t) = N(0)e^{-\omega t}$$

donde ω es la probabilidad de que el estado X pase al Y por segundo. Esto nos lleva a introducir el concepto de *vida media* $\tau = \omega^{-1}$, es decir, el tiempo que la cantidad de núcleos supervivientes es una e^{-1} parte de la original. También hablamos de la *vida media* $t_{1/2} = \tau ln2$, que es el tiempo en el que la cantidad inicial de núcleos se reduce a la mitad.

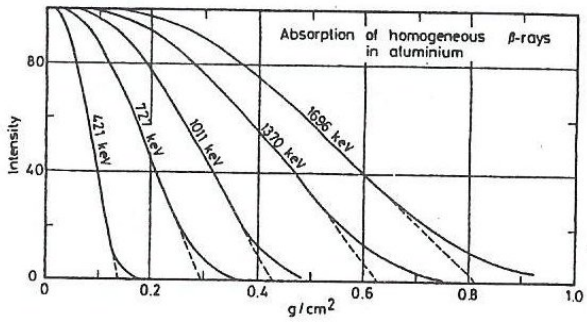

Figura 1.1: Atenuación de fotones gamma de diversas energías por el aluminio.

1.1.1. La desintegración β

En la desintegración β además del electrón o positrón se emite también un neutrino. Al entrar en juego dos partículas la energía con la que éstas se emiten se tiene que repartir. Por ello, la energía con la que el electrón o positrón salga despedido puede ir desde la máxima permitida por la desintegración si el neutrino no recibe energía hasta el caso contrario, en el que la energía cinética aportada al electrón es nula, mientras que la del neutrino es toda la disponible. Por ello el espectro de emisión en la desintegración beta es continuo.

La técnica PET se basa en la emisión β^+ y la posterior aniquilación del positrón para marcar radiactivamente compuestos químicos.

1.1.2. Absorción de los positrones β

Como el perfil de emisión β es continuo, la absorción de los electrones y positrones producto de estas desintegraciones sigue una ley exponencial: $I = I_0 e^{-\mu x}$, donde μ es el *coeficiente de absorción* β, que está directamente relacionado con la máxima energía que puede adquirir el positrón en cada desintegración. Obviamente, también depende del material.

1.1.3. Aniquilación de pares electrón-positrón

Como ya hemos visto, la desintegración β^+ implica la emisión de positrones. Si irradiamos un material con una fuente de positrones, éstos se aniquilarán con los electrones que lo conforman, produciendo dos fotones de masa equivalente a la de las dos partículas, esto es, 511 KeV, y que se propagarán en direcciones opuestas.

1.2. Detección de fotones

1.2.1. Interacción de la radiación con la materia

Puesto que son los fotones gamma producto de la aniquilación electrón-positrón los que necesitaremos detectar en un aparato de PET, debemos estudiar los cristales centelladores que se usan a tal fin. Antes de llegar a este punto es conveniente tener una visión general de los procesos que se pueden dar en estos cristales cuando los alcanza un fotón gamma. Si se quiere un

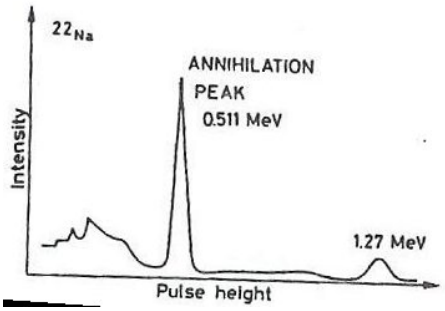

Figura 1.2: Aspecto de un pico de aniquilación de pares en un espectro de energías.

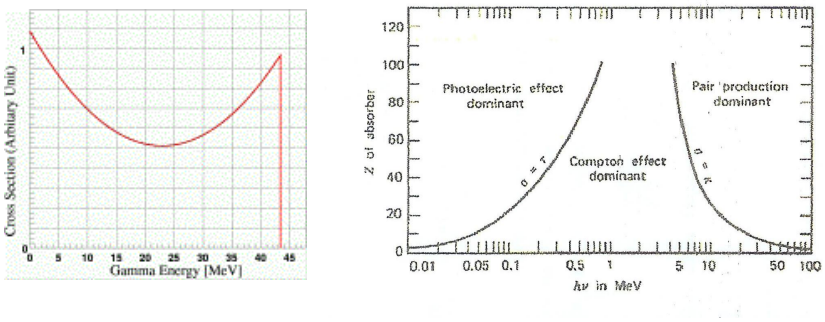

Figura 1.3: Borde Compton ideal y representación de la importancia relativa de las tres principales interacciones radiación-materia para energías y Z efectiva.

tratamiento más detallado sobre la interacción de la luz con la materia y los detectores pueden consultarse los textos [2] y [6].

Las principales interacciones entre los fotones (rayos X y γ) y la materia son:

1. **Efecto fotoeléctrico:** el fotón es absorbido por un electrón atómico y, en consecuencia, éste queda libre con una energía $E = h\nu - B.E.$, donde B.E. es la energía de ligadura del electrón.

2. **Scattering Compton (incluyendo scattering Thomson y Rayleigh):** al igual que en el efecto Compton, los fotones inciden contra los electrones de la materia, transfiriéndoles algo de energía y, por lo tanto, modificando su longitud de onda y trayectoria. La sección eficaz para este proceso está dada por la *fórmula de Klein-Nishina*: $\frac{d\sigma}{dT} = \frac{\pi r_e^2}{m_e c^2 \gamma^2}[2 + \frac{s^2}{\gamma^2(1-s^2)} + \frac{s}{1-s}(s - \frac{2}{\gamma})]$, donde $s = T/h\nu$. Esta expresión nos permite definir la energía máxima a la que se da el scattering Compton: $T_{max} = h\nu\frac{2\gamma}{1+2\gamma}$.

3. **Producción de pares:** implica la conversión de un fotón en un par electrón-positrón. Por lo tanto, la energía mínima del fotón para que esto suceda es 1,022 MeV.

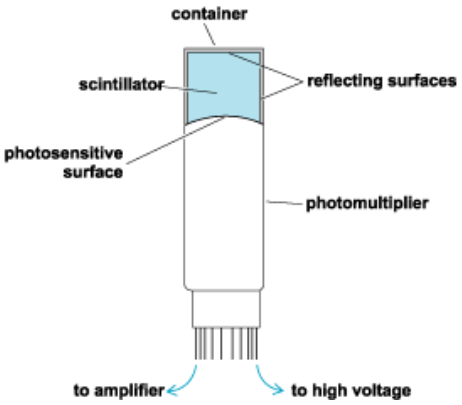

Figura 1.4: Diagrama de un detector compuesto por un cristal centellador y un fotomultplicador.

1.2.2. Cristales centelladores

Un método para detectar radiación ionizante es el estudio de la luz de centelleo que esta puede producir en el interior de ciertos materiales. Estos detectores están formados, básicamente, por un material unido ópticamente a un fotomultiplicador. La radiación, al atravesarlo, excita los átomos y moléculas, de modo que éstas vuelven a la normalidad emitiendo nuevos fotones que son transmitidos al fotomultiplicador, donde se traducen en una corriente eléctrica.

Los centelladores poseen una propiedad llamada *luminiscencia*. Los materiales luminiscentes absorben la energía a la que estén expuestos (luz, calor...) y la reemiten como fotones del espectro visible. Dentro de los materiales luminiscentes distinguimos entre los fluorescentes, que llevan a cabo esta reemisión de energía menos de $10-^{-8}$ segundos de recibirla y los fosforescentes, que tienen una respuesta algo más retardada. En una primera aproximación, el proceso de reemisión sigue una exponencial: $N = \frac{N_0}{\tau_d} e^{-t/\tau_d}$.

Las propiedades que debe presentar un material para ser un buen centellador son:

1. Debe convertir la energía cinética de las partículas cargadas en luz detectable con una gran eficacia en el centelleo.
2. Esta conversión debe ser lineal, es decir, la luz recibida debe ser proporcional a la emitida en el centelleo en el rango más amplio posible.
3. El medio ha de ser transparente a su propia longitud de emisión para que no se produzcan atenuaciones.
4. La vida media de la luminiscencia inducida debe ser corta para que puedan producirse pulsos rápidos.
5. El material debe tener una buena calidad óptica y debe poder fabricarse con facilidad en los tamaños que deseemos para nuestros instrumentos.
6. Su índice de refracción debe ser parecido al del vidrio para permitir una unión lo menos abrupta posible con el fotomultiplicador.

Según su naturaleza, podemos encontrar centelladores **orgánicos** o **inorgánicos**. Puesto que nosotros vamos a trabajar con estos últimos, vamos a comentar algunas de sus características. Los centelladores inorgánicos son cristales alcalinos con alguna impureza activadora. En

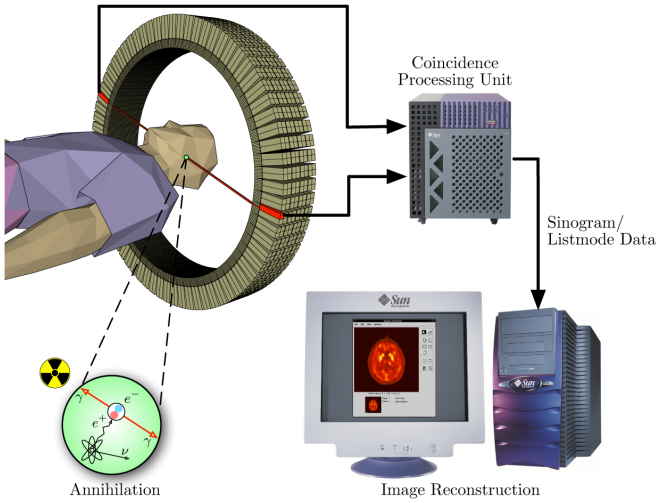

Figura 1.5: Representación esquemática del anillo detector de un aparato de PET para humanos y de la detección de dos fotones gamma en coincidencia.

general son 2 o 3 órdenes de magnitud más lentos en su respuesta que los orgánicos, pero tienen mejor resolución en energía. Esto los hace especialmente útiles en la detección de radiación gamma y electrones de alta energía.

En los centelladores orgánicos el mecanismo de centelleo es esencialmente debido a los niveles moleculares, pero en los inorgánicos es consecuencia de la estructura de bandas del cristal. Cuando una partícula nuclear se introduce en el cristal pueden suceder dos cosas: puede ionizar el cristal al excitar un electrón de la banda de valencia a la de conducción, creando un electrón y un hueco libres; o puede excitar un electrón a la banda que se encuentra justo bajo la de conducción. En este estado, el electrón y el hueco quedan unidos en un par que se puede mover libremente por el cristal. Si el cristal contiene impurezas, pueden aparecer niveles electrónicos locales en la banda prohibida. Un hueco que se encuentre libre o en un par con un electrón que se encuentre con una de estas impurezas puede ionizarla. Si, posteriormente, llega un electrón, puede ocupar el hueco dejado por la ionización. El electrón, al recombinarse con la impureza, queda en un estado excitado que, al desexcitarse emite luz en el visible.

1.3. Fundamentos de un aparato de PET

Ya hemos explicado que los positrones debidos a una desintegración β^+ se recombinan rápidamente con los electrones de la materia, emitiendo dos rayos gamma de 511 KeV en sentidos opuestos. Si conseguimos introducir una muestra que se desintegre por un canal β^+ en el seno de un anillo de detectores, podremos detectar los fotones gamma en dos puntos diametralmente opuestos de él. El fundamento del PET es introducir un emisor β^+ de corto tiempo de decaimiento en un organismo vivo e introducir éste en uno de éstos anillos detectores. Para ello se emplean isótopos radiactivos de átomos que formen parte de moléculas biológicas que sean metabolizadas por el sujeto que queremos estudiar. Dichos compuestos pueden ser ingeridos o inyectados al paciente según las necesidades. Mientras estas moléculas con los marcadores radiactivos hacen su recorrido, se emplean cristales centelladores para detectar los fotones gamma de 511 KeV que se vayan emitiendo. Se darán por válidos los fotones que se detecten en una

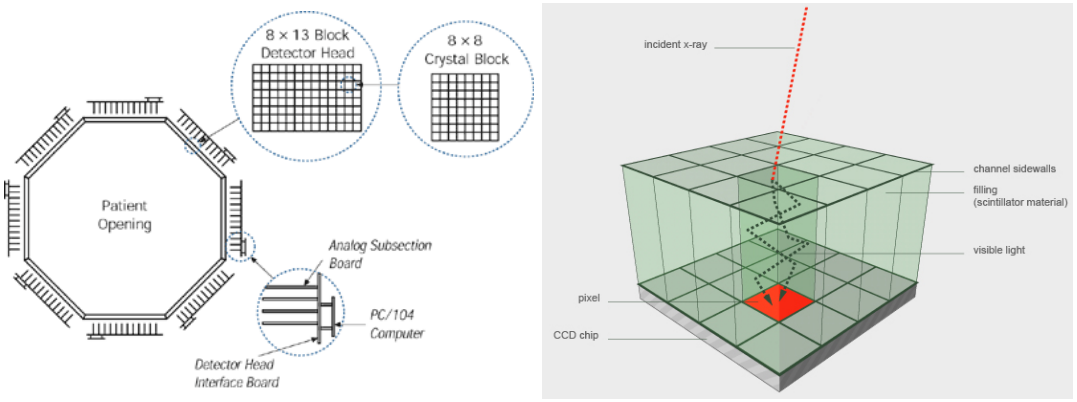

Figura 1.6: Esquemas de la distribución de los detectores en un anillo poligonal y de los cristales centelladores en cada detector.

reducida ventana temporal y en puntos opuestos de la matriz detectora. Suponiendo que las trayectorias de éstos no se han desviado o solamente lo han hecho un poco, se puede reconstruir para situar el origen del par electrón-positrón que los ha producido.

Midiendo el número de fotones en coincidencia detectados, así como calculando la posición en la que éstos se han originado, puede hacerse una reconstrucción de la actividad metabólica de la zona que hayamos estado estudiando. Veremos manchas de luz intensas allá donde se han desintegrado muchos marcadores, esto es, en zonas de alta actividad metabólica para el compuesto que hayamos utilizado. Por lo tanto, el PET es una técnica no invasiva que nos permite estudiar *in vivo* la actividad de un organismo o un órgano concreto. Esto tiene multitud de aplicaciones en oncología, cardiología y neurología.

1.3.1. Detectores de radiación en un aparato de PET

El sistema que se utiliza en los aparatos de PET para detectar los rayos gamma emitidos por los trazadores es el de un soporte (circular o cuadrado, según el aparato) de cristales centelladores. Ya hemos hablado sobre estos detectores antes y hemos visto que son los más adecuados para el estudio de la radiación gamma. Puesto que detectar los fotones y situar su origen son dos puntos fundamentales para obtener una buena imagen, el funcionamiento de los detectores debe ser óptimo, tanto en el número de fotones detectados como en la correcta situación del cristal en el que el fotón ha depositado la mayor parte de su energía.

Como se puede observar en la figura 1.6, cada detector del anillo está formado por una matriz compuesta por un gran número de cristales centelladores. Dichos cristales estarán separados entre sí por algún recubrimiento reflector, de modo que cada cristal será un píxel y estará en contacto con el fotodetector. Puesto que todos los cristales de cada matriz van a parar al mismo fototubo, no nos es posible saber a ciencia cierta en qué cristal se ha producido la llegada del fotón, pero tenemos medios estadísticos para, mediante el estudio de simulaciones anteriores, calcular el punto de la matriz en el que se ha dado la llegada. Como ya hemos dicho, esto es de capital importancia para detectar una coincidencia, puesto que si el radio del anillo es suficientemente grande, una pequeña equivocación en la situación del fotón detectado puede hacernos creer que dos fotones procedentes de la misma desintegración no se detecten en coincidencia por no encontrarse en posiciones enfrentadas.

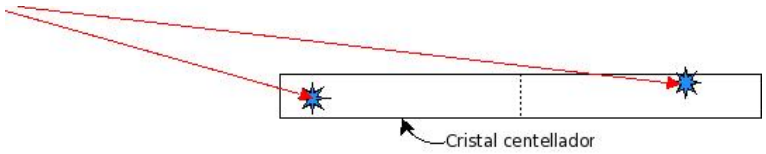

Figura 1.7: Esquema de la llegada de dos fotones con distinta inclinación al mismo cristal centellador.

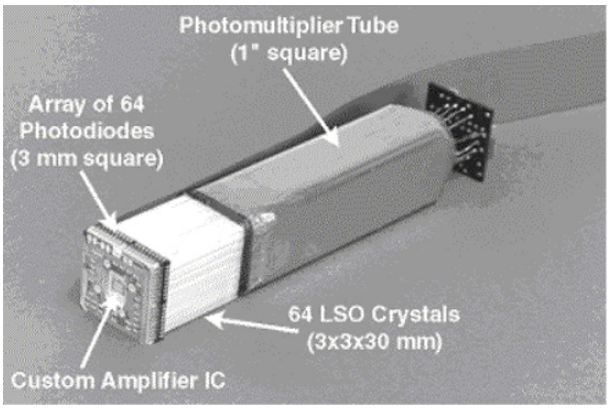

Figura 1.8: Matriz de cristales centelladores empleada en un aparato de PET.

1.4. Motivación y objetivos

1.4.1. Qué problema queremos resolver

Nuestro estudio se va a centrar en las matrices de cristales centelladores de los detectores de un aparato de PET. Queremos incrementar la precisión a la hora de situar la aniquilación del par electrón-positrón a partir de la información que obtenemos en los fotomultiplicadores. Hay muchos métodos para estimar el cristal de llegada del fotón, así que nos interesa seguir otro camino.

La figura 1.7 pone de manifiesto que fotones que, al haberse originado uno cerca del otro pero no en el mismo lugar, pueden llegar regiones distintas del mismo cristal. Saber si el fotón que detectamos ha llegado a la parte delantera o a la trasera del cristal nos ayudaría a calcular con más precisión la trayectoria y el origen de cada fotón. Por lo tanto, vamos a intentar idear algún sistema que nos ayude a saber en qué parte del cristal ha dejado el fotón la mayor parte de su energía. Para ello vamos a llevar a cabo diversas simulaciones del interior de los cristales centelladores.

Muchas matrices para detectores de PET, como la de la figura 1.8, constan de dos bloques de cristales centelladores unidos mediante un gel o una grasa óptica, además del recubrimiento reflector que los separa. Esta 'división' del cristal centellador en dos mitades iguales nos va a ser de mucha ayuda, puesto que separa naturalmente las dos partes del centellador que queremos estudiar. Nos interesa estudiar este sistema de dos cristales separados por un gel óptico y recubiertos por un material reflector para saber si el fotodetector apreciará alguna diferencia en la cantidad de fotones vistos si generamos fotones en el primer o el segundo cristal.

1.4.2. Qué herramientas vamos a utilizar

Por lo tanto, necesitamos simular lo que les sucede a los fotones que produce el cristal centellador en un sistema de cristal+recubrimiento+fotodetector. Para ello necesitamos un paquete informático de simulación óptica. Existen varias herramientas que, en principio, pueden simular este tipo de sistemas empleando métodos de Monte-Carlo y Óptica Electromagnética. Por lo tanto, vamos a empezar por presentar uno de esos programas, LITRANI, y exponer los rudimentos de su manejo que hemos aprendido durante la realización de este trabajo. Posteriormente, vamos a resolver algunos problemas sencillos de Óptica Electromagnética con él para comprobar que sus resultados coinciden con los cálculos teóricos y, por último, simularemos los fotones en el interior de los cristales centelladores.

Introducción a LITRANI

Puesto que va a ser la principal herramienta que vamos a emplear en nuestras simulaciones, vamos a hacer una exposición lo más breve y sencilla posible de las principales funciones de LITRANI y de los comandos de los que vamos a hacer uso a lo largo de este trabajo, a fin de que los resultados finales sean más claros, sobre todo de cara a futuros usuarios.

2.1. ¿Qué es LITRANI?

Las siglas LITRANI proceden de *LIght TRansmission in ANIsotropic media*. LITRANI es un paquete de simulación Monte Carlo construido sobre ROOT que simula el paso de luz a través de un montaje definido por formas y superficies previamente construidas en ROOT. Cada una de estas formas puede estar constituida por un material diferente, cuyas propiedades ópticas pueden ser constantes o depender de la longitud de onda. Además, permite simular materiales cuyo tensor dieléctrico es anisótropo además de los recubrimientos que los envuelven.

LITRANI emplea en sus cálculos óptica de Fresnel, es decir, la Óptica Clásica, con la excepción de que que si hay capas finas de material entre una forma y su recubrimiento tiene en cuenta las posibles interferencias que se pueden dar si la anchura de la capa es comparable a la longitud de onda.

2.2. Características de LITRANI

2.2.1. Ventajas

1. Descripción detallada de las superficies: rugosidad, recubrimientos, huecos, etc.
2. Es fácil de ampliar usando macros de ROOT.
3. Incluye varios modelos de fuentes y detectores de fotones comerciales, pudiéndose incluir nuevos modelos fácilmente.
4. Es válido en medios isótropos y anisótropos.

2.2.2. Inconvenientes

1. Sólo se dispone de formas simples (no pueden extenderse). No se pueden situar unas formas dentro de otras.
2. Sólo montajes fáciles (GEANT simula mejor volúmenes complejos).
3. Tiene dificultades con las superficies curvas.

4. El modelo es incorrecto para la propagación en el tiempo, pues sólo tiene en cuenta la velocidad de fase.

5. Los datos necesarios para definir un material no son los que se utilizan normalmente en la vida real y es muy difícil encontrarlos.

2.3. ¿Qué se puede hacer con LITRANI?

En primer lugar, distinguimos tres fases o módulos que componen LITRANI:

1. **LITRANI (TwoPad):** haciendo uso de las librerías y elementos que ahora describiremos, permite crear el montaje que deseamos simular.

2. **SplineFit:** módulo de interpolación de datos cuyo fin es, empleando datos dados por el usuario, realizar interpolaciones para su uso en otra simulación de TwoPad. Suele ser muy importante para introducir índices de refracción de materiales, permeabilidades o secciones eficaces en montajes con fuentes de luz no monocromáticas.

3. **VisuLitrani:** encargado de mostrar histogramas y resultados producto de las simulaciones. El número y carácter de dichos resultados puede modificarse al escribir el programa.

Litrani dispone de diversos catálogos de formas, fuentes de luz y detectores. Haremos un repaso a las principales librerías de que dispone:

1. Formas

   a) TSBRIK: caja de caras perpendiculares entre sí.

   b) TSTRD1: caja trapezoidal con la dimensión x variable a lo largo de z.

   c) TSTRD2: caja trapezoidal con las dimensiones x e y variando a lo largo de z.

   d) TSPARA: paralelepípedo.

   e) TSTRAP: trapezoide general.

   f) TS8PTS: volumen general que se determina mediante sus ocho vértices.

   g) TSCONE: cono.

   h) TSCYL: cilindro.

   i) TSTUBE: tubo con hueco para introducir un cilindro en su interior.

2. Fuentes de luz

   a) Espontánea: emite fotones desde cualquier punto de un volumen o una superficie.

   b) Fibra óptica: los fotones proceden de una fibra óptica en el interior del volumen.

   c) Haz de partículas: genera haces de, por ejemplo, muones, capaces de producir luz Cherenkov en el intetior del material.

   d) Gammas: fotones de entre 0,1 y 1 MeV, teniendo en cuenta el efecto Compton y el efecto fotoeléctrico en el interior del material.

   e) Cascadas electromagnéticas.

3. Detectores

a) Detectores de superficie.

b) Detectores de volumen.

c) Phototubos.

d) APD's.

e) Diodos PIN.

Todos estos elementos pueden emplearse en el montaje, siempre y cuando se sigan algunas normas que explicaremos en las páginas siguientes. Además, es posible añadirles recubrimientos, es decir, simular la presencia de materiales metálicos adheridos a las caras de los volúmenes implicados en cada montaje. La simulación de recubrimientos es uno de los puntos más dinámicos de LITRANI, puesto que pueden simularse distintas bandas de material sobre una cara o zonas recubiertas con formas geométricas como círculos de metal. Además, es posible añadir caras erosionadas a las superficies.

2.4. Elementos de un montaje en LITRANI

Un montaje de LITRANI puede hacer uso de todos los objetos básicos que hemos descrito, así como de numerosos elementos adicionales que escapan de la intención de esta introducción; pero dichos objetos han de distribuirse forma de ordenada y siguiendo diversas reglas. La mejor forma de explicar cómo se debe realizar uno de estos montajes es mediante un ejemplo:

```
void ejemplo1a(void)

{
  char *name    = "EJEMPLO 1a. Un cristal de 1cm^3 de  LSO recubierto
de material totalmente reflector";
  char *listing = "La cara última del cristal es un fotocátodo ";
  char *upcom   = "La fuente es puntual, isotropica y espontanea, y la
situo dentro del cristal ";

  char *downcom = "¿Cuántos fotones detecto?";
```

En primer lugar, cargamos LITRANI y especificamos el título y características del programa:

```
gROOT->ProcessLine(".x InitLitrani.C(8,name,listing,upcom,downcom,kTRUE,kFALSE
,kTRUE)");
```

También necesitamos llamar a ROOT y establecer algunas de las características de la generación de números aleatorios. Hemos empleado poco esta función, y lo poco que se ha manejado queda descrito en los siguientes capítulos. Tras esto, definimos las semilongitudes de los cuerpos que van a intervenir:

```
//1.- GEOMETRÍA

        const Double_t semilongitud_en_x = 0.5;
        const Double_t semilongitud_en_y = 0.5;
        const Double_t semilongitud_en_z = 0.5;
```

Acto seguido introducimos las propiedades de los materiales y del revestimiento que intervienen en el montaje, en nuestro caso LSO con recubrimiento totalmente reflector.

```
//2.- MATERIALES

        TOpticMaterial *LSO;
        LSO = new TOpticMaterial("LSO","Titulito,¡es LSO!",kFALSE,1.0,21);
        LSO->IsIsotropic(1.82);
```

Para definir el material debemos introducir los siguientes campos, entre paréntesis y separados por comas.

1. Nombre con el que nos referiremos a él de ahora en adelante. Cada vez que utilicemos el material lo llamaremos mediante esta designación. Hay que ser cuidadoso, pues distingue mayúsculas y minúsculas.
2. Título que adjuntamos al material para recordar.
3. Sensibilidad, es decir, si queremos que sea un detector, kTRUE y si no lo es, kFALSE. Por ejemplo, si definimos el silicio para que sea la parte sensible de un detector APD, entonces estableceremos kTRUE.
4. Permeabilidad magnética, que si es constante bastará con una cifra concreta para definirla y si varía con la longitud de onda se llama a al programa con sus valores interpolados escribiendo el nombre de éste entre comillas.

Para definir el índice de refracción del material utilizamos:

```
LSO->IsIsotropic(1.82);
```

El material puede ser isotrópo o anisótropo. En este último caso se definen cada uno de los elementos del tensor dieléctrico con los comandos `IsUniAxialNegBirefr()`, `IsUniAxialPosBirefr()` si el tensor presenta inhomogeneidad negativa o positiva, respectivamente, en uno solo de sus ejes o `IsBirefr()` si cada eje posee una constante dieléctrica distinta. Podríamos definir otras características del material tal que si es fluorescente o también definir una material que envuelva a todo nuestro experimento.

Los recubrimientos se definen de forma ligeramente diferente:

```
        TRevetment *reflectortotal;
        reflectortotal = new TRevetment("reflectortotal","Recubrimiento reflector total",
"none",0.00, 0.0,-1.0,1.0,0.0,90.0);
```

Los elementos que describen al recubrimiento en este caso son:

1. Nombre.
2. Título.
3. Declaramos si hay una capa fina de algún material entre el cristal y el recubrimiento. Si es así escribiremos su nombre (tal y como lo hayamos definido antes) entre comillas. Si no queremos tener en cuenta las interferencias bastará con que escribamos `"none"`.
4. Proporción de fotones que se difunden en vez de reflejarse.
5. Parte real del índice de refracción.
6. Parte imaginaria del índice de refracción.

7. Permeabilidad magnética.

8. Absorción extra. Por defecto, el programa supone 0.0, lo que significa que la absorción de los fotones en el revestimiento se calcula mediante la fórmula de Fresnel con el índice de refracción complejo. Si ponemos un número distinto de cero, por ejemplo 0.5, estamos aumentando la probabilidad de absorción en el 50 %. Permite estudiar cómo el recubrimiento se va deteriorando.

9. Ángulo máximo de difusión de los fotones. Es 90º por defecto.

Tras esto, definiremos la forma de los elementos que van a tomar parte en el montaje:

```
//3.- FORMAS

        TSBRIK *prisma;
        prisma = new TSBRIK("prisma","Cristal de LSO","LSO","reflectortotal",
semilongitud_en_x,semilongitud_en_y,semilongitud_en_z);
```

En este caso los elementos que definen el volumen son:

1. Nombre.

2. Título.

3. Material.

4. Revestimiento.

5. Semilongitudes de los lados (x, y, z).

Muchas veces necesitamos declarar las propiedades de alguna cara en particular como, por ejemplo, rugosidades (`SetDepolished()`), biselados (`SetBevellings()`) o si actúa como detector, pudiendo definir detectores de superficie y fototubos (estos últimos solamente para formas cilíndricas). También es posible establecer propiedades que afecten a toda la forma y no sólo a una cara, como los daños sufridos al radiarla o establecer detectores de volumen (es decir, si el fotón se para dentro de él será detectado), APD's o diodos PIN.

Por ejemplo, establecemos que la cara interior de nuestro cristal perpendicular al eje X y segunda por el orden de llegada de los fotones que se propagan por él de izquierda a derecha, es un detector de superficie. Hemos de tener cuidado con qué número va con cada cara.

```
prisma->fSuppl->SetSurfDet("detector","Detector de superficie",1,"none");
```

Donde:

1. Nombre.

2. Título.

3. Número de la cara que queremos establecer como detector.

4. Eficiencia cuántica: si es total basta con poner `"none"`.

Lo siguiente que debemos hacer es ordenar los elementos del montaje:

```
TSNode *nodo1;
nodo1 = new TSNode("nodo1","Nodo asociado al cristal 1",prisma1);
nodo1->cd();

TSNode *nodo2;
nodo2 = new TSNode("nodo2","Nodo asociado al cristal 2",prisma2,0.5,0.0,0.0 );
```

En este caso hemos definido `prisma1` como centro del montaje, en torno al cual colocaremos los demás cuerpos. Los datos que debemos introducir en este caso son:

1. Nombre.
2. Título.
3. Nombre del volumen que vayamos a colocar.
4. Distancia entre el centro del cuerpo en cuestión y el cuerpo que hemos definido como origen del montaje.

Después de esto tenemos que especificar los elementos del montaje que se encuentran en contacto. Hay que recordar que no es necesario especificar qué caras son las que están en contacto, puesto que la posición indicada por los nodos da la información suficiente como para que el programa pueda deducir qué caras están implicadas. Para ello tiene que haber, obviamente, más de un volumen en juego. Para definir los contactos empleamos un texto como el siguiente:

```
//5.- CONTACTOS

    TContact *contacto;
    contacto = new TContact("entre_cristales","Contacto entre cristales",
"prisma1", "prisma2",identical,"none");
```

El penúltimo argumento indica qué tipo de contacto es: identical si las dos caras son de iguales dimensiones, overlapping si están a distintas alturas y solamente se produce un solapamiento parcial, containing si la primera cara es mayor que la segunda o contained si la primera es menor. El último parámetro nos dice si existe una capa fina de material (a especificar cuál) entre las 2 caras en contacto. Por defecto no hay. Es importante tener en cuenta que:

1. ThinSlice no utiliza las ecuaciones de Fresnel, sino que tiene en cuenta las interferencias.
2. Es mejor, si tenemos 3 elementos en contacto, poner como nodo de referencia el elemento central.
3. Al poner las caras rugosas, si hay recubrimiento, hay que añadir una capa fina de aire entre el material y éste.
4. Al poner los cristales en contacto, si tienen revestimiento, éste se elimina en las caras en contacto.

El siguiente paso es la introducción de una o varias fuentes de luz. Ha de hacerse siempre después de establecer **todos** los nodos. Para ello, escribimos:

```
//6.- FUENTES

    TSpontan *fuente;
```

```
    fuente = new TSpontan("fuente","fuente","prisma2",-0.2499,0.0,0.0,600.0);

    for (Int_t i=1;i<=100;i++) {

     fuente->Gen(i,100,-2.0,true,false);
   }
//Generación de 100 fotones en 100 pasos (i.e., de 10000 fotones)
```

En este caso hemos definido una fuente de luz puntual y espontánea, en el centro del cristal, aunque también podría haber sido un láser, para lo que se definiría un prisma de aire con una "semilla"de generación de fotones.

El tercer argumento está relacionado con el parámetro x usado como abcisa al dibujar los resultados por la clase TPublication. Por defecto, o si se da un valor menor que $-1{,}0 \cdot 10^{20}$, todas las gráficas de TPublication tendrán el número de la etapa o paso como coordenada x. Los argumentos cuarto y quinto están relacionados con si se muestran o no los histogramas y estadísticas de cada paso, i.e., las asociadas al pointer 'gGp'. Las globales sí que se muestran y no están relacionadas con estos argumentos.

Por último, podemos especificar los resultados que queremos ver impresos en la pantalla al finalizar el programa.

```
cout << "fNbPart:    " << gGs->fNpGener    << endl;//Nº total de fotones generados.
cout << "fNpSeen:    " << gGs->fNpSeen    << endl;//Nº de fotones detectados.
cout << "fNpLossAny: " << gGs->fNpLossAny << endl;//Nº de fotones perdidos por alguna razón.
cout << "fNpAbsMat:  " << gGs->fNpAbsMat  << endl;//Nº de fotones absorbidos en los materiales.
cout << "fNpAbsRvt:  " << gGs->fNpAbsRvt  << endl;//Nº de fotones perdidos en el revesitmiento.
cout << "fNpOutSide: " << gGs->fNpOutSide << endl;//Nº de fotones que abandonan el montaje.
cout << "fNpAbnorm:  " << gGs->fNpAbnorm  << endl;//Nº de fotones que son destruidos antes de generarse.
cout << "fNpLossQE:  " << gGs->fNpLossQE  << endl;//Nº de fotones perdidos porque el detector no es  perfecto

  exit();

  }
```

Con esto tendríamos escrito un programa básico de LITRANI. Para ponerlo en marcha basta con que nos dirijamos (en cualquier distribución de GNU/Linux en la que tengamos instalado el paquete LITRANI) al terminal y, desde la carpeta en la que tengamos guardado el programa, abrir LITRANI y llamarlo.

2.5. Interpolaciones con SplineFit

Aunque no hemos hecho gran uso de SplineFit a lo largo del trabajo (solamente en el último capítulo), es importante tener alguna noción de su uso, puesto que en algunos casos es imprescindible utilizar estas interpolaciones para hacer que LITRANI funcione. Por ello explicaremos el ejemplo que se utiliza en el capítulo de simulación de la llegada de fotones gamma al interior del centellador. En primer lugar, llamamos al programa y le asignamos un nombre a la interpolación:

```
TSplineFit* nai_cross_section(Bool_t todraw = kFALSE, Bool_t infile = kFALSE,
Bool_t firstinfile = kFALSE)
```

1. todraw: kTRUE si queremos dibujar la interpolación.
2. infile: kTRUE si la interpolación va a crear un archivo SplineFitDB.rdb.

```
[adri@nuc9 programas]$ q
bash: q: command not found
[adri@nuc9 programas]$ litrani gammas04.C
  ******************************************
  *                                        *
  *        W E L C O M E  to  R O O T      *
  *                                        *
  *   Version   5.19/02     11 March 2008  *
  *                                        *
  *  You are welcome to visit our Web site *
  *          http://root.cern.ch           *
  *                                        *
  ******************************************

ROOT 5.19/02 (trunk@22600, Mar 12 2008, 09:24:00 on linuxx8664gcc)

CINT/ROOT C/C++ Interpreter version 5.16.29, Jan 08, 2008
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0]
Processing gammas04.C...

size     : 5
name     : gammas04
listing  : Paso de fotones gamma por un centellador cilíndrico
upcom    : La última cara es un detector de superficie
downcom  : ¿Qué detecto?
otherseq : 0
indebug  : 0
WithDate : 1

Start from Litrani, libLitrani is already loaded
calling TLitGlob

  ****************************************
  *                                      *
  *     W E L CO M E to L I T R A N I    *
  *                                      *
  *           Version 3_9                *
  *                                      *
  ****************************************

(int)0
Run : 101
```

Figura 2.1: Ejemplo de cómo iniciar un programa en LITRANI en un entorno Linux.

3. firstinfile: kTRUE si queremos borrar todas las interpolaciones presentes en el archivo SplineFitDB.rdb y sustuirlos por la interpolación presente.

Acto seguido hay que crear una tabla para los datos que vamos a introducir y determinar los parámetros de la interpolación:

```
Int_t k1;
  Int_t k2 = -100;
  k1 = TClassTable::GetID("TSplineFit");
  if (k1<0) k2 = gSystem.Load("libSplineFit");
  const Int_t M = 74;
  Int_t i;
```

Posteriormente, llamamos a los datos que vamos a interpolar:

```
TSplineFit *nai_cross_section;
  Double_t x[M] = { 0.001,  0.001017,  0.001035,  0.001053 };

  Double_t y[M] = { 602625.1, 581738.2, 560851.3, 540738.0, 521398.3 };
  nai_cross_section = new TSplineFit("PhotoEl_nai", "Photo-Electric Cross Section | nai",
 18, M, x, y, 0.001,  1.4);
```

Por último solamente necesitamos añadir algunos comentarios, asignar nombres a cada eje y cerrar el programa:

```
  nai_cross_section->SetSource("http://physics.nist.gov/PhysRefData/Xcom/Text/XCOM.html");
  nai_cross_section->SetMacro("nai_cross_section.C");
  nai_cross_section->SetXLabel("Gamma Energy [MeV]");
  nai_cross_section->SetVLabel("Cross Section x10-24 cm2");
  return nai_cross_section;
}
```

Para poder utilizar estas interpolaciones en otros programas de LITRANI debemos abrirlas antes de correr éstos o bien guardarlas en la carpeta Macros del programa, pues si no LITRANI no es capaz de llamarlas por sí mismo.

2.6. Representación de resultados con VisuLitrani y TwoPadDisplay

En todo lo que hemos hecho hasta ahora no hemos empleado VisuLitrani, puesto que es posible extraer resultados del programa sin necesidad de ponerlo en marcha. De hecho, esto es lo que hemos hecho a lo largo de gran parte del trabajo pero, sin lugar a dudas, VisuLitrani puede ser muy útil, como se pone de manifiesto en el último capítulo del trabajo. Emplearlo no es complicado: basta con añadir algunos comandos extra al programa que contiene el montaje.

En primer lugar, para que VisuLitrani se ponga en marcha hay que dibujar el montaje desde LITRANI. Para ello debemos añadir algunos comandos adicionales a la hora de definir los nodos:

```
TSNode *node1;
 node1=new TSNode("node1","node1",centellador);
 node1->SetLineColor(1);
 node1->SetLineWidth(2);
 node1->cd();
```

A partir de esto necesitaremos utilizar las librerías asociadas a la clase TResults para llamar a los diversos histogramas que podemos reproducir e incluso a punteros concretos en el interior de éstos:

- Histogramas de detector: Según el tipo de detectores que tengamos en cada montaje podemos llamar a diversas clases que permitan buscar los histogramas más adecuados para cada detector:

  1. **TStatSurfD:** Para detectores de superficie y fototubos.
  2. **TStatVolD:** Para detectores de volumen.
  3. **TStatAPD:** Para APD's.

  Para utilizar estas estadísticas tenemos que llamar a los detectores que necesitemos (es muy importante haberlos numerado previamente si hay más de uno) mediante los comandos GetSurfDet(), GetVolDet() y GetAPD().

La óptica que utiliza LITRANI

En esta sección vamos a hacer un repaso de los principales conceptos de Óptica que van a intervenir en los casos que vamos a estudiar y que, por lo tanto, han de estar englobadas en LITRANI. Es importante remarcar que la óptica que abarca el programa de simulación que nos ocupa es bastante más amplia que la que aquí se va a exponer, pues abarca a los medios anisótropos, de los cuales nosotros no vamos a hacer uso.

3.1. Tratamiento electromagnético de la luz

Si se desea más información sobre este apartado, puede consultarse la referencia [4]. Si se trata a la luz incidente sobre una interfase como una onda electromagnética de la forma $\vec{E}_i = \vec{E}_{0i} exp\left[i\left(\vec{k}_i \cdot \vec{r}\right) - \omega_i t\right]$ podemos llegar a expresiones más complejas que las dadas por la Óptica Geométrica, tales como la Ley de Snell, además de confirmar los resultados por ella dados. El tratamiento electromagnético de la luz ha de comenzar con la exposición de las condiciones de frontera de los campos eléctricos y magnéticos en una interfase conocidas del Electromagnetismo, a saber:

- La componente tangencial del campo eléctrico se mantiene constante al pasar de un medio de constante dieléctrica ϵ_1 a uno con ϵ_2:

$$\hat{n}_{12} \times \left(\vec{E}_2 - \vec{E}_1\right) = 0$$

- En presencia de una densidad de carga libre ρ en la superficie, l.a componente normal del vector de desplazamiento eléctrico cambia según

$$\hat{n}_{12} \cdot \left(\vec{D}_2 - \vec{D}_1\right) = 4\pi\rho$$

- La componente normal del campo magnético se mantiene constante al pasar de un medio con permeabilidad magnética μ_1 a uno con permeabilidad μ_2:

$$\hat{n}_{12} \cdot \left(\vec{B}_2 - \vec{B}_1\right) = 0$$

- En presencia de una densidad de corriente $\vec{j}$ en la superficie, la componente tangencial del vector magnético cambia según la fórmula:

$$\hat{n}_{12} \times \left(\vec{H}_2 - \vec{H}_1\right) = \frac{4\pi}{c}\vec{j}$$

Con estos conocimientos, y puesto que vamos a empezar estudiando la propagación de la luz en medios dieléctricos en los que no hay ni densidad de carga ni densidad de corriente eléctrica, podemos escribir las ondas reflejada y transmitida como

$$\vec{E}_r = \vec{E}_{0r} \cos\left[i\left(\vec{k}_r \cdot \vec{r}\right) - \omega_r t + \epsilon_r\right]$$

$$\vec{E}_t = \vec{E}_{0t} \cos\left[i\left(\vec{k}_t \cdot \vec{r}\right) - \omega_t t + \epsilon_t\right]$$

donde ϵ_r y ϵ_t son las constantes de fase relativas a $\vec{E}_i$.

3.2. Deducción de las Ecuaciones de Fresnel

Para un desarrollo más detallado de las Ecuaciones de Fresnel véase [3]. Con las Ecuaciones de Fresnel vamos a encontrar una dependencia entre las amplidutes de las ondas incidente, reflejada y transmitida, $\vec{E}_{0i}$, $\vec{E}_{0r}$ y $\vec{E}_{0t}$ a un lado y otro de una interfase que separe dos medios con distinto índice de refracción. Para deducir las Ecuaciones de Fresnel nos planteamos dos situaciones básicas.

3.2.1. Caso 1: $\vec{E}$ perpendicular al plano de incidencia

La relación entre las direcciones de los campos electromagnéticos y el vector de Poynting (equivalentemente, la propagación $\hat{\vec{k}}$) impondrá que en este caso el campo magnético sea paralelo al plano de incidencia. Como sabemos que $E = vB$, podemos decir que

$$\hat{\vec{k}} \times \vec{E} = vB$$

Haciendo uso de la continuidad de las componentes tangenciales de los campos electromagnéticos en cualquier punto de la interfase tendremos

$$\vec{E}_{0i} + \vec{E}_{0r} = \vec{E}_{0t}$$

$$-\frac{\vec{B}_i}{\mu_i}\cos\theta_i + \frac{\vec{B}_r}{\mu_i}\cos\theta_r = -\frac{\vec{B}_t}{\mu_t}\cos\theta_t$$

Podemos simplificar esto todavía más tomando $B_i = \frac{E_i}{v_i}$, $B_r = \frac{E_r}{v_r}$,$B_r = \frac{E_r}{v_r}$, lo cual nos llevará, empleando la Ley de Snell para la reflexión y teniendo en cuenta que $v_i = v_r$ a

$$\frac{(E_i - E_r)\cos\theta_i}{\mu_i v_i} = \frac{E_t \cos\theta_t}{\mu_t v_t}$$

Esto, finalmente, permite obtener los cocientes entre la amplitud de las ondas reflejadas y transmitidas respecto de la de la incidente.

$$r_\perp \equiv \left(\frac{E_{0r}}{E_{0i}}\right)_\perp = \frac{\frac{n_i}{\mu_i}\cos\theta_i - \frac{n_t}{\mu_t}\cos\theta_t}{\frac{n_i}{\mu_i}\cos\theta_i + \frac{n_t}{\mu_t}\cos\theta_t}$$

$$t_\perp \equiv \left(\frac{E_{0t}}{E_{0i}}\right)_\perp = \frac{2\frac{n_i}{\mu_i}\cos\theta_i}{\frac{n_i}{\mu_i}\cos\theta_i + \frac{n_t}{\mu_t}\cos\theta_t}$$

3.2.2. Caso 2: $\vec{E}$ paralelo al plano de incidencia

El procedimiento es, en este caso, completamente idéntico al del caso anterior: volvamos a aplicar la continuidad en las componentes tangenciales de los campos a las tres componentes, incidente, transmitida y reflejada:

$$E_{0i}\cos\theta_i - E_{0r}\cos\theta_r = E_{0t}\cos\theta_t$$
$$\frac{E_{0i}}{\mu_i v_i} + \frac{E_{0r}}{\mu_r v_r} = \frac{E_{0t}}{\mu_t v_t}$$

De nuevo, como el medio de la onda incidente y el de la reflejada es el mismo, así como la Ley de Snell para la reflexión, obtenemos unas expresiones equivalentes a las anteriores para las componentes paralelas al plano de incidencia de las ondas reflejada y transmitida.

$$r_\parallel \equiv \left(\frac{E_{0r}}{E_{0i}}\right)_\parallel = \frac{\frac{n_t}{\mu_t}\cos\theta_i - \frac{n_i}{\mu_i}\cos\theta_t}{\frac{n_i}{\mu_i}\cos\theta_t + \frac{n_t}{\mu_t}\cos\theta_i}$$
$$t_\parallel \equiv \left(\frac{E_{0t}}{E_{0i}}\right)_\parallel = \frac{2\frac{n_i}{\mu_i}\cos\theta_i}{\frac{n_i}{\mu_i}\cos\theta_t + \frac{n_t}{\mu_t}\cos\theta_i}$$

Con esto podemos decir que hemos encunciado las cuatro ecuaciones de Fresnel, que quedan pendientes de diversas simplificaciones asociadas a la homogeneidad de las permeabilidades magnéticas de los materiales implicados, así como a los casos de incidencia normal, que veremos más adelante al comprobar la bondad de los resultados ofrecidos por LITRANI en comparación con los cálculos teóricos.

3.2.3. Reflectancia y transmitancia

Puesto que en Óptica lo que sabemos medir son intensidades, no amplitudes de campo, debemos trasladar estas expresiones para hacerlas compatibles con la medida de la densidad de flujo radiante $I =< S >= \frac{c\epsilon_0}{2}E_0^2$, donde $\vec{S}$ es el vector de Poynting. Esta expresión da la energía promedio por unidad de tiempo que cruza un área unitaria normal a $\vec{S}$. Por lo tanto, la porción de energía que incide normalmente sobre dicha área por segundo será $I_i\cos\theta_i$, añ igual que las energías reflejada y transmitida. Definimos la *reflectancia* R como la razón entre la energía reflejada y la incidente:

$$R \equiv \frac{I_r\cos\theta_r}{I_i\cos\theta_i} = \frac{I_r}{I_i} = \left(\frac{E_{0r}}{E_{0i}}\right)^2 = r^2$$

De igual modo, la *transmitancia* T es el cociente entre el flujo de energía transmitida y la incidente:

$$T \equiv \frac{I_t\cos\theta_t}{I_i\cos\theta_i} = \frac{\mu_i n_t\cos\theta_t}{\mu_t n_i\cos\theta_i}\left(\frac{E_{0t}}{E_{0i}}\right)^2 = \frac{n_t\cos\theta_t}{n_i\cos\theta_i}t^2$$

3.3. Absorción por parte de un dieléctrico isótropo

La absorción en un medio no conductor puede describirse mediante la adición de una parte imaginaria a la constante dieléctrica (o al tensor dieléctrico en caso de tener un medio anisótropo, cosa que no vamos a tratar): $\epsilon_C = \epsilon - i\phi$. Sabemos que $n_C^2 = \epsilon_c\mu$, de modo que, tomando partes imaginarias, $(n_R - in_I)^2 = \mu(\epsilon - i\phi)$. Esto lleva, por lo tanto, a

$$\epsilon = \frac{n_R^2 - n_I^2}{\mu}; \phi = \frac{2n_R n_R}{\mu}$$

$$n_R = \sqrt{\frac{\mu}{2}\left(\epsilon + \sqrt{\epsilon^2 + \phi^2}\right)}$$

$$n_I = \frac{\mu\phi}{\sqrt{2\mu\left(\epsilon + \sqrt{\epsilon + \phi}\right)}}$$

La aparición de una componente imaginaria en el índice de refracción se pone de manifiesto en nuestra forma de expresar la onda plana:

$$\vec{E} = \vec{E}_0 e^{i\left(\omega t - \vec{k}\cdot\vec{x}\right)} = \vec{E}_0 e^{i\left[\omega t - \frac{\omega n_R}{c}\vec{\kappa}\cdot\vec{x}\right] - \frac{\omega n_I}{c}\vec{\kappa}\cdot\vec{x}}$$

de modo que, para la intensidad de una onda, el factor de absorción es $e^{-\frac{2\omega n_I}{c}\vec{\kappa}\cdot\vec{x}}$.

Esto permite definir la *longitud de absorción* como $L_a = \frac{c}{2\omega n_I} = \frac{\lambda}{4\pi n_I}$, de modo que para una onda plana, la intensidad decrecerá a medida que penetre en el material según

$$I(x) = I_0 e^{x/L_a}$$

3.4. Propiedades ópticas de los metales

Puede encontrarse un estudio mucho más detallado de las propiedades ópticas de los metales en [4]. Puesto que los cristales en cuyo seno vamos a estudiar el movimiento de los fotones van a poseer diversos recubrimientos metálicos a fin de que los fotones no abandonen el sistema, es conveniente tener alguna idea sobre cómo se comportarán dichos fotones al incidir sobre materiales conductores, puesto que su comportamiento ante la luz es muy especial debido a posible interacción de las cargas libres que contienen con los campos electromagnéticos asociados a la luz. Por lo tanto, hemos de tener en cuenta que se puede producir absorción de la energía radiante por un material en función de su conductividad σ.

3.4.1. Ondas en un metal

Suponiendo que el medio es continuo las Ecuaciones de Maxwell dicen que

$$\nabla^2\vec{E} = \mu\epsilon\frac{\partial^2\vec{E}}{\partial t^2} + \mu\sigma\frac{\partial\vec{E}}{\partial t}$$

lo cual no es más que la ecuación de ondas con un factor de amortiguamiento asociado a la derivada de primer órden en el tiempo. Esto implica directamente la aparición de la absorción en el índice de refracción, lo cual se pone de manifiesto en la adición de una parte imaginaria a éste: $n_C = n_R + in_I$. Esto nos lleva a una solución amortiguada de la ecuación de ondas tal y como la hemos venido utilizando:

$$\vec{E} = \vec{E}_0 e^{(-\omega n_I y/c)} e^{i\omega(t - n_R y/c)}$$

o, equivalentemente,

$$\vec{E} = \vec{E}_0 e^{(-\omega n_I y/c)} \cos\omega\left(t - n_R y/c\right)$$

Obviamente, este amorgiguamiento se extiende también a la irradiancia que, debido a que depende cuadráticamente del campo eléctrico, adquirirá la forma

$$I(y) = I_0 e^{-\alpha y}$$

donde $\alpha \equiv 2\omega n_I/c$.

3.4.2. Ecuación de dispersión

En un metal, la gran parte de las propiedades ópticas están determinadas por los electrones de conducción. Si consideramos que los electrones están unidos elásticamente a los núcleaos, su ecuación de movimiento será $x(t) = \frac{q_e}{m_e(\omega_0^2-\omega^2)}E(t)$. Los electrones libres oscilando fuera de fase con la luz incidente irradiarán nuevas ondas que también se verán amortiguadas por las propiedades del metal. Haciendo diversas suposiciones y simplificaciones que desprecian la contribución de los electrones ligados, podemos obtener una ecuación de dispersión sencilla para los metales:

$$n(\omega) = 1 - \frac{Nq_e^2}{\epsilon_0 m_e \omega^2} = 1 - \left(\frac{\omega_p}{\omega}\right)^2$$

Los electrones libres y los iones positivos dentro de un metal se pueden imaginar como un plasma cuya densidad oscila a una frecuencia natural ω_p llamada *frecuencia de plasma*, que nos sirve como un valor crítico por debajo del cual el índice es complejo y la onda transmitida decae exponencialmente, mientras que para frecuencias superiores a ella la absorción es pequeña y el conductor se hace prácticamente transparente. Lo más frecuente es que el índice de refracción de un metal sea complejo y que la onda incidente sufra absorción en una cantidad dependiente de su frecuencia.

3.4.3. Reflexión en un metal

Empleando $n_C^2 = \mu\epsilon$ podemos llegar a dos ecuaciones que relacionen las componentes del índice de refracción con la permeabilidad magnética, la constante dieléctrica y la conductividad del metal:

$$n^2 = \frac{1}{2}\left(\sqrt{\mu^2\epsilon^2 + \frac{4\mu^2\sigma^2}{\nu^2}} + \mu\epsilon\right)$$

$$n^2\kappa^2 = \frac{1}{2}\left(\sqrt{\mu^2\epsilon^2 + \frac{4\mu^2\sigma^2}{\nu^2}} - \mu\epsilon\right)$$

Aunque las formas funcionales de la parte real y compleja del índice de refracción son complicadas, se puede estudiar la reflexión por parte de metales retomando los resultados que obtuvimos para la reflectividad en dieléctricos pero teniendo en cuenta la presencia de una parte imaginaria en el índice de refracción. Por lo tanto, las fórmulas de la reflectividad de las componentes polarizadas paralela y perpendicularmente al eje óptico se reducirán a las que se obtuvieron para los dieléctricos, pero teniendo en cuenta que, por ser uno de los índices implicados complejo, elevar los coeficientes de reflexión al cuadrado implcia, en realidad, obtener su módulo tal que $R_{\|} = r_{\|}r_{\|}^*$, lo cual modifica la forma final de la reflectividad. El ejemplo más sencillo (y que después utilizaremos) es el de la reflexión normal desde un medio con índice de refracción y permeabilidad magnética unidad, que toma la forma de

$$R = \frac{(n_R - \mu_t)^2 + n_I}{(n_R + \mu_t)^2 + n_I}$$

Validación de problemas sencillos con LITRANI

Antes de emplear LITRANI para ejecutar simulaciones en el interior de los cristales centelladores de un aparato de PET, es necesario comprobar que sus resultados concuerdan con los que conocemos de Óptica y que, por lo tanto, son útiles. Para ello, plantearemos una serie de problemas sencillos, de modo que podamos comparar el resultado obtenido por LITRANI y el que nos dé la aplicación de los conocimientos expuestos en la sección anterior. Todos los programas aquí comentados se pueden consultar en el anexo adjunto al final del trabajo.

4.1. Ejemplo 1: comprobación de la equivalencia de montajes

En primer lugar, vamos a comprobar si dos montajes que son físicamente equivalentes lo son también para LITRANI. Por ejemplo, tomemos un cristal de unas determinadas dimensiones con una de sus caras definida como un detector de superficie y un revestimiento totalmente reflector. Introducimos una fuente espontánea de fotones en un punto concreto de su interior y contamos la cantidad de fotones vistos.

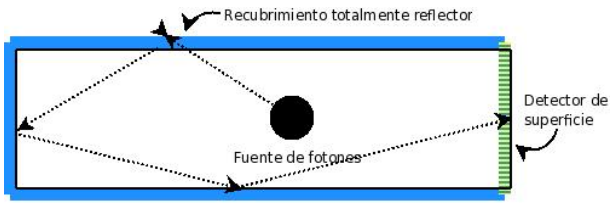

Figura 4.1: Montaje con un solo cristal

Si hacemos exactamente el mismo experimento pero con dos cristales pegados perfectamente (esto es, sin ningún tipo de material entre ellos), con la fuente y el detector en el mismo lugar, los resultados deberían ser los mismos.

En una primera aproximación al problema, corrimos los dos programas con total normalidad, obteniendo los resultados que se muestran en la siguiente tabla.

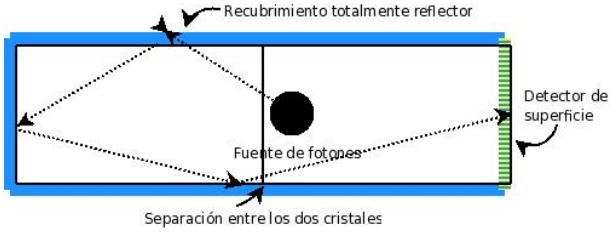

Figura 4.2: Montaje con dos cristales consecutivos

	Un solo cristal	Dos cristales	Fluctuación aproximada
Fotones emitidos	10000	10000	-
Fotones vistos	8376	8396	91
Fotones perdidos en el material	1624	1604	40

Se aprecia que, si bien los resultados son bastante parecidos, hay una diferencia de 20 fotones. Podríamos suponer que, dado que la fluctuación estadística asociada a la generación de números aleatorios es del orden de $\sqrt{N}$, una diferencia de 20 fotones puede englobarse en dicha fluctuación. De todas maneras, hemos querido comprobar que esto es realmente así, para lo cual hemos tenido que estudiar la modificación de los resultados para los mismos programas modificando la semilla generadora de números aleatorios. Esto puede hacerse en LITRANI activando "otherseq" en el la llamada que arranca el programa (InitLitrani) y llamando al comando "SetSeed()". De esta manera, y con la ayuda de un programa *.sh, podemos automatizar la implementación de ambos programas utilizando diferentes semillas comprendidas entre 0 y 10000 con los siguientes resultados.

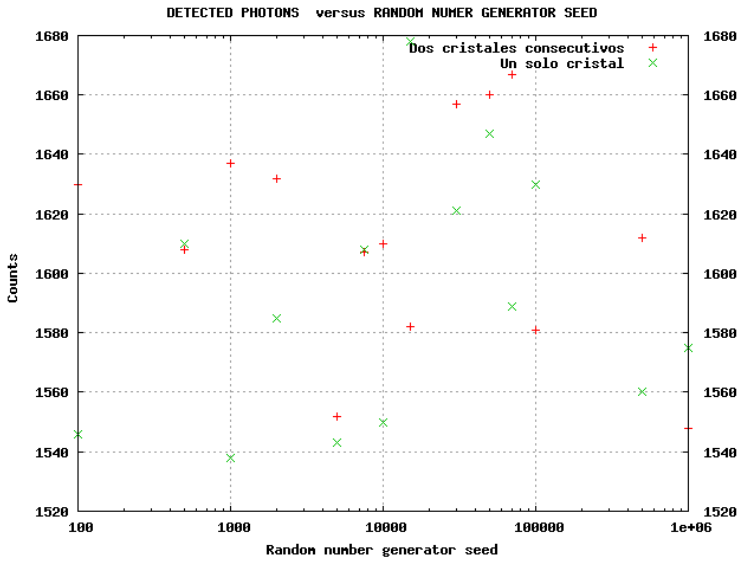

Figura 4.3: Comparación de los resultados para ambos montajes con diferentes semillas en la generación de números aleatorios.

Esta gráfica pone de manifiesto que, si bien no suelen obtenerse resultados idénticos para ambos montajes, la diferencia entre estos nunca supera el orden de magnitud de la fluctuación estadística esperada, por lo que se pueden considerar dichas diferencias como flucuaciones asociadas a la generación de los números aleatorios, por lo que podemos aceptar como equivalentes, dentro del error estadístico, los resultados de LITRANI para los dos montajes. Saber esto será importante a la hora de analizar nuestros resultados en el futuro.

4.2. Ejemplo 2: Reflectancia y transmitancia

En segundo lugar, vamos a estudiar los resultados que LITRANI da para un problema muy simple de haces reflejados y transmitidos en un dieléctrico. Para ello, ideamos un par de montajes en LITRANI. El primero de ellos consiste en un haz láser de longitud de onda 600 nm que incide desde el aire sobre un cristal de índice de refracción real 1,5, sin componente compleja, cuya última cara es una superficie detectora. La longitud de absorción de dicho cristal se ha establecido lo suficientemente grande como para que pueda ser despreciable.

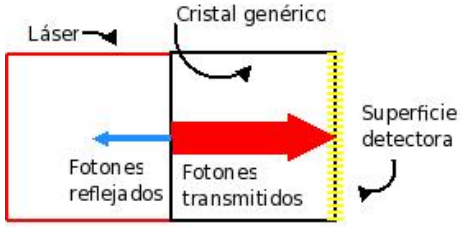

Figura 4.4: Montaje para la detección de los fotones transmitidos

Puesto que estamos suponiendo incidencia normal, la reflectancia y la transmitancia se calculan mediante

$$R = \left(\frac{n_t - n_i}{n_t + n_i}\right)^2 ; T = \frac{4 n_t n_i}{(n_t + n_i)^2}$$

Con esto podemos comparar los resultados de la simulación en LITRANI con los dados por las fórmulas de Fresnel.

	LITRANI	Resultado teórico
Fotones emitidos	10000	10000
Fotones transmitidos	9600	9600
Fotones reflejados	400	400

Hemos supuesto que los fotones reflejados son los que LITRANI considera que se escapan del montaje. De este modo, el resultado ofrecido por LITRANI es exactamente igual al esperado.

Una forma alternativa de este mismo montaje permitiría ver los fotones reflejados en lugar de los transmitidos. Para ello, hacemos que el detector de superficie sea la primera cara del bloque que contiene al láser (hecho de aire), y añadimos un recubrimiento totalmente absorbente al cristal. De este modo, todos los fotones transmitidos deben ser absorbidos por el recubrimiento, mientras que los reflejados deben ir todos al detector.

Sorprendentemente, este montaje solamente nos permite ver 363 fotones reflejados, frente 9635 absorbidos por el recubrimiento y 2 absorbidos por el cristal. Parece que estamos ante un

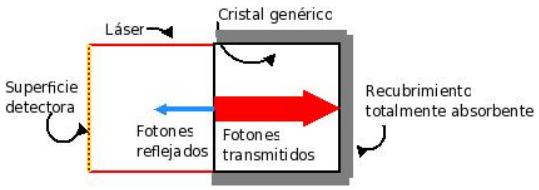

Figura 4.5: Montaje para la detección de los fotones reflejados.

nuevo caso en el que la fluctuación estadística enmascara en número esperado de fotones. Podemos comprobar que esto es así volviendo a llamar al comando `SetSeed` y probando algunos nuevos números para la semilla de la generación de números aleatorios. Obtenemos

Semilla	Por defecto	0	10	100
Vistos (reflejados)	363	390	418	411
Absorbidos en el recubrimiento (transmitidos)	9635	9609	9579	9588
Absorbidos por el material	2	1	3	1

Por lo tanto, comprobamos que la desviación del resultado se debe, una vez más, a las fluctuaciones estadísticas que ya vimos en el caso anterior.

4.3. Ejemplo 3: Estudio de la absorción de un dieléctrico

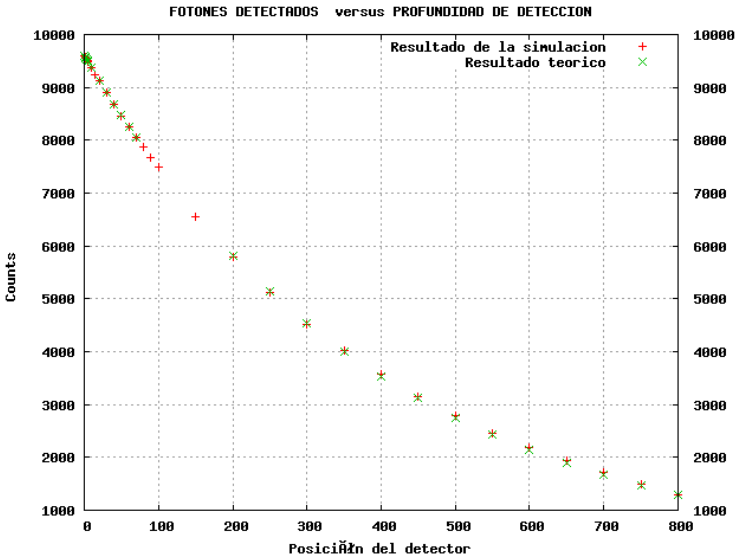

Figura 4.6: Resultados teóricos y simulados para el montaje de absorción.

Nuestro siguiente paso es comprobar los resultados de LITRANI en lo que a la absorción de luz por parte de un material dieléctrico se refiere. Este asunto ha sido tratado en el capítulo anterior, de modo que podremos comprar los resultados simulados por LITRANI con un cálculo

teórico. El montaje ideado es muy similar al primero de los empleados en el ejemplo anterior: un láser de longitud de onda de 600 nm incide normalmente desde un medio tipo aire sobre un cristal de índice de refracción 1,5, similar al del caso anterior. Volveremos a colocar un detector de superficie en la última de sus caras y, para comprobar que la absorción de la intensidad lumínica en función de la profundidad que ésta alcanza en el cristal sigue una exponencial, estableceremos como variable la longitud de éste, de modo que podremos realizar diversas iteraciones del programa de LITRANI para distintas longitudes y, por lo tanto, profundidades de detección. Paralelamente, empleando los resultados vistos en la sección 2.3, podemos hacer un cálculo teórico empleando los mismos datos que en la simulación para, así, comparar los resultados. La figura 4.6 pone de manifiesto que podemos tener en cuenta fenómenos de absorción en nuestras simulaciones con LITRANI con la seguridad de que sus resultados serán correctos.

4.4. Ejemplo 4: Estudio de la variación de la permeabilidad magnética

En un experimento análogo al primero del ejemplo 2, variando la permeabilidad magnética del cristal, mantenemos constante su índice de refracción en 1,5. La fórmula que obtuvimos para la transmitancia en la sección 3.2.3 se simplifica enormemente para casos de incidencia normal, de modo que podemos calcular el porcentaje de fotones que serán transmitidos a medida que variemos la permeabilidad. De nuevo, la concordancia entre los resultados teóricos y los simulados es buena.

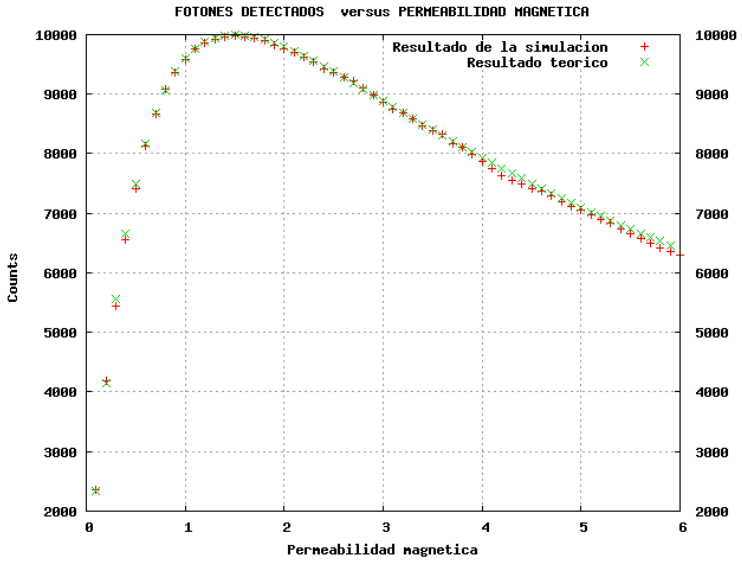

Figura 4.7: Resultados teóricos y simulados para la variación de la permeabilidad del magnética dieléctrico.

4.5. Ejemplo 5: Estudio de la variación del índice de refracción

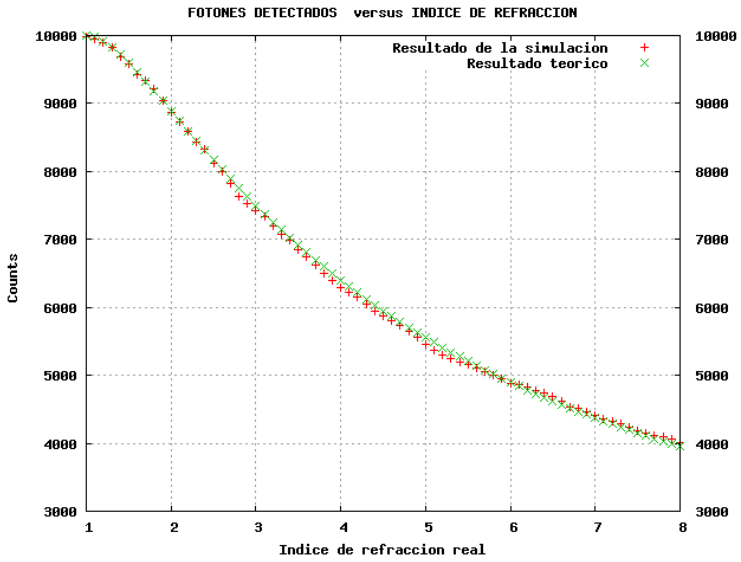

Figura 4.8: Resultados teóricos y simulados para la variación del índice de refracción en un dieléctrico.

Igual que hemos variado la permeabilidad magnética, podemos modificar la parte real del índice de refracción para compararla con los resultados teóricos. Nuevamente, los resultados reflejan que LITRANI funciona como hemos supuesto.

4.6. Ejemplo 6: Reflexión en metales

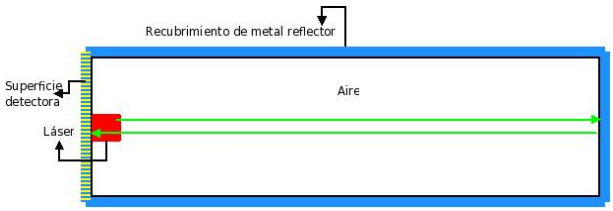

Figura 4.9: Montaje para el estudio de la reflexión de los fotones por parte de un metal.

De acuerdo con la sección 2.4.3, la reflectividad de un metal depende de su índice de refracción y su permeabilidad magnética de manera análoga a como lo hacen los dieléctricos, con la salvedad de que se añade una parte imaginaria al índice de refracción. Es por esto que, para los recubrimientos, LITRANI pide como datos las partes real e imaginaria del índice de refracción, así como la permeabilidad magnética. Para estudiar la reflexión de fotones en incidencia

normal por parte de un recubrimiento metálico en aire hemos implementado en LITRANI el montaje representado en la figura 4.9. En este caso nos ha parecido más conveniente dejar la parte imaginaria del índice de refracción y la permeabilidad magnética como variables libres al mismo tiempo, de modo que podemos representar en tres dimensiones la variación de la cantidad de fotones reflejados en función de ellas, asdemás de comparar este resultado con el teórico visto en el capítulo anterior. En este caso, la concordancia entre los resultados teóricos y los simulados es tan elevada que, al representarlos simultáneamente no se distinguen los unos de los otros. Es por ello que omitiremos esta última gráfica comparativa, puesto que no permite apreciar diferencias entre los dos métodos.

Simulaciones en el interior de centelladores de LSO

5.1. Descripción del montaje

Después de aprender a utilizar LITRANI y de comprobar que su funcionamiento se ajusta a los resultados teóricos esperados, podemos empezar a simular los centelladores de LSO que se emplean en PET. Para ello, en primer lugar describiremos el montaje que hemos implementado en LITRANI. Disponemos de dos tipos de cristales diferentes: unos de 2cmx1cmx1cm y otros de 7mmx1,4mmx1,4mm, ambos de LSO. Nuestro plan es estudiar cómo varía la cantidad de fotones detectados en la última cara de uno de éstos cristales en función de la posición a lo largo del eje x en la que situemos una fuente isótropa de fotones de 600 y 400 nm en función del recubrimiento que rodee al montaje o de si, en vez de un solo cristal de 4 cm (14 mm) de largo tenemos dos cristales de 2 cm (7 mm) separados por una capa de aire o bien por un gel óptico.

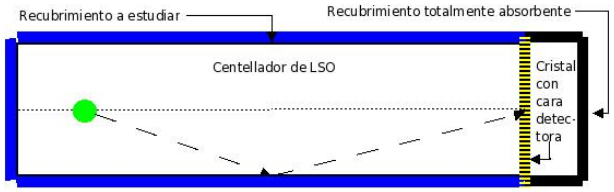

Figura 5.1: Montaje con un solo cristal.

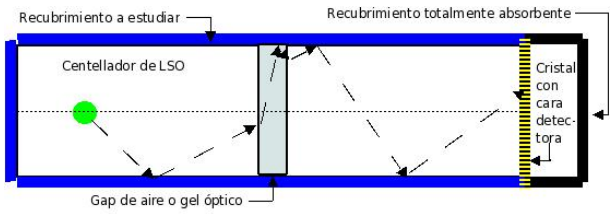

Figura 5.2: Montaje para dos cristales separados por una capa de aire o gel.

Posteriormente introduciremos modificaciones sobre este montaje para obtener resultados más complejos, pero por ahora constituye un buen punto de inicio para las simulaciones por su sencillez y por la facilidad que presenta a la hora de añadir nuevos elementos.

5.2. Descripción de los materiales

Presentaremos en este punto todos los materiales de los que vamos a hacer uso, así como de los datos necesarios para utilizarlos en LITRANI. Cabe remarcar que los datos que LITRANI requiere para simular un material poco tienen que ver con las características que se dan generalmente tanto para los centelladores como para los recubrimientos, de modo que no está claro que LITRANI sea capaz de simular satisfactoriamente las propiedades de dichos materiales. Todos los datos se presentan para una longitud de onda de 600 nm.

5.2.1. Cristales centelladores

Cristal	n_R	μ_r	L_a (nm)
LSO	1.82	1	21

5.2.2. Recubrimientos

Puesto que no es sencillo encontrar datos tabulados de los índices de refracción y permeabilidad magnética para la mayoría de los recubrimientos comerciales, hemos recurrido a elementos más simples, cuyas propiedades pueden encontrarse en [5]. Además de estos recubrimientos, hemos realizado también simulaciones haciendo rugosas las caras externas del los cristales centelladores, cosa posible en LITRANI, así como con un recubrimiento genérico (similar al teflón) con una difusividad del 70 %, para comparar el efecto de las caras rugosas y el de la activación de la difusividad en un recubrimiento.

Los códigos empleados para definir todos estos recubrimientos se pueden consultar en el apéndice correspondiente al presente capítulo, así como una muestra reducida de los programas escritos a tal fin.

Recubrimiento	n_R	n_I	μ_r
Totalmente absorbente	0.5	0.5	1.0
Reflector total	0.0	-1.0	1.0
Aluminio	1.30	7.48	1.56
Teflón	1.32	0.00	0.83
Cobre	0.27	3.24	0.18
Oro	0.13	3.16	0.22
Níquel	1.92	3.65	1.36
Wolframio	3.60	2.89	1.25

5.3. Resultados de las simulaciones con desplazamiento de la posición longitudinal de la fuente

Las primeras simulaciones que hemos llevado a cabo han tenido como fin estudiar el comportamiento de los fotones ante diversos tipos de recubrimiento, tanto a lo largo de un solo cristal como en dos cristales separados por sendas capas de aire o gel. Presentaremos los resultados para cada grupo de cristales, en primer lugar comparando los resultados para las tres situaciones (un solo cristal o dos separados por gel o aire) en cada recubrimiento y, posteriormente, una comparativa de todos los materiales empleados.

5.3.1. Fotones de 600 nm en cristales de 2cmx1cmx1cm

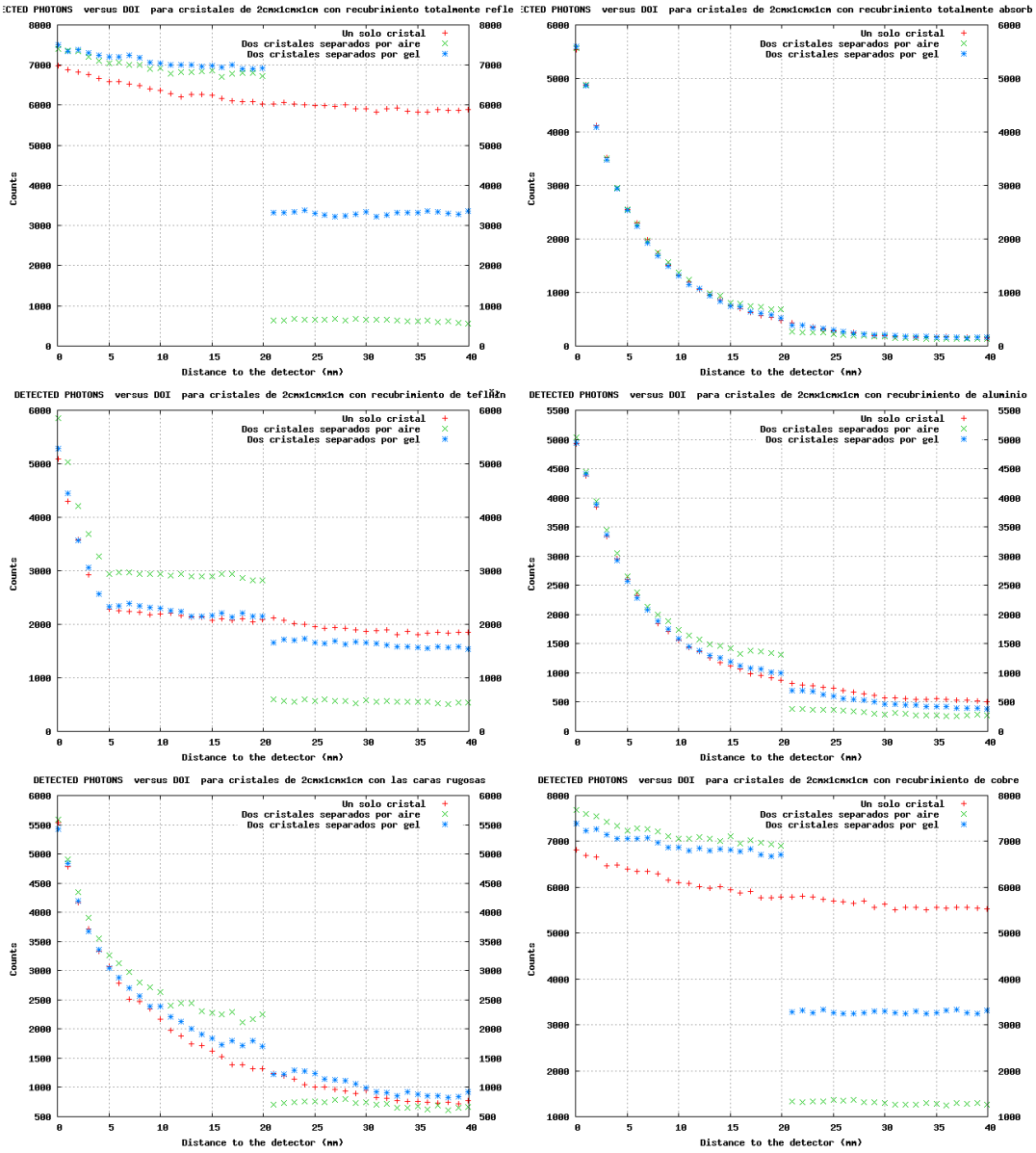

Figura 5.3: Simulaciones de un solo recubrimiento sobre cristales 2cmx1cmx1cm.

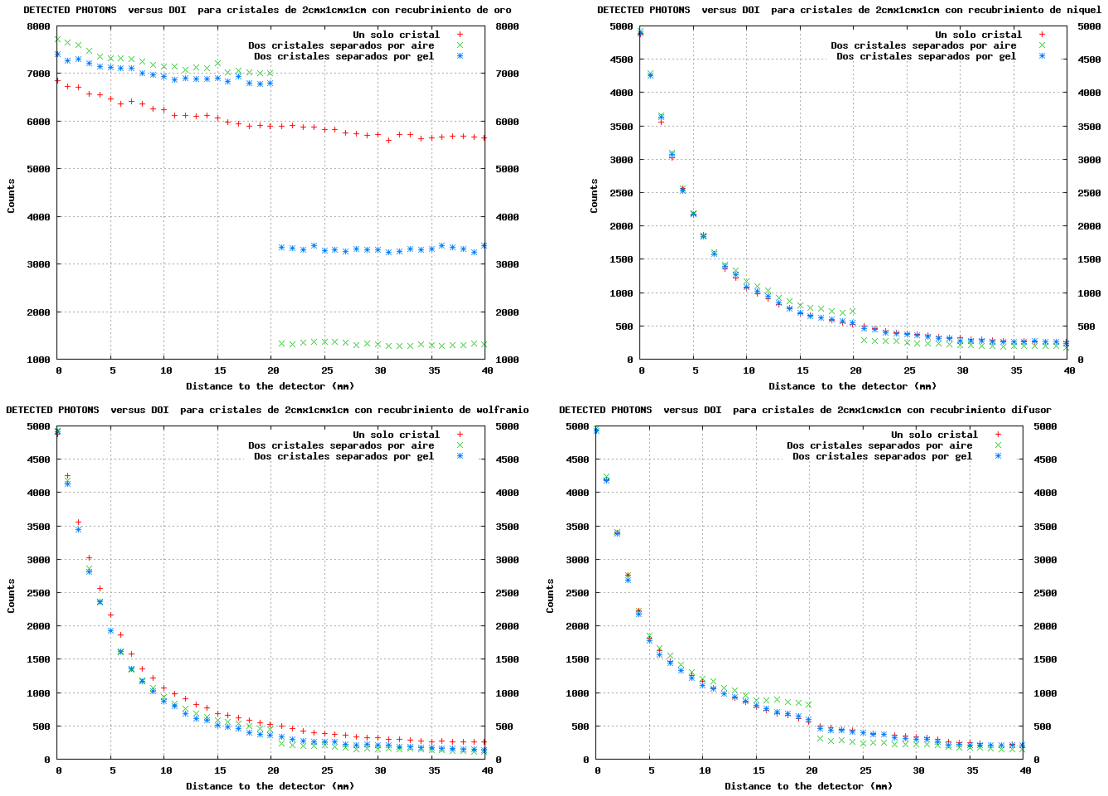

Figura 5.4: Simulaciones de un solo recubrimiento sobre cristales 2cmx1cmx1cm.

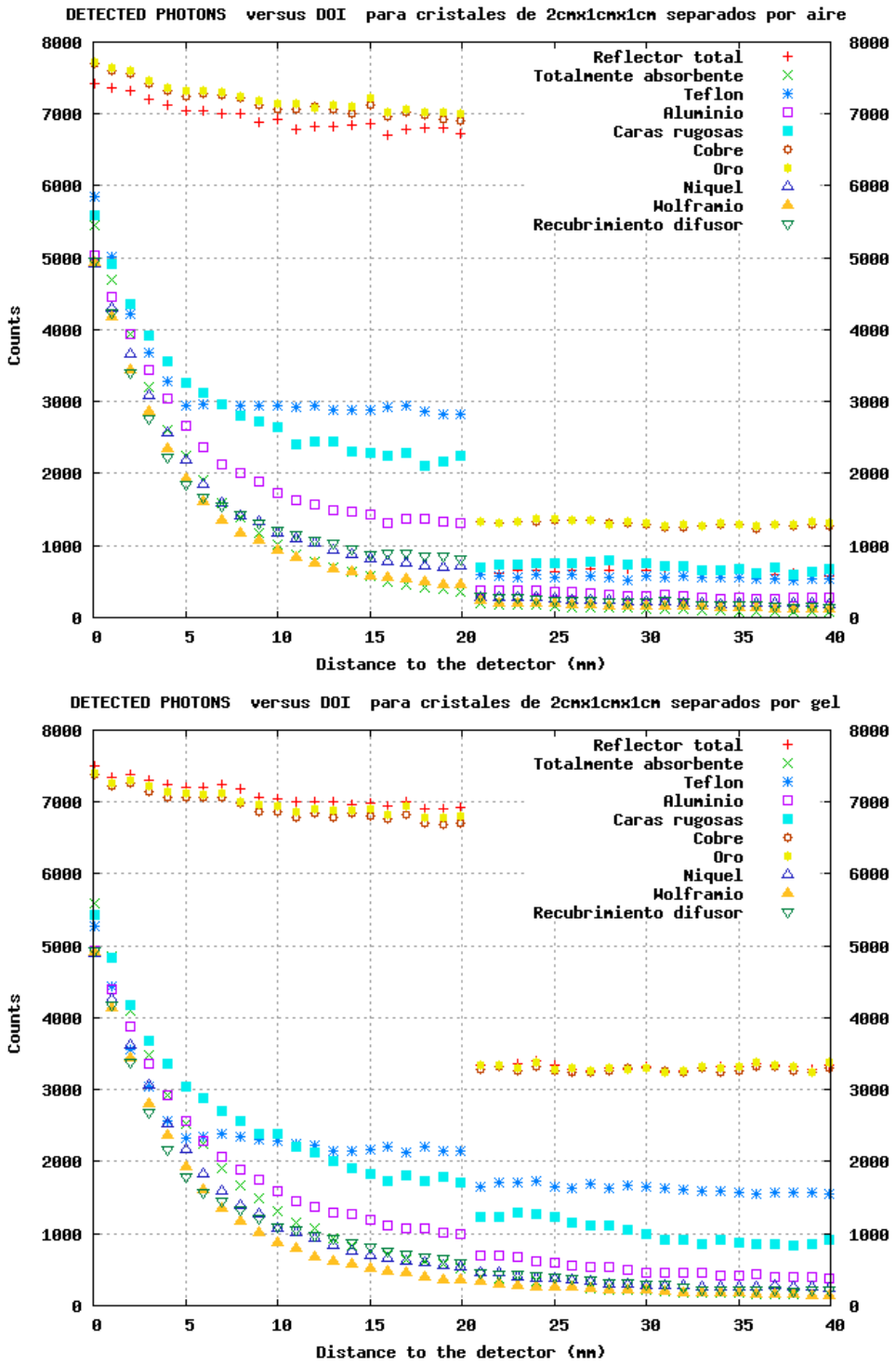

Figura 5.5: Comparación entre las simulaciones para distintos recubrimientos sobre cristales 2cmx1cmx1cm.

5.3.2. Fotones de 600 nm en cristales de 7mmx1,4mmx1,4mm

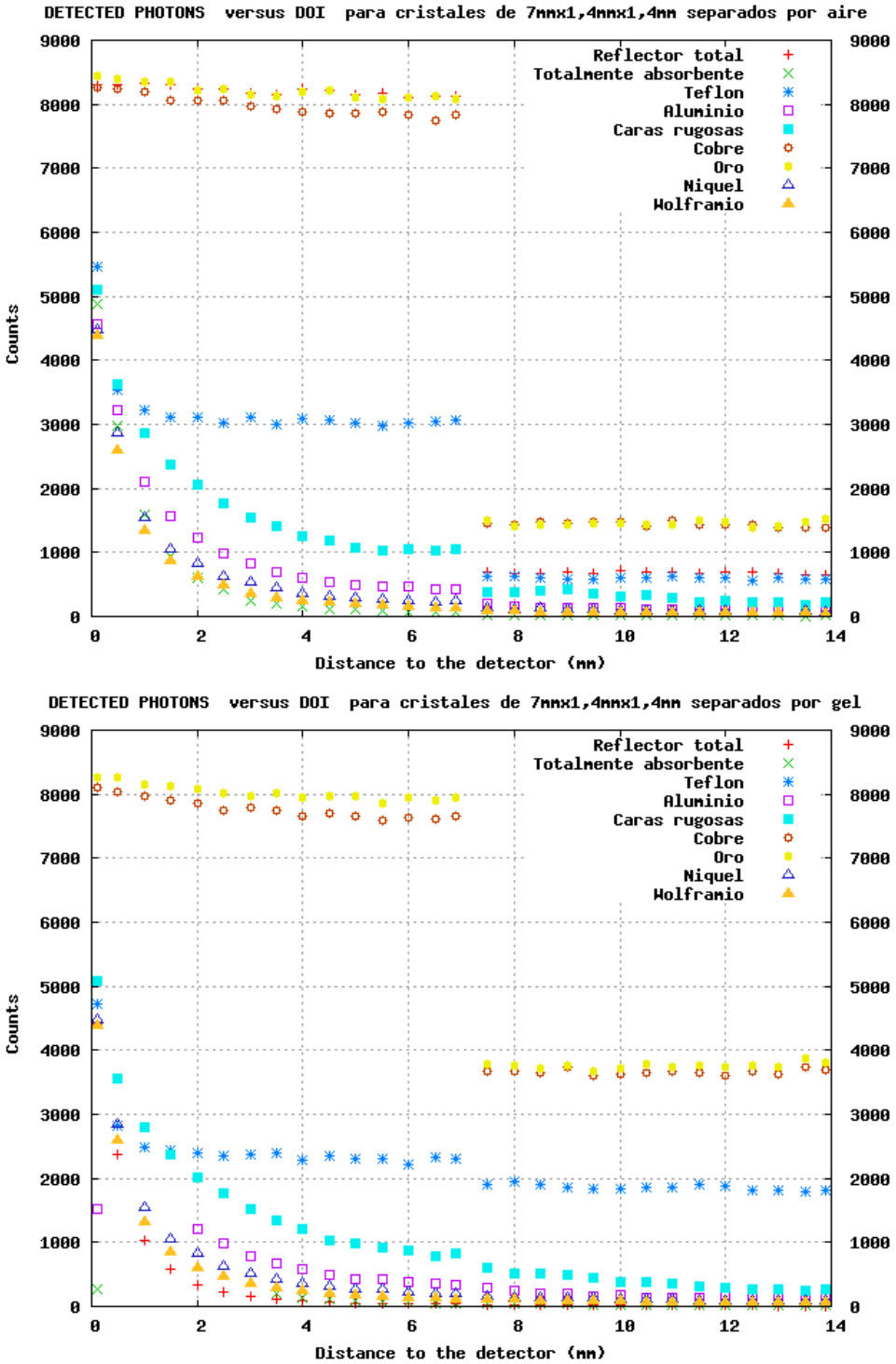

Figura 5.6: Comparación entre las simulaciones para distintos recubrimientos sobre cristales 7mmx1,4mmx1,4mm.

5.3.3. Fotones de 400 nm en cristales de 2cmx1cmx1cm

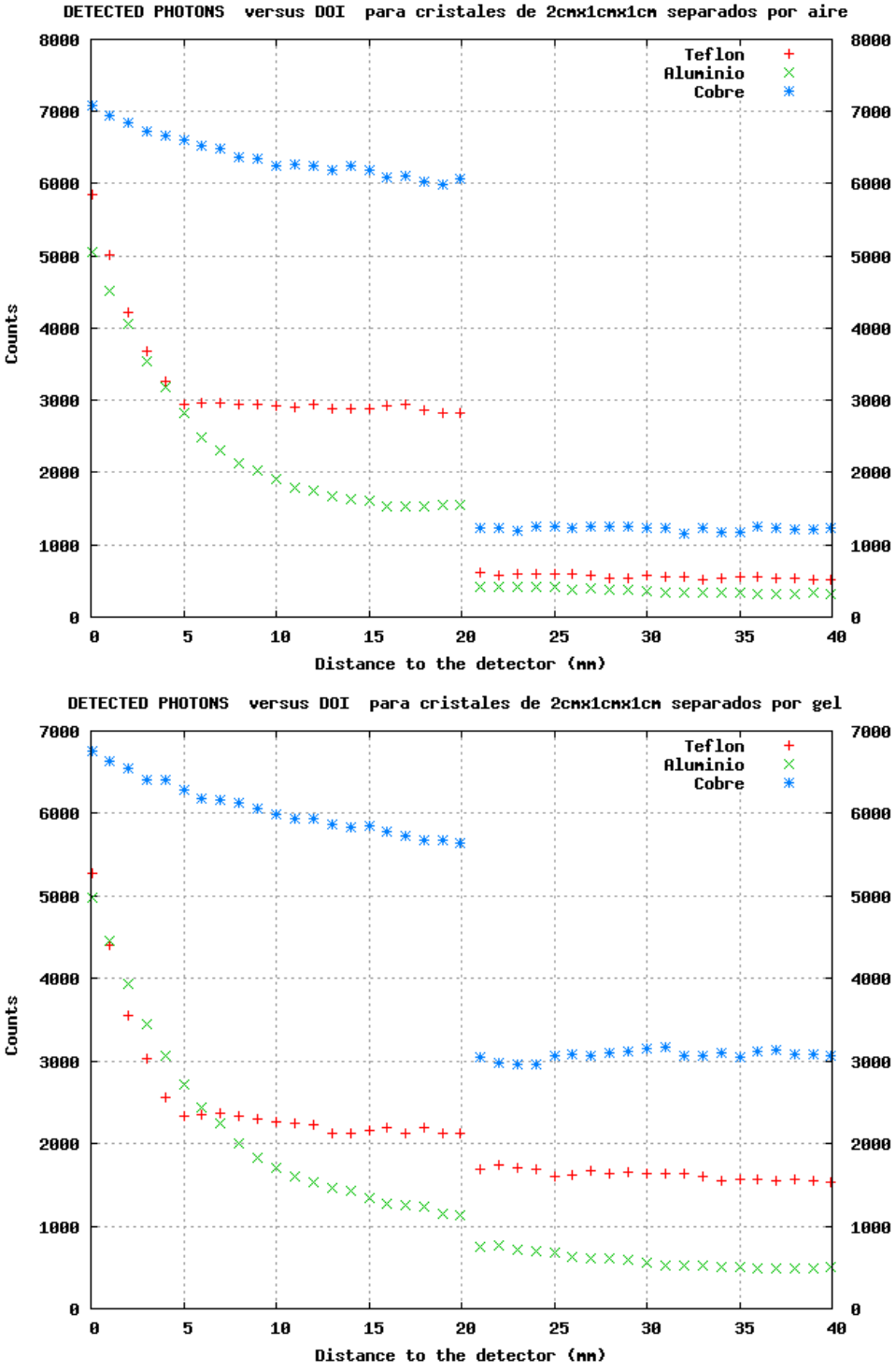

Figura 5.7: Simulaciones de un solo recubrimiento para fotones de 400 nm en cristales de 2cmx1cmx1cm.

5.3.4. Fotones de 400 nm en cristales de 7mmx1,4mmx1,4mm

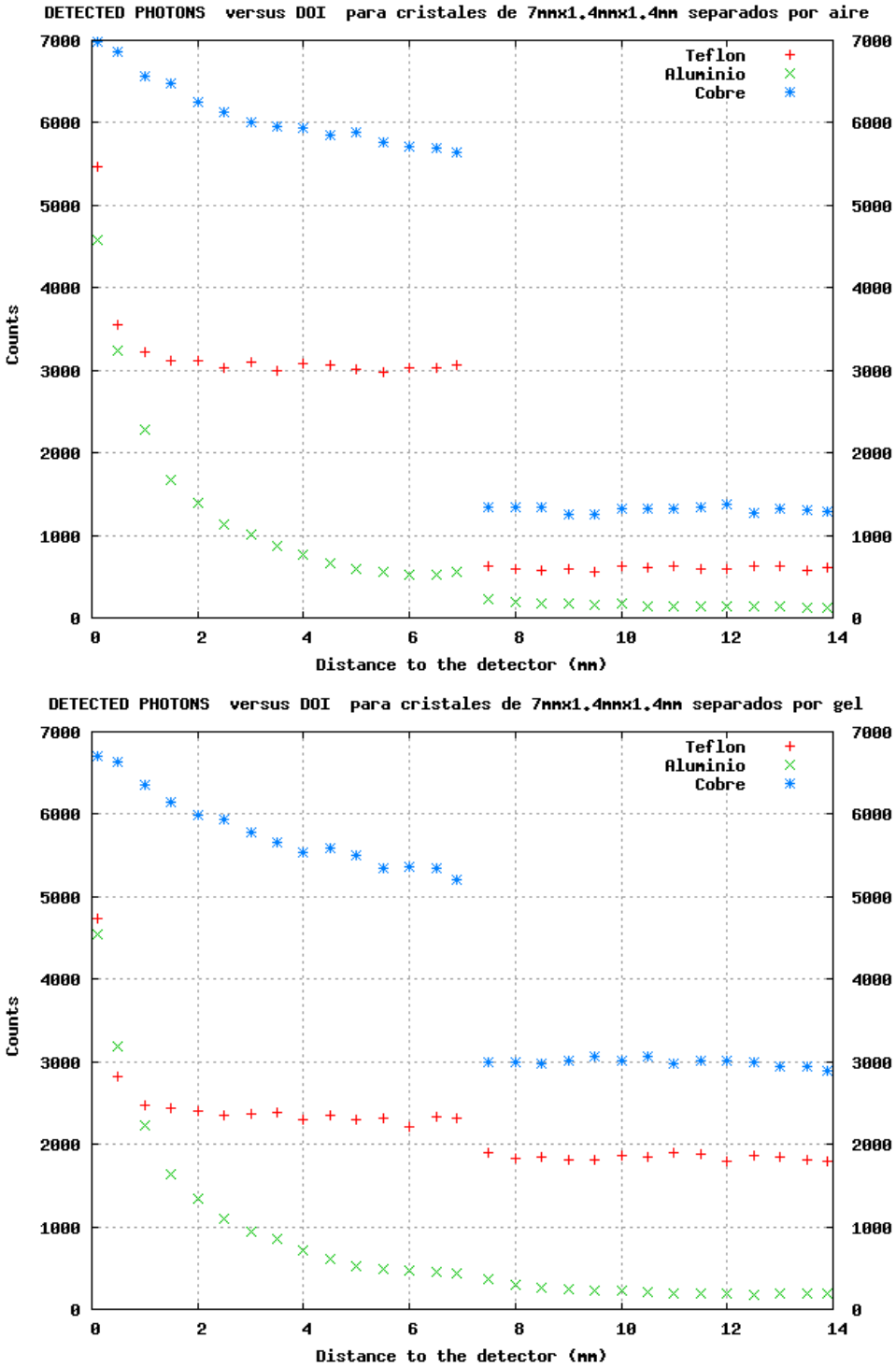

Figura 5.8: Simulaciones de un solo recubrimiento para fotones de 400 nm en cristales de 7mmx1,4mmx1,4mm.

5.3.5. Conclusiones

Como se puede ver, para los fotones de 600 nm los mejores resultados, es decir, los casos en los que es más apreciable la diferencia entre los fotones generados en el primer o el segundo cristal, son los de los buenos reflectores, concretamente el oro y el cobre, tal y como los hemos definido. A primera vista parecen más favorables los montajes en los que no hay gel entre los cristales, puesto que la capa de aire, por fina que sea, supone un cambio brusco del índice de refracción, lo cual hace que el número de fotones vistos generados en el primer cristal descienda hasta poco más de los 1000. Aunque este cambio tan acusado beneficia a nuestros propósitos, una cantidad tan reducida de fotones puede llegar a no apreciarse en el detector, lo cual podría acabar desembocando en no recibir señal. El color rojizo de esta longitud de onda puede ser la responsable de que el cobre y el oro, que tienen brillos de este color, sean tan buenos reflectores.

Por otro lado, si introducimos un gel el cambio en el índice de refracción es menos brusco, y el número de fotones que, tras generarse en el primer cristal, pueden verse se reduce a la mitad de los que se ven en el segundo cristal. En este caso, los buenos reflectores dan una variación en el número de fotones detectados bastante satisfactoria y, aún así, permiten que llegue una señal aceptable al detector. Este sería, a priori, el caso ideal en el que querríamos encontrarnos: que la intensidad que llegue al detector sea apreciablemente diferente según el fotón se produzca en el primer o en el segundo cristal pero que, a su vez, podamos detectar dichas llegadas con nuestra sensibilidad.

Otra conclusión que podemos extraer para los fotones de 600 nm es que hay una diferencia muy notable entre poner caras rugosas o un recubrimiento difusor, confiriendo las caras rugosas un resultado más satisfactorio para nuestro propósito.

Cuando pasamos a los 400 nm, que es una longitud más azulada (más parecida a la que emite realmente el LSO) los resultados cambian, pero no demasiado: el cobre ya no es un reflector tan bueno, pero sigue siendo, con diferencia, el más beneficioso de los tres que hemos estudiado. En los casos separados por aire el aluminio también mejora ligeramente sus resultados, pero es casi inapreciable. La reflectividad del teflón, según [5], no cambia apreciablemente al pasar a los 400 nm, de modo que su comportamiento es el mismo que para 600 nm.

5.4. Resultado de las simulaciones con desplazamiento tridimensional de la fuente

Para complementar los resultados de las simulaciones anteriores, vamos a estudiar el número de fotones vistos si, además de a lo largo del eje x, variamos la posición de la fuente en z e y. Para simplificar y, puesto que el cristal tiene simetría, solamente estudiaremos una de sus esquinas y con una selección de los recubrimientos que hemos estado utilizando. De las gráficas anteriores se desprende que el aluminio, el teflón y el cobre constituyen una buena muestra de la variedad de fenómenos que podemos observar en el interior de los cristales.

Para simplificar solamente tomaremos unos pocos puntos en el interior del cristal. Las gráficas posteriores constatarán que, por ahora, serán suficientes puesto que hay mucha simetría. Además de las gráficas, nos interesa obtener los ficheros con los resultados del número de fotones visto para cada posición en el interior de los cristales para futuras consultas y/o tomas de datos sin necesidad de correr los programas.

Veamos, pues, cómo varía el número de fotones vistos en el interior de los cristales, con o sin separaciones entre ellos.

5.4.1. Simulaciones para un solo cristal

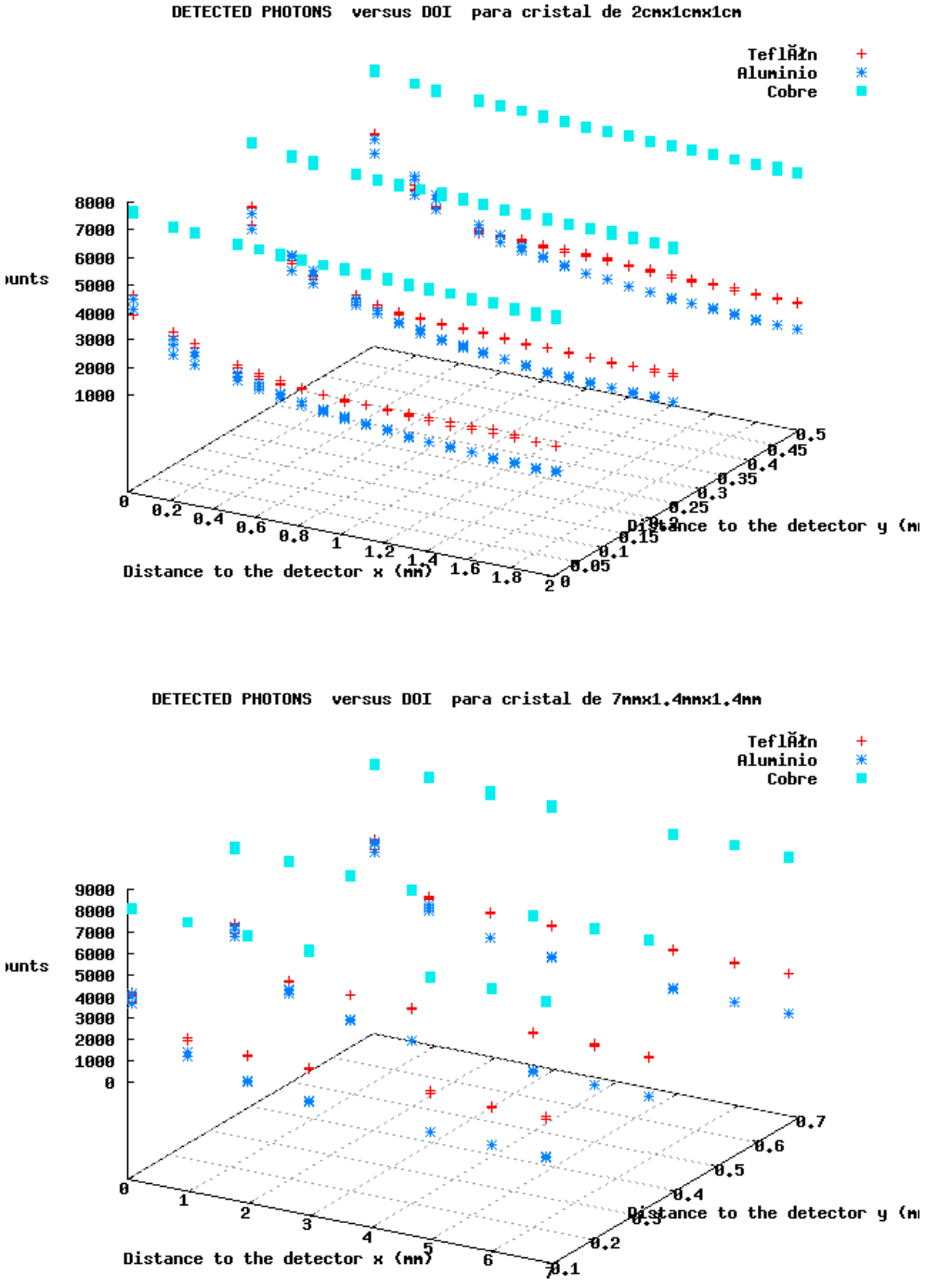

Figura 5.9: Comparación entre las simulaciones para distintos recubrimientos sobre los dos tipos de cristales de que disponemos, sin separación de ningún tipo entre ellos.

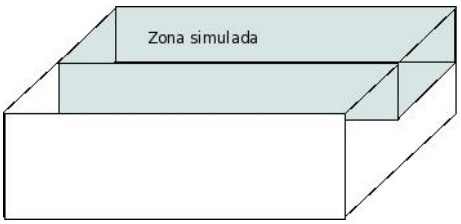

Figura 5.10: Representación genérica de la región de los cristales que hemos estudiado.

Como se puede ver, no existe diferencia apreciable entre estos nuevos resultados y los que ya obtuvimos moviendo la fuente solamente en el eje x. Aunque se han realizado y los resultados numéricos se pueden consultar, hemos omitido las gráficas restantes porque no aportan nada a las que ya se han visto.

5.4.2. Conclusiones

En estas simulaciones no se aprecian grandes diferencias con los resultados que ya obtuvimos con los desplazamientos longitudinales. El comportamiento ante los tres recubrimientos que hemos estudiado en este caso es idéntico al que ya habíamos visto y no parecen producirse deformaciones de las curvas en las proximidades de los bordes de los centelladores. No obstante, la realización de estas simulaciones y la obtención de sus resultados numéricos es importante, puesto que las tablas de resultados pueden ponerse a disposición de cualquiera que, con objeto de emplearlos en otra simulación de PET, como alguna que se pueda llevar a cabo con PENELOPET, necesite consultarlas y, bien mediante interpolación o bien mediante la ejecución de los programas que se han escrito.

5.5. Comparación con los resultados experimentales

Samuel España ha llevado a cabo un pequeño experimento en el laboratorio del Grupo de Física Nuclear para hacer una estimación del número de fotones que llegan al detector con cada recubrimiento, aunque sin tener en cuenta el lugar en el que los fotones son producidos. Empleó para ello cristales de LYSO de 5cmx1cmx1cm individualmente. La longitud de onda de los fotones que ha estudiado está en torno a los 400 nm, y ha comparado teflón, aluminio y un reflector perfecto. Cabe mencionar que también intentó hacer el experimento con un recubrimiento de pan de oro, pero los resultados eran extraordinariamente malos, hasta tal punto que apenas se podía distinguir el fotopico. Sus resultados, tomando el número de fotones vistos con un recubrimiento de teflón como la unidad, se presentan en la siguiente tabla:

Recubrimiento	Fotones vistos respecto al teflón
Totalmente absorbente	0.45
Vicuiti (reflector total)	0.95
Aluminio	0.73
Teflón	1.00

Sin atender al número de fotones visto en realidad, sino más bien a las tendencias que presentan estos resultados, podemos compararlos con los de nuestras simulaciones. En la figura 5.9

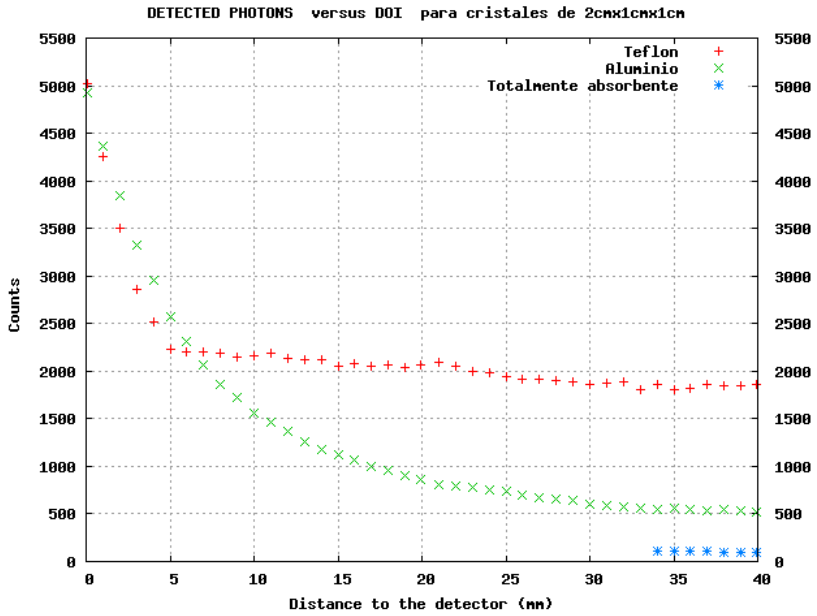

Figura 5.11: Simulaciones de un solo recubrimiento sobre cristales 2cmx1cmx1cm para fotones de 400 nm.

podemos ver los resultados de nuestra simulación. Comparemos el número de fotones vistos cuando éstos se han producido en la región más alejada del detector.

El número de fotones visto para el teflón está en torno a los 2000, mientras que para el aluminio se queda alrededor de los 500, lo cual discrepa claramente con los resultados de Samuel. Puede verse que en el caso del recubrimiento completamente absorbente pasa algo parecido: el número de fotones visto en comparación con el tefón es mucho menor que el del experimento.

Simulación de los fotones gamma

6.1. La clase TPhotoElecCompton

Es posible emplear LITRANI para simular el paso de fotones gamma de entre 0.1 y 1 MeV, que es el espectro que nos interesa en PET, a través de un material, depositando energía a consecuencia del efecto Compton. Puesto es es esto lo que se da en el interior de un aparato de PET, vamos a intentar ampliar el horizonte de nuestras simulaciones con un montaje más completo que abarque la llegada de los fotones de 511 KeV al cristal centellador, de modo que podamos estudiar el número de fotones que produce. La clase `TPhotoElecCompton` es la encargada de simular la producción de efecto Compton y efecto fotoeléctrico como consecuencia del paso de los fotones gamma. Presenta, no obstante, un inconveniente principal, y es que el cálculo sobre el que está basada está resuelto para una simetría cilíndrica. En consecuencia, las únicas formas que permiten la utilización de esta clase son los tubos y los cilindros. Ello imposibilita el estudio de la llegada de fotones gamma a los centelladores que hemos estado viendo hasta ahora.

Es por esto que el cristal centellador que vamos a estudiar en este capítulo es muy diferente a los que hemos empleado en el anterior. Disponemos de centelladores de NaI dopado con talio, esto es, NaI(Tl) de forma cilíndrica con una pulgada de altura y diámetro. El recubrimiento que emplearemos será aluminio. Puesto que la interacción de los fotones con el material es más fina en el efecto Compton que en las simulaciones meramente ópticas que hemos llevado a cabo hasta ahora, es necesario ser cuidadoso con las características de los materiales que simulemos, así como dar más información de la que hemos requerido hasta ahora. Además, es necesario establecer el material como fluorescente y dar sus características de fluorescencia.

Material	n_R	L_a (nm)	μ_r	A	Z	ρ (g/cm^3)	λ_{max} (nm)	τ (ns)
NaI (Tl)	1.85	400	1.0	46.57	64.3	3.67	415	230

La sección eficaz del NaI(Tl) ha sido obtenida de la página web del NIST [9], puesto que se necesitaba un conjunto notable de valores para realizar una interpolación con TSplineFit. De igual modo, los datos del aluminio son similares a los utilizados en capítulos anteriores, pero empleando unos valores interpolados para los índices de refracción contenidos en las librerías de LITRANI.

6.2. Fotones vistos en un centellador

Vamos a suponer que la fuente de gammas se encuentra en el centro del cilindro, siendo la última de sus caras un detector de superficie. En estas condiciones, y empleando VisuLitrani podemos estudiar el espectro de fotones que llegan al detector, así como los que son absorbidos por el material y por el recubrimiento.

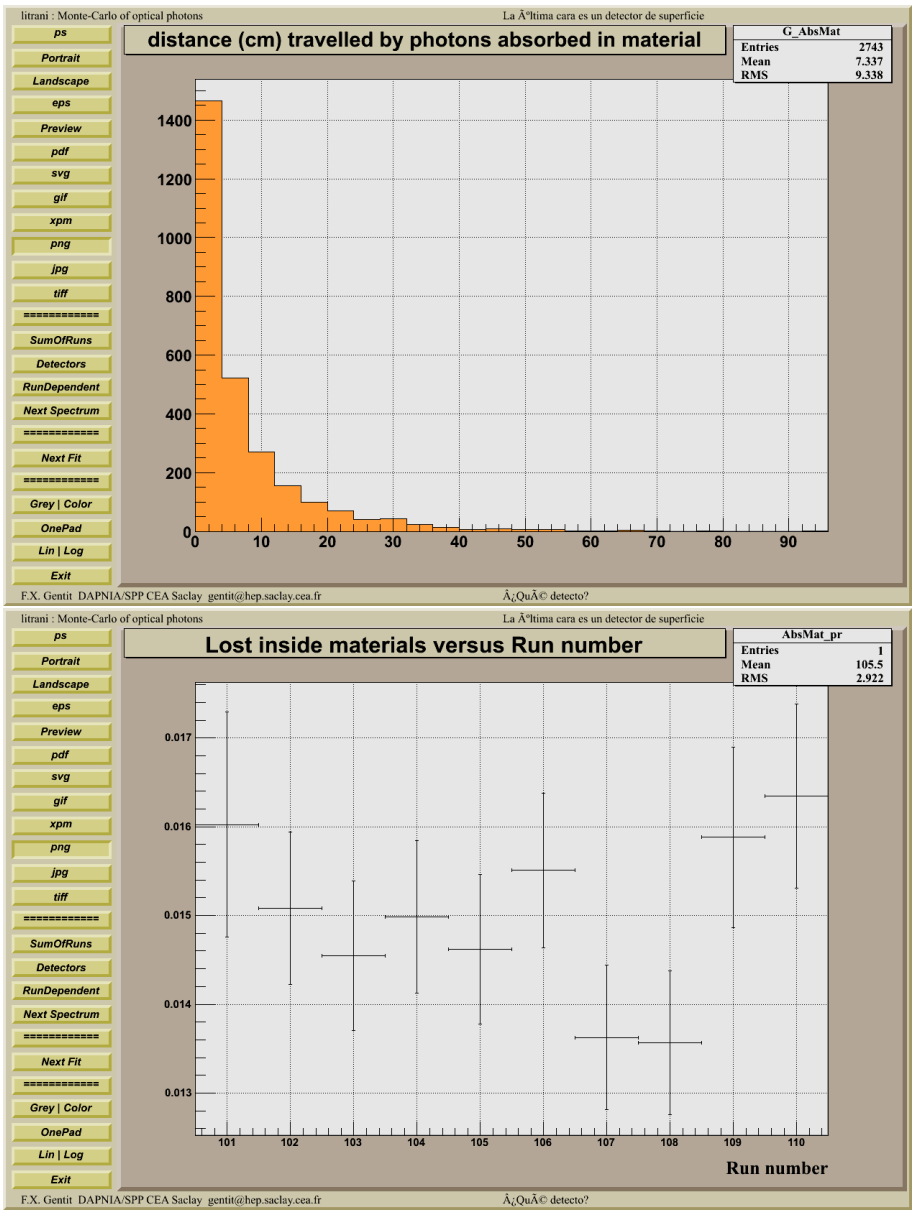

Figura 6.1: Fotones absorbidos por el material.

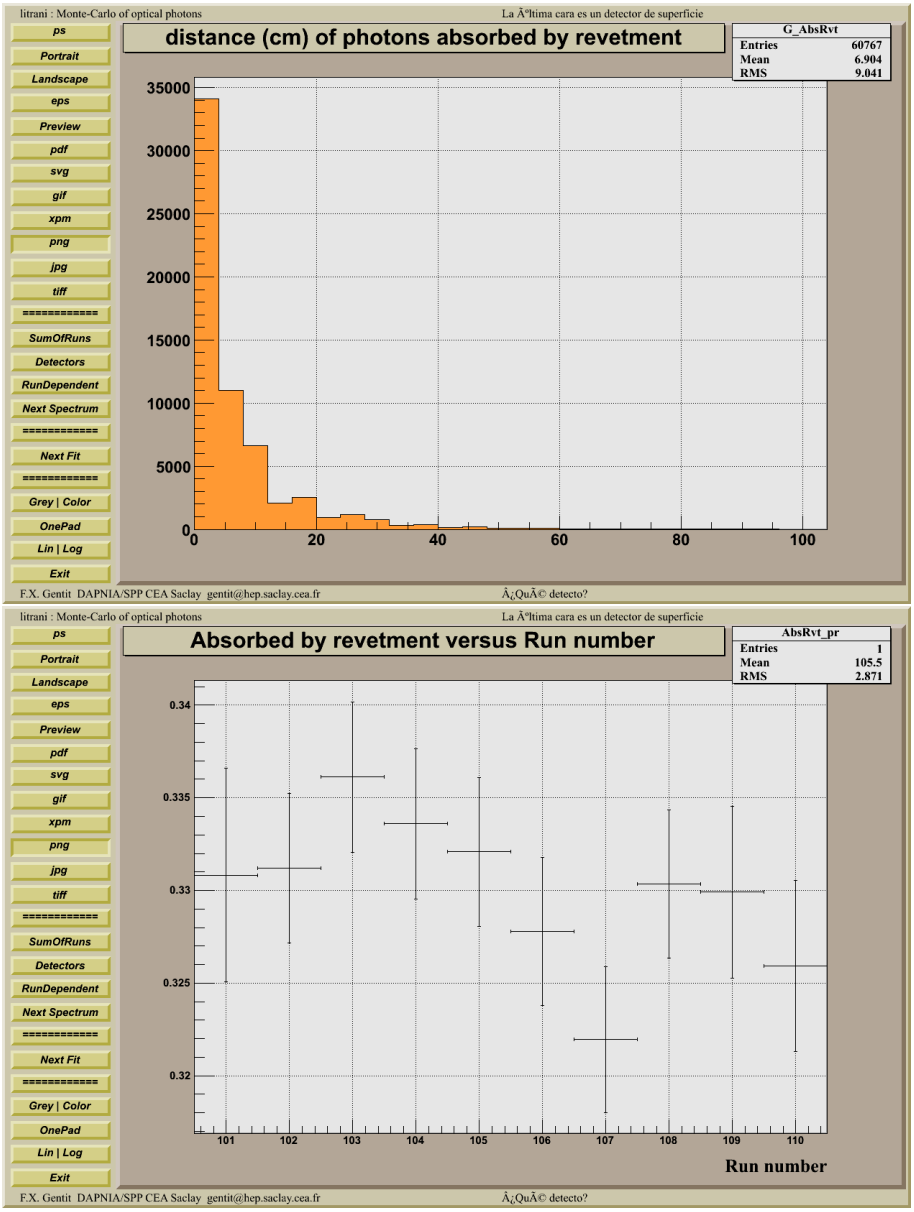

Figura 6.2: Fotones absorbidos por el revestimiento.

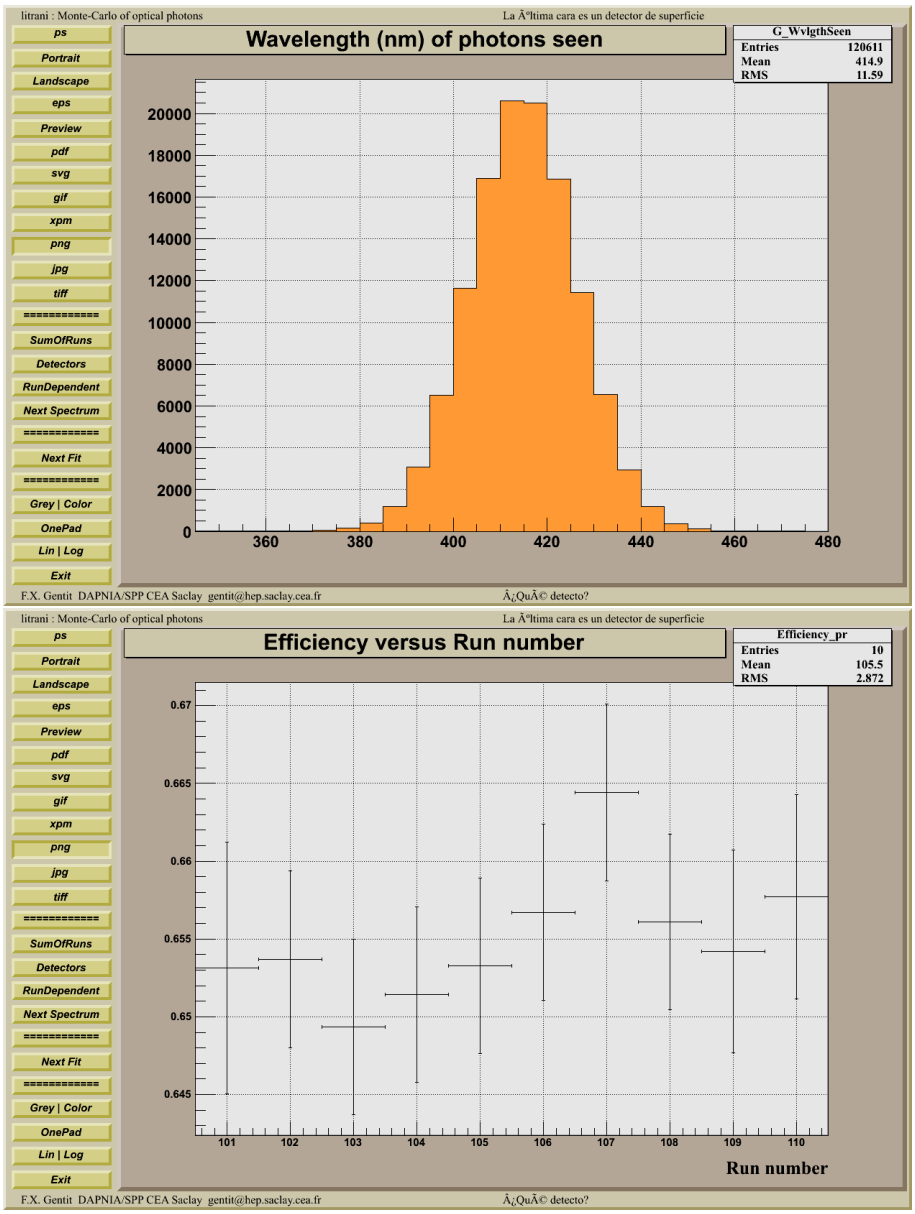

Figura 6.3: Fotones vistos.

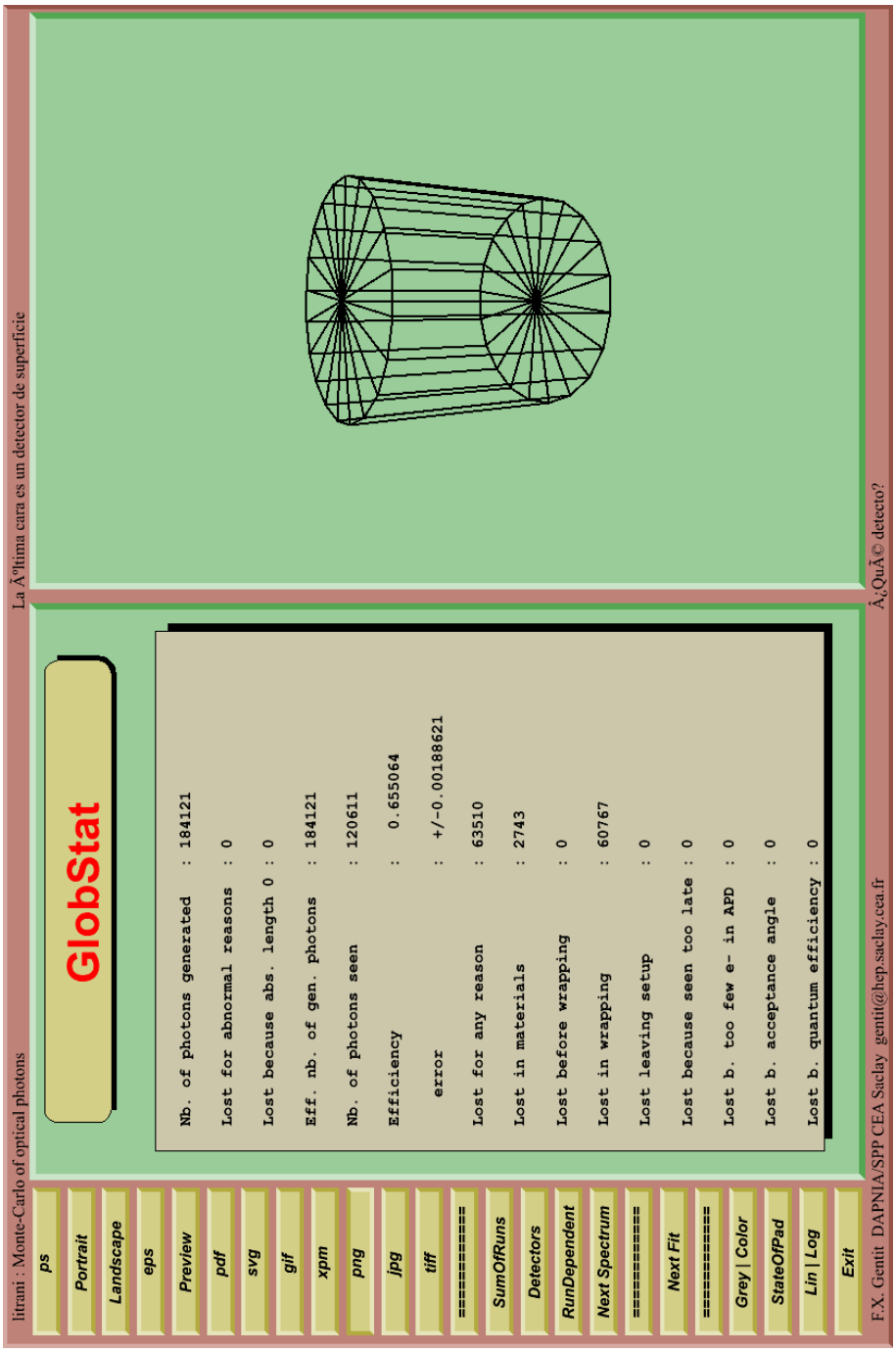

Figura 6.4: Estadísticas globales.

6.3. Conclusiones

Tras la obtención de estos resultados el siguiente paso sería, a semejanza de lo que hemos hecho en el capítulo anterior, estudiar otros casos en los que la fuente de rayos gamma no esté en el centro del cilindro. De ese modo se podría observar la variación en el número de fotones vistos o en la longitud de onda de éstos en función del punto en el que el fotón alcance el cristal. Desgraciadamente, modificar esta simulación para variar la posición de la fuente ha sido imposible, puesto que las iteraciones quedaban truncadas por errores en la ejecución. Además, en los pocos casos en que se pudo implementar la modificación con éxito los resultados eran exactamente los mismos que los que aquí hemos presentado. Todo esto, por supuesto, da que pensar sobre la fiabilidad y la utilidad de los cálculos que LITRANI pueda hacer con la clase `TPhotoElecCompton`.

Dicho esto, podemos pasar a analizar los resultados que hemos obtenido. En primer lugar tenemos la distancia recorrida por los fotones en el interior del material. El histograma presenta una tendencia claramente exponencial, lo cual entra dentro de lo que ya vimos en el capítulo 3. También es interesante poder acceder a las estadísticas de cada ejecución del programa, pues nos permiten ver que las fluctuaciones de, en este caso, el número de fotones perdidos en el material son reducidas.

El siguiente resultado que hemos obtenido es el de la distancia recorrida pòr los los fotones que son absorbidos por el recubrimiento de aluminio, así como las fluctuaciones del número de éstos en cada iteración. En este caso, las fluctuaciones son un orden de magnitud mayores que en el anterior, lo cual se refleja en el histograma final, que no marca una tendencia exponencial tan clara como aquél.

La longitud de onda de los fotones vistos sigue una distribución gaussiana centrada en la longitud de emisión de NaI. La distribución es perfectamente simétrica y las fluctuaciones en la eficiencia del detector son reducidas. Por último, hemos presentado las estadísticas globales tal u como las ofrece VisuLitrani. De ellas se desprende que 10000 fotones gamma generan 184121 fotones en el espectro visible en de la luminiscencia del centellador, el 65,5 % de los cuales son vistos por el detector de superficie que hemos situado en una de las caras planas del cilindro.

Resultados y conclusiones

Durante la realización de este trabajo hemos aprendido, en primer lugar, a utiliar un software completamente desconocido para nosotros, hasta llegar a simular montajes de cierta complejidad con él. Además, de este aprendizaje hemos podido escribir un pequeño manual de iniciación para los posibles interesados en el manejo de LITRANI, así como una buena cantidad de programas sencillos para entrenarse en su utilización. Conocer su funcionamiento, posibilidades y limitaciones es de gran utilidad para futuros trabajos en los que se necesite un software de simulación óptica.

Lamentablemente, los resultados que hemos obtenido en nuestras simulaciones son difícilmente contrastables y, a nuestro parecer, no demasiado fiables. Además, la pequeña comprobación experimental de la bondad de nuestros resultados ha arrojado conclusiones bastante negativas. Los buenos datos obtenidos para, por ejemplo, recubrimientos de cobre y oro sobre cristales de LSO caracen de lógica, pues estos materiales no son buenos reflectores en el espectro azulado del visible. La única explicación que podemos encontrar a este hecho es que hayamos cometido algún error a la hora de definir estos materiales. Y es que una de las principales limitaciones de LITRANI es en modo en el que programamos los materiales y recubrimientos: los datos que debemos introducir para ejecutar el programa (la constante dieléctrica, la parte compleja del índice de refracción...) no se encuentran tabulados ni estudiados para la mayoría de los materiales comerciales, de los que se conocen otras cosas, como su reflectividad. Es una lástima que encontrar todos estos números tabulados para los materiales a simular sea tan complicado. Esta falta de seguridad con respecto a los datos que hemos utilizado es una de las principales causas de la poca fiabilidad de nuestros resultados.

Tampoco hemos tenido mucho éxito al simular la llegada de fotones gamma a un centellador, puesto que no hemos sido capaces de estudiar distintos puntos de llegada de los fotones en el seno del cristal, aunque bien es cierto que los resultados obtenidos para la simulación que logramos hacer con la fuente en el centro del cilindro fueron muy razonables al comparar sus tendencias con las predicciones teóricas.

Programas empleados en el capítulo 4

A.1. Ejemplo 1: comprobación de equivalencia de montajes

A.1.1. Un solo cristal

LITRANI

```
void ejemplo1c(Double_t seed)

{
  char *name    = "EJEMPLO 1c. Un cristal de 1cm^3 de  LSO recubierto de material totalmente reflector";
  char *listing = "La cara última del cristal es un fotocátodo ";
  char *upcom   = "La fuente es puntual, isotropica y espontanea, y la situo dentro del cristal ";

  char *downcom = "¿Cuántos fotones detecto?";

  gROOT->ProcessLine(".x InitLitrani.C(8,name,listing,upcom,downcom,kTRUE,kFALSE,kTRUE)");
  gRandom3->SetSeed(seed);

//1.- GEOMETRÍA

        const Double_t semilongitud_en_x = 0.5;
        const Double_t semilongitud_en_y = 0.5;
        const Double_t semilongitud_en_z = 0.5;

//2.- MATERIALES

        TOpticMaterial *LSO;
        LSO = new TOpticMaterial("LSO","Titulito,¡es LSO!",kFALSE,1.0,21);
        LSO->IsIsotropic(1.82);

        TRevetment *reflectortotal;
        reflectortotal = new TRevetment("reflectortotal","Recubrimiento reflector total",
"none",0.00, 0.0,-1.0,1.0,0.0,90.0);

//3.- FORMAS

         TSBRIK *prisma;
         prisma = new TSBRIK("prisma","Cristal de LSO","LSO","reflectortotal",
semilongitud_en_x,semilongitud_en_y,semilongitud_en_z);

   prisma->fSuppl->SetSurfDet("detector","Detector de superficie",1,"none");

//4.- ORDENAR ELEMENTOS DEL MONTAJE

    TSNode *nodo1;//Declaro el nodo1 como un elemento de la clase TSNode()

    nodo1 = new TSNode("nodo1","Nodo asociado al cristal",prisma);
    nodo1->cd();

//5.- CONTACTOS

//6.- FUENTES
```

```
    TSpontan *fuente;
    fuente = new TSpontan("fuente","fuente","prisma",0.0001,0.0,0.0,600.0);

   for (Int_t i=1;i<=100;i++) {

     fuente->Gen(i,100,-2.0,true,false);
   }

cout << "fNbPart:    " << gGs->fNpGener    << endl;//Nº total de fotones generados.
cout << "fNpSeen:    " << gGs->fNpSeen     << endl;//Nº de fotones detectados.
cout << "fNpLossAny: " << gGs->fNpLossAny  << endl;//Nº de fotones perdidos por alguna razón.
cout << "fNpAbsMat:  " << gGs->fNpAbsMat   << endl;//Nº de fotones absorbidos en los materiales.
cout << "fNpAbsRvt:  " << gGs->fNpAbsRvt   << endl;//Nº de fotones perdidos en el revesitmiento.
cout << "fNpOutSide: " << gGs->fNpOutSide  << endl;//Nº de fotones que abandonan el montaje.
cout << "fNpAbnorm:  " << gGs->fNpAbnorm   << endl;//Nº de fotones que son destruidos antes de generarse.
cout << "fNpLossQE:  " << gGs->fNpLossQE   << endl;//Nº de fotones perdidos porque el detector no es  perfecto

  exit();

  }
```

SH

```
#!/bin/tcsh

  echo "" > hola
  echo "" > hola2
  echo "" > hola3
  echo "" > hola4
  echo "" > hola5
  echo "" > hola6
  echo "" > hola7

   for seed in 0 10 50 100 200 500 750 1000 1500 3000 5000 7000 10000 50000 100000
do
    echo running $seed
    echo $seed >> hola2
    command='litrani ejemplo1c.C('$seed')'
    $command > output
    cat output | grep fNpSeen  | sed 's/fNpSeen://' >> hola
    cat output | grep fNpLossQE  | sed 's/fNpLossQE://' >> hola3
    cat output | grep fNpAbsRvt  | sed 's/fNpAbsRvt://' >> hola4
    cat output | grep fNpAbsMat  | sed 's/fNpAbsMat://' >> hola5
    cat output | grep fNbPart  | sed 's/fNbPart://' >> hola6
    cat output | grep fNpOutSide  | sed 's/fNpOutSide://' >> hola7
   done

  paste hola2 hola hola3 hola4 hola5 hola6 hola7 > ejemplo1c.txt
```

A.1.2. Dos cristales consecutivos

LITRANI

```
void ejemplo1c(Double_t seed)

{
  char *name    = "EJEMPLO 1c. Un cristal de 1cm^3 de  LSO recubierto de material totalmente reflector";
  char *listing = "La cara última del cristal es un fotocátodo ";
  char *upcom   = "La fuente es puntual, isotropica y espontanea, y la situo dentro del cristal ";

  char *downcom = "¿Cuántos fotones detecto?";

  gROOT->ProcessLine(".x InitLitrani.C(8,name,listing,upcom,downcom,kTRUE,kFALSE,kTRUE)");
  gRandom3->SetSeed(seed);

//1.- GEOMETRÍA

        const Double_t semilongitud_en_x1 = 0.25;
        const Double_t semilongitud_en_y1 = 0.5;
        const Double_t semilongitud_en_z1 = 0.5;
```

```
        const Double_t semilongitud_en_x2 = 0.25;
        const Double_t semilongitud_en_y2 = 0.5;
        const Double_t semilongitud_en_z2 = 0.5;

//2.- MATERIALES

        TOpticMaterial *LSO;
        LSO = new TOpticMaterial("LSO","Titulito,¡es LSO!",kFALSE,1.0,21);
        LSO->IsIsotropic(1.82);

        TRevetment *reflectortotal;
        reflectortotal = new TRevetment("reflectortotal","Recubrimiento reflector total",
"none",0.00, 0.0,-1.0,1.0,0.0,90.0);

//3.- FORMAS

        TSBRIK *prisma1;
         prisma1 = new TSBRIK("prisma1","Cristal de LSO 1","LSO","reflectortotal",
semilongitud_en_x1,semilongitud_en_y1,semilongitud_en_z1);

        TSBRIK *prisma2;
         prisma2 = new TSBRIK("prisma2","Cristal de LSO 2","LSO","reflectortotal",
semilongitud_en_x2,semilongitud_en_y2,semilongitud_en_z2);

      prisma2->fSuppl->SetSurfDet("detector","Detector de superficie",1,"none");

//4.- ORDENAR ELEMENTOS DEL MONTAJE

    TSNode *nodo1;
    nodo1 = new TSNode("nodo1","Nodo asociado al cristal 1",prisma1);
    nodo1->cd();

    TSNode *nodo2;
    nodo2 = new TSNode("nodo2","Nodo asociado al cristal 2",prisma2,0.5,0.0,0.0 );

//5.- CONTACTOS

    TContact *contacto;
    contacto = new TContact("entre_cristales","Contacto entre cristales","prisma1",
"prisma2",identical,"none");

//6.- FUENTES

    TSpontan *fuente;
    fuente = new TSpontan("fuente","fuente","prisma2",-0.2499,0.0,0.0,600.0);

    for (Int_t i=1;i<=100;i++) {

     fuente->Gen(i,100,-2.0,true,false);
   }

cout << "fNbPart:    " << gGs->fNpGener    << endl;//Nº total de fotones generados.
cout << "fNpSeen:    " << gGs->fNpSeen    << endl;//Nº de fotones detectados.
cout << "fNpLossAny: " << gGs->fNpLossAny << endl;//Nº de fotones perdidos por alguna razón.
cout << "fNpAbsMat:  " << gGs->fNpAbsMat  << endl;//Nº de fotones absorbidos en los materiales.
cout << "fNpAbsRvt:  " << gGs->fNpAbsRvt  << endl;//Nº de fotones perdidos en el revesitmiento.
cout << "fNpOutSide: " << gGs->fNpOutSide << endl;//Nº de fotones que abandonan el montaje.
cout << "fNpAbnorm:  " << gGs->fNpAbnorm  << endl;//Nº de fotones que son destruidos antes de generarse.
cout << "fNpLossQE:  " << gGs->fNpLossQE  << endl;//Nº de fotones perdidos porque el detector no es  perfecto

  exit();

  }
```

GNU DISPLAY

```
set ylabel 'Counts '
set xlabel 'Random number generator seed'
set title 'DETECTED PHOTONS  versus RANDOM NUMER GENERATOR SEED'
set grid
set logscale x
set y2tics
```

```
set ytics nomirror
#set term post color solid
set term png
set out 'ejemplo1.png'

p 'ejemplo1d.txt' u ($1*10):($5) t   'Dos cristales consecutivos',
'ejemplo1c.txt' u ($1*10):($5) t   'Un solo cristal'
set term x11
set out
rep

pause -1
```

A.2. Ejemplo 2: Reflectancia y transmitancia

A.2.1. Montaje para la detección de fotones transmitidos

LITRANI

```
void ejemplo2a(void)

{
  char *name    = "EJEMPLO 2a. La luz de un láser atraviesa un bloque de aire y
luego otro de índice de refracción 1,5.";
  char *listing = "La cara última del cristal es un fotocátodo.";
  char *upcom   = "La fuente es un láser pegado al bloque de aire ";

  char *downcom = "¿Cuántos fotones detecto?";

  gROOT->ProcessLine(".x InitLitrani.C(8,name,listing,upcom,downcom,kFALSE,kFALSE,kTRUE)");

//1.- GEOMETRÍA

        const Double_t semilongitud_en_x1 = 0.5;
        const Double_t semilongitud_en_y1 = 0.5;
        const Double_t semilongitud_en_z1 = 0.5;

        const Double_t semilongitud_en_x2 = 0.5;
        const Double_t semilongitud_en_y2 = 0.5;
        const Double_t semilongitud_en_z2 = 0.5;

// 2.- MATERIALES

        TOpticMaterial *cristal;
        cristal = new TOpticMaterial("cristal","Un cubo de algo",kFALSE,1.0,10000.0);
        cristal->IsIsotropic(1.50);

        TOpticMaterial *aire;
        aire = new TOpticMaterial("aire","Un cubo de aire",kFALSE,1.0,10000.0);
        aire->IsIsotropic(1.00);

              TRevetment *reflectortotal;
        reflectortotal = new TRevetment("reflectortotal","Recubrimiento reflector total",
"none",0.0, 0.0,-1.0,1.0,0.0,90.0);


 // 3.- FORMAS

         TSBRIK *laserbrick;
         laserbrick = new TSBRIK("laserbrick","Bloque de aire con láser","aire","none",
semilongitud_en_x1,semilongitud_en_y1,semilongitud_en_z1);

         TSBRIK *detectorbrick;
         detectorbrick = new TSBRIK("detectoribrick","Cristal","cristal","none",
semilongitud_en_x2,semilongitud_en_y2,semilongitud_en_z2);

   detectorbrick->fSuppl->SetSurfDet("detector","Detector de superficie",1,"none");

// 4.- ORDENAR LOS ELEMENTOS DEL MONTAJE
```

```
    TSNode *nodo1;//Declaro el nodo1 como un elemento de la clase TSNode()

    nodo1 = new TSNode("nodo1","Nodo asociado al láser",laserbrick);
    nodo1->cd();

    TSNode *nodo2;
    nodo2 = new TSNode("nodo2","Nodo asociado al cristal",detectorbrick,1.0,0.0,0.0 );

// 5.- CONTACTOS

 TContact *contacto;
 contacto = new TContact("laser-detector","Contacto entre el láser y el cristal con detector",
"laserbrick","detectoribrick",identical,"none");

// 4.- FUENTES

    TSpontan *laser;
    laser = new TSpontan("laser","laser","laserbrick",0.0,0.0,0.0,600.0);
    laser->DefineCradle("laserbrick",point);
    laser->DefineBeam(flat,0.0,90.0,0.0);
    for (Int_t i=1;i<=100;i++) {
    laser->Gen(i,100,-2.0,true,false);

  }

cout << "fNbPart:    " << gGs->fNpGener   << endl;//Nº total de fotones generados
cout << "fNpSeen:    " << gGs->fNpSeen    << endl;//Nº de fotones detectados
cout << "fNpLossAny: " << gGs->fNpLossAny << endl;//Nº de fotones perdidos por alguna razón.
cout << "fNpAbsMat:  " << gGs->fNpAbsMat  << endl;//Nº de fotones absorbidos en los materiales.
cout << "fNpAbsRvt:  " << gGs->fNpAbsRvt  << endl;//Nº de fotones perdidos en el revesitmiento.
cout << "fNpOutSide: " << gGs->fNpOutSide << endl;//Nº de fotones que abandonan el montaje????
cout << "fNpAbnorm:  " << gGs->fNpAbnorm  << endl;//Nº de fotones destruidos antes de generarse
cout << "fNpLossQE:  " << gGs->fNpLossQE  << endl;//Nº de fotones perdidos porque el detector no es perfecto
  exit();

  }
```

A.2.2. Montaje para la detección de fotones reflejados

LITRANI

```
void ejemplo2b(void)

{
  char *name    = "EJEMPLO 2b. La luz de un láser atraviesa un bloque de aire y
luego otro de índice de refracción 1,5.";
  char *listing = "La cara primera del bloque de aire es un fotocátodo.";
  char *upcom   = "La fuente es un láser pegado al bloque de aire ";
  char *downcom = "¿Cuántos fotones detecto?";

  gROOT->ProcessLine(".x InitLitrani.C(8,name,listing,upcom,downcom,kTRUE,kFALSE,kTRUE)");
  gRandom3->SetSeed();

//1.- GEOMETRÍA

        const Double_t semilongitud_en_x1 = 0.5;
        const Double_t semilongitud_en_y1 = 0.5;
        const Double_t semilongitud_en_z1 = 0.5;

        const Double_t semilongitud_en_x2 = 0.5;
        const Double_t semilongitud_en_y2 = 0.5;
        const Double_t semilongitud_en_z2 = 0.5;

// 2.- MATERIALES

        TOpticMaterial *cristal;
        cristal = new TOpticMaterial("cristal","Un cubo de algo",kFALSE,1.0,10000.0);
        cristal->IsIsotropic(1.50);
```

```
        TOpticMaterial *aire;
        aire = new TOpticMaterial("aire","Un cubo de aire",kFALSE,1.0,10000.0);
        aire->IsIsotropic(1.00);

              TRevetment *totabsorbing;
        totabsorbing = new TRevetment("TotAbsorbing","Totally absorbing revetment","none",1.0,5,0.5,1.0,1.0,90.0);

 // 3.- FORMAS

         TSBRIK *laserbrick;
         laserbrick = new TSBRIK("laserbrick","Bloque de aire con láser","aire","none",
semilongitud_en_x1,semilongitud_en_y1,semilongitud_en_z1);

         TSBRIK *detectorbrick;
         detectorbrick = new TSBRIK("detectoribrick","Cristal","cristal","none",
semilongitud_en_x2,semilongitud_en_y2,semilongitud_en_z2);

   laserbrick->fSuppl->SetSurfDet("detector","Detector de superficie",3,"none");

// 4.- ORDENAR LOS ELEMENTOS DEL MONTAJE

    TSNode *nodo1;//Declaro el nodo1 como un elemento de la clase TSNode()

    nodo1 = new TSNode("nodo1","Nodo asociado al láser",laserbrick);
    nodo1->cd();

    TSNode *nodo2;
    nodo2 = new TSNode("nodo2","Nodo asociado al cristal",detectorbrick,1.0,0.0,0.0 );

// 5.- CONTACTOS

 TContact *contacto;
 contacto = new TContact("laser-detector","Contacto entre el láser y el cristal con detector",
"laserbrick","detectoribrick",identical,"none");

// 4.- FUENTES

    TSpontan *laser;
    laser = new TSpontan("laser","laser","laserbrick",0.0,0.0,0.0,600.0);
    laser->DefineCradle("laserbrick",point);
    laser->DefineBeam(flat,0.0,90.0,0.0);
    for (Int_t i=1;i<=100;i++) {
    laser->Gen(i,100,-2.0,true,false);

  }

cout << "fNbPart:    " << gGs->fNpGener   << endl;//Nº total de fotones generados
cout << "fNpSeen:    " << gGs->fNpSeen    << endl;//Nº de fotones detectados
cout << "fNpLossAny: " << gGs->fNpLossAny << endl;//Nº de fotones perdidos por alguna razón.
cout << "fNpAbsMat:  " << gGs->fNpAbsMat  << endl;//Nº de fotones absorbidos en los materiales.
cout << "fNpAbsRvt:  " << gGs->fNpAbsRvt  << endl;//Nº de fotones perdidos en el revesitmiento.
cout << "fNpOutSide: " << gGs->fNpOutSide << endl;//Nº de fotones que abandonan el montaje????
cout << "fNpAbnorm:  " << gGs->fNpAbnorm  << endl;//Nº de fotones destruidos antes de generarse
cout << "fNpLossQE:  " << gGs->fNpLossQE  << endl;//Nº de fotones perdidos porque el detector no es perfecto
  exit();

  }
```

A.3. Ejemplo 3: Estudio de la absorción de un dieléctrico

LITRANI

```
void ejemplo3(Double_t x_var)

{
  char *name    = "EJEMPLO 3. La luz de un láser atraviesa un bloque de aire y
luego otro de índice de refracción real 1,5 y de longitud variable.";
  char *listing = "La última cara del cristal es un fotocátodo.";
  char *upcom   = "La fuente es un láser pegado al bloque de aire ";
  char *downcom = "¿Cuántos fotones detecto?";
```

```
  gROOT->ProcessLine(".x InitLitrani.C(8,name,listing,upcom,downcom,kFALSE,kFALSE,kTRUE)");

//1.- GEOMETRÍA

        const Double_t semilongitud_en_x1 = 0.5;
        const Double_t semilongitud_en_y1 = 0.5;
        const Double_t semilongitud_en_z1 = 0.5;

        const Double_t semilongitud_en_y2 = 0.5;
        const Double_t semilongitud_en_z2 = 0.5;

// 2.- MATERIALES

        TOpticMaterial *cristal;
        cristal = new TOpticMaterial("cristal","Un cubo de algo",kFALSE,1.0,400.0);
        cristal->IsIsotropic(1.50);

        TOpticMaterial *aire;
        aire = new TOpticMaterial("aire","Un cubo de aire",kFALSE,1.0,10000.0);
        aire->IsIsotropic(1.00);

               TRevetment *totabsorbing;
        totabsorbing = new TRevetment("TotAbsorbing","Totally absorbing revetment","none",
1.0,5,0.5,1.0,1.0,90.0);

 // 3.- FORMAS

         TSBRIK *laserbrick;
         laserbrick = new TSBRIK("laserbrick","Bloque de aire con láser","aire","none",
semilongitud_en_x1,semilongitud_en_y1,semilongitud_en_z1);

         TSBRIK *detectorbrick;
         detectorbrick = new TSBRIK("detectoribrick","Cristal","cristal","none",
x_var,semilongitud_en_y2,semilongitud_en_z2);

   detectorbrick->fSuppl->SetSurfDet("detector","Detector de superficie",1,"none");

// 4.- ORDENAR LOS ELEMENTOS DEL MONTAJE

    TSNode *nodo1;//Declaro el nodo1 como un elemento de la clase TSNode()

    nodo1 = new TSNode("nodo1","Nodo asociado al láser",laserbrick);
    nodo1->cd();

    TSNode *nodo2;
    nodo2 = new TSNode("nodo2","Nodo asociado al cristal",detectorbrick,
semilongitud_en_x1+x_var,0.0,0.0 );

// 5.- CONTACTOS

 TContact *contacto;
 contacto = new TContact("laser-detector","Contacto entre el láser y el cristal con detector",
"laserbrick","detectoribrick",identical,"none");

// 4.- FUENTES

    TSpontan *laser;
    laser = new TSpontan("laser","laser","laserbrick",0.0,0.0,0.0,600.0);
    laser->DefineCradle("laserbrick",point);
    laser->DefineBeam(flat,0.0,90.0,0.0);
    for (Int_t i=1;i<=100;i++) {
    laser->Gen(i,100,-2.0,true,false);

  }

cout << "fNbPart:    " << gGs->fNpGener   << endl;//Nº total de fotones generados
cout << "fNpSeen:    " << gGs->fNpSeen    << endl;//Nº de fotones detectados
cout << "fNpLossAny: " << gGs->fNpLossAny << endl;//Nº de fotones perdidos por alguna razón.
cout << "fNpAbsMat:  " << gGs->fNpAbsMat  << endl;//Nº de fotones absorbidos en los materiales.
cout << "fNpAbsRvt:  " << gGs->fNpAbsRvt  << endl;//Nº de fotones perdidos en el revesitmiento.
cout << "fNpOutSide: " << gGs->fNpOutSide << endl;//Nº de fotones que abandonan el montaje????
cout << "fNpAbnorm:  " << gGs->fNpAbnorm  << endl;//Nº de fotones destruidos antes de generarse
```

```
cout << "fNpLossQE:  " << gGs->fNpLossQE  << endl;//Nº de fotones perdidos porque el detector no es perfecto
  exit();

  }
```

SH

```
#!/bin/tcsh

  echo "" > hola
  echo "" > hola2
  echo "" > hola3
  echo "" > hola4
  echo "" > hola5
  echo "" > hola6
  echo "" > hola7

    for x_var in 0.1 0.2 0.5 0.75 1.00 1.25 1.50 2.00 2.50 5.00
7.50 10.00 15.00 20.00 25.00 30.00 35.00 40.00 45.00  50.00 75.00
100.00 125.00 150.00 175.00  200.00 225.00 250.00 275.00 300.00
325.00 350.00 375.00 400.00
do
    echo running $x_var
    echo $x_var >> hola2
    command='litrani ejemplo3.C('$x_var')'
    $command > output
    cat output | grep fNpSeen  | sed 's/fNpSeen://' >> hola
    cat output | grep fNpLossQE  | sed 's/fNpLossQE://' >> hola3
    cat output | grep fNpAbsRvt  | sed 's/fNpAbsRvt://' >> hola4
    cat output | grep fNpAbsMat  | sed 's/fNpAbsMat://' >> hola5
    cat output | grep fNbPart  | sed 's/fNbPart://' >> hola6
    cat output | grep fNpOutSide  | sed 's/fNpOutSide://' >> hola7
   done

  paste hola2 hola hola3 hola4 hola5 hola6 hola7 > ejemplo3.txt
  gnuplot ejemplo3.gnu
```

GNU DISPLAY

```
set ylabel 'Counts '
set xlabel 'Posición del detector'
set title 'FOTONES DETECTADOS  versus PROFUNDIDAD DE DETECCION'
set grid

set y2tics
set ytics nomirror
#set term post color solid
set term png
set out 'ejemplo3.png'

p 'ejemplo3.txt' u ($1*2):($2) t   ' Resultado de la simulacion ',
'ejemplo3teo.txt' u ($1):($2) t ' Resultado teorico '
set term x11
set out
rep

pause -1
```

A.4. Ejemplo 4: Estudio de la variación de la permeabilidad magnética

LITRANI

```
void ejemplo4(Double_t mu)

{
  char *name    = "EJEMPLO 4. La luz de un láser atraviesa un bloque de aire y
luego otro de índice de refracción real 1,5 y permeabilidad magnética variable.";
  char *listing = "La última cara del cristal es un fotocátodo.";
  char *upcom   = "La fuente es un láser pegado al bloque de aire ";
  char *downcom = "¿Cuántos fotones detecto?";
```

```
  gROOT->ProcessLine(".x InitLitrani.C(8,name,listing,upcom,downcom,kFALSE,kFALSE,kTRUE)");
  //gRandom3->SetSeed();

//1.- GEOMETRÍA

        const Double_t semilongitud_en_x1 = 0.5;
        const Double_t semilongitud_en_y1 = 0.5;
        const Double_t semilongitud_en_z1 = 0.5;

        const Double_t semilongitud_en_x2 = 0.5;
        const Double_t semilongitud_en_y2 = 0.5;
        const Double_t semilongitud_en_z2 = 0.5;

// 2.- MATERIALES

        TOpticMaterial *cristal;
        cristal = new TOpticMaterial("cristal","Un cubo de algo",kFALSE,mu,400.0);
        cristal->IsIsotropic(1.50);

        TOpticMaterial *aire;
        aire = new TOpticMaterial("aire","Un cubo de aire",kFALSE,1.0,10000.0);
        aire->IsIsotropic(1.00);

              TRevetment *totabsorbing;
       totabsorbing = new TRevetment("TotAbsorbing","Totally absorbing revetment",
"none",1.0,5,0.5,1.0,1.0,90.0);

 // 3.- FORMAS

         TSBRIK *laserbrick;
         laserbrick = new TSBRIK("laserbrick","Bloque de aire con láser","aire","none",
semilongitud_en_x1,semilongitud_en_y1,semilongitud_en_z1);

         TSBRIK *detectorbrick;
         detectorbrick = new TSBRIK("detectoribrick","Cristal","cristal","none",
semilongitud_en_x2,semilongitud_en_y2,semilongitud_en_z2);

   detectorbrick->fSuppl->SetSurfDet("detector","Detector de superficie",1,"none");

// 4.- ORDENAR LOS ELEMENTOS DEL MONTAJE

    TSNode *nodo1;//Declaro el nodo1 como un elemento de la clase TSNode()
    nodo1 = new TSNode("nodo1","Nodo asociado al láser",laserbrick);
    nodo1->cd();

    TSNode *nodo2;
    nodo2 = new TSNode("nodo2","Nodo asociado al cristal",detectorbrick,1.0,0.0,0.0 );

// 5.- CONTACTOS

 TContact *contacto;
 contacto = new TContact("laser-detector","Contacto entre el láser y el cristal con detector",
"laserbrick","detectoribrick",identical,"none");

// 4.- FUENTES

    TSpontan *laser;
    laser = new TSpontan("laser","laser","laserbrick",0.0,0.0,0.0,600.0);
    laser->DefineCradle("laserbrick",point);
    laser->DefineBeam(flat,0.0,90.0,0.0);
    for (Int_t i=1;i<=100;i++) {
    laser->Gen(i,100,-2.0,true,false);

  }

cout << "fNbPart:    " << gGs->fNpGener   << endl;//Nº total de fotones generados
cout << "fNpSeen:    " << gGs->fNpSeen    << endl;//Nº de fotones detectados
cout << "fNpLossAny: " << gGs->fNpLossAny << endl;//Nº de fotones perdidos por alguna razón.
cout << "fNpAbsMat:  " << gGs->fNpAbsMat  << endl;//Nº de fotones absorbidos en los materiales.
cout << "fNpAbsRvt:  " << gGs->fNpAbsRvt  << endl;//Nº de fotones perdidos en el revesitmiento.
cout << "fNpOutSide: " << gGs->fNpOutSide << endl;//Nº de fotones que abandonan el montaje????
cout << "fNpAbnorm:  " << gGs->fNpAbnorm  << endl;//Nº de fotones destruidos antes de generarse
```

```
cout << "fNpLossQE:  " << gGs->fNpLossQE  << endl;//Nº de fotones perdidos porque el detector no es perfecto
  exit();

  }
```

SH

```
#!/bin/tcsh

  echo "" > hola
  echo "" > hola2
  echo "" > hola3
  echo "" > hola4
  echo "" > hola5
  echo "" > hola6
  echo "" > hola7

    for mu in 0.1 0.20 0.30 0.40  0.50 0.60  0.70 0.80 0.90 1.00 1.10
1.20 1.30 1.40  1.50 1.60 1.70 1.80 1.90 2.00 2.10 2.20 2.30 2.40 2.50
2.60 2.70 2.80 2.90 3.00 3.10 3.20 3.30 3.40 3.50 3.60 3.70 3.80 3.90
4.00 4.10 4.20 4.30 4.40 4.50 4.60 4.70 4.80 4.90 5.00 5.1 5.2 5.3 5.4
5.5 5.6 5.7 5.8 5.9 6.0
do
    echo running $mu
    echo $mu >> hola2
    command='litrani ejemplo4.C('$mu')'
    $command > output
    cat output | grep fNpSeen  | sed 's/fNpSeen://' >> hola
    cat output | grep fNpLossQE  | sed 's/fNpLossQE://' >> hola3
    cat output | grep fNpAbsRvt  | sed 's/fNpAbsRvt://' >> hola4
    cat output | grep fNpAbsMat  | sed 's/fNpAbsMat://' >> hola5
    cat output | grep fNbPart  | sed 's/fNbPart://' >> hola6
    cat output | grep fNpOutSide  | sed 's/fNpOutSide://' >> hola7
   done

  paste hola2 hola hola3 hola4 hola5 hola6 hola7 > ejemplo4.txt
  gnuplot ejemplo4.gnu
```

A.5. Ejemplo 5: Estudio de la variación del índice de refracción

Omitiremos estos programas puesto que son muy similares a los del ejemplo 4.

A.6. Ejemplo 6: Reflexión en metales

LITRANI

```
void ejemplo6(Double_t index, Double_t mu)

{
  char *name    = "EJEMPLO 6. La luz de un láser atraviesa un bloque de aire
 y se refleja en un metal de índice de refracción imaginario y permeanilidad
 magnétoca variables.";
  char *listing = "La primera cara del bloque de aire es un fotocátodo.";
  char *upcom   = "La fuente es un láser pegado al bloque de aire ";
  char *downcom = "¿Cuántos fotones detecto?";

  gROOT->ProcessLine(".x InitLitrani.C(8,name,listing,upcom,downcom,kFALSE,kFALSE,kTRUE)");

//1.- GEOMETRÍA

        const Double_t semilongitud_en_x = 0.5;
        const Double_t semilongitud_en_y = 0.5;
        const Double_t semilongitud_en_z = 0.5;
```

```
// 2.- MATERIALES

        TOpticMaterial *aire;
        aire = new TOpticMaterial("aire","Un cubo de aire",kFALSE,1.0,10000.0);
        aire->IsIsotropic(1.00);

               TRevetment *metal;
        metal = new TRevetment("metal","Recubrimiento metálico","none",
                               0.0,1.0,index,mu,0.0,90.0);

 // 3.- FORMAS

         TSBRIK *laserbrick;
         laserbrick = new TSBRIK("laserbrick","Bloque de aire con láser","aire","metal",
semilongitud_en_x,semilongitud_en_y,semilongitud_en_z);

         laserbrick->fSuppl->SetSurfDet("detector","Detector de superficie",3,"none");

// 4.- ORDENAR LOS ELEMENTOS DEL MONTAJE

    TSNode *nodo1;//Declaro el nodo1 como un elemento de la clase TSNode()

    nodo1 = new TSNode("nodo1","Nodo asociado al láser",laserbrick);
    nodo1->cd();

// 5.- CONTACTOS

// 4.- FUENTES

    TSpontan *laser;
    laser = new TSpontan("laser","laser","laserbrick",0.0,0.0,0.0,600.0);
    laser->DefineCradle("laserbrick",point);
    laser->DefineBeam(flat,0.0,90.0,0.0);
    for (Int_t i=1;i<=100;i++) {
    laser->Gen(i,100,-2.0,true,false);

  }

cout << "fNbPart:    " << gGs->fNpGener   << endl;//Nº total de fotones generados
cout << "fNpSeen:    " << gGs->fNpSeen    << endl;//Nº de fotones detectados
cout << "fNpLossAny: " << gGs->fNpLossAny << endl;//Nº de fotones perdidos por alguna razón.
cout << "fNpAbsMat:  " << gGs->fNpAbsMat  << endl;//Nº de fotones absorbidos en los materiales.
cout << "fNpAbsRvt:  " << gGs->fNpAbsRvt  << endl;//Nº de fotones perdidos en el revesitmiento.
cout << "fNpOutSide: " << gGs->fNpOutSide << endl;//Nº de fotones que abandonan el montaje
cout << "fNpAbnorm:  " << gGs->fNpAbnorm  << endl;//Nº de fotones destruidos antes de generarse
cout << "fNpLossQE:  " << gGs->fNpLossQE  << endl;//Nº de fotones perdidos porque el detector no es perfecto
  exit();

  }
```

SH

```
#!/bin/tcsh

  echo "" > hola
  echo "" > hola2
  echo "" > hola3
  echo "" > hola4
  echo "" > hola5
  echo "" > hola6
  echo "" > hola7

    for index in 1.00 1.10 1.20 1.30 1.40  1.50 1.60 1.70 1.80 1.90
 2.00 2.10 2.20 2.30 2.40 2.50 2.60 2.70 2.80 2.90 3.00 3.10 3.20
3.30 3.40 3.50 3.60 3.70 3.80 3.90 4.00 4.10 4.20 4.30 4.40 4.50 4.60
 4.70 4.80 4.90 5.00 5.1 5.2 5.3 5.4 5.5 5.6 5.7 5.8 5.9 6.0
    do

    for mu in 0.1 0.20 0.30 0.40  0.50 0.60  0.70 0.80 0.90 1.00 1.10
 1.20 1.30 1.40  1.50 1.60 1.70 1.80 1.90 2.00 2.10 2.20 2.30 2.40 2.50
 2.60 2.70 2.80 2.90 3.00 3.10 3.20 3.30 3.40 3.50 3.60 3.70 3.80 3.90
 4.00 4.10 4.20 4.30 4.40 4.50 4.60 4.70 4.80 4.90 5.00 5.1 5.2 5.3
```

```
5.4 5.5 5.6 5.7 5.8 5.9 6.0
    do
    echo running $index $mu
    echo $index $mu >> hola2
    command='litrani ejemplo6.C('$index,$mu')'
    $command > output
    cat output | grep fNpSeen  | sed 's/fNpSeen://' >> hola
    cat output | grep fNpLossQE  | sed 's/fNpLossQE://' >> hola3
    cat output | grep fNpAbsRvt  | sed 's/fNpAbsRvt://' >> hola4
    cat output | grep fNpAbsMat  | sed 's/fNpAbsMat://' >> hola5
    cat output | grep fNbPart  | sed 's/fNbPart://' >> hola6
    cat output | grep fNpOutSide  | sed 's/fNpOutSide://' >> hola7
   done

done

  paste hola2 hola hola3 hola4 hola5 hola6 hola7 > ejemplo6.txt
  gnuplot ejemplo6.gnu
```

GNU DISPLAY

```
set ylabel ' Counts '
set xlabel ' Indice de refraccion complejo '
set zlabel ' Permeabilidad magnetica '
set title 'FOTONES DETECTADOS  versus INDICE DE REFRACCION COMPLEJO Y PERMEABILIDAD MAGNETICA'
set grid
set ytics nomirror
set term png


set contour both
set out 'ejemplo6.png'

sp 'ejemplo6.txt' u 1:2:3 t   ' Resultado de la simulacion ', 'ejemplo6.txt' u 1:2:
((10000-$6)*((1-$2)**2+($1)**2)/((1+$2)**2+($1)**2)) t ' Resultado teorico '
set term x11
set out
rep

pause -1
```

Programas empleados en el capítulo 5

Dada la gran cantidad de programas similares empeados en este capítulo, incorporaremos solamente una muestra reducida de ellos para comprender su funcionamiento.

RECUBRIMIENTOS

```
  TRevetment *teflon;
  revestimiento = new TRevetment("Revestimiento","Revestimiento","none"
,0.00, 1.32,0.00001,0.83,0.0,90.0);

  TRevetment *totabsorbing;
  totabsorbing = new TRevetment("TotAbsorbing","Totally absorbing revetment","none",
1.0,0.5,0.5,1.0,0.0,90.0);

  TRevetment *reflectortotal;
  reflectortotal = new TRevetment("reflectortotal","reflectortotal","none",
0.00, 0.0,-1.0,1.0,0.0,90.0);

  TRevetment *aluminio;
  aluminio = new TRevetment("aluminio","Revestimiento de aluminio","none",
0.00, 1.304,-7.479,1.56,0.0,90.0);

  TRevetment *cobre;
  cobre = new TRevetment("cobre","Revestimiento de cobre","none",
0.00, 0.27,-3.24,0.18,0.0,90.0);

  TRevetment *oro;
  oro = new TRevetment("oro","Revestimiento de oro","none",
0.00, 0.13,-3.16,0.22,0.0,90.0);

  TRevetment *niquel;
  niquel = new TRevetment("niquel","Revestimiento de niquel","none",
0.00, 1.92,-3.65,1.36,0.0,90.0);

  TRevetment *wolframio;
  wolframio = new TRevetment("wolframio","Revestimiento de wolframio","none",
0.00, 3.60,-2.89,1.25,0.0,90.0);

  TRevetment *difusor;
  difusor = new TRevetment("difusor","Revestimiento de teflón","none",
0.70, 1.32,0.00001,0.83,0.0,90.0);

  TFace *face10;
  TSupplShape *ps10 = gapbrick1->fSuppl;
  face10 = ps10->GetFace(0);
  face10 ->SetDepolished(70.00);

  TFace *face12;
  TSupplShape *ps12 = gapbrick1->fSuppl;
  face12 = ps12->GetFace(2);
  face12 ->SetDepolished(70.00);

  TFace *face13;
  TSupplShape *ps13 = gapbrick1->fSuppl;
```

```
  face13 = ps13->GetFace(3);
  face13 ->SetDepolished(70.00);

  TFace *face14;
  TSupplShape *ps14 = gapbrick1->fSuppl;
  face14 = ps14->GetFace(4);
  face14 ->SetDepolished(70.00);

  TFace *face15;
  TSupplShape *ps15 = gapbrick1->fSuppl;
  face15 = ps15->GetFace(5);
  face15 ->SetDepolished(70.00);

  TFace *face20;
  TSupplShape *ps20 = gapbrick2->fSuppl;
  face20 = ps20->GetFace(0);
  face20 ->SetDepolished(70.00);

  TFace *face22;
  TSupplShape *ps22 = gapbrick2->fSuppl;
  face22 = ps22->GetFace(2);
  face22 ->SetDepolished(70.00);

  TFace *face24;
  TSupplShape *ps24 = gapbrick2->fSuppl;
  face24 = ps24->GetFace(4);
  face24 ->SetDepolished(70.00);

  TFace *face25;
  TSupplShape *ps25 = gapbrick2->fSuppl;
  face25 = ps25->GetFace(0);
  face25 ->SetDepolished(70.00);
```

B.1. Desplazamiento longitudinal de la fuente

LITRANI

```
void centelladores02g(Double_t source_x)

{
  char *name    = "2 cristales de LSO separados por capa extrafina de aire";
  char *listing = "¿cuantos fotones llegan al detector en funcion de DOI";
  char *upcom   = "Cristales reucbiertos de oro";

  char *downcom = "Con capa de aire finisimo entre ellos";


  gROOT->ProcessLine(".x InitLitrani.C(8,name,listing,upcom,downcom,kFALSE,kFALSE,kTRUE)");

  gLit->SetPrintFreq(1000);

  const Double_t zero= 0.0;

// 1.- GEOMETRÍA


        const Double_t gap1_dx   = 1.0;
        const Double_t gap1_dy   = 0.5;
        const Double_t gap1_dz   = 0.5;

        const Double_t gap2_dx   = 1.0;
        const Double_t gap2_dy   = 0.5;
        const Double_t gap2_dz   = 0.5;

        const Double_t gap3_dx   = 0.0005;
        const Double_t gap3_dy   = 0.5;
        const Double_t gap3_dz   = 0.5;

        const Double_t gap4_dx   = 0.0005;
```

```
        const Double_t gap4_dy   = 0.5;
        const Double_t gap4_dz   = 0.5;

// 2.- MATERIALES

  TOpticMaterial *aire;
  aire = new TOpticMaterial("aire","aire",kFALSE,1.0,10000.0);
  aire->IsIsotropic(1.0);
  aire->SetAbsLa(1000);

  TOpticMaterial *LSO;
  LSO = new TOpticMaterial("LSO","LSO",kFALSE,1.0,21);
  LSO->IsIsotropic(1.82);

  TRevetment *oro;
  oro = new TRevetment("oro","Revestimiento de oro","none",
0.00, 0.13,-3.16,0.22,0.0,90.0);

  TRevetment *totabsorbing;
  totabsorbing = new TRevetment("TotAbsorbing","Totally absorbing revetment",
"none",1.0,0.5,0.5,1.0,0.0,90.0);

  TRevetment *reflectortotal;
  reflectortotal = new TRevetment("reflectortotal","reflectortotal",
"none",0.00, 0.0,-1.0,1.0,0.0,90.0);

// 3.- FORMAS

  TSBRIK *gapbrick1;
  gapbrick1 = new TSBRIK("gapbrick1","gapbrick1","LSO","oro",
gap1_dx,gap1_dy,gap1_dz);

  TSBRIK *gapbrick2;
  gapbrick2 = new TSBRIK("gapbrick2","gapbrick2","LSO","oro",
gap2_dx,gap2_dy,gap2_dz);

  TSBRIK *gapbrick3;
  gapbrick3 = new TSBRIK("gapbrick3","gapbrick3","aire","oro",
gap3_dx,gap3_dy,gap3_dz);

  TSBRIK *gapbrick4;
  gapbrick4 = new TSBRIK("gapbrick4","gapbrick4","LSO","TotAbsorbing",
gap4_dx,gap4_dy,gap4_dz);

  gapbrick2->fSuppl->SetSurfDet("pm","Phototube de eficiencia cuántica total",1,"");

// 4.- ORDENAR LOS ELEMENTOS DEL MONTAJE

  TSNode *node1;

  node1 = new TSNode("node1","node1",gapbrick3);

  TSNode *node2 = new TSNode("node2","node2",gapbrick2,gap2_dx+gap3_dx,zero,zero);

  TSNode *node3 = new TSNode("node3","node3",gapbrick1,-gap1_dx-gap3_dx,zero,zero);

  TSNode *node4 = new TSNode("node4","node4",gapbrick4,
gap3_dx+2*gap2_dx+gap4_dx,zero,zero);

// 5.- CONTACTOS

 TContact *contact13;
 contact13 = new TContact("cristal1-aire","cristal1-aire","gapbrick1",
"gapbrick3",identical);

 TContact *contact32;
 contact32 = new TContact("aire-cristal2","aire-cristal2","gapbrick3",
"gapbrick2",identical);

 TContact *contact24;
 contact24 = new TContact("cristal2-detector","cristal2-detector",
"gapbrick2","gapbrick4",identical);
```

```
// 6.- FUENTE

const Double_t wavelength = 600.0;
 if  (source_x>=0.0 &&  source_x<=2.0) {

 TSpontan *fuente1;
  fuente1 = new TSpontan("fuente1","fuente1","gapbrick2",
1.0-source_x,zero,zero,wavelength);

  for (Int_t i=1;i<=100;i++) {

    fuente1->Gen(i,100,-2.0,true,false);
   }
 }


  if (source_x>2.0 &&  source_x<=4.0){

    TSpontan *fuente2;
   fuente2 = new TSpontan("fuente2","fuente2","gapbrick1",
3.001-source_x,zero,zero,wavelength);
   for (Int_t i=1;i<=100;i++) {

    fuente2->Gen(i,100,-2.0,true,false);
   }
  }

  cout << "fNbPart:    " << gGs->fNpGener    << endl;
  cout << "fNpSeen:    " << gGs->fNpSeen    << endl;
  cout << "fNpLossAny: " << gGs->fNpLossAny << endl;
  cout << "fNpAbsMat:  " << gGs->fNpAbsMat  << endl;
  cout << "fNpAbsBef:  " << gGs->fNpAbsBef  << endl;
  cout << "fNpAbsRvt:  " << gGs->fNpAbsRvt  << endl;
  cout << "fNpOutSide: " << gGs->fNpOutSide << endl;
  cout << "fNpAbnorm:  " << gGs->fNpAbnorm  << endl;
  cout << "fNpTooLate: " << gGs->fNpTooLate << endl;
  cout << "fNpTooFew:  " << gGs->fNpTooFew  << endl;
  cout << "fNpLossAng: " << gGs->fNpLossAng << endl;
  cout << "fNpLossQE:  " << gGs->fNpLossQE  << endl;
  cout << "fPhweigth:  " << gGs->fPhweigth  << endl;


  exit();

}
```

Programas empleados en el capítulo 6

C.1. Fotones vistos en un centellador

LITRANI

```
void gammas04(void)

{
  char *fname   = "gammas04";
  char *listing = "Paso de fotones gamma por un centellador cilíndrico";
  char *upcom   = "La última cara es un detector de superficie";
  char *downcom = "¿Qué detecto?";
  gROOT->ProcessLine(".x InitLitrani.C(5,fname,listing,upcom,downcom,kFALSE,kFALSE,kTRUE)");
  gLit->SetPrintFreq(2000);

  TPhotoElecCompton *source;

  //1.- GEOMETRÍA Y CONSTANTES

  Int_t nRuns = 10;
  Double_t gammaE = 0.511;
  Double_t nPhotMeV = 10000;
  Int_t nBins = (int)(gammaE*nPhotMeV/2.0);
  const Double_t radio1 = 1.27;
  const Double_t semilong1_dz = 1.27;

  //2.- MATERIALES

  TOpticMaterial *nai;
  nai = new TOpticMaterial("nai","NaI",kFALSE,1.0,400.0,-1,-1,46.57,64,3.67);
  nai->IsIsotropic(1.85);
  nai->FluoComponent(410.0, 25.0, 0.01, 250.0);
  nai->FluoComponent(415.0, 11.3, 0.99, 230.0);
  nai->SetPhotMev(nPhotMeV);
  nai->SetXSectnPE("nai_cross_section");
  gLitPhys->SetLate(1000000000);

  TOpticMaterial* air = new TOpticMaterial("Air", "Air", kFALSE, 1.0, 10000.0);
  air->IsIsotropic(1.0);

  TRevetment* totabsorbing = new TRevetment("TotAbs", "Totally absorbing revetment",
 "none", 0.0, 2.0, 6.0, 1.0, 1.0);

  TRevetment *reflectortotal;
  reflectortotal = new TRevetment("reflectortotal","reflectortotal","none",0.00,0.0,
-1.0,1.0,0.0,90.0);

  TRevetment* al = new TRevetment("al", "Aluminium", "none", 0.0,"RIndexRev_Aluminium",
"IIndexRev_Aluminium",1.0);

  //3.- FORMAS

  TSCYL* centellador = new TSCYL("centellador","Centellador cilíndrico de NaI","nai",
"al",radio1,semilong1_dz);
```

```
  centellador->fSuppl->SetSurfDet("detector0", "Cara detectora", 0);
  centellador->fSuppl->GetFace(0)->SetDetNumber(0);

  //4.- ORDENAR LOS ELEMENTOS DEL MONTAJE

  TSNode *node1;
  node1=new TSNode("node1","node1",centellador);
  node1->SetLineColor(1);
  node1->SetLineWidth(2);
  node1->cd();

  //5.- CONTACTOS

  //6.- FUENTES

  T3Vector StartCasc(0.0, 0.0, 0.0);
  const Double_t Theta0 = 0.01;
  Double_t sTheta0,cTheta0;
  sTheta0 = TMath::Sin(Theta0);
  cTheta0 = TMath::Cos(Theta0);
  const Double_t Phi0 = 0.0;
  Double_t sPhi0,cPhi0;
  sPhi0 = TMath::Sin(Phi0);
  cPhi0 = TMath::Cos(Phi0);
  T3Vector AxisCasc(sTheta0*cPhi0,sTheta0*sPhi0,cTheta0);
  source = new TPhotoElecCompton("source","Fuente de fotones gamma","centellador",
gammaE,StartCasc,AxisCasc);

  TH1F* spectrum = new TH1F("spectrum", "spectrum", nBins, 0, nBins*1.0);

  for (Int_t i=0; i<nRuns; ++i){
    source->Gen(101+i,4);
    if (gCs->fNpSeen) {spectrum->Fill(gCs->fNpSeen)};

    }

  //7.- DIBUJO Y RESULTADOS

  gLit->BookCanvas();
  gTwoPad->SelectPad2();
  node1->Draw();
  gTwoPad->ChangePad();
  spectrum->Draw();
  gGp->SetTitle("Run number");
  gGp->Summary();
  gGs->DoStat();
  gLit->CloseFiles();

  cout << "fNbPart:    " << gGs->fNpGener   << endl;
  cout << "fNpSeen:    " << gGs->fNpSeen    << endl;
  cout << "fNpLossAny: " << gGs->fNpLossAny << endl;
  cout << "fNpAbsMat:  " << gGs->fNpAbsMat  << endl;
  cout << "fNpAbsBef:  " << gGs->fNpAbsBef  << endl;
  cout << "fNpAbsRvt:  " << gGs->fNpAbsRvt  << endl;
  cout << "fNpOutSide: " << gGs->fNpOutSide << endl;
  cout << "fNpAbnorm:  " << gGs->fNpAbnorm  << endl;
  cout << "fNpTooLate: " << gGs->fNpTooLate << endl;
  cout << "fNpTooFew:  " << gGs->fNpTooFew  << endl;
  cout << "fNpLossAng: " << gGs->fNpLossAng << endl;
  cout << "fNpLossQE:  " << gGs->fNpLossQE  << endl;
  cout << "fPhweigth:  " << gGs->fPhweigth  << endl;


}
```

TSPLINEFIT

```
TSplineFit* nai_cross_section(Bool_t todraw = kFALSE, Bool_t infile = kFALSE,
Bool_t firstinfile = kFALSE)
//
// Arguments:
//
```

```
//  todraw       : true if the fit is to be drawn
//  infile       : true if the fit is to be put in file SplineFitDB.rdb
//  firstinfile  : if true,  [BE CAREFUL: RISK OF LOOSING FILE SplineFitDB.rdb ]
//                 delete all fits present in file SplineFitDB.rdb and place
//                 this fit as the first fit in a new file SplineFitDB.rdb
//                 default false !!!
// All defaults for arguments correspond to the case where one calls this CINT
//macro from within a Litrani CINT macro to define a needed fit,  instead of trying
//to find it in the database file SplineFitDB.rdb using TSplineFit::FindFit()
//
//   Photo-Electric Effect Cross Section for nai
//
//   Source: NIST http://physics.nist.gov/PhysRefData/Xcom/Text/XCOM.html
//
{
  Int_t k1;
  Int_t k2 = -100;
  k1 = TClassTable::GetID("TSplineFit");
  if (k1<0) k2 = gSystem.Load("libSplineFit");
  const Int_t M = 74;
  Int_t i;
  TSplineFit *nai_cross_section;
  Double_t x[M] = { 0.001,  0.001017,  0.001035,  0.001053,  0.001072,  0.001072,
0.001072, 0.001072, 0.001072, 0.001072, 0.001286, 0.0015, 0.00175, 0.002, 0.0025,
 0.003, 0.0035, 0.004, 0.004279, 0.004557, 0.004557, 0.00463, 0.004702, 0.004777,
 0.004852, 0.004852, 0.004852, 0.004926, 0.005, 0.005094, 0.005188, 0.005188,
0.005594, 0.006, 0.007, 0.008, 0.009, 0.01, 0.0125, 0.015, 0.0175, 0.02, 0.025,
0.03, 0.03159, 0.03317, 0.03317, 0.03317, 0.03658, 0.04, 0.045, 0.05, 0.055, 0.06,
 0.07, 0.08, 0.09, 0.1, 0.125, 0.150, 0.175, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5,
 0.5055, 0.511, 0.5555, 0.6, 0.8, 1 };

  Double_t y[M] = { 602625.1, 581738.2, 560851.3, 540738.0, 521398.3, 521398.3,
 521398.3, 521398.3, 521398.3, 521398.3, 412322.4, 293963.5, 204227.2, 148528.9,
 85094.7, 54151.2, 36513.4, 25915.2, 21737.8, 18488.8, 18488.8, 49277.6, 47730.4,
46183.2, 44636.0, 44636.0, 44636.0, 57941.7, 56317.2, 53609.7, 51134.2, 51134.2,
48813.4, 41000.2, 27307.7, 19262.3, 14079.3, 10675.5, 5801.9, 3535.3, 2305.3, 1593.6,
850.9, 511.3, 441.7, 2305.3, 2305.3, 2305.3, 1787.0, 1407.9, 1036.6, 781.3, 600.3,
472.7, 308.7, 212.7, 152.4, 112.9, 59.9, 35.5, 22.8, 15.6, 8.4, 5.0, 3.3, 2.3, 1.7,
1.3, 1.3, 1.2, 1.0, 0.8, 0.4, 0.3 };
  nai_cross_section = new TSplineFit("PhotoEl_nai", "Photo-Electric Cross Section | nai",
18, M, x, y, 0.001,  1.4);
  nai_cross_section->SetSource("http://physics.nist.gov/PhysRefData/Xcom/Text/XCOM.html");
  nai_cross_section->SetMacro("nai_cross_section.C");
  nai_cross_section->SetXLabel("Gamma Energy [MeV]");
  nai_cross_section->SetVLabel("Cross Section x10-24 cm2");
  if (todraw) {
    nai_cross_section->SetDefaultLabels();
    nai_cross_section->DrawFit();
    nai_cross_section->Print();
  }
  if (infile) {
    if (firstinfile) nai_cross_section->UpdateFile(kTRUE);
    else             nai_cross_section->UpdateFile(kFALSE);
  }
  return nai_cross_section;
}
```

Índice de programas

Puesto que una de las finalidades de este trabajo es dejar abierta la posibilidad a cualquier otra persona de trabajar con LITRANI, bien sea sobre la base de los programas aquí expuestos o bien desde cualquier otro punto, consideramos conveniente poner a disposición de cualquier persona interesada los programas que se han escrito a lo largo de la realización del trabajo. El manejo y comprensión de dichos programas exige que su orden y organización sea conocido para el usuario. Es por ello que aquí explicitaremos la lista de programas y extensiones que se han expuesto en estas páginas.

D.1. Extensiones que aparecen en este texto

En la inmensa mayoría de los casos cada programa se implementa mediante varios subprogramas cuyas extensiones vamos a explicar en primer lugar, de modo que daremos por supuesto que cada uno de los apartados que vamos a exponer contiene dicho abanico de extensiones.

1. *.C: Código de fuente en lenguaje C++, que es el utilizado por LITRANI. Constituyen, en cada caso, el programa de LITRANI escrito para cada caso que se va a estudiar y que, por lo tanto, es el motor que los programas subsiguientes utilizarán para obtener resultados.

2. *.sh: Ejecutan los programas de LITRANI anteriores realizando varias iteraciones en las que se va modificando una o varias variables que se hayan dejado libres en el fichero inicial de LITRANI. Al correrlos generan un fichero *.txt con los resultados obtenidos ordenados en columnas.

3. *.txt: Archivos de texto en los que los programas *.sh escriben los resultados de las diversas iteraciones de cada programa de LITRANI.

4. *.gnu: Haciendo uso del programa GNUPlot toman los resultados grabados en los ficheros de texto para representarlos gráficamente. Al correrlos generan una imagen *.png con el mismo nombre.

5. *.png: Representaciones gráficas de los resultados obtenidos de las sucesivas implementaciones de un programa de LITRANI al modificar alguna variable.

D.2. Lista detallada de programas

Todos estos archivos están contenidos en la carpeta 'programas' del paquete.

D.2.1. Validación de problemas sencillos con LITRANI

1. Ejemplo 1: Comprobación de equivalencia de montajes.
   a) ejemplo1a (*.C): Estudia el la llegada de fotones a un detector tras atravesar un solo cristal con recubrimiento totalmente reflector. No hay ningún parámetro variable.
   b) ejemplo1b (*.C): Estudia el la llegada de fotones a un detector tras atravesar dos cristales sin separación con recubrimiento totalmente reflector. No hay ningún parámetro variable.
   c) ejemplo1c (*.C, *.sh, *.txt): Equivalente a ejemplo1a con diversas implementaciones para variar la semilla de números aleatorios.
   d) ejemplo1d (*.C, *.sh, *.txt): Equivalente a ejemplo1b con diversas implementaciones para variar la semilla de números aleatorios.
   e) ejemplo1 (*.gnu, *.png): Aúna los resultados de ejemplo1c.txt y ejemplo1d.txt en una gráfica comparativa de resultados.
2. Ejemplo 2: Reflectancia y transmitancia.
   a) ejemplo2a (*.C): Montaje para la detección de los fotones transmitidos. No hay ningún parámetro variable.
   b) ejemplo2b (*.C): Montaje para la detección de los fotones reflejados. No hay ningún parámetro variable.
3. Ejemplo 3: Estudio de la absorción en un dieléctrico.
   a) ejemplo3 (*.C, *.sh, *.txt, *.gnu, *.png): Mide el número de fotones detectados en función de la posición del detector. El archivo *.gnu compara los resultados simulados con los teóricos.
   b) ejemplo3teo (*.txt): Tabla de resultados teóricos para el montaje.
4. Ejemplo 4: Estudio de la variación de la permeabilidad magnética.
   a) ejemplo4 (*.C, *.sh, *.txt, *.gnu, *.png): Mide el número de fotones detectados en función de la permeabilidad magnética del dieléctrico. El archivo *.gnu compara los resultados simulados con los teóricos.
   b) ejemplo4teo (*.txt): Tabla de resultados teóricos para el montaje.
5. Ejemplo 5: Estudio de la variación del índice de refracción.
   a) ejemplo5 (*.C, *.sh, *.txt, *.gnu, *.png): Mide el número de fotones detectados en función del índice de refracción del dieléctrico. El archivo *.gnu compara los resultados simulados con los teóricos.
   b) ejemplo5teo (*.txt): Tabla de resultados teóricos para el montaje.
6. Ejemplo 6: Reflexión en metales.
   a) ejemplo6 (*.C, *.sh, *.txt, *.gnu, *.png): Estudia los fotones vistos al variar la parte compleja del índice de refracción y la permeabilidad magnética del recubrimiento metálico de manera simultánea.

D.2.2. Simulaciones en el interior de los cristales de LSO

A cada uno de los materiales que vayamos a utilizar en estas simulaciones le asociamos una letra. Las características de cada uno de los materiales se explica en el capítulo correspondiente. La correspondencia se explicita en la siguiente lista:

a.- Recubrimiento totalmente reflector.

b.- Recubrimiento totalmente absorbente.

c.- Teflón.

d.- Aluminio.

e.- Caras rugosas.

f.- Cobre.

g.- Oro.

h.- Níquel.

i.- Wolframio.

j.- Teflón difusor.

De este modo, puesto que necesitamos escribir un programa de LITRANI para cada uno de estos recubrimientos, añadiremos la letra correspondiente al material al final del título en cada uno de los montajes estudiados. Así las cosas, los programas empleados en esta sección son:

1. centelladores01 (.*C, *.sh, *.txt) (todos los materiales): Fotones vistos para un cristal de LSO de 2cmx1cmx1cm en el que la fuente se desplaza longitudinalmente.

2. centelladores02 (.*C, *.sh, *.txt) (todos los materiales): Fotones vistos para dos cristales de LSO de 2cmx1cmx1cm separados por una finísima capa de aire y en los que la fuente se desplaza longitudinalmente.

3. centelladores03 (.*C, *.sh, *.txt) (todos los materiales): Fotones vistos para dos cristales de LSO de 2cmx1cmx1cm separados por una fina capa de gel y en los que la fuente se desplaza longitudinalmente.

4. centelladores04 (.*C, *.sh, *.txt) (todos los materiales): Fotones vistos para un cristal de LSO de 7mmx1,4mmx1,4mm en el que la fuente se desplaza longitudinalmente.

5. centelladores05 (.*C, *.sh, *.txt) (todos los materiales): Fotones vistos para dos cristales de LSO de 7mmx1,4mmx1,4mm separados por una finísima capa de aire y en los que la fuente se desplaza longitudinalmente.

6. centelladores06 (.*C, *.sh, *.txt) (todos los materiales): Fotones vistos para dos cristales de LSO de 7mmx1,4mmx1,4mm separados por una fina capa de gel y en los que la fuente se desplaza longitudinalmente.

7. centelladores13 (.*C, *.sh, *.txt) (c, d, f): Fotones vistos para un cristal de LSO de 2cmx1cmx1cm en el que la fuente se desplaza en x, y, z.

8. centelladores14 (.*C, *.sh, *.txt) (c, d, f): Fotones vistos para dos cristales de LSO de 2cmx1cmx1cm separados por una finísima capa de aire y en los que la fuente se desplaza en x, y, z.
9. centelladores15 (.*C, *.sh, *.txt) (c, d, f): Fotones vistos para dos cristales de LSO de 2cmx1cmx1cm separados por una fina capa de gel y en los que la fuente se desplaza en x, y, z.
10. centelladores16 (.*C, *.sh, *.txt) (c, d, f): Fotones vistos para un cristal de LSO de 7mmx1,4mmx1,4mm en el que la fuente se desplaza x, y, z.
11. centelladores17 (.*C, *.sh, *.txt) (c, d, f): Fotones vistos para dos cristales de LSO de 7mmx1,4mmx1,4mm separados por una finísima capa de aire y en los que la fuente se desplaza en x, y, z.
12. centelladores18 (.*C, *.sh, *.txt) (c, d, f): Fotones vistos para dos cristales de LSO de 7mmx1,4mmx1,4mm separados por una fina capa de gel y en los que la fuente se desplaza en x, y, z.
13. centelladores_01 (*.gnu, *.png) (equivalente para todos los números anteriores): Representación de todos los resultados asociados a centelladores01 (equivalente para todos los números anteriores).

D.2.3. Simulación de los fotones gamma

1. gammas04 (*.C): Programa que simula la llegada de fotones gamma de 0.511 MeV a un centellador cilíndrico, así como la energía que en él dejan y los fotones secundarios que producen.
2. nai_cross_section (*.C): Programa que realiza la interpolación de los datos de la sección eficaz del NaI extraídos de las tablas del NIST.

Bibliografía

[1] W. S. C. Williams. "Nuclear and Particle Physics". *Oxford Science Publications*.

[2] W. R. Leo. "Thechniques for Nuclear and Particle Physics Experiments. A how-to approach". *Springer-Verlag*.

[3] Eugene Hecht. Alfred Zajac. "Óptica". *Fondo Educativo Interamericano*.

[4] Max Born. Emil Wolf. "Principles of Optics". *Cambridge University Press*.

[5] David R. Lide, editor-in.chief. "Handbook of Chemestry and Physics, 88º Edition".*CRC Press*.

[6] Glenn F. Knoll. Radiation detection and measurement, Second Edition".*John Wiley and Sons*.

[7] J.S. Huber, W.W. Moses, M.S. Andreaco, O. Petterson. .ª LSO scintillator array for a PET detector module DOI measurement". *IEEE Trans. Nucl. Sci., vol 48, pp. 684-688*, 2001.

[8] Catherine Michelle Pepin, Philippe Bérard, Roger Lecomte. .ªssessment of reflective separator films for small crystal arrays".

[9] NIST: http://physics.nist.gov/PhysRefData/Xcom/Text/XCOM.html