UNIVERSITAT POLITÈCNICA DE CATALUNYA

UNIVERSITAT DE BARCELONA

Proyecto Final de Estudios
MASTER
EN
INGENIERÍA BIOMÉDICA

EB

Diseño y optimización del aplicador de un acelerador tipo microtrón para radioterapia intraoperatoria

Barcelona, 18 de Julio del 2008

Autora: **Immaculada Martínez Rovira**
Directores: Josep Sempau, José M. Fernández-Varea

Realizado en: Institut de Tècniques Energètiques (UPC)

Resumen

La primera parte del trabajo trata del diseño y optimización del aplicador de un acelerador tipo microtrón de pista dedicado a radioterapia intraoperatoria. Después de una pequeña introducción sobre los elementos a evaluar y las características dosimétricas que se deben cumplir, se ha realizado un estudio exhaustivo del tipo de blindaje y láminas dispersoras a usar. Finalmente, se ha llegado a la conclusión de que un tubo de lucita y un tubo deslizante externo de latón es la mejor opción para mantener la transparencia del aplicador, evitar la contaminación por radiación y cumplir todas las normativas.

El objetivo de la segunda parte del trabajo es presentar una adaptación del programa de simulación Monte Carlo PENELOPE que permite la utilización de la base de datos del Organismo Internacional de la Energía Atómica (OIEA) sobre ficheros de espacios de fase de aceleradores lineales y unidades de cobaltoterapia. En primer lugar se describe la información contenida en dicha base de datos, su formato y el proceso de validación para su aceptación. En segundo lugar se detallan los pasos a seguir para que los usuarios de PENELOPE puedan generar y leer ficheros de espacios de fase en el formato del OIEA.

Índice general

Prefacio

He cursado la especialidad de investigación del Master en Ingeniería Biomédica (UB-UPC) con el fin de poder adquirir los conocimientos necesarios para investigar en el campo de la física médica; el presente trabajo pretende poner de manifiesto que he conseguido mi objetivo. Es por eso que estoy satisfecha de presentar este proyecto como el final de mi primera etapa como investigadora en el *Institut de Tècniques Energètiques* (*Universitat Politècnica de Catalunya*), la de aprendizaje.

El presente trabajo es una combinación de dos partes complementarias; la primera está basada en una aplicación de la simulación Monte Carlo en física médica y la segunda está relacionada con el desarrollo de nuevas herramientas para los usuarios de este tipo de simulaciones. Además, mi trabajo ha trascendido a la comunidad científica ya que ha dado lugar a un poster en un congreso de ámbito nacional y a una publicación científica, todavía en proceso de aceptación.

A lo largo del proyecto he aprendido mucho sobre la descripción teórica y la simulación del transporte de radiación aplicado a la física médica, el uso del programa de simulación Monte Carlo PENELOPE y su programa principal penEasy, la programación en Fortran, el manejo de Linux,... Creo que todos los conocimientos adquiridos a lo largo del desarrollo de este proyecto, ya sean prácticos o teóricos, me van a ser de gran ayuda para mi segunda etapa como investigadora, que estoy deseando empezar.

Introducción

Capítulo 1

La *Universitat Politècnica de Catalunya* (UPC), en colaboración con otros centros científicos, está llevando a cabo un proyecto para diseñar un microtrón de pista para radioterapia intraoperatoria.

La radioterapia intraoperatoria (*IntraOperative Radiation Therapy*, IORT) con electrones de alta energía es una modalidad de tratamiento que tiene como objetivo combinar los esfuerzos de dos disciplinas, cirugía y radioterapia, para incrementar las tasas de control tumoral local. Consiste en la administración, en el lecho quirúrgico, de una alta dosis de radiación en un cierto volumen tumoral delimitado por el radioterapeuta; esto permite disminuir las posibilidades de cometer errores en la estimación del volumen de tratamiento, además de proteger las estructuras anatómicas correspondientes a tejido normal mediante su separación o protección.

La utilización en IORT de equipos basados en aceleradores lineales para radioterapia externa tiene muchos problemas, al igual que los aceleradores móviles diseñados hasta el momento. Es por eso que la UPC se ha propuesto diseñar un microtrón de pista compacto, fiable y de bajo peso y coste para IORT.

En el presente capítulo se evaluarán las características de la fuente justo a la salida del microtrón y se estudiarán y optimizarán todos los elementos modificadores del haz a fin de tener un microtrón apto para IORT; todo el estudio se realizará con el programa de simulación Monte Carlo (MC) de transporte de radiación PENELOPE.

Capítulo 2

La simulación MC es un método extensamente utilizado para la simulación por computador de los componentes de los aceleradores lineales, la generación y colimación del haz y la interacción de los haces de radiación con el paciente u otros materiales. Este tipo de simulaciones suele dividirse en dos etapas con el fin de optimizar los tiempos de cálculo: primero se caracteriza el campo de radiación en un cierto plano, para lo cual se almacena el estado de las partículas en un fichero informático llamado fichero del espacio de fase (*Phase-Space File*) o PSF. Posteriormente se simula el transporte de las partículas almacenadas en el PSF a través de los modificadores del haz adicionales que haya y del objeto irradiado.

Para generar PSFs realistas es necesario disponer de información muy detallada sobre la geometría y composición de los elementos que constituyen los cabezales de tratamiento, así como de una amplia experiencia en el uso de programas MC y en las técnicas de validación más apropiadas. A fin de evitar la repetición del costoso cálculo de PSFs, el Organismo Internacional de la Energía Atómica (OIEA) ha creado una base de datos pública de PSFs, que contempla tanto aceleradores lineales como unidades de ^{60}Co.

El objetivo de este capítulo del trabajo es presentar unas nuevas subrutinas que, utilizadas conjuntamente con PENELOPE y penEasy (programa principal para PENELOPE), permiten al usuario final generar y leer PSFs en formato OIEA sin necesidad de conocer los detalles de los interfaces de las rutinas del OIEA ni desarrollar programas adicionales.

Aunque el objetivo final de las rutinas es permitir el uso de la base de datos del OIEA, el presente trabajo también se ha empleado para guardar los PSFs en el caso del microtrón. De esta forma, no es necesario repetir completamente las simulaciones para las diferentes configuraciones y el tamaño de los PSFs resultantes es razonable.

Capítulo 1

Diseño del aplicador de un microtrón de pista para radioterapia intraoperatoria

1.1. Microtrón de pista para radioterapia intraoperatoria

1.1.1. Radioterapia Intraoperatoria

La radioterapia intraoperatoria (*IntraOperative Radiation Therapy*, IORT) es una técnica terapéutica que permite la administración única de una alta dosis de radiación ionizante en tejido tumoral durante la cirugía con la finalidad de mejorar su control local. La gran ventaja de esta modalidad radioterapéutica es que permite realizar una demarcación visual y palpable del tumor, así como excluir físicamente del campo las estructuras más sensibles a la dosis, sea por desplazamiento o por protección de las mismas [1, 2, 3, 4]. La IORT permite combinar las ventajas de las técnicas quirúrgicas con las de la radioterapia en diferentes aspectos:

- Reduce la posibilidad de presencia de tumor residual en el lugar de la cirugía, eliminando los focos de tumor microscópico.
- Permite una mejor definición del volumen a irradiar, reduciendo al máximo los daños producidos en el tejido sano.

- Maximiza el efecto radiobiológico con una dosis única, alta y localizada.
- Optimiza el tiempo de cirugía combinada con radioterapia externa debido la irradiación inicial con IORT.

La IORT es una técnica usada en Europa, Asia y América desde hace más de 30 años y se ha observado que es muy eficaz en algunos tipos de cáncer (cáncer de mama, tumores en el tracto digestivo, cáncer pancreático, etc.), algunas veces combinado con radioterapia externa o con quimioterapia. En la Figura 1.1 vemos dos imágenes de tratamientos con IORT.

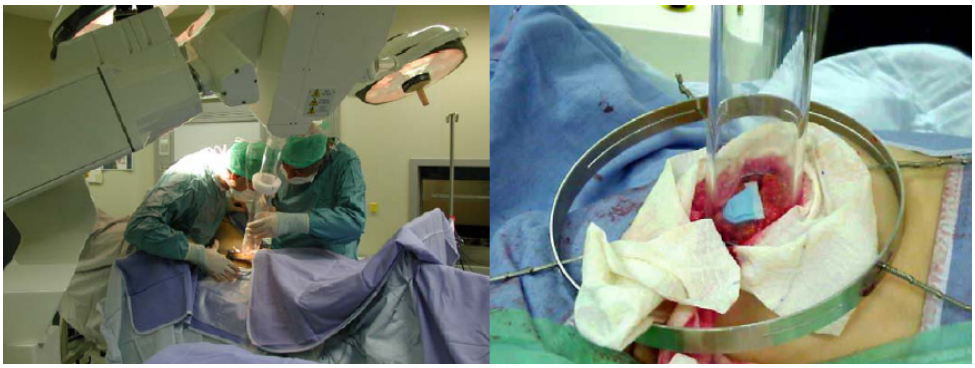

Figura 1.1: *Tratamiento con IORT de un cáncer pélvico y un cáncer de mama con un acelerador lineal móvil (extraído de Ref. [3]).*

Los requerimentos dosimétricos para IORT son los siguientes:

- Se requieren haces de electrones de energías entre 4 MeV y 12 MeV.
- Diferencia de 2 MeV entre haces de diferentes energías.
- La tasa de dosis absorbida debe ser del orden de 10 - 20 Gy/min.
- Tamaño de campo: de 5 cm a 15 cm.

El uso de IORT en sistemas convencionales de tratamiento para radioterapia externa tiene muchos inconvenientes ya que supone operar al paciente dentro del bunker del acelerador o bien transportar al paciente anestesiado de la sala de cirugía al bunker, irradiarlo y volver a la sala de cirugía. Para resolver este problema se han diseñado aceleradores de electrones móviles.

Existen diferentes aceleradores móviles en el mercado pero la idea de usar un microtrón de pista para IORT se propuso en Ref. [5]. Los microtrones de pista presentan ciertas ventajas respecto los otros sistemas actuales dedicados a IORT, en particular:

- El consumo de potencia es mucho menor.
- La energía de los electrones se determina con el campo magnético de los imanes curvadores con gran precisión, sin necesidad de ningún circuito de control especial; por lo tanto, no es necesaria una calibración diaria del haz. Eso permite reducir la dosis de radiación producida en el quirófano y aumentar el número de operaciones anual permitido.
- Con la misma energía del haz de salida, el peso y las dimensiones del microtrón de pista son muy inferiores.
- El coste de fabricación es mucho más bajo.
- El principio de cambio de la energía del haz de salida es muy simple (y no cambian los tamaños transversales del haz ni su espectro de energía).

1.1.2. Microtrón de pista

Se podría decir que un microtrón es un ciclotrón de electrones pequeño. Un ciclotrón es una cámara de alto vacío en la que mediante campos magnéticos y un sistema de radiofrecuencia es posible acelerar partículas elementales.

Básicamente, constan de una fuente de partículas y de dos imanes curvadores que permiten la recirculación del haz. Los imanes estan conectados a los bornes de un circuito eléctrico que crea una diferencia de potencial alterna con una cierta frecuencia; estas oscilaciones en frecuencia producen un campo eléctrico oscilante en la región entre ambos imanes. Si el campo magnético se ajusta de tal forma que el tiempo que necesita la partícula para recorrer la trayectoria semicircular dentro del imán sea igual al semiperiodo de las oscilaciones, el campo eléctrico se invertirá justo cuando los electrones acaben la mitad de un ciclo y los acelerará en la zona entre los imanes. Si repetimos el proceso más veces, las partículas van ganando energía a medida que recorren una

espiral creciente y finalmente salen proyectadas en línea recta del acelerador. En la Figura 1.2 podemos ver el principio de funcionamiento de forma gráfica.

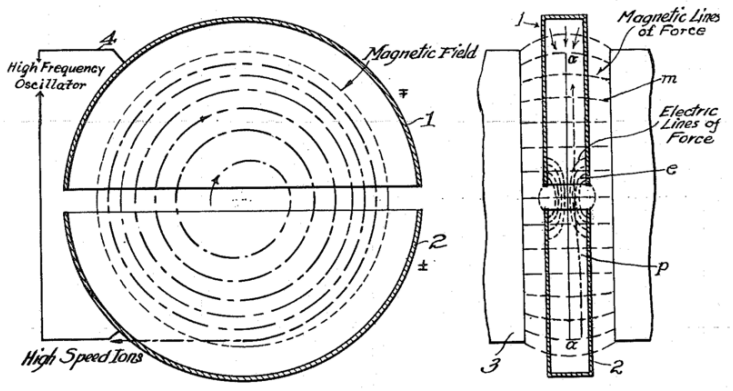

Figura 1.2: *Principio de funcionamiento de un ciclotrón (extraído de Ref. [6])*

En los microtrones, la frecuencia está en el rango de las microondas y el campo magnético de los imanes suele ser unas diez veces menor que en los ciclotrones.

En el caso particular de los microtrones de pista (*race-track microtron*, RTM), los electrones se aceleran con un acelerador lineal y éstos son readmitidos en el acelerador después de cada vuelta. Vemos un esquema general en la Figura 1.3.

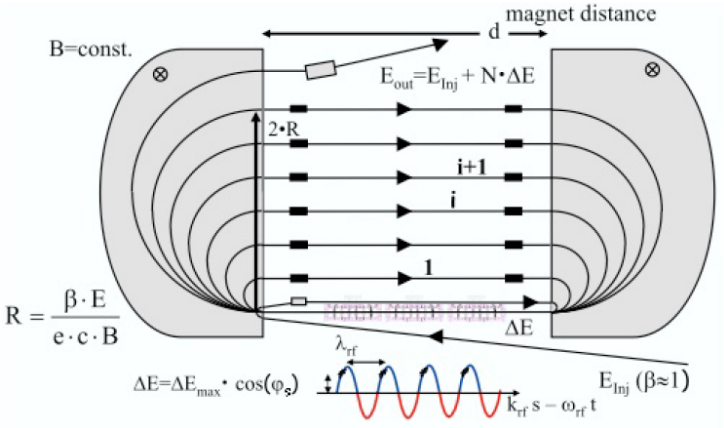

Figura 1.3: *Principio de funcionamiento de un microtrón de pista (extraído de Ref. [7])*

En la Figura 1.4 se puede ver un esquema del RTM de la UPC en proceso de diseño. Está compuesto por un cañón de electrones (fuente), el acelerador lineal que hemos comentado, los

imanes curvadores (compuestos por los polos principales, que curvan la órbita de los electrones, y los polos inversos, encargados de cerrar la órbita de 2 MeV), el imán de focalización (para evitar que el haz se abra debido a las interacciones electrostáticas entre electrones) y el imán de extracción (que desvía los electrones para que no vuelvan a entrar en el acelerador una vez tienen la energía deseada).

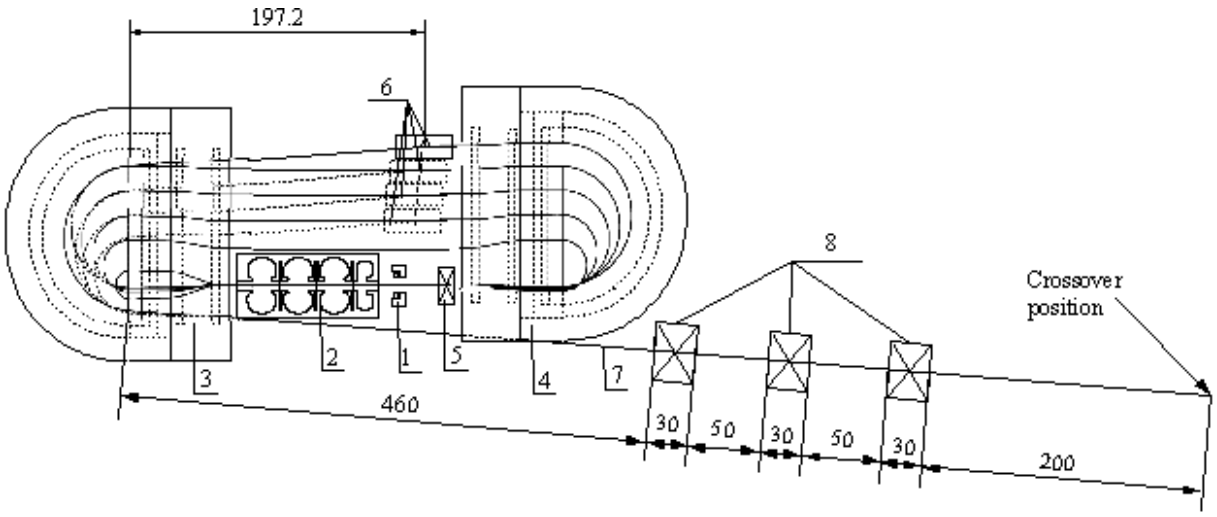

Figura 1.4: *Esquema del microtrón de pista de la UPC: (1) cañón de electrones, (2) acelerador lineal, (3) y (4) imanes curvadores, (5) imán de focalización, (6) imán de extracción, (7) haz de salida, (8) cuadrupolos del haz de salida (extraído de Ref. [8]).*

En la Figura 1.4 también se pueden ver los cuadrupolos del haz de salida, usados para evitar la desfocalización del haz en la salida del RTM. De todas formas, veremos que no son necesarios porque sin ellos podemos generar un campo de irradiación apto para IORT. Así, evitaremos tener unas dimensiones y peso mayores y un uso más complicado del RTM.

Para el funcionamiento del RTM se requiere, además del cabezal con el RTM (posicionado por un brazo robótico), de un sistema de generación de radiofrecuencia, una fuente de alimentación, una cámara de vacío, un estabilizador térmico, un sistema de control y posicionamiento, el aplicador y el blindaje (véase Figura 1.5).

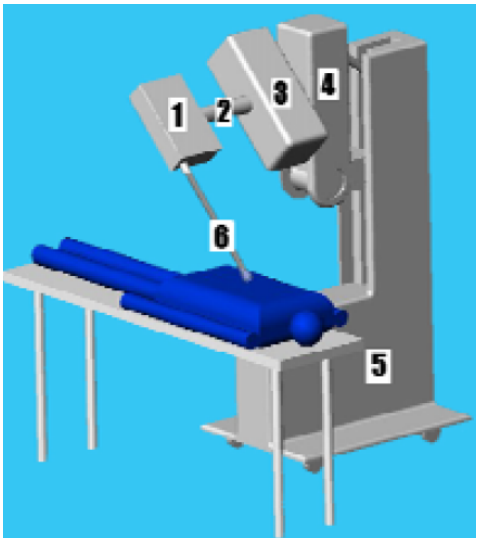

Figura 1.5: *Esquema general del microtrón de la UPC: (1) Cabezal con el RTM; (2) tubo de bombeo y la guía de onda; (3) unidad con la bomba de vacío y el bloque de alto voltaje del cañón de electrones; (4) unidad con la fuente de radiofrecuencia; (5) base con el modulador, elementos del sistema de refrigeración y el sistema de alimentación (extraído de Ref. [8]).*

La Tabla 1.1 muestra las características básicas del RTM de la UPC. Los parámetros han sido escogidos llegando a un compromiso entre el peso, las dimensiones y la efectividad.

Energías del haz de salida	6,8,10,12 MeV
Ganancia de energía por paso en linac	2 MeV
Potencia de radiofrecuencia en pulso	<800 kW
Duración del pulso	1 μs
Frecuencia de repetición	0-300 Hz
Intensidad del haz nominal	50-150 nA
Corriente máxima del haz	5 μA
Campo magnético del imán	0.8 T
Dimensiones del acelerador	50x20x11 cm^3
Masa del acelerador (sin blindaje)	$\sim$ 60 kg
Masa del equipo	350-400 kg
Altura máxima del equipo	$\sim$ 230 cm
Tasa de dosis generada	10-20 Gy/min
Dimensiones del campo de irradiación	Φ = 5,10,15 cm

Tabla 1.1: *Características principales del RTM de la UPC (extraído de Ref. [5, 8]).*

Una vez visto el esquema general del microtrón, nos vamos a centrar primero en las características de la fuente a la salida del microtrón y después en los modificadores del haz (láminas dispersonas, aplicador, blindaje) que debemos incorporar para tener un RTM apto para IORT.

1.2. El haz de salida

El haz de partículas cargadas representado en el espacio de fases se modeliza con una elipse [9, 10]. En el caso del microtrón tenemos un haz de electrones y por lo tanto su representación en el espacio de fases a una cierta distancia z vendrá dada por una elipse en el plano (x,x') y otra en el plano (y,y'). Las variables x e y representan coordenadas de posición (unidad SI: [m]), teniendo en cuenta que la dirección de z es la del haz y x e y son perpendiculares al eje z. Las variables x' y y' son las coordenadas de momento normalizadas:

$$x' = \tan\left(\frac{p_x}{p}\right) \approx \frac{p_x}{p} \qquad y' = \tan\left(\frac{p_y}{p}\right) \approx \frac{p_y}{p} \tag{1.1}$$

siendo p_x, p_y las componentes del momento lineal en la dirección x e y respectivamente y p el momento lineal total. Al haber poca dispersión se cumple que $p \approx p_z$ y podemos aproximar la tangente por su ángulo (unidad SI: [rad]).

En cinemática relativista, la relación que guarda el momento lineal total p con la energía total E es

$$E = \sqrt{p^2c^2 + m_e^2c^4} \tag{1.2}$$

y con la energía cinética K

$$K = \sqrt{p^2c^2 + m_e^2c^4} - m_ec^2 \tag{1.3}$$

siendo m_e la masa en reposo del electrón y c la velocidad de la luz en el vacío.

La ecuación usada en dinámica de haces para una elipse en el plano (x,x') con ejes no paralelos a los coordenados es

$$\gamma_x x^2 + 2\alpha_x x x' + \beta_x x'^2 = \epsilon_x \tag{1.4}$$

donde, de forma general, los parámetros α, β y γ se denominan *parámetros Courant-Snyder o Twiss*. γ está relacionada con α y β de la siguiente forma:

$$\beta\gamma - \alpha^2 = 1 \tag{1.5}$$

Vemos la relación gráfica de algunos de estos parámetros en la Figura 1.6

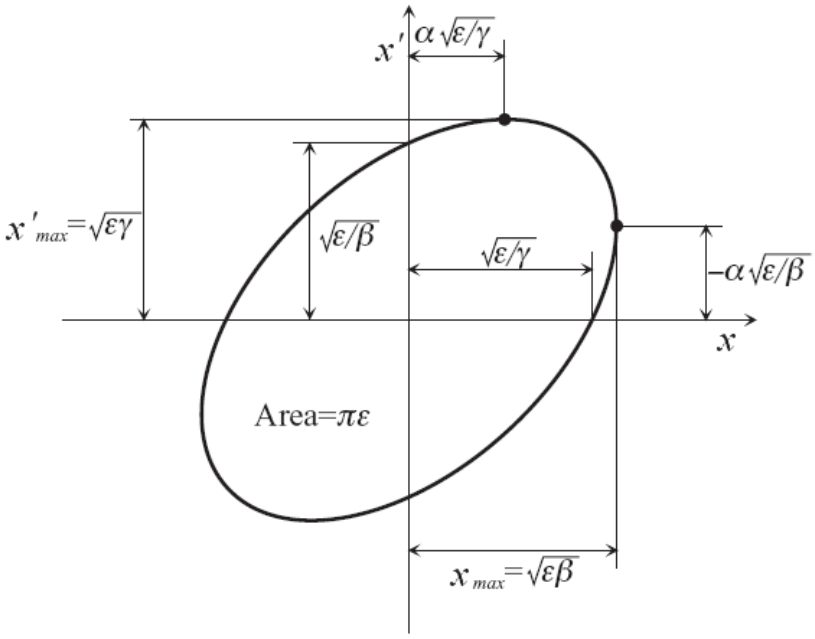

Figura 1.6: *Parámetros de la elipse en el espacio de fases (adaptado de Ref. [11]).*

El area ocupada por la elipse es $\pi\epsilon$, donde ϵ es la emitancia. Lo ideal sería que el area en una sección transversal fuera nula y que todas las partículas se dirigieran en la misma dirección; así, todas ocuparían el mismo punto en el espacio de fases transversal. Sin embargo, un haz de electrones tiene una cierta sección en los planos tranversales y sigue una distribución gaussiana.

Por convenio, la elipse a la que nos referiremos es la que incluye el 68 % de las partículas. En la Figura 1.7 vemos un ejemplo de cómo es el haz real a la salida de un RTM y cómo sigue una distribución gaussiana.

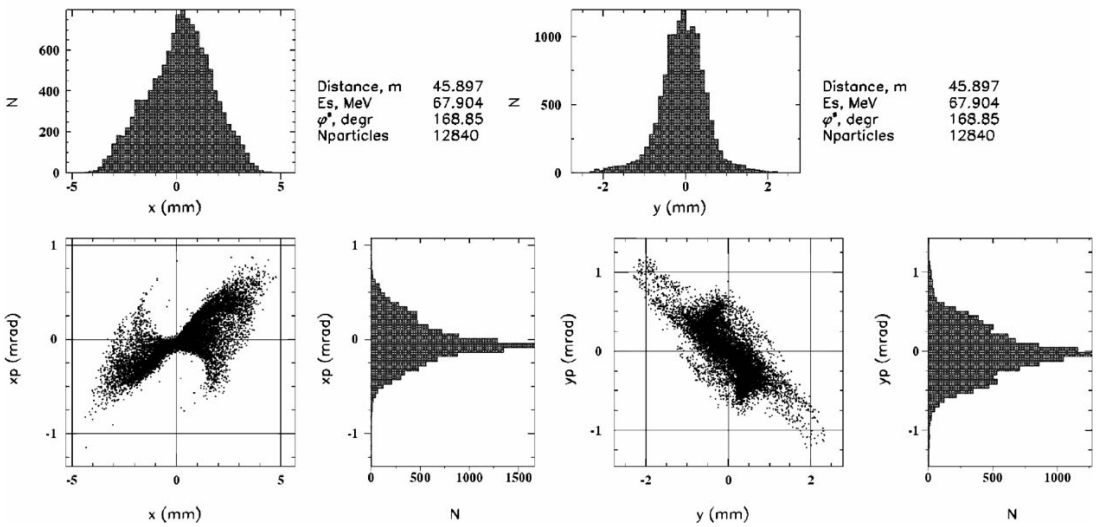

Figura 1.7: *Salida de un RTM en el espacio de fases (extraído de Ref. [12]).*

El código RTMTRACE [13] simula la dinámica de haces recirculando en aceleradores y nos proporciona los parámetros de las elipses en el espacio de fases. La Tabla 1.2 muestra los parámetros en el caso del RTM de la UPC.

p [MeV/c]	K [MeV]	α_x	β_x [mm/mrad]	γ_x [mrad/mm]	ϵ_x [mm.mrad]	α_y	β_y [mm/mrad]	γ_y [mrad/mm]	ϵ_y [mm.mrad]
6.51	6.02	-1.90	2.57	1.79	1.010	-7.59	4.79	12.24	0.272
8.56	8.06	-0.39	1.43	0.81	0.970	-9.05	6.67	12.43	0.326
10.6	10.10	-0.65	1.01	1.41	0.809	-8.76	6.17	12.60	0.378
12.6	12.10	-1.26	1.81	1.43	0.735	-10.3	8.56	12.51	0.543

Tabla 1.2: *Parámetros de las elipses en el espacio de fases del haz de electrones en la salida (Datos cedidos por el Dr. Yuri Kubyshin).*

A partir de ahora, nos centraremos en el plano (x,x') (en (y,y') será equivalente). Para encontrar los parámetros naturales de la elipse debemos aplicar una rotación de un cierto ángulo ϕ (ver Figura 1.8).

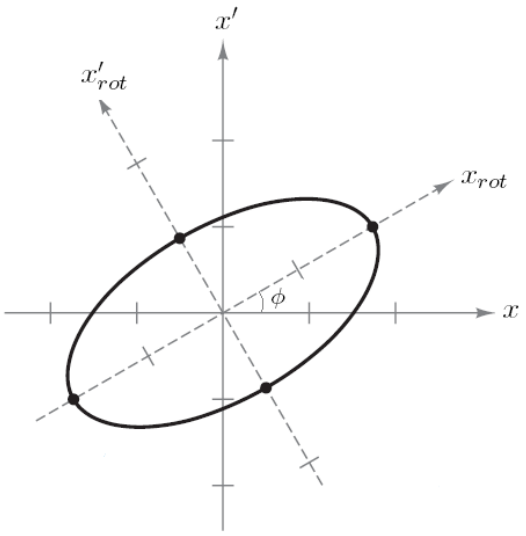

Figura 1.8: *Rotación de la elipse en el espacio de fases.*

Para simplificar, primero haremos un cambio de variable: $x = \lambda\bar{x}$, $x' = \mu\bar{x}'$, donde $\lambda = 1$ mm y $\mu = 1$ mrad, de tal forma que las nuevas variables sean adimensionales. Ahora, aplicamos la matriz de rotación [14] con ángulo de rotación ϕ:

$$\begin{pmatrix} \bar{x} \\ \bar{x}' \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \bar{x}_{rot} \\ \bar{x}'_{rot} \end{pmatrix} \tag{1.6}$$

La ecuación final de la elipse centrada en los ejes (x_{rot},x'_{rot}) tendrá la forma

$$\frac{\bar{x}^2_{\mathrm{rot}}}{{a_x}^2} + \frac{\bar{x}'^2_{\mathrm{rot}}}{{b_x}^2} = 1 \tag{1.7}$$

donde los parámetros a_x y b_x son los semiejes de la elipse. Podremos determinar el ángulo de rotación ϕ y los valores de a_x y b_x introduciendo la Ecuación (1.6) en (1.4) y comparando con (1.7). El resultado de hacer todo el desarrollo es

$$\tan 2\phi = \frac{2\alpha_x\lambda\mu}{\gamma_x\lambda^2 - \beta_x\mu^2} \tag{1.8}$$

$$\frac{1}{a_x^2} = \frac{1}{\epsilon_x}\left[\gamma_x\lambda^2(\cos\phi)^2 + 2\alpha_x\lambda\mu\cos\phi\sin\phi + \beta_x\mu^2(\sin\phi)^2\right]$$
$$\frac{1}{b_x^2} = \frac{1}{\epsilon_x}\left[\gamma_x\lambda^2(\sin\phi)^2 - 2\alpha_x\lambda\mu\cos\phi\sin\phi + \beta_x\mu^2(\cos\phi)^2\right] \quad (1.9)$$

En la Tabla 1.3 se presentan los resultados obtenidos para cada energía.

p [MeV/c]	ϕ_x [rad]	a_x	b_x	ϕ_y	a_y	b_y
6.51	0.685	2.040	0.495	-0.557	0.127	2.148
8.56	0.448	1.253	0.774	-0.631	0.131	2.492
10.6	-0.637	0.654	1.236	-0.610	0.142	2.660
12.6	0.710	1.458	0.504	-0.691	0.161	3.379

Tabla 1.3: *Parámetros de las elipses rotadas en el espacio de fases.*

1.3. Definición de la fuente en penEasy

Nuestro objetivo es transportar las partículas de la salida del RTM al paciente. Para ello, utilizaremos el paquete de subrutinas de simulación Monte Carlo de transporte de radiación PENELOPE [15]. Como paquete de subrutinas, requiere de un programa principal para funcionar: penEasy [16]. En el Apéndice A se encuentran detalles del funcionamiento básico de la simulación Monte Carlo, de PENELOPE y de penEasy.

1.3.1. Sorteo de una distribución normal bidimensional

Hemos visto que el haz de salida se puede modelizar como dos gaussianas bidimensionales en el espacio de fases. Así pues, para incorporar la fuente dentro de un programa de simulación MC, tenemos que generar números aleatorios que sigan la distribución de probabilidad de esta fuente. Hablaremos de variables genéricas x e x' y luego nos centraremos en nuestro caso. Para una distribución normal con valor medio 0 y varianza 1, la función de densidad de probabilidad es

$$p(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{1.10}$$

En dos dimensiones

$$p(x, x') = \frac{1}{2\pi} \exp\left(-\frac{x^2 + x'^2}{2}\right) \tag{1.11}$$

Si hacemos un cambio de coordenadas de cartesianas a polares $x = r\cos\theta$ y $x' = r\sin\theta$, la relación entre la distribución de probabilidad $p(x, x')$ y $p(r, \theta)$ será la siguiente

$$p(r, \theta)\,\mathrm{d}r\,\mathrm{d}\theta = p(x, x')\,\mathrm{d}x\,\mathrm{d}x' = \frac{1}{2\pi} \exp\left(-\frac{r^2}{2}\right) r\,\mathrm{d}r\,\mathrm{d}\theta \tag{1.12}$$

donde podemos ver dos partes diferenciadas, una corresponde a la función de distribución de probabilidad de r y la otra a la de θ

$$p_r(r, \theta) = r \exp\left(-\frac{r^2}{2}\right) \qquad p_\theta(r, \theta) = \frac{1}{2\pi} \tag{1.13}$$

La función de distribución de probabilidad acumulada $\mathcal{P}(x) = \int_{x_{\min}}^{x} p(x')\,\mathrm{d}x'$ para las dos variables es

$$\mathcal{P}_r(r, \theta) = 1 - \exp\left(-\frac{r^2}{2}\right) \qquad \mathcal{P}_\theta(r, \theta) = \frac{1}{2\pi}\theta \tag{1.14}$$

Utilizando el método de la transformada inversa (ver Apéndice A), $\mathcal{P}(x) = \xi$, donde ξ es un número aleatorio distribuido uniformemente entre 0 y 1. Si se invierte la Ecuación (1.14), podemos generar r y θ con su correspondiente distribución de probabilidad

$$r = \sqrt{-2\ln\xi_1} \qquad \theta = 2\pi\xi_2 \tag{1.15}$$

Si deshacemos el cambio de coordenadas y seguimos los mismos pasos para x' obtenemos

$$x = \sqrt{-2\ln\xi_1}\cos(2\pi\xi_2)$$
$$x' = \sqrt{-2\ln\xi_1}\sin(2\pi\xi_2) \tag{1.16}$$

Hasta ahora hemos considerado que la media y la varianza de la variable son $\langle x\rangle = 0$ y $\mathrm{var}(x) = 1$. Si no es el caso, con la transformación (1.17) podemos generar una nueva variable normalmente distribuida con $\langle x\rangle = m$ y $\mathrm{var}(x) = \sigma$

$$X = m_1 + \sigma_1 x$$
$$X' = m_2 + \sigma_2 x' \tag{1.17}$$

En este caso, la función de distribución de probabilidad para X (y también para X') es la que sigue

$$P(X) = p(x)\frac{\mathrm{d}x}{\mathrm{d}X} = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(X-m)^2}{2\sigma^2}\right) \tag{1.18}$$

La media m nos indica el centro de la gaussiana y la desviación estándar σ nos da una medida de la dispersión. Veamos la probabilidad que la variable X se encuentre entre $-\sigma$ y σ [6] es

$$p\left(X\in(-\sigma,\sigma)\right) = \int_{-\sigma}^{\sigma} P(X)\,\mathrm{d}X = 2\int_{0}^{\sigma} P(X)\,\mathrm{d}X = \mathrm{erf}\left(\frac{x-m}{\sigma\sqrt{2}}\right) = 68\,\% \tag{1.19}$$

donde erf es la función error y se encuentra tabulada en Ref. [17]. Si ahora nos centramos en nuestro caso, tenemos que $\bar{x}_{\mathrm{rot}} = X$ y $\bar{x}'_{\mathrm{rot}} = X'$ con una media m=0 porque tenemos las elipses centradas en (0,0) y una desviación estándar σ igual a los valores de los semiejes de la Tabla 1.3. Es decir,

$$\bar{x}_{\mathrm{rot}} = a_x\sqrt{-2\ln\xi_1}\cos(2\pi\xi_2)$$
$$\bar{x}'_{\mathrm{rot}} = b_x\sqrt{-2\ln\xi_1}\sin(2\pi\xi_2) \tag{1.20}$$

Aplicando la matriz de rotación (Ecuación (1.6)) tenemos que

$$\begin{aligned} \bar{x} &= \bar{x}_{\text{rot}} \cos\phi - \bar{x}'_{\text{rot}} \sin\phi \\ \bar{x}' &= \bar{x}_{\text{rot}} \sin\phi + \bar{x}'_{\text{rot}} \cos\phi \end{aligned} \tag{1.21}$$

Y finalmente, recuperamos la dimensionalidad de nuestras variables: $x = \lambda\bar{x}$, $x = \mu\bar{x}'$.

Las Figuras 1.9 y 1.10 presentan el resultado gráfico para el caso de 12 MeV.

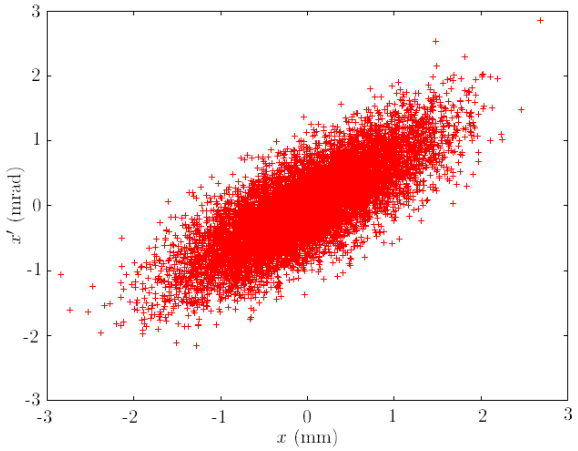

Figura 1.9: *Representación de la fuente en el espacio de fases (x, x') a la salida del microtrón.*

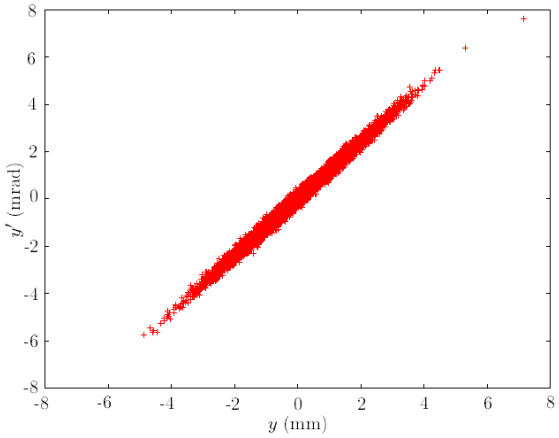

Figura 1.10: *Representación de la fuente en el espacio de fases (y, y') a la salida del microtrón.*

1.3.2. Implementación en penEasy

Una vez detallado el formalismo matemático, se ha creado una nueva fuente dentro de penEasy (se puede encontrar información detallada de la estructura de penEasy en el Apéndice A). Vemos los parámetros que se requieren en el fichero de entrada para esta nueva fuente:

```
[RTM SOURCE]
(ON )                          STATUS (ON or OFF)
 1                             PARTICLE TYPE (1=electron, 2=photon, 3=positron)
 Moment(eV/c) Probability      MOMENTUM SPECTRUM
```

```
 12.6e6      1                  Channel 1 Prob 1
 12.6e6      -1
 0.000e0  0.000e0               MEAN X X'
 1.458e0  0.504e0               SIGMA X X'
 0.000e0  0.000e0               MEAN Y Y'
 0.161e0  3.379e0               SIGMA Y Y'
 61.0000e0                      Z SOURCE
 -1                             Z DIRECTION
 0.710                          ROTATION ANGLE X X'
 -0.691                         ROTATION ANGLE Y Y'
 0                              MATERIAL (0=DON'T CARE)
[END OF RTM SOURCE]
```

- Tipo de partícula.
- El momento lineal p (posteriormente el programa lo transforma a energía cinética siguiendo la Ecuación (1.3)). Consideraremos que la energía no tiene dispersión ya que el espectro es aproximadamente monocromático. Es por eso que vemos que sólo existe un canal de momento.
- La media (=0 para todos los casos) y la desviación estándar de las gaussianas en x, x', y, y' (parámetros de la Tabla 1.3).
- Posición en z de la fuente (más adelante veremos el esquema de la geometría).
- Ángulo de rotación ϕ para la elipse en el plano (x, x') y la elipse en el plano (y, y').
- Es posible especificar en qué material queremos generar las partículas que vamos a simular. Si = 0 la partícula se acepta sea cual sea el material donde esté localizada.

Una vez el programa lee todos los parámetros de entrada, se llama dos veces a la subrutina *gauss2* (con parámetros de entrada: media, desviación estándar y ángulo de rotación) y nos devuelve la posición y dirección de movimiento de una partícula de la fuente en x ($xsrc$,$usrc$) y en y ($ysrc$,$vsrc$).

```
call gauss2(mx1,sx1,mx2,sx2,rotax,xsrc,usrc)
call gauss2(my1,sy1,my2,sy2,rotay,ysrc,vsrc)
```

Podemos ver la programación de la subrutina *gauss2* junto con la programación completa de la nueva fuente del RTM para penEasy en el Apéndice B.

La velocidad normalizada en z de la partícula la determinamos de la siguiente forma:

```
wsrc=-sqrt(abs(1.0d0-usrc**2-vsrc**2))
```

ya que la suma cuadrática de las componentes de la velocidad normalizada es igual a la unidad.

La Figura 1.11 muestra la forma de la fuente en el plano (x,y) obtenida con penEasy.

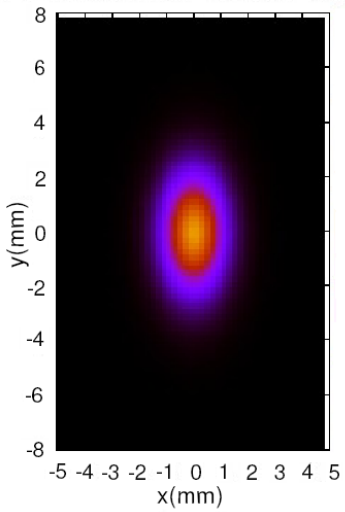

Figura 1.11: *Tamaño de la fuente a la salida del RTM.*

Debido a su pequeño tamaño tendremos que usar diferentes elementos para modificar el haz de tal forma que en el paciente tengamos una cierta dosis absorbida de cierta manera y en una cierta zona. Por lo tanto, antes de diseñar la geometría debemos saber qué condiciones requiere la IORT.

1.4. Características dosimétricas

En los siguientes apartados vamos a hablar de *dosis absorbida*, un concepto importante que debemos definir previamente. La dosis absorbida es el cociente de la energía media impartida por la radiación ionizante $\mathrm{d}\bar{\epsilon}$ (suma de todos los depósitos de energía en el volumen infinitesimal considerado) en un material de masa $\mathrm{d}m$ [18]. Es decir,

$$D = \frac{\mathrm{d}\bar{\epsilon}}{\mathrm{d}m} \tag{1.22}$$

Su unidad en el SI es el gray (1 Gy = 1 J/kg).

Las simulaciones transportan el haz de electrones desde la fuente (situada en $x = 0$, $y = 0$ y $z =$ - 61 cm) hasta el maniquí de agua (situado en $z \geq 0$). El sistema de referencia utilizado se representa en la Figura 1.12

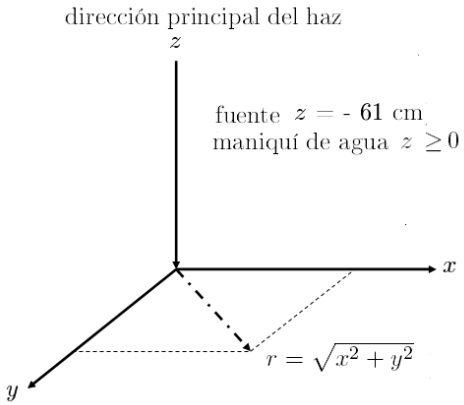

Figura 1.12: *Sistema de referencia utilizado.*

donde z es el eje que indica la dirección principal del haz y está dirigido hacia dentro del maniquí con origen en la superficie de éste. El radio r es la distancia desde el eje central en dirección ortogonal a z; la relación de r con las coordenadas transversales x e y es $r = \sqrt{x^2 + y^2}$.

Una vez definido el sistema de referencia, las características dosimétricas que debemos considerar son las siguientes [19, 20, 21, 22]:

1. La curva de **dosis en profundidad en el eje central** es una representación de la dosis absorbida en un cierto material a lo largo del eje central del haz. También se puede representar respecto la dosis máxima en este eje (en este caso se llama *Percentual Depth Dose Curve*, PDD). A partir de esta curva podemos determinar diferentes parámetros:

   - $d_{\max}$: profundidad donde la dosis absorbida en el eje central es máxima ($D_0(d_{\max})$).
   - d_{90}: profundidad superior a $d_{\max}$ donde la dosis absorbida en el eje central se ve reducida al 90 % de la dosis $D_0(d_{\max})$.
   - d_{50}: profundidad donde la dosis absorbida en el eje central se ve reducida al 50 % de $D_0(d_{\max})$.
   - *Contaminación por rayos X*: dosis medida en el eje central a d_{10} + 10 cm (d_{10}: profundidad donde la dosis absorbida en el eje central se ve reducida al 10 % de $D_0(d_{\max})$).
   - R_{p} (*practical range*): profundidad de interacción entre el fondo de contaminación por rayos X y la tangente de la curva PDD en el punto de inflexión.

   La Figura 1.13 ilustra los parámetros de la PDD de forma gráfica y la Figura 1.14 muestra una curva PDD en un caso real de IORT.

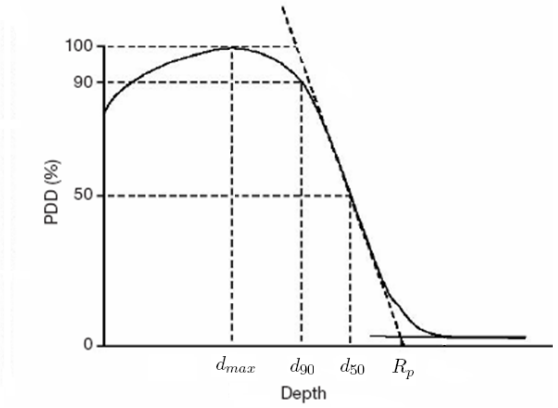

Figura 1.13: *Parámetros de la PDD.*

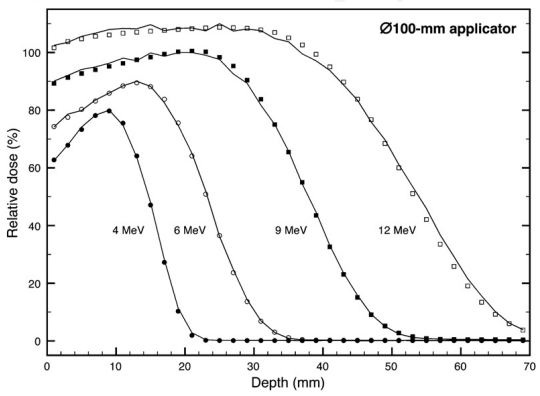

Figura 1.14: *PDD (extraído de Ref. [23])*

2. El **perfil lateral de dosis** muestra la dosis absorbida a una cierta profundidad versus el radio r (o bien x o y). Se suele representar el perfil de dosis a $d_{\max}$, d_{90} y en superficie y también se puede expresar en valor absoluto o como un % respecto la dosis máxima en el eje central $D_0(d_{\max})$. Los parámetros que podemos obtener son:

- La *asimetría* y la *uniformidad* de la distribución de dosis (ver siguiente apartado).
- *Dosis superficial*: dosis en la superficie medida a partir del perfil lateral de dosis en superficie.
- *Dosis periférica*: dosis absorbida a 2 cm más allá del diámetro interno del aplicador a cierta profundidad (normalmente $d_{\max}$ y d_{90}).

La Figura 1.15 muestra un ejemplo real de perfil lateral de dosis en IORT.

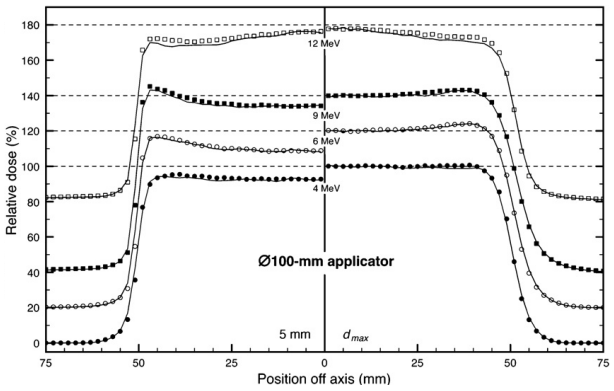

Figura 1.15: *Perfil lateral de dosis (extraído de Ref. [23]).*

3. Las **curvas de idososis** representan un mapa de distribución de dosis absorbida en función de la profundidad y de la distancia al eje r (o x o y). Las curvas que se suelen representar son la 90 %, 50 % y 20 % respecto $D_0(d_{\max})$. Parámetros de interés que podemos extraer son:

   - *Campo de irradiación*: región del plano $z = d_{\max}$ contenida dentro de la curva de isodosis del 90 %. También se puede determinar con el perfil lateral de dosis.
   - *Tamaño del campo*: región del plano $z = d_{\max}$ contenida dentro de la curva de isodosis del 50 %. Esta definición coincide con el tamaño nominal del campo.

La Figura 1.16 muestra un ejemplo real de curvas de isodosis en IORT.

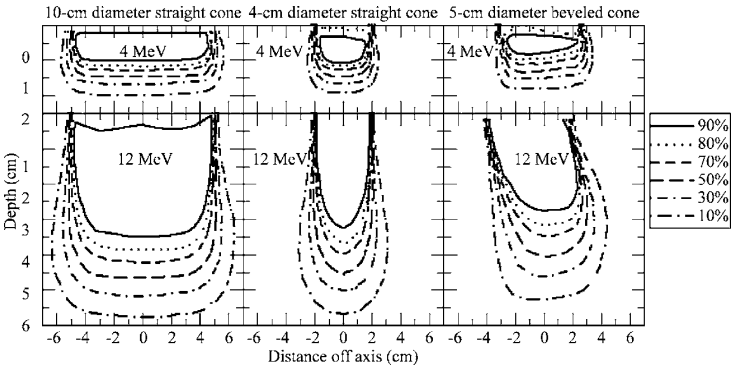

Figura 1.16: *Curvas de isodosis (extraído de Ref. [24]).*

1.5. Condiciones dosimétricas

Una vez definidas las características dosimétricas, las condiciones a cumplir para que el diseño de los modificadores finales del haz sea apto para IORT son las siguientes:

1. La **dosis absorbida en la superficie** en todo el campo de irradiación tiene que ser menor que el 85 % de la dosis $D_0(d_{\max})$.

2. **Uniformidad**: la diferencia entre la dosis máxima $D_{r,\max}(d_{\max})$ y mínima $D_{r,\min}(d_{\max})$ a la profundidad $d_{\max}$ en la región que cubre el 80 % de la extensión lateral del campo de irradiación respecto $D_0(d_{\max})$ no puede superar el 10 %,

$$H = \frac{D_{r,\max}(d_{\max}) - D_{r,\min}(d_{\max})}{D_0(d_{\max})} < 10\,\% \tag{1.23}$$

3. **Asimetría**: la diferencia en la dosis dada por el perfil lateral de dosis a una profundidad $d_{\max}$ de cualquier par de puntos situados simétricamente a igual distancia del eje central, dentro del 80 % de la extensión lateral del campo, tiene que ser inferior al 2 %.

4. El criterio para la **dosis periférica** es que debe ser inferior al 5 % de $D_0(d_{\max})$ para los perfiles laterales de dosis en $d_{\max}$ y d_{90}.

5. **Contaminación por rayos X**: la dosis absorbida debida a los rayos X debe ser inferior o igual al 1 % de $D_0(d_{\max})$.

6. **Normativa International Standard IEC 60601-2-1** [25], que hace referencia a la dosis generada fuera del campo de irradiación. En la Tabla 1.4 se presenta un resumen de las normas que se deben cumplir.

Radio r	**Dosis absorbida en agua respecto** $D_0(d_{\max})$
R_i+2 cm $<r<$ R_i+10 cm ($z = d_{\max}$)	Máxima <10 %
R_i+4 cm $<r<$ R_i+10 cm ($z = d_{\max}$)	Media <1 %
R_i+10 cm$<r<$ R_i+ 200 cm ($z = d_{\max}$)	Máxima <0.2 %
R_i+10 cm$<r<$ R_i+ 200 cm ($z = d_{\max}$)	Media <0.1 %
R_e + 2 cm ($\forall z < 0$)	Máxima <10 %

Tabla 1.4: *Normativa IEC 60601-2-1. Notación:* R_i *es el radio interno del aplicador y* R_e *el exterior.*

1.6. Diseño de los modificadores finales del haz

La distribución de dosis final depende fuertemente del diseño de los modificadores del haz, así que debemos analizar cuidadosamente cada elemento que incorporemos.

1.6.1. Componentes y características

El sistema final de transporte del haz consta de diferentes partes:

- **Ventana de salida**

  La ventana de salida del RTM está hecha de titanio y tiene un grosor de 20 μm.

- **Aplicador**

  En IORT, el haz de salida se colima hacia el paciente mediante aplicadores, sistemas que limitan el campo a irradiar y a la vez lo hacen más uniforme en esta zona. Vemos una representación gráfica en la Figura 1.17. Sus características son las siguientes:

  - Suelen tener una sección lateral circular o rectangular. Nos centraremos en los de sección circular ya que son los más comunes.
  - Pueden ser cortados de forma recta u oblicua para evitar los *gaps* de aire (ver Figura 1.18). En el presente trabajo consideraremos únicamente los de corte recto.

- Suelen estar hechos de lucita o latón.
  - La lucita (PMMA, polimetilmetracrilato) es un plástico transparente con densidad 1.19 g/cm^3 que facilita la visualización de la zona de irradiación. Aun así, con el uso pierde su transparencia. Un aplicador de lucita es suficiente para conformar el campo, pero no para blindar el exterior de la radiación.
  - El latón es una aleación de cobre y zinc que puede tener composiciones diferentes dependiendo de su aplicación; en nuestro caso, usaremos un 65 % de Cu y un 35 % de Zn con una densidad de 8.0 g/cm^3. El latón es opaco pero nos proporciona una protección mayor ya que evita que la radiación se escape fuera del paciente.
- El grosor suele ser de 3 mm, aunque podemos incrementarlo incluso hasta 8 mm.
- El diámetro de los aplicadores es variable y puede ir desde 5 a 15 cm o más, dependiendo del tamaño del tumor y su localización.
- La altura del aplicador es variable, aunque se requiere un mínimo de 50 cm y puede llegar a ser de más de 1 m. En nuestro caso, intentaremos que sea lo más corto posible ya que el RTM será menos aparatoso; usaremos una longitud de 60 cm para todos los casos evaluados.

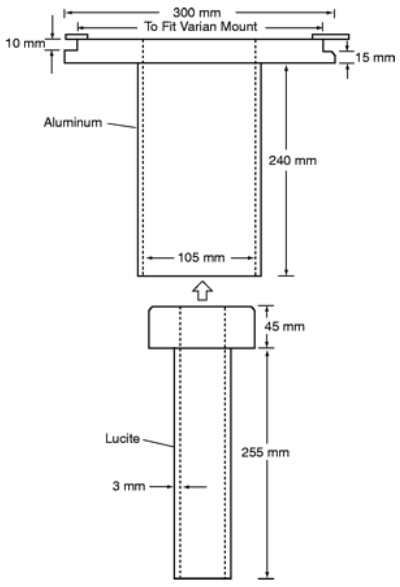

Figura 1.17: *Aplicador para IORT* (extraído de Ref. [26]).

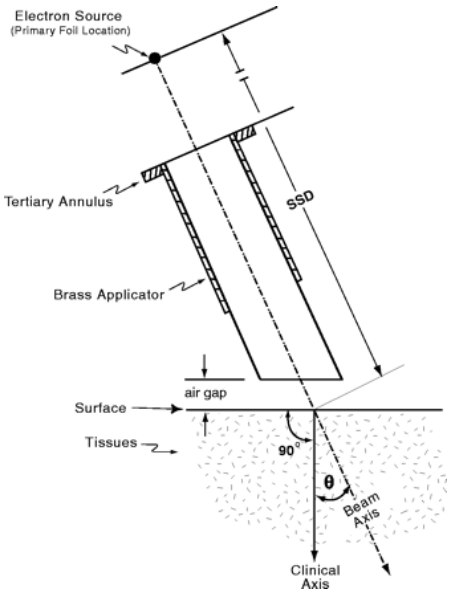

Figura 1.18: *Aplicador oblicuo* (extraído de Ref. [26]).

- **Láminas dispersoras** (*scattering foil(s)*)

  Las láminas dispersoras son necesarias para abrir el haz (recordemos que el tamaño de nuestra fuente es muy pequeño) y reducir la longitud del aplicador. La elección de las láminas dispersoras requiere conocimientos teóricos de la interacción de la radiación con la materia; en el Apéndice A se puede encontrar una breve descripción de los diferentes mecanismos de interacción.

  - Se usan láminas de aluminio (Al, Z=13) para abrir los haces de electrones. En principio, cuanto mayor sea el número atómico Z de la lámina, más poder de dispersión por unidad de grosor presenta el material. Pero debido a la producción paralela de bremsstrahlung, creciente con el cuadrado de Z del material, es desaconsejable el uso de elementos pesados en contacto con la fuente emisora. Por eso material de la primera lámina debe tener un Z bajo.
  - Debajo de la lámina de Al, pondremos una lámina fina de oro (Au, Z=79) para atenuar el bremsstrahlung originado previamente. Como la sección eficaz de bremsstrahlung depende de $1/E_\gamma$, donde E_γ es la energía del fotón emitido, será más probable la emisión de fotones poco energéticos en la lámina de Al. Como el mecanismo de interacción mayoritario de fotones en Au para energías menores a 1 MeV es el efecto fotoeléctrico, los fotones de Bremsstrahlung producidos en la primera lámina padecerán efecto fotoeléctrico, dando lugar a electrones con energía $E_\mathrm{e} = E_\gamma - U_i$ (la energía de ligadura U_K de un electron de la capa K del oro es de aproximadamente 81 keV [27]).
  - La producción de fotones de bremsstrahlung aumenta con la energía del haz incidente. Es por eso que en el caso de las energías bajas del RTM no será necesario usar la lámina de oro.
  - Con un radio de las láminas dispersoras de 2 cm será suficiente porqué el haz en la salida del RTM es del orden de mm. Respecto al grosor de las láminas, lo analizaremos más adelante con detalle.

- **Blindaje**

  En el caso de las energías bajas del RTM se cumplen los requisitos detallados en apartados anteriores sin necesidad de blindaje. De todas formas, para energías altas necesitamos blindaje. En apartados posteriores analizaremos con detalle el tipo de blindaje a usar.

- En el presente estudio no se incluyen más elementos, aunque sería posible añadir una cámara de ionización para monitorizar la radiación que sale del RTM.

Cabe decir que aunque la energía incidente del haz de electrones después de la ventana de salida presente poca dispersión energética, la energía del haz en la superficie del paciente será menor y con más dispersión debido a la pérdida de energía de los electrones al interaccionar con las paredes del aplicador y los otros accesorios entre la ventana de salida y la región a tratar. La calidad del haz se podrá determinar mediante la PDD.

En la Figura 1.19 vemos un esquema de los modificadores del haz (sin blindaje). Debido a la gran cantidad de parámetros libres que existen en el diseño del aplicador, se decidió estudiar sólo algunos casos. Mi trabajo, a parte de hacer toda la parte de programación y adaptación de penEasy para todas las energías, consiste en estudiar con detalle el caso de 12 MeV, el más complicado debido a que las partículas tienen mayor energía y se requiere también diseño del blindaje. Para acotar más el trabajo, se decidió evaluar sólo el caso de un campo de irradiación de 15 cm, que es el que se prevé que tenga más dificultades ya que supone una apertura mayor del campo. De esta forma, si se consigue cumplir las normas para este caso, se conseguirá cumplirlas para los demás casos. En la Figura 1.20 y en la Tabla 1.5 se presentan los parámetros de la geometría que se va a estudiar.

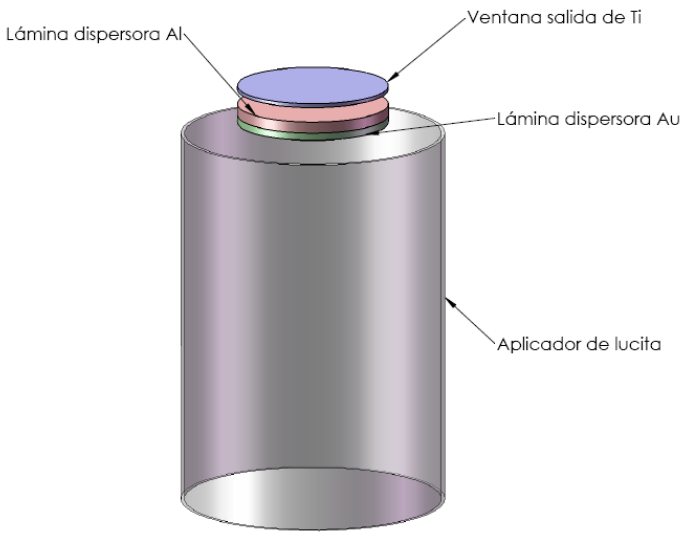

Figura 1.19: *Esquema de los modificadores del haz.*

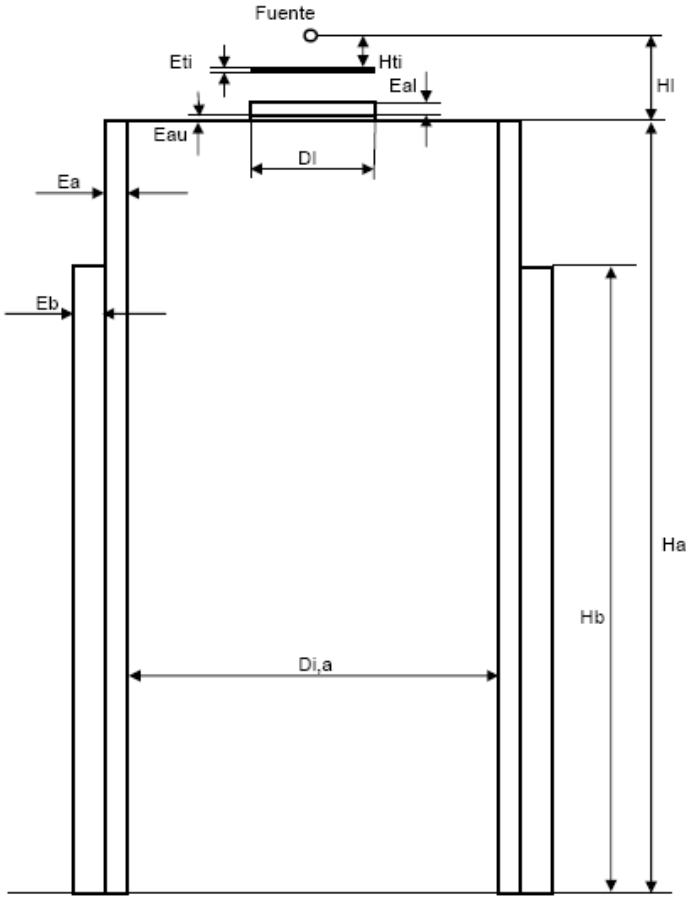

Figura 1.20: *Parámetros de la geometría.*

Parámetro	**Notación**	**12 MeV [cm]**
Diámetro interno aplicador	$D_{\mathrm{i,a}}$	15
Longitud del aplicador	H_{a}	60
Grosor del aplicador	E_{a}	0.3
Distancia entre la fuente y la ventana de salida	H_{ti}	0.5
Grosor de la ventana de salida de Ti	E_{ti}	0.002
Distancia entre la fuente y la base de las láminas dispersoras	H_{l}	1
Diámetro de las láminas dispersoras	D_{l}	4
Grosor de la lámina dispersora de Al	E_{al}	a estudiar
Grosor de la lámina dispersora de Au	E_{au}	a estudiar
Longitud del blindaje	H_{b}	a estudiar
Grosor de blindaje	E_{b}	a estudiar

Tabla 1.5: *Parámetros de la geometría.*

1.6.2. Análisis teórico inicial para las láminas dispersoras

Un grosor típico de láminas dispersoras para aceleradores lineales de 12 MeV es el que se presenta en la Tabla 1.6. Antes de hacer todas las simulaciones con estos parámetros, vamos a analizar si son aceptables para nuestro caso o no.

Parámetro	Notación	12 MeV [cm]
Grosor de la lámina dispersora de Al	E_{al}	0.0800
Grosor de la lámina dispersora de Au	E_{au}	0.0150

Tabla 1.6: *Grosor de las láminas dispersoras.*

Las deflexiones angulares de las trayectorias de los electrones son en gran medida debidas a las dispersiones elásticas. Si consideramos teorías de dispersión elástica múltiple, se define

$$\lambda_1^{-1} = \mathcal{N}\int_{-1}^{+1}\sigma(\omega)[1-\omega]\mathrm{d}\omega = \lambda^{-1}[1-\langle\omega\rangle] \tag{1.24}$$

donde $\mathcal{N}$ es el número de moléculas por unidad de volumen, $\omega = \cos\theta$, θ es el ángulo de dispersión, $\sigma(\omega)$ es la sección eficaz diferencial elástica y $\lambda^{-1} = \mathcal{N}\sigma$ es la distancia media recorrida por una partícula entre colisiones (*mean free path*). Según la teoría de dispersión múltiple de Lewis [28], despreciando las pérdidas de energía

$$\langle\cos\Theta\rangle = \exp\left(-\int_0^s \frac{\mathrm{d}s'}{\lambda_1(s')}\right) \tag{1.25}$$

siendo Θ el ángulo de dispersión acumulado y s el camino recorrido por la partícula. Para un recorrido s pequeño comparado con λ_1,

$$\langle\cos\Theta\rangle \simeq 1 - \frac{\langle\Theta^2\rangle}{2} \qquad \langle\cos\Theta\rangle = exp\left(-\frac{s}{\lambda_1}\right) \simeq 1 - \frac{s}{\lambda_1} \tag{1.26}$$

Combinando las ecuaciones anteriores tenemos

$$\langle \Theta^2 \rangle \simeq \frac{2s}{\lambda_1} \tag{1.27}$$

El valor de λ_1, que depende de la energía cinética del electron, está tabulado en la base de datos de PENELOPE.

Los electrones del RTM salen de la fuente con una energía cinética de 12.1 MeV. Considerando dispersión inelástica múltiple, entra en juego el poder de frenado S (*stopping power*)

$$S = -\frac{\mathrm{d}K}{\mathrm{d}s} \tag{1.28}$$

que es la energía cinética media perdida por los electrones por unidad de recorrido; en el caso de los electrones es del orden de 2 MeV.cm^2/g. Por lo tanto, la energía perdida en un cierto material de densidad ρ y grosor s es

$$\triangle E \approx 2\,\frac{\mathrm{MeV\,cm^2}}{\mathrm{g}}\,\rho s \tag{1.29}$$

A la salida de la lámina de Ti, los electrones tendrán una energía de 12.1 MeV-$\triangle E_{\mathrm{Ti}}$=12.08 MeV

$$\triangle E_{\mathrm{Ti}} \approx 2\,\frac{\mathrm{MeV\,cm^2}}{\mathrm{g}}\,4.54\,\frac{\mathrm{g}}{\mathrm{cm^3}}\,0.002\,\mathrm{cm} = 18.2\,\mathrm{keV}$$

Una vez sabemos la energía de entrada en la lámina de Al, hacemos una interpolación de los valores que se presentan en los archivos de salida del programa *tables*, escalando por la densidad del Al ($\rho_{\mathrm{Al}} = 2.6989$ g/cm^3)

```
# PENELOPE >>> Aluminum. Electron mean free paths and range.
#
#   Energy       mfp_tot     1st tmfp       range
#    (eV)       (g/cm**2)    (g/cm**2)    (g/cm**2)
#-------------------------------------------------
```

```
  1.13634E+07  2.74876E-05  2.11123E+01  6.56439E+00
  1.23650E+07  2.74647E-05  2.45550E+01  7.06567E+00
```

y tenemos que $\lambda_{1,\mathrm{Al}} = 8.76$ cm. Ahora calculemos la energía perdida por los electrones en la lámina de Al de grosor s_{Al}=0.0800 cm

$$\triangle E_{\mathrm{Al}} \approx 2\,\frac{\mathrm{MeV\,cm^2}}{\mathrm{g}}\,2.6989\frac{\mathrm{g}}{\mathrm{cm^3}}\,0.0800\,\mathrm{cm} = 431.8\,\mathrm{keV}$$

Así pues, la energía de entrada de los electrones en la lámina de Au es 12.08 MeV-$\triangle E_{\mathrm{Al}}$=11.65 MeV. Si seguimos los mismos pasos que con la lámina de Al, pero con Au (ρ_{Au} = 19.32 g/cm^3) y un grosor de s_{Au}=0.0150 cm

```
 # PENELOPE >>>  Gold. Electron mean free paths and range.
 #
 #   Energy       mfp_tot     1st tmfp       range
 #    (eV)       (g/cm**2)    (g/cm**2)    (g/cm**2)
 #-------------------------------------------------
   1.13634E+07  3.39307E-05  4.05873E+00  6.70783E+00
   1.23650E+07  3.39099E-05  4.74125E+00  7.08718E+00
```

el resultado es $\lambda_{1,\mathrm{Al}} = 0.22$ cm. Una vez tenemos los valores de $\lambda_{1,\mathrm{Al}}$=8.76 cm y $\lambda_{1,\mathrm{Au}}$=0.23 cm, volvemos a la teoría de dispersión elástica múltiple, donde el ángulo de dispersión acumulado promedio para el caso descrito es

$$\langle\Theta\rangle = \sqrt{\langle\Theta^2\rangle_{\mathrm{Al}} + \langle\Theta^2\rangle_{\mathrm{Au}}} = 0.388\,\mathrm{rad} \tag{1.30}$$

Si consideramos que queremos abrir el haz 7.5 cm (radio del aplicador) a una altura respecto el maniquí de 40 cm (el resto servirá para homogeneizar la dosis en el paciente), con una simple relación trigonométrica $\tan\langle\Theta'\rangle$=radio/longitud, encontramos que el ángulo que necesitamos

es de 0.359 rad. Por lo tanto, comparando con la Ecuación (1.30) vemos que nos movemos en el rango correcto de grosores de láminas dispersoras. Ahora ya podemos empezar con las simulaciones.

1.7. Simulaciones del RTM de 12 MeV con penEasy

1.7.1. Características de la simulación

En la presente sección especificaremos los valores escogidos para cada uno de los parámetros de entrada del programa (en el Apéndice A se detallan cuáles son y su significado).

- Parámetros de la simulación.

  Para los parámetros del transporte de partículas, seguiremos las recomendaciones de la Ref. [29]. En el caso del material agua (que representa el paciente y es donde queremos calcular las magnitudes de interés):

  - EABS(γ)=10 keV, ya que a este nivel de energías el efecto predominante es el fotoeléctrico y por lo tanto, el fotón da lugar a un electrón. La energía de absorción para el electrón es:
  - EABS($e^{\pm}$)

    Consideraciones a tener en cuenta:

    - El poder de frenado de los electrones es aproximadamente 2 MeV cm^2/g. Si los voxels para calcular la dosis en las simulaciones realizadas son de un mínimo de $\triangle x$= 5 mm, la energía de absorción tiene que ser menor que 2 MeV cm^2/g $\frac{\triangle x}{2}$ = 500 keV. De esta forma, es una buena aproximación considerar que cuando el electron llega a la energía de absorción (por ejemplo en la mitad del voxel) se deposite toda su energía porque no llegaría a cruzar el voxel para depositar su energía en otro contiguo. Así pues, no distorsionamos la distribución de dosis final. Siendo un poco conservadores, se ha considerado EABS($e^{\pm}$)=300 keV.
    - Pero los electrones también producen fotones en la interacción con la materia y éstos depositan su energía lejos del voxel de interacción. Por esta razón se define

el rendimiento radiativo, la fracción de energía disipada en forma de fotones; es recomendable que sea menor que el 1 % para la energía de absorción escogida. Los valores del rendimiento radiativo para todos los materiales se pueden encontrar en la Ref. [30]. En el caso de EABS($e^{\pm}$)=300 keV para el agua, tenemos que el rendimiento radiativo es 0.13 %.

Para el resto de materiales, donde no queremos calcular ninguna magnitud, tendremos que poner energías de absorción lejos de la energía que tienen las partículas en esa zona para no absorber más partículas de las debidas, pero los parámetros no son tan críticos.

- C1=C2=0.1
- WCC=min(E0/100,EABS(e^-))
- WRC=min(E0/1000,EABS(γ))
- DSMAX=0.1*grosor

- Características de los resultados que queremos obtener, que los vamos a ir especificando a medida que se presenten los resultados.

- Fichero de materiales presentes en la geometría, que se genera fácilmente con el programa *material* de PENELOPE.

- El archivo de geometría. Podemos encontrar archivo de geometría generado (con blindaje) en el Apéndice C (en el Apéndice A se explica el funcionamiento general de la geometría). La Figura 1.21 muestra la geometría con el visualizador de PENELOPE.

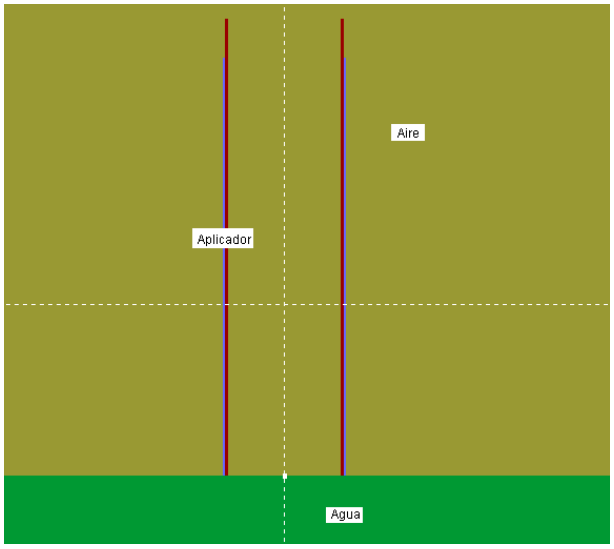

Figura 1.21: *Esquema de la geometría con el visualizador de PENELOPE.*

En todos los casos calcularemos la dosis absorbida en un cierto material. Para ello, utilizaremos varios de los *tallies* implementados en penEasy:

- ***Tally* de distribución de dosis espacial**, que calcula la dosis absorbida por historia simulada en el intervalo [XMIN,XMAX] usando NXBIN bins (para las tres dimensiones). Los parámetros de entrada de este *tally* son

```
[SECTION TALLY SPATIAL DOSE DISTRIB v.2006-08-01]
(ON )                              STATUS (ON or OFF)
 -20.0 20.0 80                     XMIN,XMAX(cm),NXBIN (0 for DX=infty)
 -20.0 20.0 80                     YMIN,YMAX(cm),NYBIN (0 for DY=infty)
 -18.0 0.0 36                      ZMIN,ZMAX(cm),NZBIN (0 for DZ=infty)
 1                                 PRINT COORDINATES IN REPORT (1=yes,0=no)
 0.5                               RELATIVE UNCERTAINTY (%) REQUESTED
[END OF SDD SECTION]
```

Es decir, el tamaño de los voxels donde queremos calcular la dosis y la incertidumbre relativa requerida. En la mayor parte de las simulaciones, la incertidumbre requerida será del orden de 0.5 % (se especificará cuando no sea así). Para llegar a tal precisión, una

simulación individual completa requiere de aproximadamente 7 días con un PC con un procesador de 3.0 GHz y 2 Gb de memoria RAM.

Para reducir el tiempo de simulación, en los estudios preliminares utilizaremos otras técnicas: absorber toda la radiación en una lámina de agua considerando que es un material completamente absorbente (es decir, la energía de absorción de los fotones y de los electrones es superior a la energía inicial y cuando una partícula interacciona con el material, quedará absorbida instantáneamente), conformarnos con una incertidumbre menor o bien usar:

- ***Tally* de distribución de dosis cilíndrica**, que calcula la dosis absorbida por historia simulada en el intervalo dr r [RMIN,RMAX] usando NRBIN bins (lo mismo para la coordenada z). Con estos elementos de volumen podremos calcular la distribución de dosis cilíndrica. Los parámetros de entrada para este *tally* son:

```
[SECTION TALLY CYLINDRICAL DOSE DISTRIB v.2006-08-01]
(OFF)                              STATUS (ON or OFF)
 6.0 200.0 97                      RMIN,RMAX(cm),NRBIN (>0)
 -9.0 0.0 18                       ZMIN,ZMAX(cm),NZBIN (0 for DZ=infty)
 1                                 PRINT COORDINATES IN REPORT (1=yes,0=no)
 0.5                               RELATIVE UNCERTAINTY (%) REQUESTED
[END OF CDD SECTION]
```

También he hecho una modificación al programa para poder calcular más de un tally cilíndrico (en concreto 3) en la misma simulación y así poder obtener más información de una sola vez.

1.7.2. Estudio inicial sin blindaje

En primer lugar, calculamos la dosis absorbida depositada en agua en el plano $z = 0$ considerando que es un material totalmente absorbente. Las Figuras 1.22 y 1.23 muestran la dosis absorbida por historia simulada a lo largo de x e y, respectivamente.

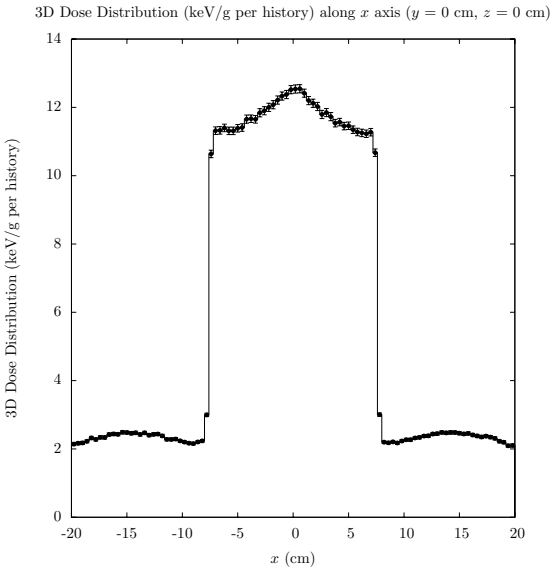

Figura 1.22: *Estudio inicial. Perfil lateral en x para agua completamente absorbente.*

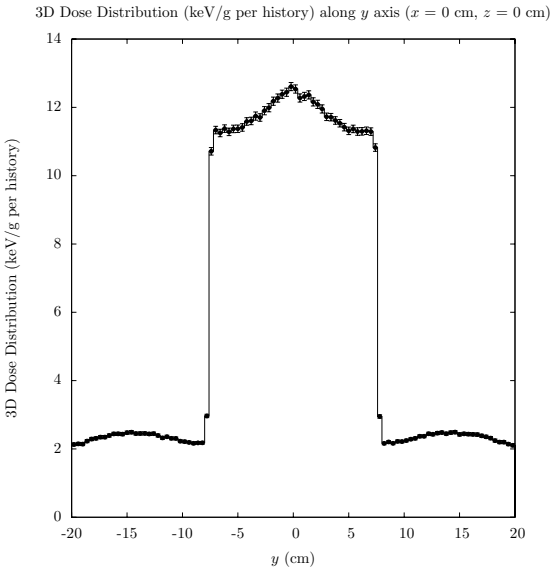

Figura 1.23: *Estudio inicial. Perfil lateral en y para agua completamente absorbente.*

He hecho una pequeña modificación al programa para calcular separadamente la dosis debida a electrones y la debida a fotones. Las Figuras 1.24 y 1.25 representan el perfil de dosis en x debido a fotones y electrones en agua totalmente absorbente.

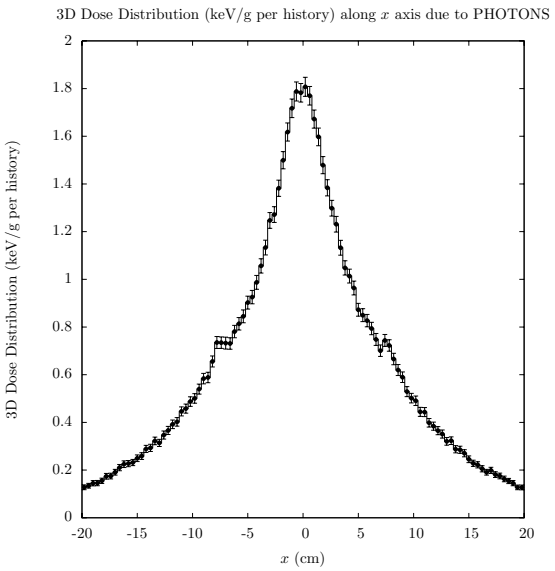

Figura 1.24: *Estudio inicial. Perfil de dosis en x debida a fotones.*

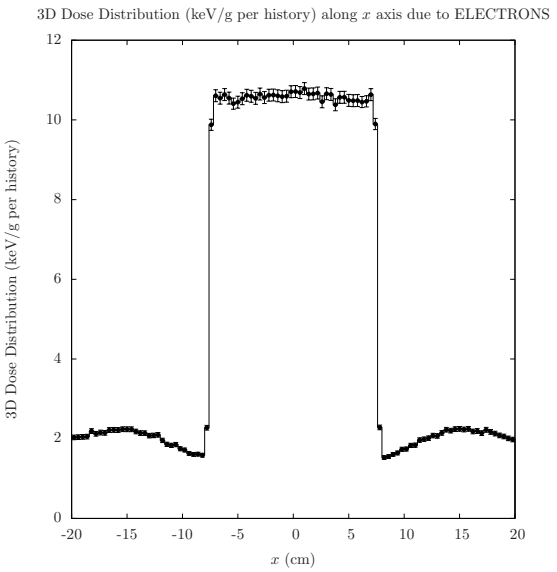

Figura 1.25: *Estudio inicial. Perfil de dosis en x debida a electrones.*

Vemos que la punta central es debida a fotones, que no nos causará problemas ya que penetran distancias mucho mayores que los electrones. Si miramos la dosis depositada por electrones vemos que la uniformidad en la zona de irradiación es alta, pero no cumple los requisitos dosimétricos porqué la dosis periférica, prácticamente toda debida a electrones, es muy elevada. Con esto también nos hacemos una idea del tipo de blindaje que deberemos usar.

Repetimos la simulación pero teniendo en cuenta una distribución 3D de dosis. Además, analizaremos este caso con detalle para ver cuales serán las condiciones más difíciles de cumplir.

1. **Dosis en profundidad en el eje central**

   La Figura 1.26 representa la dosis en profundidad en el eje central y la Tabla 1.7 los parámetros que se pueden obtener a partir de ella.

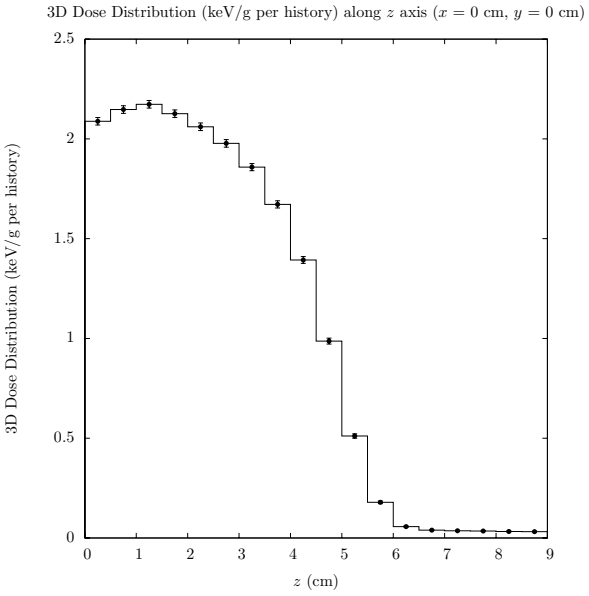

Figura 1.26: *Estudio inicial. Dosis en profundidad en el eje central.*

Parámetro	Normativa	Valor
$d_{\max}$		1.25 ± 0.3 cm
d_{90}		2.7 ± 0.3 cm
R_p		5.5 ± 0.5 cm
$D_{d_{10}+10\text{cm}}/D_0(d_{\max})$	1 %	0.90 ± 0.04 %

Tabla 1.7: *Estudio inicial. Características dosimétricas obtenidas a partir de la PDD.*

2. **Perfil lateral de dosis**

Podemos ver el perfil lateral en x e y a la profundidad $d_{\max}$ en las Figuras 1.27 y 1.28. Como vemos que no existen diferencias significativas entre las dos figuras, a partir de ahora sólo se mostrará el gráfico para x (o y) ya que se puede obtener la misma información.

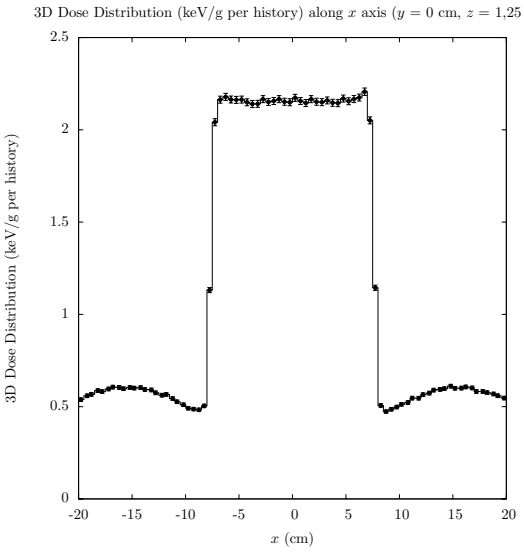

Figura 1.27: *Estudio inicial. Perfil lateral en x a $z = d_{\max}$.*

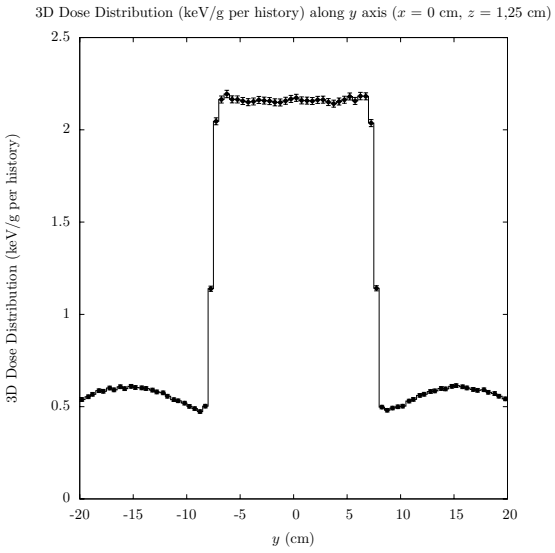

Figura 1.28: *Estudio inicial. Perfil lateral en y a $z = d_{\max}$.*

El perfil lateral en x en la superficie (z=0) se representa en la Figura 1.29, mientras que las Figuras 1.30 y 1.31 muestran el perfil lateral a la profundidad $z = d_{50}$=4.75 cm y en la cola de la curva PDD (z=8.25 cm), respectivamente.

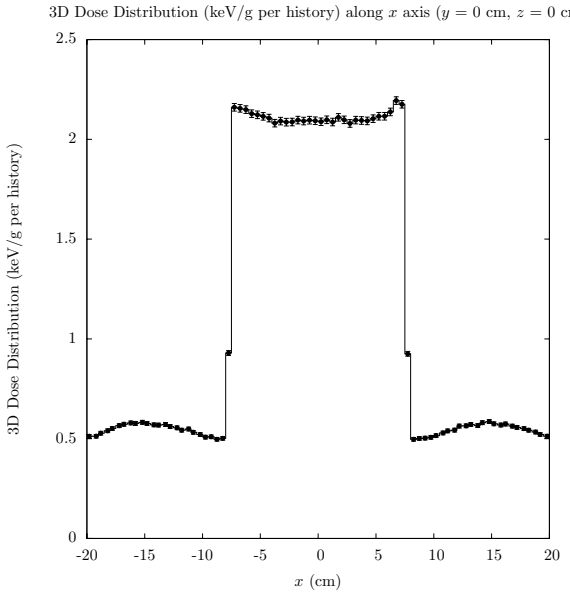

Figura 1.29: *Estudio inicial. Perfil lateral en x en la superficie.*

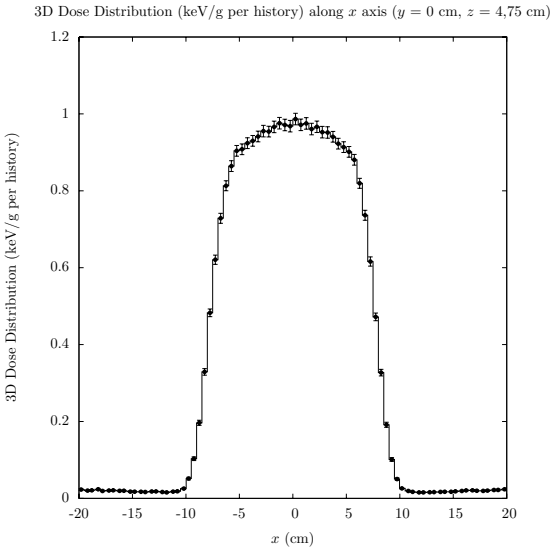

Figura 1.30: *Estudio inicial. Perfil lateral en x a z=4.75 cm.*

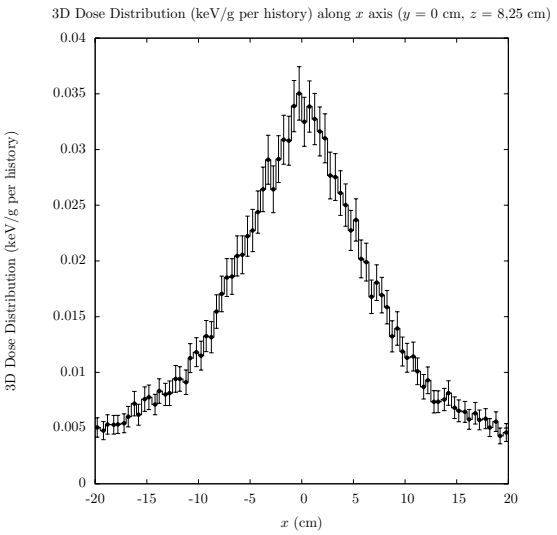

Figura 1.31: *Estudio inicial. Perfil lateral en x a z=8.25 cm.*

A partir de las figuras anteriores, obtenemos las características dosimétricas de la Tabla 1.8.

Parámetro	Normativa	Valor
Uniformidad	10 %	2.6 ± 0.4 %
Asimetría	2 %	<0.9 %
Dosis superficial	>85 %	>95 %
Dosis periférica a $d_{\max}$	<5 %	22.6 ± 0.9 %
Dosis periférica a d_{90}	<5 %	14.7 ± 0.2 %

Tabla 1.8: *Estudio inicial. Características dosimétricas obtenidas a partir del perfil lateral (en rojo, valores que no cumplen la normativa).*

Evaluamos el cumplimiento de la normativa IEC 60601-2-1 partir de la Figura 1.32, que muestra el perfil lateral de dosis hasta 200 cm. La Figura 1.33 muestra si se cumple el último punto de la normativa referente a la dosis a 2 cm del radio externo del aplicador en agua (o en aire pero comparando respecto la dosis máxima en aire).

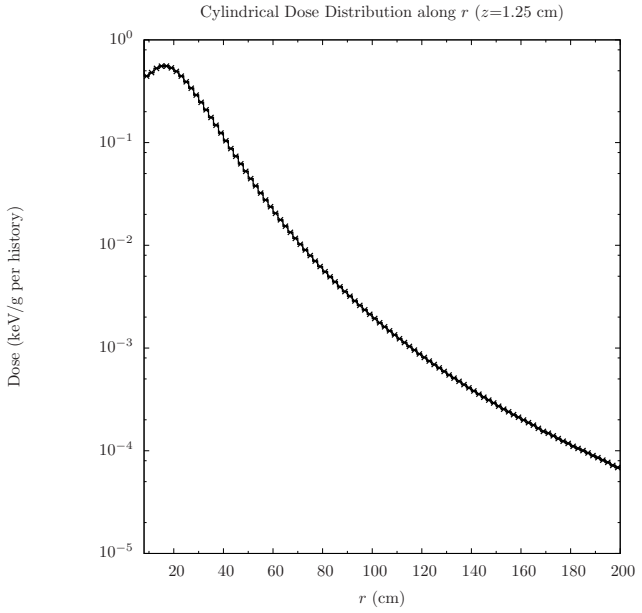

Figura 1.32: *Estudio inicial. Perfil lateral en r a $d_{\max}$.*

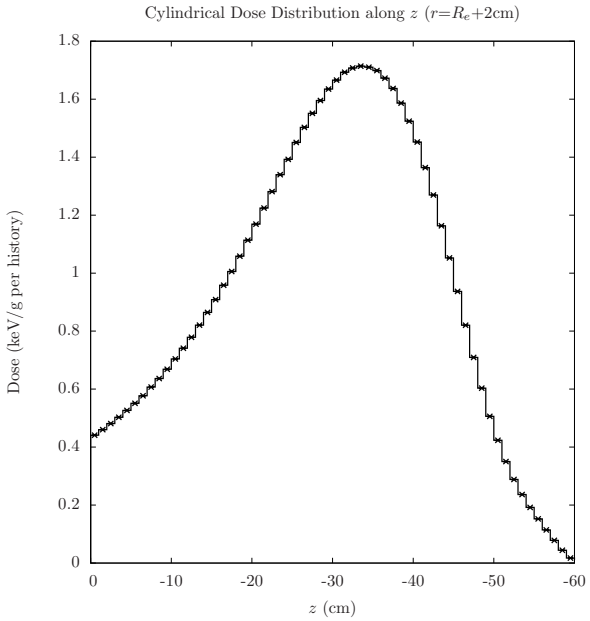

Figura 1.33: *Estudio inicial. Dosis absorbida en z en aire para $r = R_e$+2 cm.*

La Tabla 1.9 muestra como no se cumple la normativa.

Radio r	**Normativa**	**Valor**
R_i+2 cm $< r <$ R_i+10 cm (a d_{max})	Máxima <10 %	27.3 ± 0.1 %
R_i+4 cm $< r <$ R_i+10 cm (a d_{max})	Media <1 %	26.8 ± 0.1 %
R_i+10 cm$< r <$ R_i+ 200 cm (a d_{max})	Máxima <0.2 %	25.7 ± 0.1 %
R_i+10 cm$< r <$ R_i+ 200 cm (a d_{max})	Media <0.1 %	2.32 ± 0.01 %
R_e + 2 cm ($\forall z < 0$)	Máxima <10 %	38 ± 1 %

Tabla 1.9: *Estudio inicial. Normativa IEC 60601-2-1 (en rojo, valores que no cumplen la normativa).*

3. Curvas de isodosis

A partir de las Figuras 1.34 y 1.35 podemos obtener los valores del campo de irradiación y del tamaño de campo (véase Tabla 1.10).

Parámetro	**Valor**
Campo de irradiación	7.5 cm
Tamaño de campo	7.75 cm

Tabla 1.10: *Estudio inicial. Características dosimétricas obtenidas a partir de las curvas de isodosis.*

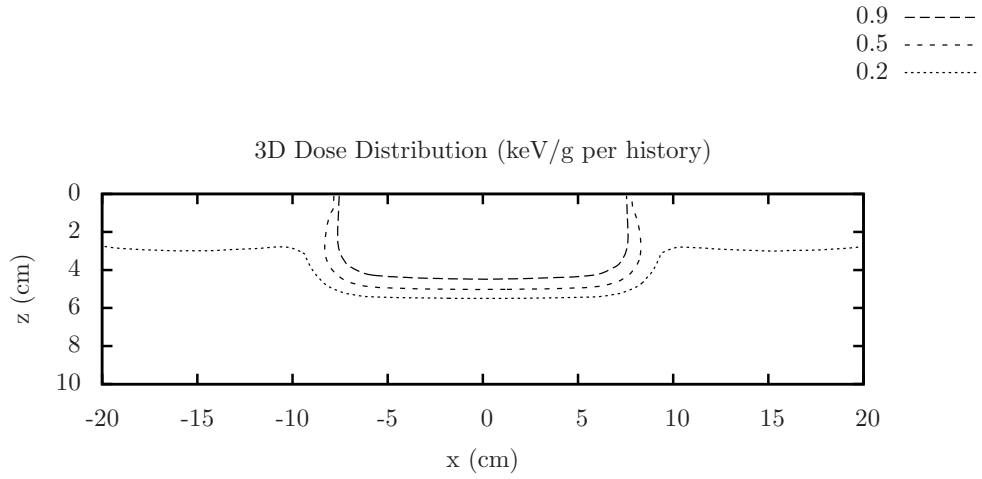

Figura 1.34: *Estudio inicial. Curva de isodosis x-z (y=0).*

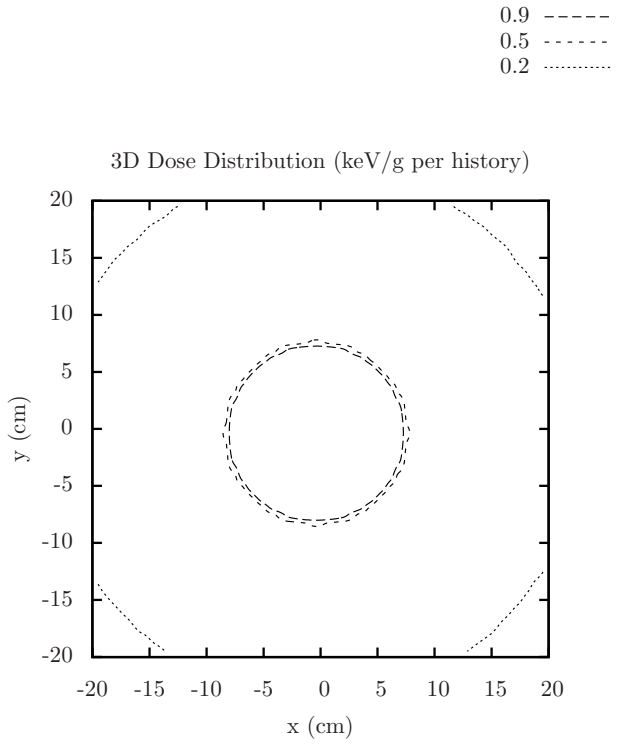

Figura 1.35: *Estudio inicial. Curva de isodosis x-y ($z = d_{max}$).*

Por lo tanto, vemos que muchas de las características dosimétricas no se cumplen. Las más difíciles de cumplir van a ser la dosis periférica y la normativa IEC 60601-2-1. Como sin blindaje no se cumplen las normativas, el siguiente objetivo es evaluar diferentes tipos de blindaje.

1.7.3. Estudio del blindaje

En todo el apartado usaremos el grosor de las láminas dispersoras que se muestra en la Tabla 1.6.

a. Grosor de lucita

Lo primero que uno se plantea a la hora de diseñar el blindaje es aumentar el grosor de lucita. Para los mismos parámetros que el caso anterior, he aumentado el grosor hasta 5, 7 y 9 mm. Vemos un ejemplo gráfico en la Figura 1.36.

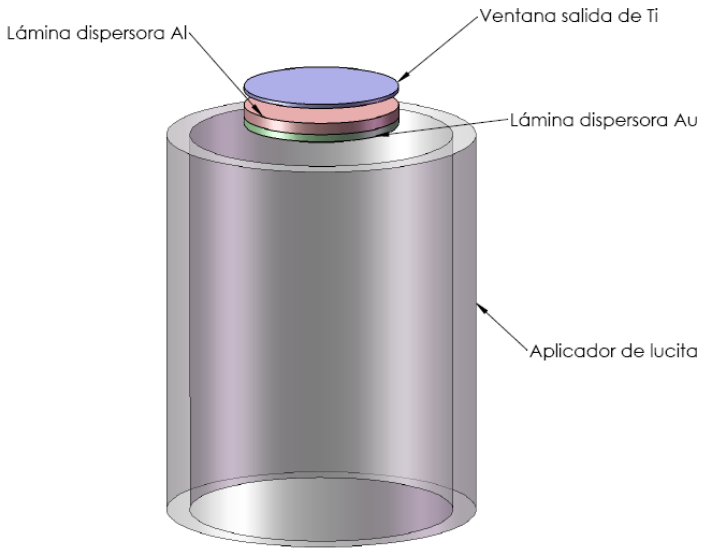

Figura 1.36: *Esquema del blindaje con un aumento del grosor de lucita.*

Para acortar el tiempo de simulación, hemos calculado la dosis en agua como si fuera un material completamente absorbente y hemos considerado aceptable una incertidumbre relativa total del 2 %.

Se puede observar el perfil lateral en x para los diferentes casos en las Figuras 1.37, 1.38 y 1.39.

- **Caso 1**. Grosor de lucita = 5 mm

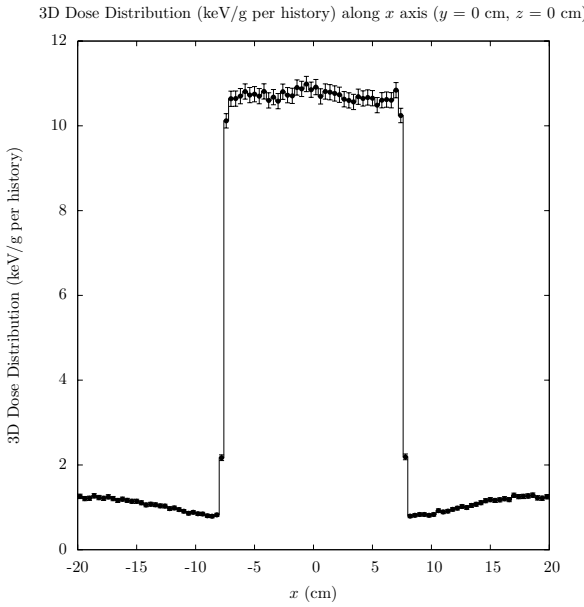

Figura 1.37: *Blindaje. Perfil lateral en x para un grosor de lucita de 5 mm.*

- **Caso 2**. Grosor de lucita = 7 mm

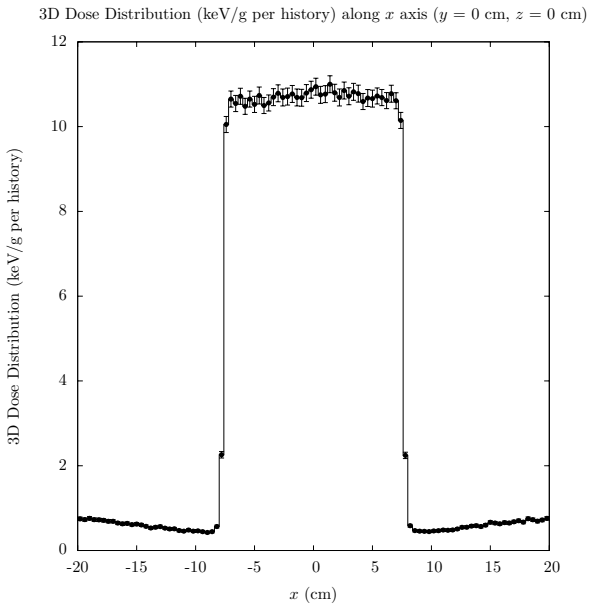

Figura 1.38: *Blindaje. Perfil lateral en x para un grosor de lucita de 7 mm.*

- **Caso 3**. Grosor de lucita = 9 mm

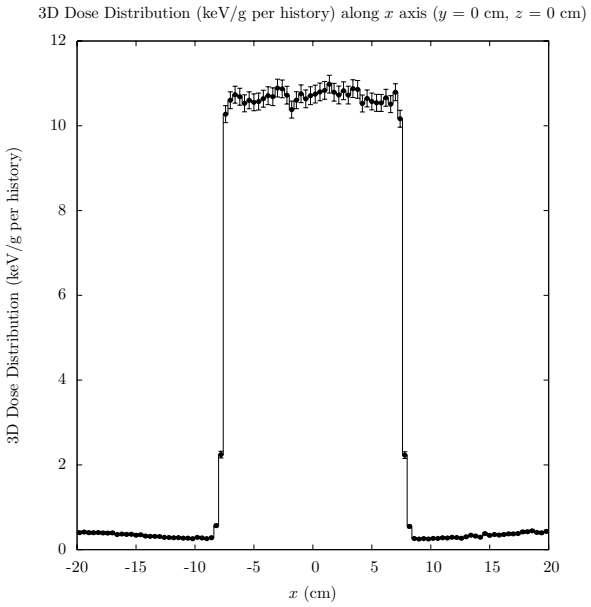

Figura 1.39: *Blindaje. Perfil lateral en x para un grosor de lucita de 9 mm.*

Si analizamos los datos vemos que la relación dosis a 2 cm versus dosis máxima es del orden del 8 % en el caso de un grosor de 5 mm, del 3 % en el caso de 7 mm y del 2 % en el caso de 9 mm. Los casos 7 y 9 mm de grosor podrían ser interesantes para estudiar con más detalle pero incumplen la norma IEC 60601-2-1 ya que la relación dosis en la zona R_{i}+4 cm $<r<R_{\mathrm{i}}$+10 cm respecto la máxima es muy superior al 1 %. Además, el médico del grupo ve inviable usar cilindros de lucita con 9 mm de grosor para IORT. Así pues, descartamos esta opción.

b. Lámina transversal de Al

Se planteó también de poner una lámina transversal (ver Figura 1.40) de aluminio Al (Z=13) a añadir justo en el momento de la irradiación. Los resultados para la dosis absorbida en agua completamente absorbente con una incertidumbre relativa total del 1 % se muestran en las Figuras 1.41, 1.42, 1.43 y 1.44 para los diferentes casos tratados.

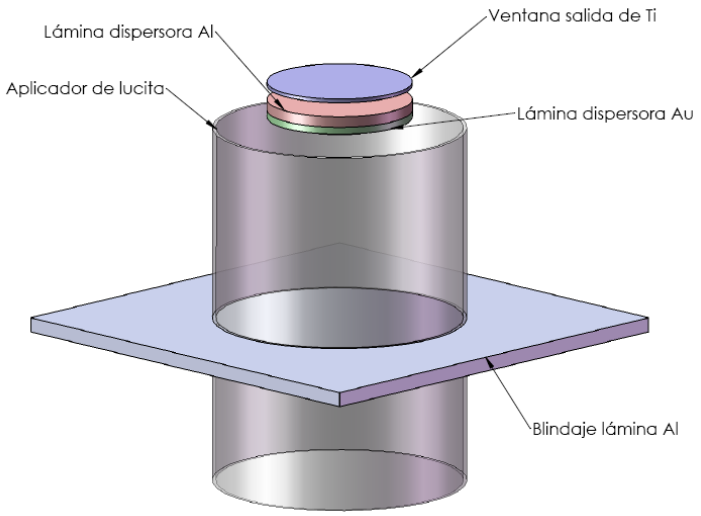

Figura 1.40: *Esquema del blindaje con una lámina transversal de Al.*

- **Caso 1**. Lámina de Al. z = -30 cm, grosor = 0.5 cm

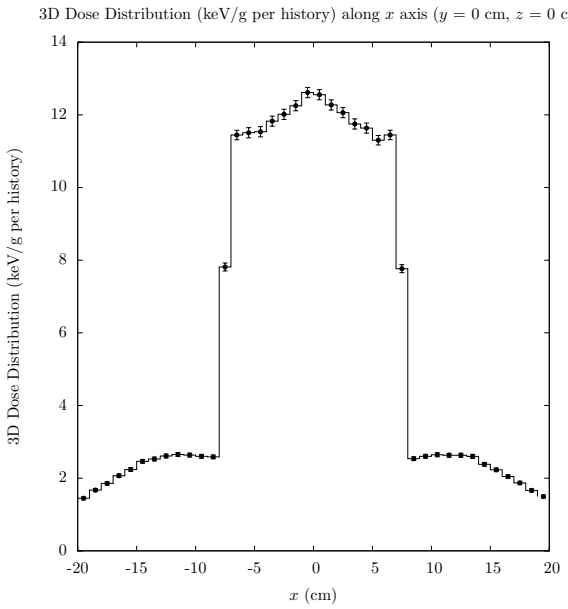

Figura 1.41: *Blindaje. Lámina transversal de Al. Perfil lateral en x para una lámina de Al a z=-30 cm y un grosor de 0.5 cm.*

- **Caso 2**. Lámina de Al. z = -30 cm, grosor = 2 cm

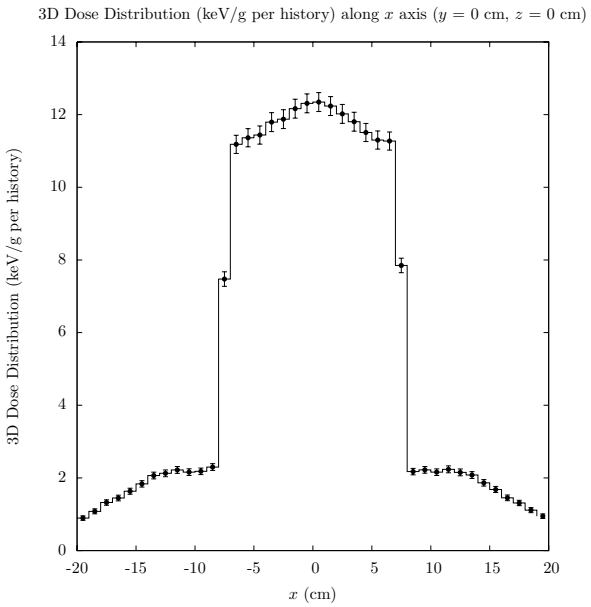

Figura 1.42: *Blindaje. Lámina transversal de Al. Perfil lateral en x para una lámina de Al a z=-30 cm y un grosor de 2 cm.*

- **Caso 3**. Lámina de Al. z = -5 cm, grosor = 1 cm

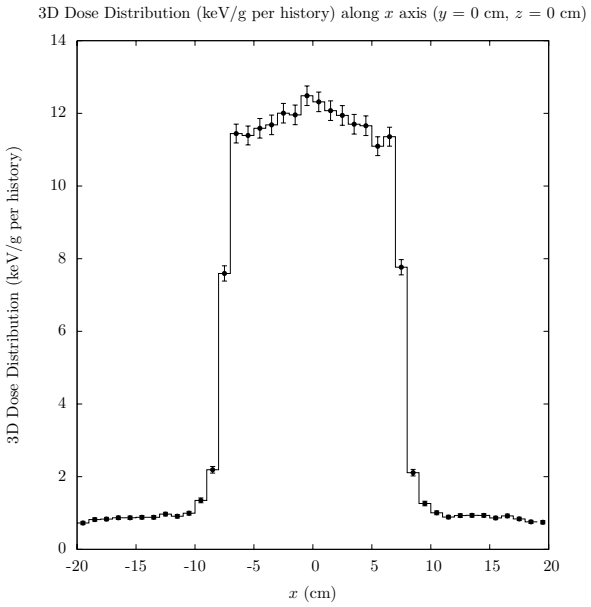

Figura 1.43: *Blindaje. Lámina transversal de Al. Perfil lateral en x para una lámina de Al z=-5 cm y un grosor de 1 cm.*

- **Caso 4**. Lámina de Al. z = -5 cm, grosor = 2 cm

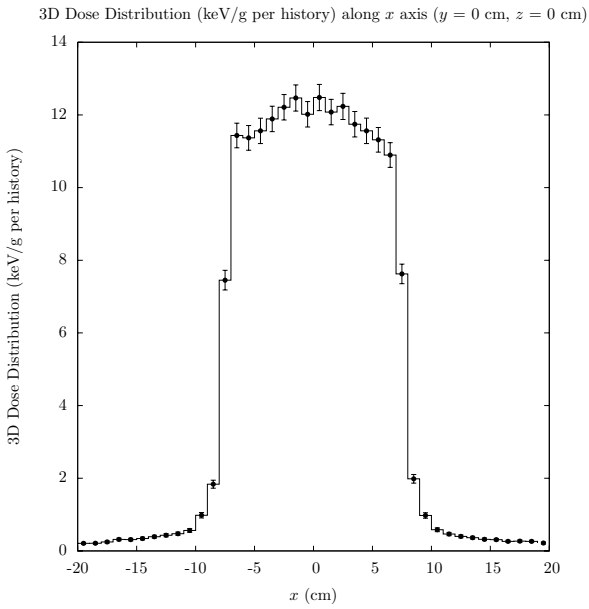

Figura 1.44: *Blindaje. Lámina transversal de Al. Perfil lateral en x para una lámina de Al z=-5 cm con y un grosor de 2 cm.*

También hemos probado con una lámina de Pb a 5 y 30 cm del maniquí de agua con grosores de 0.2 cm y 0.4 cm y los resultados son parecidos a los presentados.

En todos los casos, la relación dosis a 2 cm del radio interno del aplicador respecto la dosis máxima (sin considerar la contribución de los fotones y suponiendo que en el campo de irradiación la distribución de dosis es plana) es del 23 % en el caso 1, del 19 % en el caso 2, del 11 % en el caso 3 y del 9 % en el caso 4. Por tanto, descartamos también este tipo de blindaje porqué no se cumplen los requisitos dosimétricos y porqué es un blindaje muy aparatoso.

c. Tubo de lucita y latón

Otra posibilidad para el blindaje es hacer la parte de arriba del tubo con latón y parte de abajo con lucita (para poder visualizar la zona a irradiar). Vemos un esquema de la geometría en la Figura 1.45.

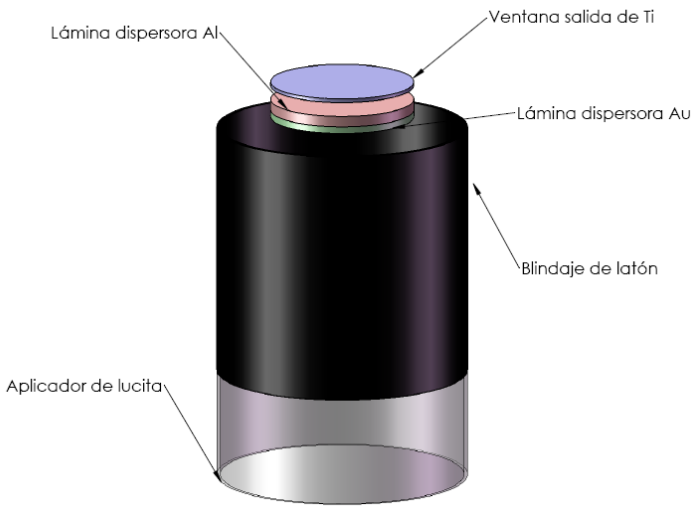

Figura 1.45: *Esquema de blindaje con un tubo compuesto de lucita y latón.*

El estudio de la distribución de dosis tridimensional para cada caso es el siguiente:

- **Caso 1**. Lucita: de z=0 (maniquí de agua) al plano z=-15 cm. Latón: del plano z=-15 al plano z=-60 cm (grosor de 3 mm) (Figuras 1.46 y 1.47).

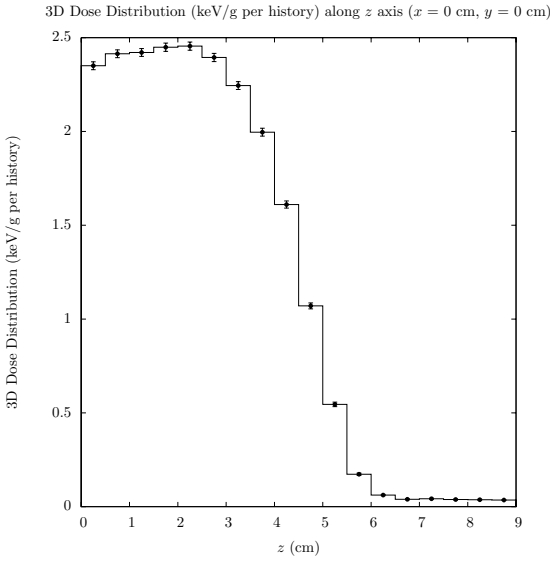

Figura 1.46: *Blindaje. Tubo de lucita y latón. Dosis en profundidad para una el Caso 1.*

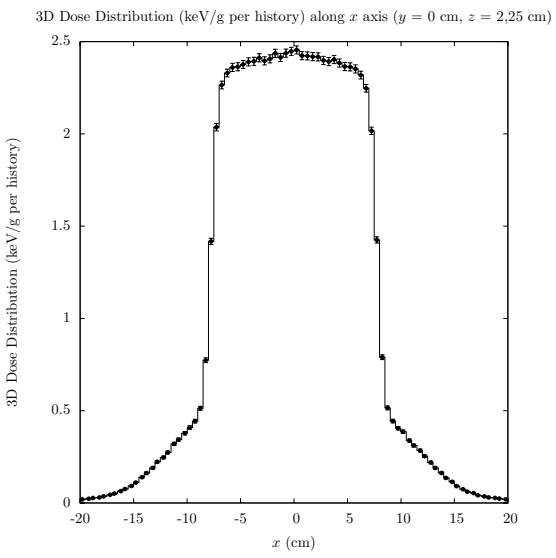

Figura 1.47: *Blindaje. Tubo de lucita y latón. Perfil lateral en x a $d_{\max}$ para una el Caso 1.*

- **Caso 2**. Lucita: de z=0 (maniquí de agua) al plano z=-10 cm. Latón: del plano z=-10 al plano z=-60 cm (grosor de 5 mm) (Figuras 1.48 y 1.49).

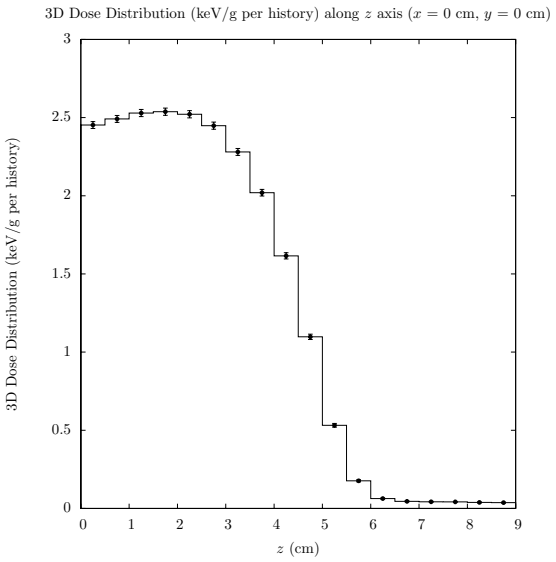

Figura 1.48: *Blindaje. Tubo de lucita y latón. Dosis en profundidad para una el Caso 2.*

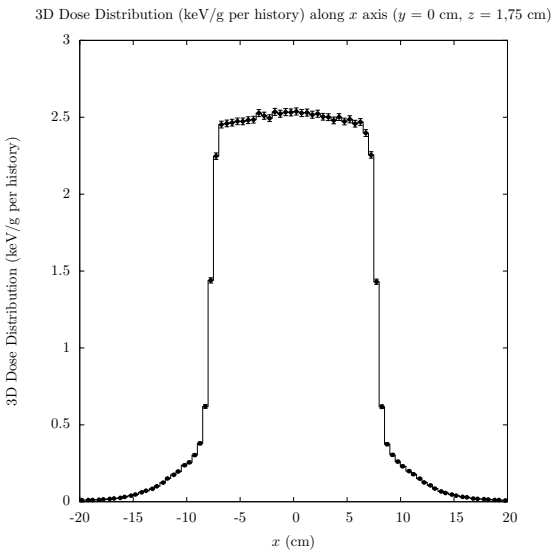

Figura 1.49: *Blindaje. Tubo de lucita y latón. Perfil lateral en x a $d_{\max}$ para una el Caso 2.*

Para el Caso 1, la dosis periférica es 18.1 ± 0.4 %, mientras que para el Caso 2 es 12.1 ± 0.3 %. Por lo tanto, en ambos casos no se cumplen las normas. Si alargamos el latón sería imposible ver nada y por tanto, pierde sentido poner la lucita.

d. Tubo completo de latón

Sabemos que este caso no nos será útil porqué tenemos visibilidad nula. Aun así, veremos si un tubo entero de latón es suficiente para blindar toda la radiación. La Figura 1.50 muestra un dibujo de la geometría.

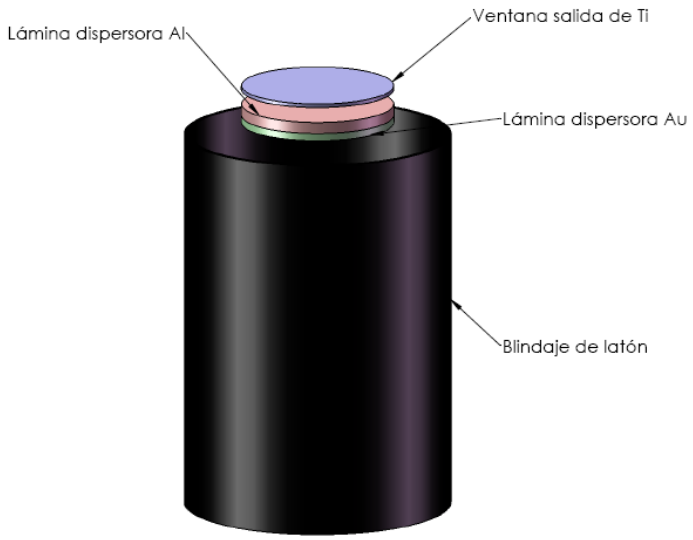

Figura 1.50: *Esquema de blindaje con un tubo de latón.*

- **Caso 1**. Tubo de latón con un grosor de 2 mm (Figuras 1.51 y 1.52).

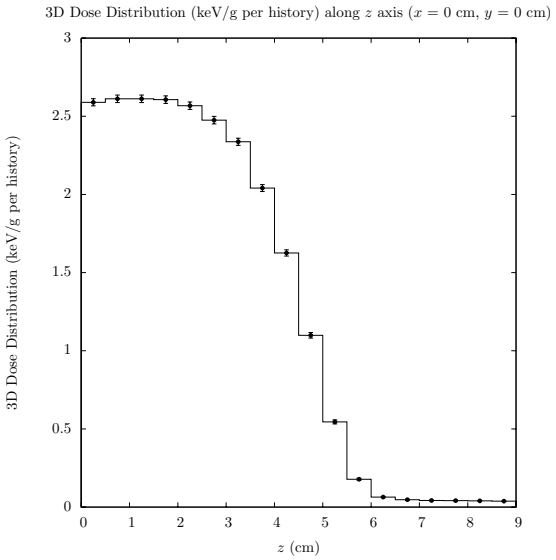

Figura 1.51: *Blindaje. Tubo de latón. Dosis en profundidad para el Caso 1.*

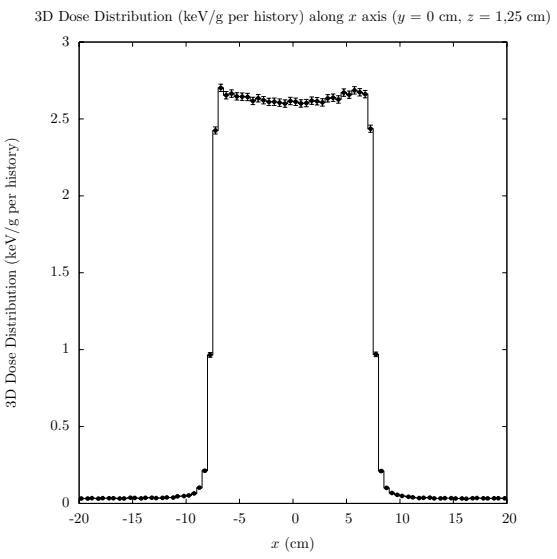

Figura 1.52: *Blindaje. Tubo de latón. Perfil lateral en x a $d_{\max}$ para el Caso 1.*

- **Caso 2**. Tubo de latón con un grosor de 3 mm (Figuras 1.53 y 1.54).

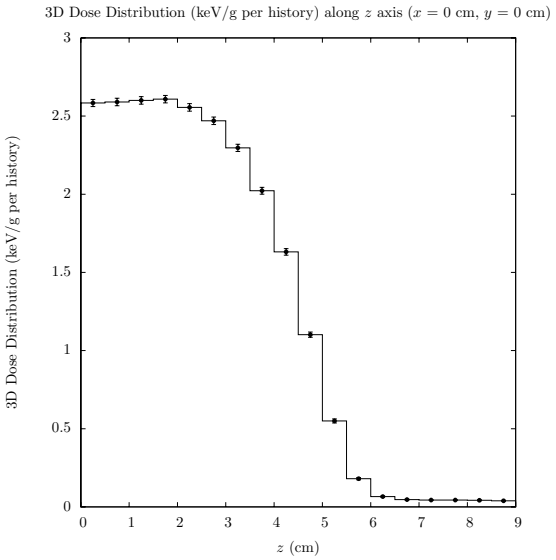

Figura 1.53: *Blindaje. Tubo de latón. Dosis en profundidad para el Caso 2.*

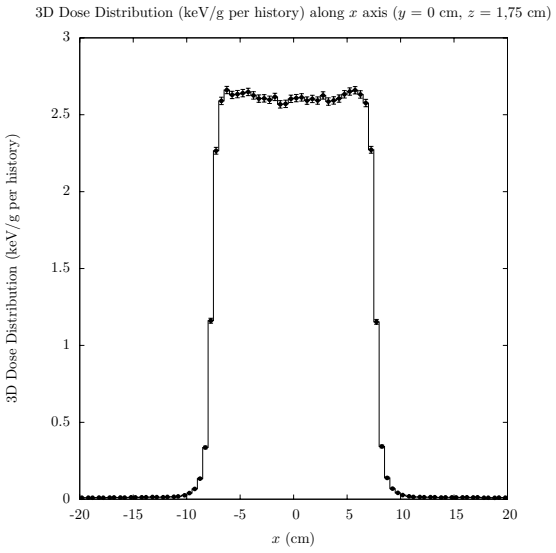

Figura 1.54: *Blindaje. Tubo de latón. Perfil lateral en x a $d_{\max}$ para el Caso 2.*

Los resultados para los dos casos son bastante buenos ya que la dosis fuera del paciente se ve reducida hasta cumplir las normativas en el Caso 2 (en el Caso 1 roza los límites). No hemos hecho ningún estudio dosimétrico completo con esta geometría porqué el médico del grupo asegura que es importante la transparencia del aplicador. Aun así, es un primer paso para el diseño final del blindaje, que veremos en el siguiente punto.

e. Un tubo de lucita y un tubo externo de latón

Si queremos tener un aplicador transparente para ver las estructuras a irradiar y también queremos un buen blindaje, nuestra propuesta es utilizar un aplicador de lucita y uno de latón concéntricos de la forma que se muestra en la Figura 1.55. Así pues, cuando el doctor quiera ver la zona a irradiar, el aplicador de latón se deslizaría encima del de lucita (Figura 1.56).

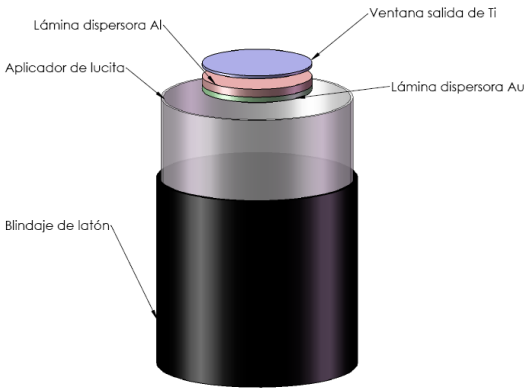

Figura 1.55: *Esquema de blindaje durante la irradiación.*

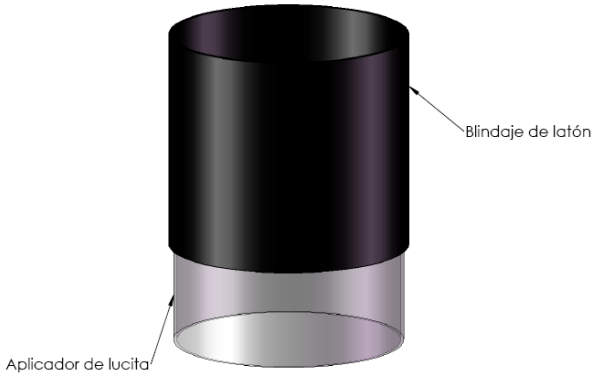

Figura 1.56: *Esquema de blindaje antes de la irradiación.*

La longitud de deslizamiento será variable en función del diseño final del RTM, pero seguro que serán unos centímetros. Además, acortemos la longitud del cilindro de latón por la parte de arriba (ver Figura 1.55) para poder subir algunos centímetros más. La Tabla 1.11 presenta el valor de todas las características dosimétricas para:

- **Caso 1**. Tubo de latón hasta z=-48 cm con un grosor de 3 mm.
- **Caso 2**. Tubo de latón hasta z=-48 cm con un grosor de 4 mm.

Parámetro	**Normativa**	**Caso 1**	**Caso 2**
$d_{\max}$		0.25 ± 0.25 cm	0.25 ± 0.25 cm
$D_{d_{10}+10\text{cm}}/D_0(d_{\max})$	1 %	0.80 ± 0.04 %	0.80 ± 0.04 %
Uniformidad	10 %	2.90 ± 0.04 %	2.93 ± 0.04 %
Dosis superficial	>85 %	>99 %	>99 %
Dosis periférica a $d_{\max}$	<5 %	0.81 ± 0.9 %	0.48 ± 0.04 %
Dosis periférica a d_{90}	<5 %	2.43 ± 0.08 %	2.38 ± 0.08 %
IEC 60601-2-1			
R_i+2 cm <r <R_i+10 cm (a $d_{\max}$)	Máxima <10 %	0.549 ± 0.003 %	0.337 ± 0.003 %
R_i+4 cm <r <R_i+10 cm (a $d_{\max}$)	Media <1 %	0.500 ± 0.003 %	0.339 ± 0.003 %
R_i+10 cm<r <R_i+ 200 cm (a $d_{\max}$)	Máxima <0.2 %	1.299 ± 0.006 %	0.280 ± 0.002 %
R_i+10 cm<r <R_i+ 200 cm (a $d_{\max}$)	Media <0.1 %	0.437 ± 0.002 %	0.280 ± 0.001 %

Tabla 1.11: *Blindaje. Tubos de lucita y latón concéntricos. Características dosimétricas (en rojo, valores que no cumplen la normativa).*

Las Figuras 1.57, 1.58, 1.59 y 1.60 muestran la dosis en profundidad, la dosis lateral y las curvas de isodosis (sólo para el Caso 1 ya que para el Caso 2 los valores son similares). Vemos que la diferencia entre tener blindaje y no tenerlo (ver Sección 1.7.2) es bastante significativa.

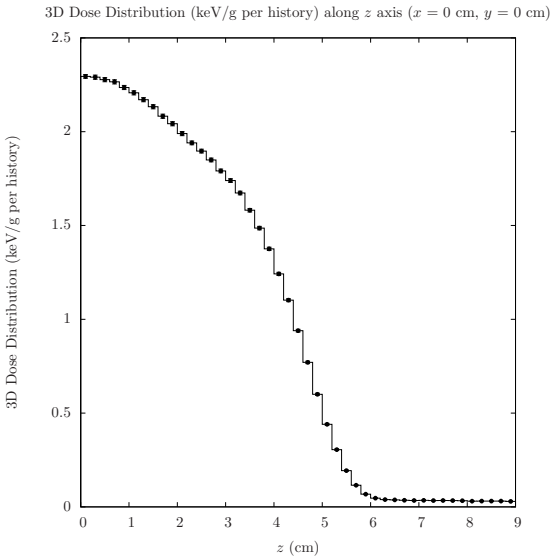

Figura 1.57: *Blindaje. Tubos de latón y lucita. Dosis en profundidad para el Caso 1.*

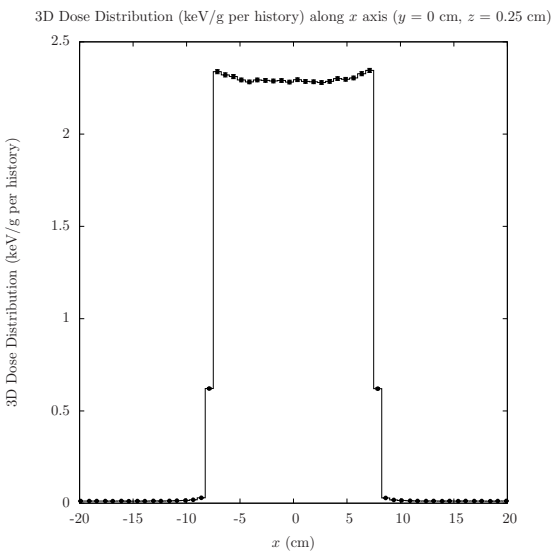

Figura 1.58: *Blindaje. Tubos de latón y lucita. Perfil lateral en x a $d_{\max}$ para el Caso 1.*

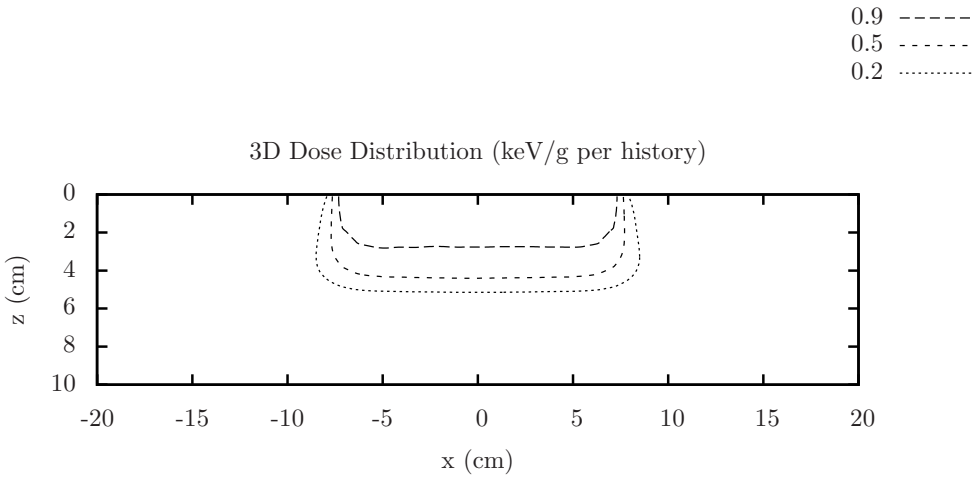

Figura 1.59: *Blindaje. Tubos de latón y lucita. Curva de isodosis x-z (y=0) para el Caso 1.*

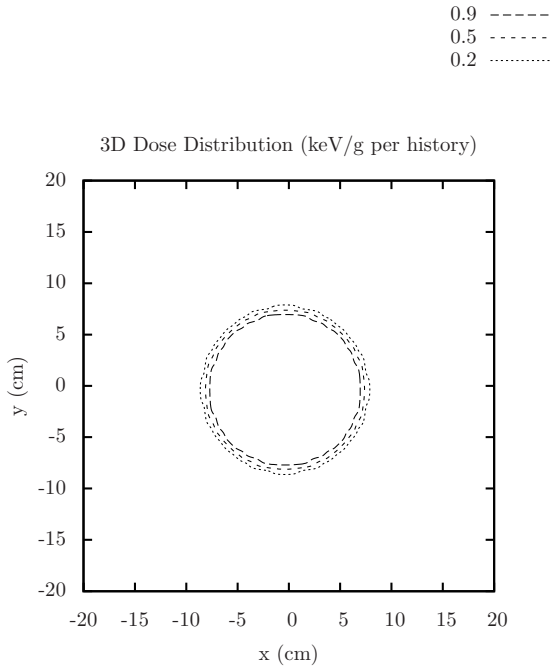

Figura 1.60: *Blindaje. Tubos de latón y lucita.*
Curva de isodosis x-y ($z = d_{\max}$) para el Caso 1.

Hemos visto que en ninguno de los casos se cumple la normativa IEC 60601-2-1 para distancias grandes (Figuras 1.61 y 1.62).

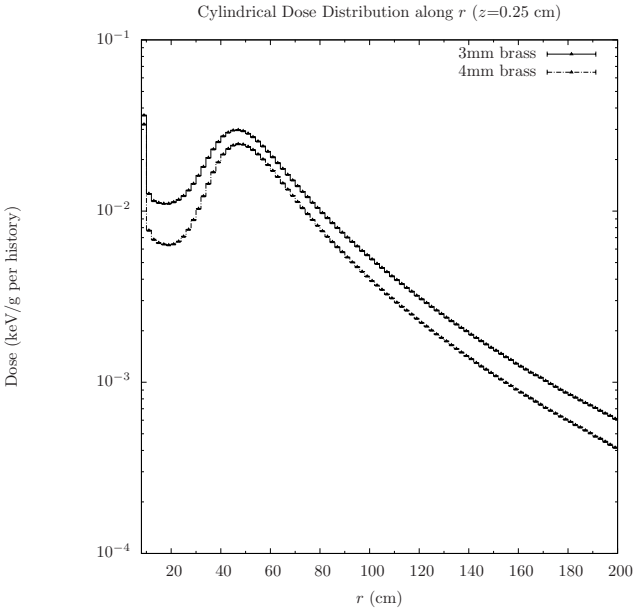

Figura 1.61: *Blindaje. Tubos de latón y lucita. Perfil lateral en $d_{\max}$. Normativa IEC 60601-2-1.*

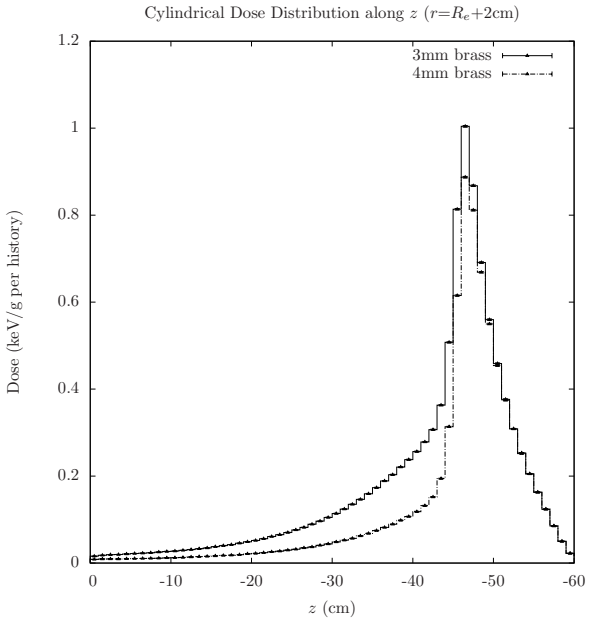

Figura 1.62: *Blindaje. Tubos de latón y lucita. Dosis absorbida en z en aire para* $r = R_e + 2$ *cm. IEC 60601-2-1.*

Deducimos que la normativa no se cumple debido a las partículas que se escapan por los 12 cm de arriba (donde sólo hay lucita y no latón) ya que 1 mm de latón disminuye muy poco la dosis. Por lo tanto, intentaremos alargar el tubo de latón unos centímetros. Antes, pero, estudiaremos el grosor de las láminas dispersoras.

1.7.4. Grosor de las láminas dispersoras

Después de este estudio del blindaje, vamos a centrarnos en el grosor de las láminas dispersoras.

a. Estudio inicial

La elección del grosor de las láminas nos va a afectar en la dosis periférica, pero sobretodo nos va a influenciar en el cálculo de la uniformidad. Lo que nos interesa son láminas lo más finas posibles (tendremos menos dispersión y por lo tanto, menos radiación fuera del paciente), siempre que se cumplan los requisitos. Además, el tener láminas finas hace que tengamos una eficiencia mayor ya que la radiación no estará tan degradada.

Para saber en el rango en el cual nos movemos, empezaremos con un estudio sencillo: no consideraremos blindaje y dejaremos la lámina de Al tal y como está, modificando sólo el grosor de la lámina de Au. Los grosores de las láminas usados para este estudio se presentan en la Tabla 1.12.

Parámetro	**Notación**	**12 MeV [cm]**
Grosor de la lámina dispersora de Al	E_{Al}	0.0800
Grosor de la lámina dispersora de Au	E_{Au}	0.005 y 0.01

Tabla 1.12: *Estudio inicial. Grosor de las láminas dispersoras.*

Los resultados para ambos casos se ven en las Figuras 1.63, 1.64, 1.65 y 1.66. En el caso de un grosor de la lámina de Au de 50 μm, la uniformidad es 12.0 $\pm$ 0.9 %. En caso de 100 μm, en cambio, la uniformidad es 3.2 $\pm$ 0.7 % (<5 %, se cumple la normativa). Por lo tanto, el grosor de láminas óptimo estará entre 50 μm y 100 μm.

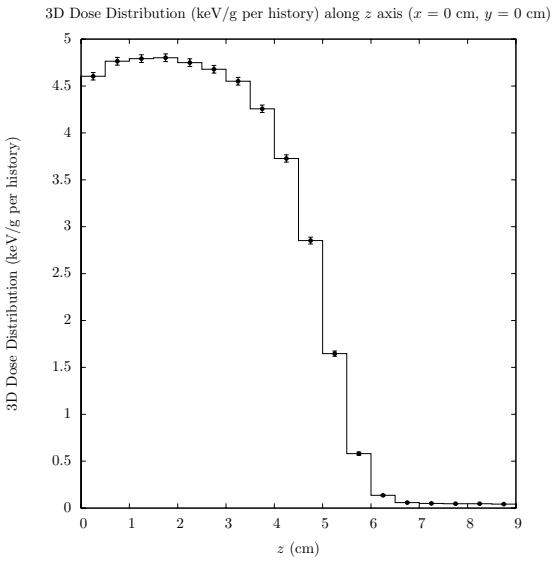

Figura 1.63: *Grosor de las láminas. Au = 50 μm. Dosis en profundidad.*

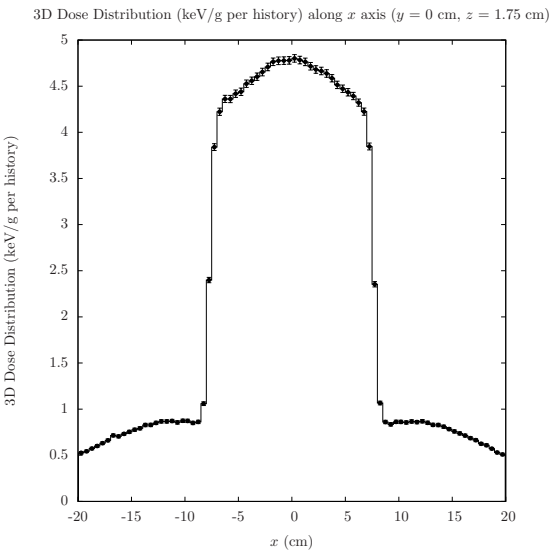

Figura 1.64: *Grosor de las láminas. Au = 50 μm. Perfil lateral en x a $d_{\max}$.*

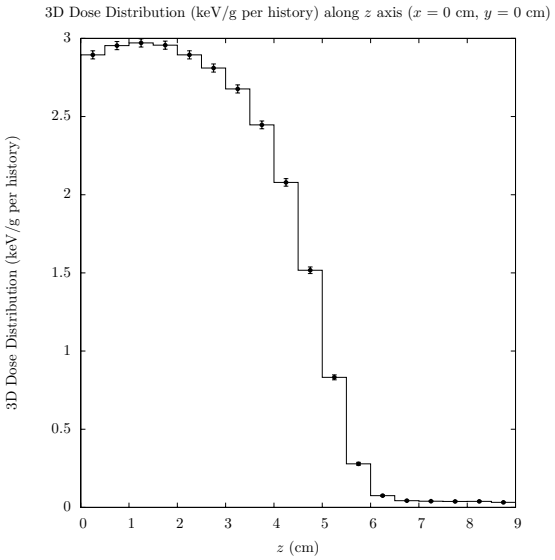

Figura 1.65: *Grosor de las láminas.*

Au = 100 µm. Dosis en profundidad.

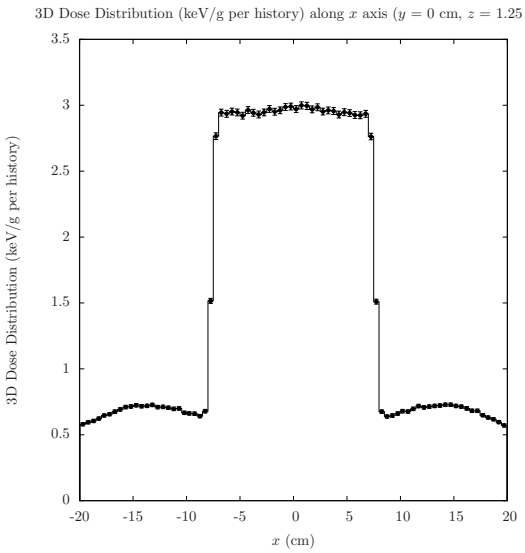

Figura 1.66: *Grosor de las láminas.*

Au = 100 µm. Perfil lateral en x a $d_{\max}$.

1.7.5. Solución geométrica final

Geometría

Parámetro	**Notación**	**12 MeV [cm]**
Diámetro interno aplicador	$D_{\mathrm{i,a}}$	15
Longitud del aplicador	H_{a}	60
Grosor del aplicador	E_{a}	0.3
Distancia entre la fuente y la ventana de salida	H_{ti}	0.5
Grosor de la ventana de salida de Ti	E_{ti}	0.002
Distancia entre la fuente y la base de las láminas dispersoras	H_{l}	1
Diámetro de las láminas dispersoras	D_{l}	4
Grosor de la lámina dispersora de Al	E_{al}	0.080
Grosor de la lámina dispersora de Au	E_{au}	0.080
Longitud del blindaje	H_{b}	55
Grosor de blindaje	E_{b}	0.4

Tabla 1.13: *Parámetros de la geometría final.*

Después de muchas pruebas y viendo los resultados anteriores, encontramos un caso (ver geometría en la Tabla 1.13) donde se llegan a cumplir todas las normas dosimétricas, manteniendo una buena uniformidad y minimizando el valor de las láminas dispersoras. Vamos a analizar este caso en detalle, presentando todos los resultados y las gráficas usadas para obtenerlos.

Condiciones dosimétricas

1. **PDD**

   A partir de la curva PDD (Figura 1.67) obtenemos la Tabla 1.14.

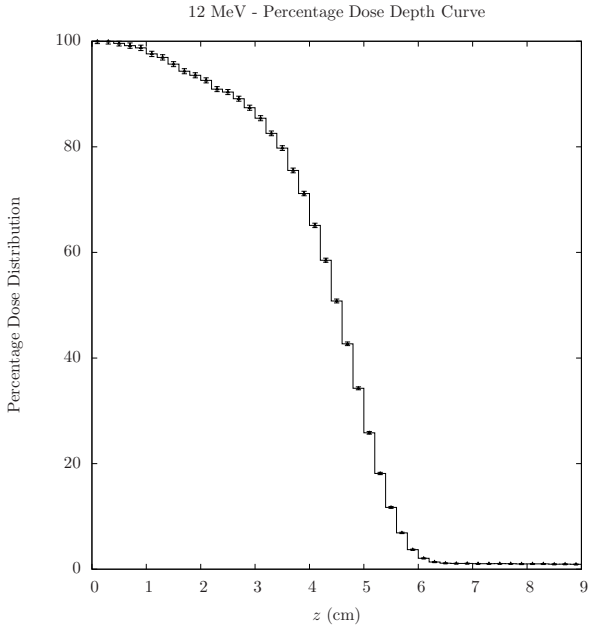

Figura 1.67: *Solución geométrica final. PDD.*

Parámetro	Normativa	Valor
$D_0(d_{\max})$		3.55 ± 0.02 keV/g por historia
$d_{\max}$		0.1 ± 0.1 cm
d_{90}		2.7 ± 0.1 cm
d_{50}		4.7 ± 0.1 cm
d_{10}		5.9 ± 0.1 cm
R_{p}		5.7 ± 0.5 cm
$D_{d_{10}+10\mathrm{cm}}/D_0(d_{\max})$	1 %	0.57 ± 0.03 %

Tabla 1.14: *Solución geométrica final. Características dosimétricas obtenidas a partir de la curva PDD.*

2. **Perfil lateral de dosis**

En la Figura 1.68 se muestra el perfil lateral en x a diferentes profundidades z.

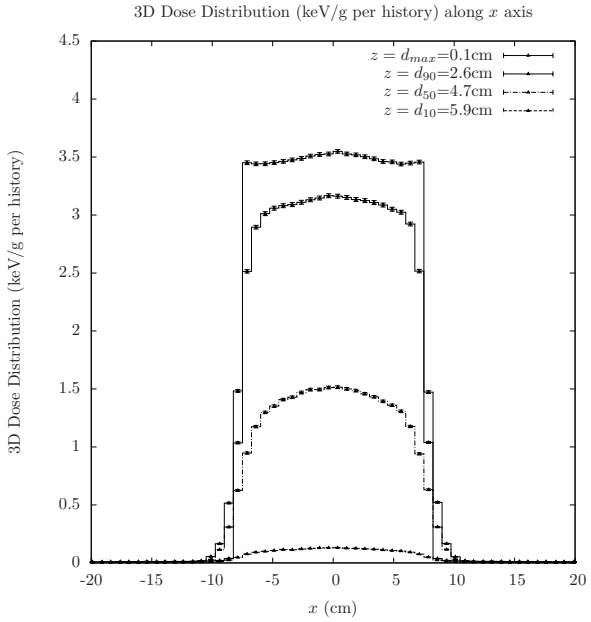

Figura 1.68: *Solución geométrica final. Perfil lateral en x a diferentes z.*

A partir de ella, podemos obtener las siguientes características dosimétricas (véase Tabla 1.15).

Parámetro	**Normativa**	**Valor**
Uniformidad	10 %	3.0 ± 0.5 %
Asimetría	2 %	<1.0 %
Dosis superficial	>85 %	>96 %
Dosis periférica a $d_{\max}$	<5 %	0.60 ± 0.04 %
Dosis periférica a d_{90}	<5 %	4.7 ± 0.1 %

Tabla 1.15: *Solución geométrica final. Características dosimétricas obtenidas a partir del perfil lateral.*

En las Figuras 1.69 y 1.70 se muestran las gráficas para comprobar el cumplimiento de la normativa IEC 60601-2-1 (véase Tabla 1.16 para resultados cuantitativos).

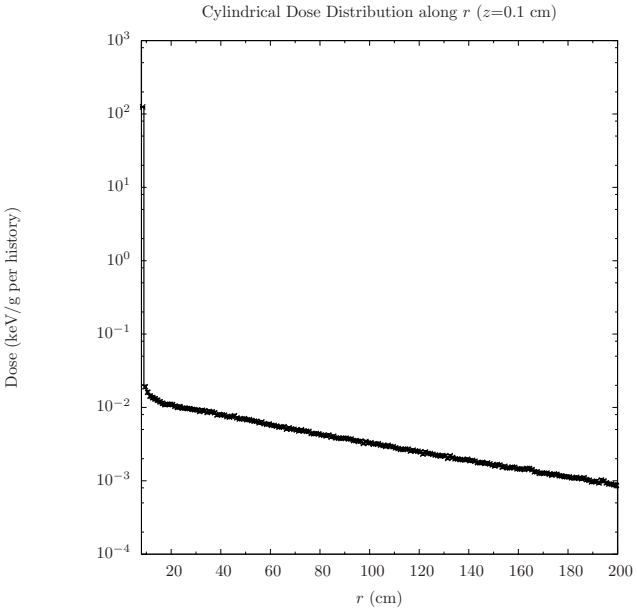

Figura 1.69: *Solución geométrica final. Perfil lateral en r a* $d_{\max}$.

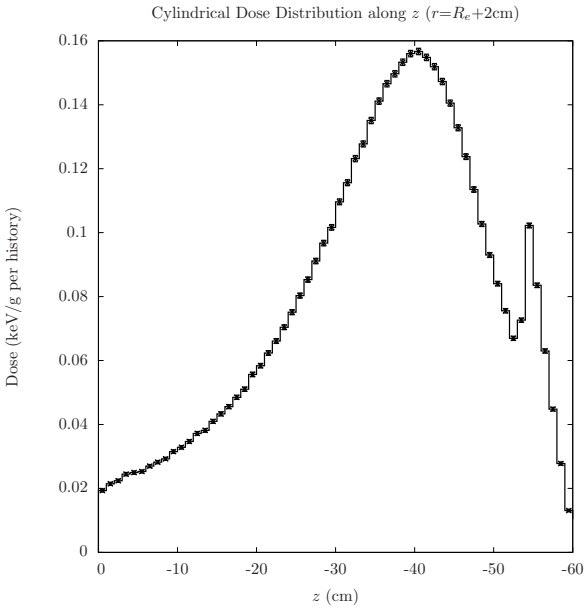

Figura 1.70: *Solución geométrica final. Dosis absorbida en z en aire para* $r = R_{\mathrm{e}}$*+2 cm.*

Vemos que el segundo pico de la Figura 1.70 corresponde al final del tubo de latón a una altura de z=-55 cm.

Radio r	**Normativa**	**Valor**
R_i+2 cm $<r<$ R_i+10 cm (a d_{max})	Máxima <10 %	0.59 ± 0.03 %
R_i+4 cm $<r<$ R_i+10 cm (a d_{max})	Media <1 %	0.36 ± 0.02 %
R_i+10 cm$<r<$ R_i+ 200 cm (a d_{max})	Máxima <0.2 %	0.03 ± 0.02 %
R_i+10 cm$<r<$ R_i+ 200 cm (a d_{max})	Media <0.1 %	0.102 ± 0.007 %
R_e + 2 cm ($\forall z < 0$)	Máxima <10 %	4.47 ± 0.09 %

Tabla 1.16: *Solución geométrica final. Normativa IEC 60601-2-1.*

3. Curvas de isodosis

La Tabla 1.17 muestra los valores del campo de irradiación y del tamaño de campo.

Parámetro	**Valor**
Campo de irradiación	7.5 cm
Tamaño de campo	7.75 cm

Tabla 1.17: *Solución geométrica final. Características dosimétricas obtenidas de las curvas de isodosis.*

Los valores de éstos parámetros se obtienen con las Figuras 1.71 y 1.72.

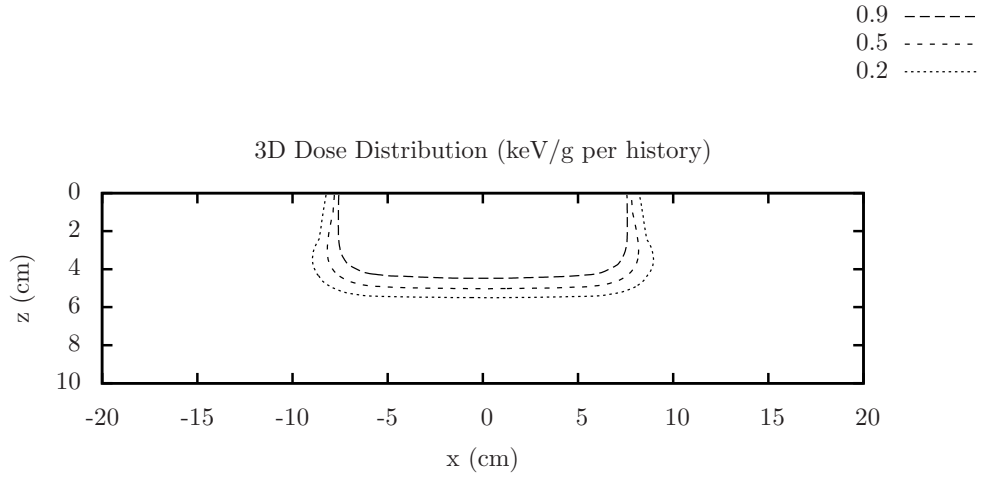

Figura 1.71: *Solución geométrica final. Curva de isodosis x-z (y=0).*

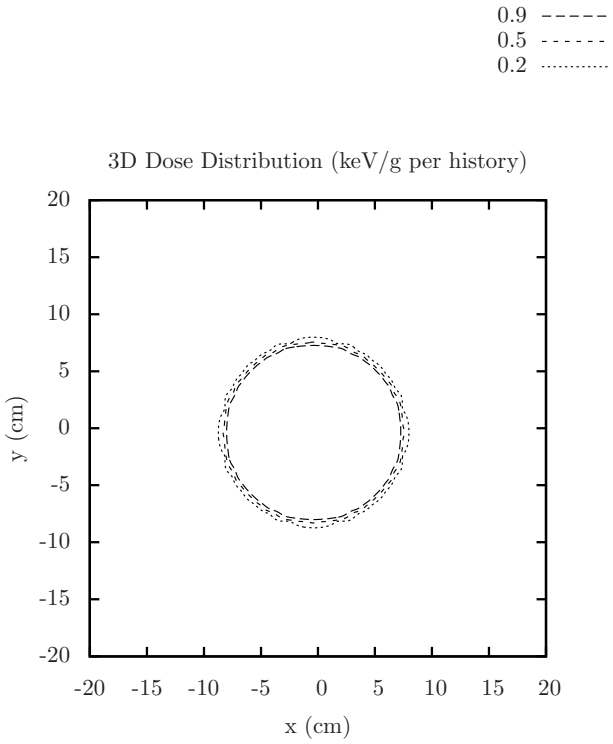

Figura 1.72: *Solución geométrica final. Curva de isodosis x-y ($z = d_{\max}$).*

Por lo tanto, la geometría propuesta es útil para IORT ya que cumple las normas y se mantiene la funcionalidad completa del aplicador (transparente y con capacidad del blindaje). No podemos disminuir más el grosor de las láminas porqué se empezaría a poner en juego la uniformidad y nos interesa que sea elevada. Tampoco podemos poner menos de 55 cm de latón porqué la condición en R_{i}+10 cm$<r<$ R_{i}+ 200 cm de la norma IEC 60601-2-1 no se cumpliría. Así pues, hemos conseguido nuestro objetivo, que era el de encontrar un aplicador adecuado y óptimo para IORT en el caso más complicado.

4. **Comparación con el caso sin blindaje**

En el presente apartado, compararemos el diseño final (Tabla 1.13) con y sin blindaje. De esta forma, justificaremos de nuevo la necesidad de blindaje.

En la Figura 1.73 se muestra la curva PDD con y sin blindaje.

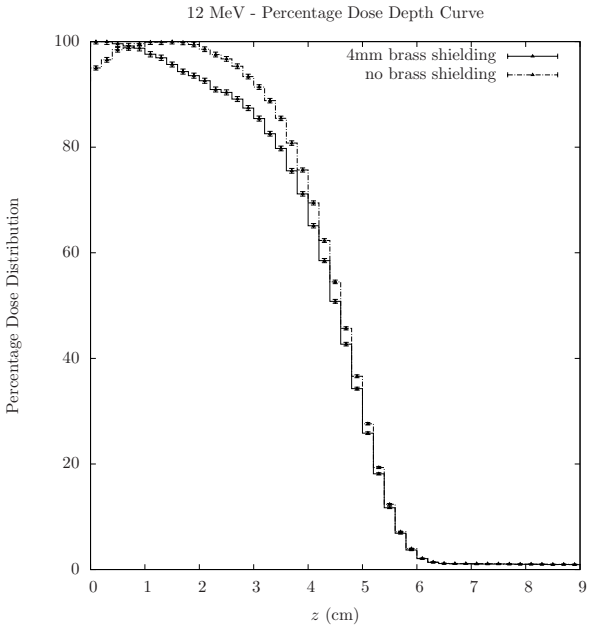

Figura 1.73: *Solución geométrica final. Comparación con y sin blindaje. PDD.*

La Figura 1.74 representa el perfil lateral en x a una profundidad $d_{\max}$ con y sin blindaje. Vemos que las diferencias son notables, sobretodo fuera del paciente.

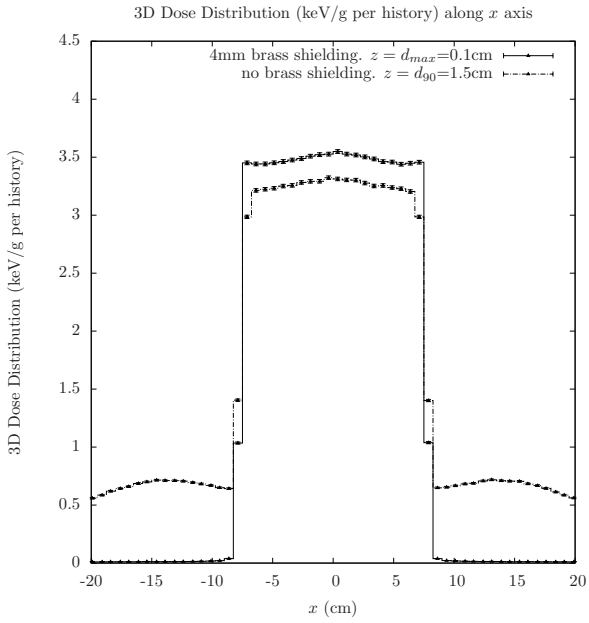

Figura 1.74: *Solución geométrica final. Comparación con y sin blindaje. Perfil lateral en x a $d_{\max}$.*

A continuación vemos el equivalente de la Figura 1.68, pero sin blindaje (Figura 1.75).

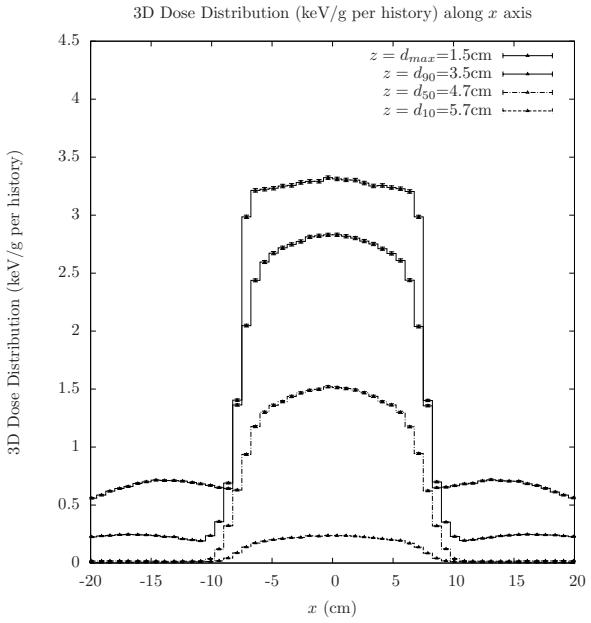

Figura 1.75: *Solución geométrica final. Caso sin blindaje. Perfil lateral en x a diferentes z.*

Para finalizar, vemos una comparativa del cumplimiento de la normativa IEC 60601-2-1 con y sin blindaje; este es un punto muy importante ya que éstas son las condiciones dosimétricas más difíciles de cumplir, además de la dosis periférica (véase Figuras 1.76 y 1.77).

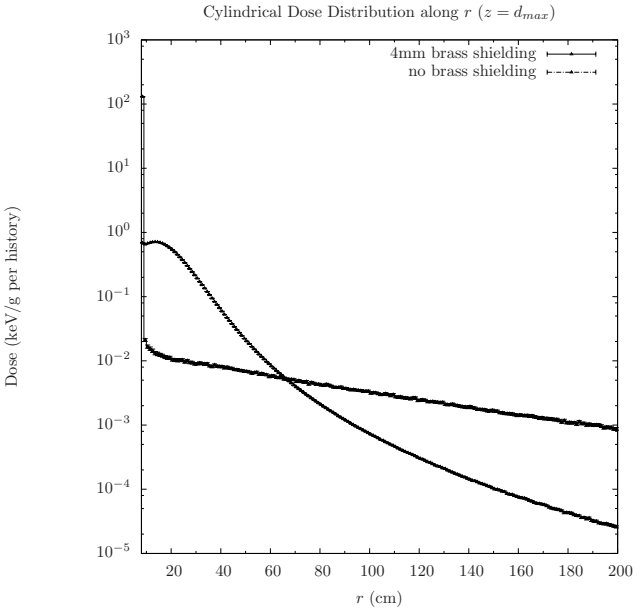

Figura 1.76: *Solución geométrica final. Comparación con y sin blindaje. Perfil lateral en r a $d_{\max}$.*

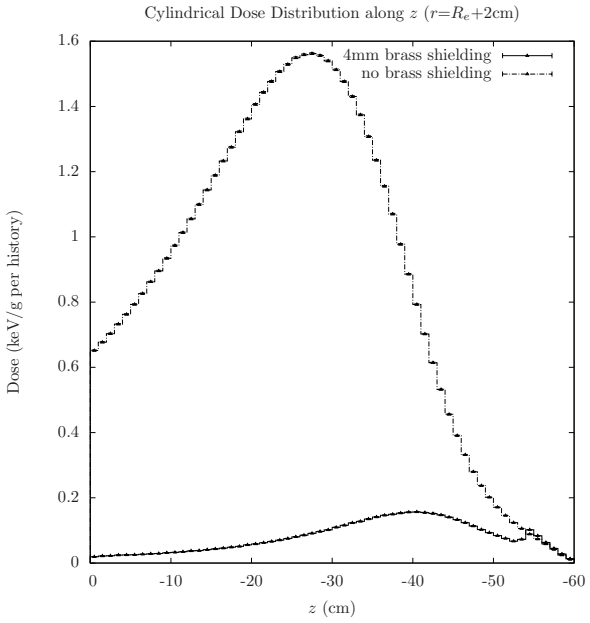

Figura 1.77: *Solución geométrica final. Comparación con y sin blindaje. Dosis en aire en* $r = R_e$*+2 cm.*

La Tabla 1.18 presenta las características dosimétricas sin blindaje de forma cuantitativa.

Parámetro	**Normativa**	**Valor**
$D_0(d_{max})$		3.31 ± 0.02 keV/g por historia
d_{max}		1.5 ± 0.1 cm
d_{90}		3.5 ± 0.1 cm
d_{50}		4.7 ± 0.1 cm
d_{10}		5.7 ± 0.1 cm
R_p		5.6 ± 0.5 cm
$D_{d_{10}+10cm}/D_0(d_{max})$	1 %	0.58 ± 0.03 %
Uniformidad	10 %	3.5 ± 0.5 %
Asimetría	2 %	<1.0 %
Dosis superficial	>85 %	>92 %
Dosis periférica a d_{max}	<5 %	19.6 ± 0.2 %
Dosis periférica a d_{90}	<5 %	10.8 ± 0.2 %
IEC 60601-2-1		
R_i+2 cm <r <R_i+10 cm (a d_{max})	Máxima <10 %	21.5 ± 0.2 %
R_i+4 cm <r <R_i+10 cm (a d_{max})	Media <1 %	20.5 ± 0.1 %
R_i+10 cm<r <R_i+ 200 cm (a d_{max})	Máxima <0.2 %	19.5 ± 0.1 %
R_i+10 cm<r <R_i+ 200 cm (a d_{max})	Media <0.1 %	1.2 ± 0.01 %
R_e + 2 cm ($\forall z < 0$)	Máxima <10 %	50 ± 1 %

Tabla 1.18: *Características dosimétricas sin blindaje (en rojo, valores que no cumplen la normativa).*

Vemos claramente que sin blindaje no podríamos haber cumplido nuestros objetivos y que por tanto, queda justificado el estudio realizado. Además, no sólo hemos conseguido tener un blindaje que cumpla las normativas, sino que permite mantener la visibilidad del tumor, propiedad importante en IORT.

Capítulo 2

Adaptación de penEasy para el uso de la base de datos de ficheros de espacios de fase del OIEA

2.1. Simulación Monte Carlo en radioterapia externa

La radioterapia utiliza la radiación ionizante para eliminar las células cancerosas depositando una dosis letal en un volumen localizado, afectando lo menos posible el tejido sano que lo rodea. La radioterapia con haces de fotones y electrones es una de las técnicas más difundidas para el control y el tratamiento de tumores malignos.

Los aceleradores lineales y las unidades de cobaltoterapia son ampliamente utilizados en los hospitales para el tratamiento de pacientes en radioterapia externa, donde la fuente de irradiación está a cierta distancia del paciente.

En el caso de las unidades de cobaltoterapia se usa la radiación emitida por el isótopo ^{60}Co. En el caso de los aceleradores lineales, en cambio, se aceleran electrones de energías entre 4 y 35 MeV utilizando microondas de alta frecuencia y se pueden usar en modalidad electrones o fotones, tal como vemos en la Figura 2.1.

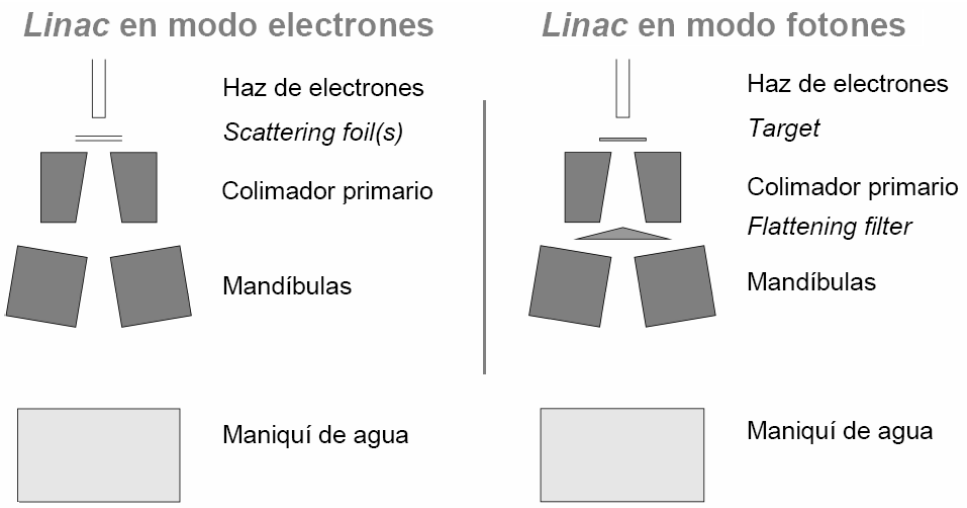

Figura 2.1: *Esquema básico de un linac en modalidad electrones y modalidad fotones (extraído de Ref. [31]).*

En modalidad electrones tenemos un haz de electrones que incide sobre una o más láminas dispersoras (*scattering foil(s)* en la Figura 2.1) para abrir el haz; posteriormente, pasan por el colimador primario y las mandíbulas. En modalidad fotones, los electrones inciden sobre un blanco (*target*) de número atómico elevado que emite bremsstrahlung. Los fotones generados pasan por un filtro aplanador (*flattening filter*) que se encarga de homogeneizar el haz.

Tal como hemos comentado, la simulación Monte Carlo es un método extensamente utilizado para la simulación por computador de los componentes de los aceleradores lineales, la generación y colimación del haz y la interacción de los haces de radiación con el paciente u otros materiales. En la Figura 2.2 se puede ver la geometría de una simulación de un cabezal de un acelerador lineal.

La simulación Monte Carlo de problemas relacionados con la radioterapia externa suele dividirse habitualmente en dos etapas con el fin de optimizar los tiempos de cálculo [33, 34]. Primero se caracteriza el campo de radiación creado por la fuente en un cierto plano, por ejemplo después de atravesar las mandíbulas. Para ello se almacena el estado de las partículas, ya sean fotones, electrones o positrones, que llegan a dicho plano en un fichero informático que se conoce como el fichero del espacio de fase (*phase-space file*) o PSF. La información recopilada incluye el tipo de partícula, su energía, posición, dirección de movimiento y otras características relevantes.

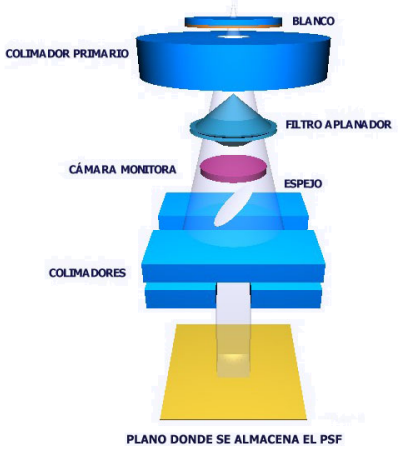

Figura 2.2: *Simulación de un acelerador lineal (extraído de Ref. [32]).*

Posteriormente se simula el transporte de las partículas almacenadas en el PSF a través de los modificadores del haz adicionales que haya y del objeto irradiado.

2.2. La base de datos del OIEA

La base de datos de PSFs del Organismo Internacional de la Energía Atómica (OIEA) Ref. [35] es apta para aplicaciones de investigación y contempla tanto aceleradores lineales como unidades de ^{60}Co. En principio, dicha base de datos no resulta adecuada para aplicaciones comerciales o de tratamiento de pacientes.

La base de datos está formada con las aportaciones realizadas por la propia comunidad científica, para lo cual se han definido unos criterios que emplea la comisión del OIEA encargada de evaluar los PSFs presentados. Para compatibilizar las aportaciones de grupos distintos, el OIEA ha definido un formato estandarizado y flexible.

2.2.1. Información contenida en la base de datos del OIEA

La base de datos del OIEA contiene, para cada espacio de fase incluido en la misma, dos ficheros, a saber, un fichero de cabecera y el PSF propiamente dicho.

El fichero de cabecera contiene información relativa al formato en el que se encuentran los datos del PSF simulado, así como todos los detalles necesarios para la interpretación del mismo. Su formato, que es de texto en ASCII, viene fijado por el OIEA y su contenido debe ser completado por los autores. Dicha información es principalmente la siguiente:

- Tipo de fichero con un código unívoco asignado por el OIEA
- Información sobre los autores / institución
- *Byte order* de los datos
- Longitud total del fichero
- Para cada variable, especifica si la variable es constante y cómo está guardada
- Número de historias simuladas
- Número de partículas de cada tipo
- Tipo de máquina donde se ha realizado la simulación
- Código de simulación utilizado para generar el PSF y su versión
- Descripción del acelerador o unidad de cobaltoterapia simulada
- Descripción de la fuente de radiación inicial (tipo de partícula, tamaño y forma del haz, dispersión angular y energética, etc.).
- Condiciones geométricas simuladas (tamaño de campo, distancia fuente-superficie, origen de coordenadas, etc.).
- Información relativa a cómo se ha realizado la simulación (modelos físicos de interacción, energías de absorción, técnicas de reducción de varianza, etc.)
- Referencia a la documentación de validación
- Información estadística del PSF (la energía mínima, máxima y media para cada partícula y las coordenadas X, Y y Z mínimas y máximas)
- Información complementaria

En el Apéndice D podemos ver un ejemplo básico de un fichero de cabecera, extraído de Ref. [36].

Por otra parte, el fichero PSF contendrá, en binario y formato estandarizado, información sobre las partículas que emergen del cabezal simulado. La Tabla 2.1 muestra las variables consideradas y su tipo.

Variable	**Tipo**
Posición en la dirección X [cm]	Real*4
Posición en la dirección Y [cm]	Real*4
Posición en la dirección Z [cm]	Real*4
Coseno director en X	Real*4
Coseno director en Y	Real*4
Energía cinética en MeV	Real*4
Peso estadístico	Real*4
Tipo de partícula	Integer*4
Signo del coseno director en Z	Logical
Indicación de si la partícula pertenece a una nueva historia	Logical
Espacio extra para variables enteras (por ejemplo, EGS LATCH, número incremental de historias, PENELOPE ILB, etc.)	n*(Integer*4); n≥0
Espacio extra para variables reales (por ejemplo, EGS ZLAST, etc.)	m*(Real*4); m≥0

Tabla 2.1: *Información contenida en el PSF (adaptado de Ref. [35]).*

El fichero de cabecera tendrá que especificar si todas las variables están guardadas en el PSF o si se tiene que usar un valor constante para una determinada variable. Por ejemplo, si todas las partículas del PSF se guardan en un cierto plano en Z, no será necesario escribir la posición en Z para todas las partículas ya que se guardará éste valor de forma genérica en el fichero de cabecera.

2.2.2. Envío y validación de PSFs

Para que un PSF sea aceptado para su inclusión en la base de datos del OIEA, sus autores deberán proporcionar, amén de los ficheros de cabecera y PSF, documentación relativa a cómo ha sido generado y también elementos para su validación. Además, se anima a los autores a

suministrar los ficheros de entrada empleados en la simulación y una especificación completa de la versión del código utilizado.

En cuanto a la validación del PSF, ésta debe basarse en el contraste con resultados experimentales. Dichas comparaciones pueden realizarse con medidas directas de la radiación emergente y/o medidas indirectas, por ejemplo, mediante la distribución de dosis absorbida. En el caso de la comparación con medidas indirectas, éstas pueden ser realizadas en aire y/o en un maniquí, para configuraciones múltiples, con distintos materiales o bien empleando distintos tipos de detectores.

Una comisión del OIEA evaluará las validaciones presentadas y juzgará la calidad de los PSFs enviados por la comunidad científica. El OIEA dispone de una página web de acceso público y gratuito donde se presentan el objetivo de la base de datos, las instrucciones para su utilización y el proceso para el envío de PSFs y su posterior evaluación. También pueden descargarse un conjunto de rutinas en C++ para la lectura y escritura de PSFs en el formato estandarizado mencionado anteriormente [36].

En la Figura 2.3 podemos ver un esquema de lo descrito en el presente apartado.

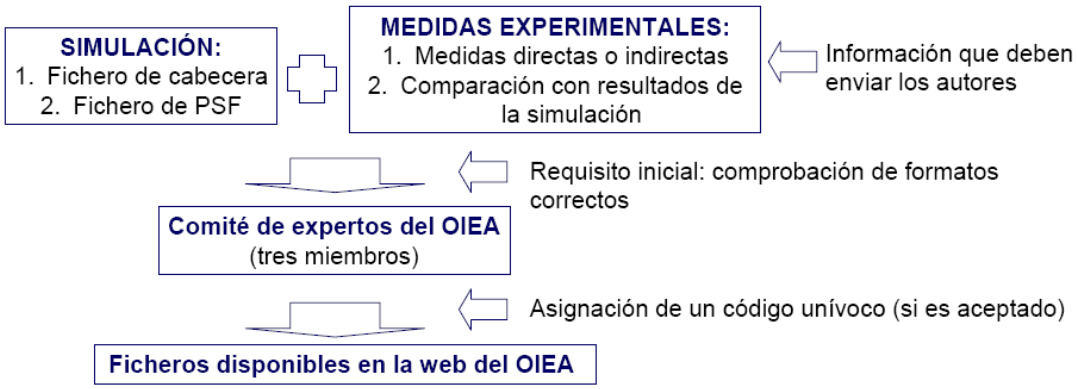

Figura 2.3: *Proceso de evaluación de un PSF.*

2.3. Subrutinas para PENELOPE

El código de simulación Monte Carlo (MC) PENELOPE [15] permite el transporte acoplado de fotones, electrones y positrones en un amplio intervalo de energías, en medios materiales

arbitrarios y geometrías complejas. El programa, que cuenta con varios centenares de usuarios en todo el mundo, ha sido empleado con éxito en la simulación de aceleradores lineales y unidades de cobalto [37, 38, 39].

Adicionalmente a PENELOPE, hemos comentado que puede disponerse también de penEasy [16], un programa principal genérico desarrollado para PENELOPE que provee modelos de fuentes de radiación y permite calcular diversas magnitudes de interés. El usuario de penEasy debe modificar los ficheros de entrada de datos para definir la fuente de radiación, la geometría del problema y la obtención de, por ejemplo, la distribución 3D/cilíndrica/radial de dosis absorbida, el espectro de fluencia de partículas en un determinado material, etc. Además, su estructura modular facilita la adaptación del código si el problema a resolver así lo requiere.

A fin de que el usuario de PENELOPE pueda utilizar la base de datos del OIEA sin necesidad de conocer los detalles de los interfaces involucrados, se han adaptado dos rutinas de penEasy, *sourcePhaseSpaceFile.f* y *tallyPhaseSpaceFile.f*, en Fortran, para la lectura y generación de PSFs, respectivamente.

En el Apéndice A podemos encontrar información más detallada de la simulación MC y del funcionamiento general de PENELOPE y penEasy.

2.3.1. Subrutina SourcePhaseSpaceFile

Esta parte de penEasy se utiliza para leer partículas de un PSF dado. A continuación se muestra la parte correspondiente del archivo de entrada.

```
[SECTION SOURCE PHASE SPACE FILE v.2008-06-01]
(OFF)                  STATUS (ON or OFF)
 0                     PHASE SPACE FILE FORMAT (0=STANDARD FORMAT,1=IAEA FORMAT)
(OIEApsf             ) PSF FILENAME
 1                     SPLITTING FACTOR
 0.0 0.0 0.0           EULER ANGLES [Rz,Ry,Rz](deg) TO ROTATE
 0.0 0.0 0.0           CARTESIAN COMPONENTS [DX,DY,DZ](cm) OF POSITION SHIFT
 1                     VALIDATE BEFORE SIMULATION (1=YES, MAY TAKE A WHILE; 0=NO)
```

```
 0.000e0              MAX PSF ENERGY (eV) (UNUSED IF VALIDATE=1 OR PSF FORMAT=1)
[END OF SPSF SECTION]
```

Vemos que como datos de entrada tenemos que proporcionar: el tipo de formato del PSF (si tenemos un PSF en formato estándar = 0, si está en formato OIEA = 1), el nombre del PSF, si queremos aplicar *particle splitting* (técnica de reducción de varianza donde una partícula se parte en S partículas idénticas con peso estadístico wght/S; se usa cuando una partícula se acerca a una región de interés para incrementar la probabilidad de tener una contribución), si queremos rotar o trasladar la posición de las partículas que leeremos, si queremos validar el PSF (no validación = 0, validación = 1) y su energía máxima en caso de que no queramos validarlo.

En el presente trabajo se ha modificado el programa para que pueda leer PSF en formato OIEA. Para ello, se ha adaptado el fichero *sourcePhaseSpaceFile.f* original y se han creado dos subrutinas nuevas:

- **IAEAread_ini (psfnam, validate, emax)** (295 líneas)
  Inicializa el programa para poder leer PSF.
  input: nombre del fichero PSF (**psfnam**) y si se quiere validar o no (**validate**).
  output: energía máxima del PSF (**emax**).
  Usa las siguientes rutinas de la OIEA en C++:

  - **iaea_new_source (source_read, psfnam, access_read, result)**
    Dados el nombre del fichero PSF (**psfnam**) y para qué lo queremos utilizar (**access_read**, en este caso para leer partículas), inicializa la fuente y le asigna una identificación unívoca al PSF (**source_read**). Si se produce algún error le asigna un número negativo a la variable **result**.
  - **iaea_get_max_particles (source_read, -1, npmaxiaea)**
    Nos proporciona el nombre máximo de partículas presentes en el PSF (**npmaxiaea**) con identificación **source_read**. Si se produce algun error, retorna un valor negativo de **npmaxiaea**.
  - **iaea_get_total_original_particles (source_read, ntopiaea)**
    Nos proporciona el nombre total de historias del PSF (**ntopiaea**) con identificación **source_read**. Si se produce algun error, retorna un valor negativo de **ntopiaea**.

- **iaea_get_constant_variable (source_read, i, constant, result)**
  Dada la identificación de la fuente y el índice de una variable (por ejemplo **i**=2 para Z), nos dice si esta variable ha sido almacenada como constante o no. En el caso que se haya guardado como constante, lee su valor en el fichero de cabecera (**constante**).

- **iaea_get_extra_numbers (source_read, extrafloat, extralong)**
  Hemos comentado que se pueden guardar variables extra en el PSF (ver Tabla 2.1). Esta subrutina nos dice cuantos enteros (**extralong**) y reales (**extrafloat**) extra se han guardado.

- **iaea_get_type_extra_variables (source_read, result, extralongtypes, extrafloattypes)**
  La presente rutina nos proporciona el tipo de cada variable extra guardada; es decir, nos dice que tenemos guardado exactamente (**extralongtypes**, **extrafloatypes**). Tiene que ser llamada tantas veces como variables extra tengamos guardadas y por tanto, después de **iaea_get_extra_numbers**.

- **iaea_get_maximum_energy (source_read, emaxiaea)**
  Da como resultado la energía máxima del PSF (**emaxiaea**) con identificación **source_read**.

Si en el *input* hemos especificado que queremos validar el PSF, se usarán también las siguientes rutinas de C++ de la OIEA:

- **iaea_check_file_size_byte_order (source_read, result)**
  Comprueba que el tamaño del fichero y el *byte order* es igual que los valores correspondientes del fichero de cabecera. De nuevo, si **result** es negativo es que se ha producido algún error.
  Además de hacer una estadística del PSF (número de partículas de cada tipo), también se hacen otras comprobaciones: que el tipo de partícula no tenga un valor sin sentido, que los valores de energía para todas las partículas no excedan los límites superiores e inferiores del programa, que la partícula tenga una dirección de movimiento lógica y que la energía máxima y el nombre de historias proporcionados por las rutinas de la OIEA sean correctos.

- **getparIAEA()** (120 líneas)

  Lee una partícula del PSF.

  Output: nos devuelve el estado de la partícula leída y si se han acabado las partículas o no.

  Rutinas de la OIEA en C++ utilizadas:

  - **call_iaea_get_particle (source_read, n_stat, kpariaea, eiaea, wghtiaea, xiaea, yiaea, ziaea, uiaea, viaea, wiaea, extra_floats, extra_ints)**

    Lee la siguiente partícula de la lista de la fuente con identificación **source_read**, dando como resultado si es una nueva historia o no (**n_stat**), el tipo de partícula (**kpariaea**), energía (**eiaea**), peso estadístico (**wghtiaea**), posición (**xiaea**, **yiaea**, **z-iaea**), dirección de movimiento (**uiaea**, **viaea**, **wiaea**) y otras variables extra guardadas (**extra_floats**, **extra_ints**). Si hay algún error durante la lectura, da como resultado un **n_stat** negativo. Si se ha alcanzado el final de fichero devuelve *false* mientras que si todavía quedan partículas *true*.

En la Figura 2.4 podemos ver un esquema de lo explicado.

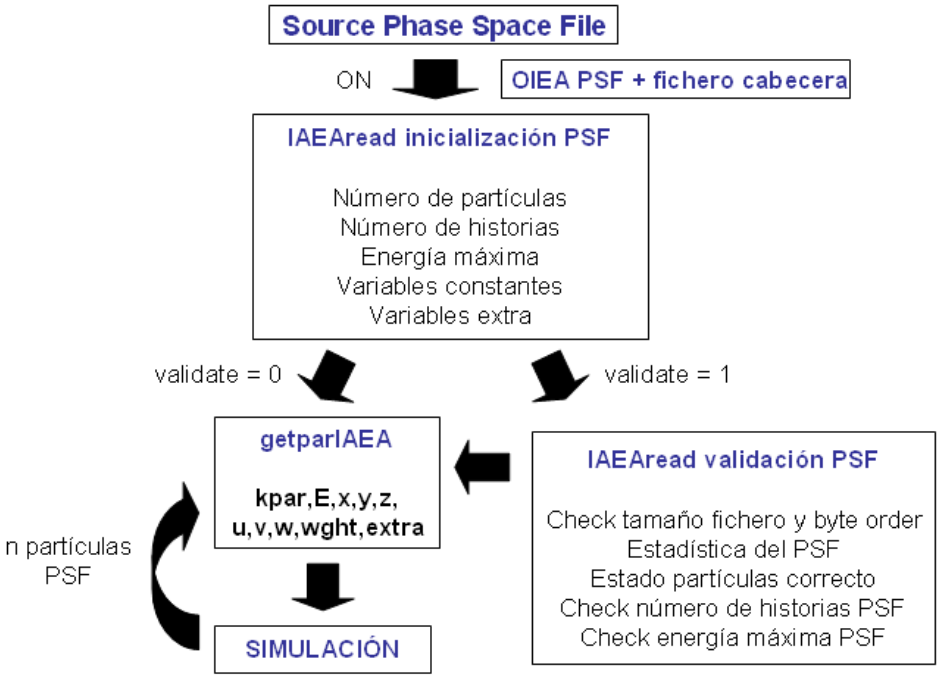

Figura 2.4: *Esquema de funcionamiento de la subrutina SourcePhaseSpaceFile.*

En el Apéndice E se puede encontrar el código completo con todo detalle.

2.3.2. Subrutina TallyPhaseSpaceFile

Esta parte de penEasy se utiliza para generar un nuevo PSF. Se muestra la parte correspondiente del archivo de entrada de penEasy.

```
[SECTION TALLY PHASE SPACE FILE v.2006-08-01]
(OFF)                 STATUS (ON or OFF)
 1                    DETECTION MATERIAL (NOT EQUAL 0)
 0                    PHASE SPACE FILE FORMAT (0=STANDARD FORMAT,1=IAEA FORMAT)
(OIEApsf            ) PSF FILENAME
[END OF PSF SECTION]
```

Vemos que como datos de entrada tenemos que proporcionar: el número de material donde queremos guardar el PSF, el tipo de formato del PSF (si tenemos un PSF en formato estándar = 0, si está en formato OIEA = 1) y el nombre del PSF.

En el presente trabajo se ha reformado el programa para que pueda leer PSF en formato OIEA. Para ello, se ha adaptado el fichero *tallyPhaseSpaceFile.f* original y se han creado tres subrutinas nuevas:

- **IAEAwrite_ini(buffer)** (65 líneas)
  Inicializa el programa para poder escribir en el PSF.
  input: nombre del fichero PSF (**buffer**)
  Usa las siguientes rutinas de la OIEA en C++:
  - **iaea_new_source (source_write, buffer, access_write, result)**
    Dados el nombre del fichero PSF (**buffer**) y para qué lo queremos utilizar (**access_write**, en este caso para escribir partículas), inicializa el PSF y le asigna una identificación unívoca al PSF (**source_write**). Si se produce algún error le asigna un número negativo a la variable **result**.
  - **iaea_set_extra_numbers (source_write, extrafloat, extralong)**
    Fija el número de variables extra enteras (**extralong**) y reales (**extrafloat**) que vamos

a guardar en el PSF con identificación **source_write**. En el caso de PSF generados con PENELOPE, **extralong** = 2, el dn (número incremental de historias) y ilb(5), y **extrafloat** = 0.

- **iaea_set_type_extralong_variable (source_write, ind, extralong_type)**
  Fija el tipo de variable extra guardada con dos parámetros: **ind** y (**extralong_type**). Para dn, **ind** = 0 y **extralong_type** = 1; para ilb(5), **ind** = 1, **extralong_type** = 3 (viene predeterminado por las rutinas en C++ de la OIEA). Si hay algún error al fijar las variables, da como resultado un valor de **extralong_type** negativo.

- **IAEAwrite** (60 líneas)
  Escribe una partícula en el PSF en formato OIEA.
  Usa las siguientes rutinas de la OIEA en C++:

  - **iaea_write_particle (source_write, n_stat, kpariaea, eiaea, wghtiaea, xiaea, yiaea, ziaea, uiaea, viaea, wiaea, extra_floats, extra_ints)**
    Escribe el estado de una partícula en el PSF con identificación **source_write**. Si se produce algún error, **n_stat** será negativo.

- **IAEAwrite_report** (30 líneas)
  Escribe el fichero de cabecera.
  Rutinas de la OIEA en C++ utilizadas:

  - **iaea_set_total_original_particles (source_write, ntopiaea)**
    Actualiza el número de historias (**ntopiaea**) cada vez que se hace un *update* del fichero de cabecera. Si se produce algun error, dará como resultado un valor de **ntopiaea** negativo.
  - **iaea_update_header (source_write, result)**
    Hace un *update* del fichero de cabecera del PSF con identificación **source_write**. Un valor de la variable **result** negativo muestra que se ha producido un error.

En el Apéndice F se puede encontrar el código completo. La Figura 2.5 muestra un esquema general de lo explicado.

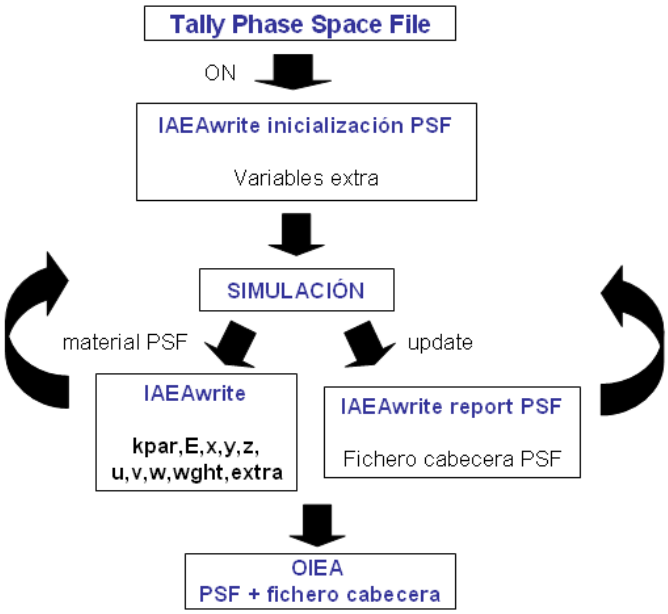

Figura 2.5: *Esquema de funcionamiento de la subrutina TallyPhaseSpaceFile.*

2.3.3. Instrucciones prácticas para su funcionamiento

Para realizar una simulación el usuario deberá disponer de los siguientes elementos:

- PENELOPE (distribuido por la *Nuclear Energy Agency* (NEA) [40].

- penEasy, junto con las rutinas adaptadas *sourcePhaseSpaceFile.f* y *tallyPhaseSpaceFile.f*, que deberán sustituir a las rutinas estándar de penEasy con el mismo nombre. Tanto penEasy como las rutinas desarrolladas son distribuidas a través de la web del *Institut de Tècniques Energètiques* [41].

- Rutinas del OIEA, escritas en C++: *iaea_ header*, *iaea_ record*, *iaea_ phsp*, *utilities* (archivos .cpp y .h más *config*.h) [36].

Así, la estructura general se presenta en en la Figura 2.6

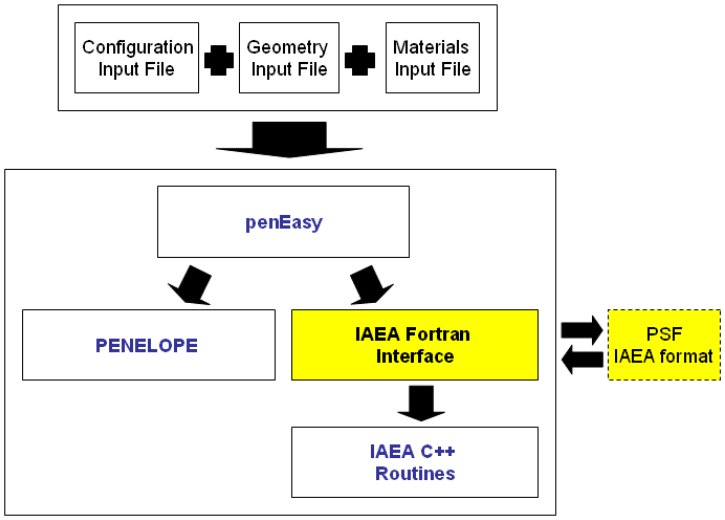

Figura 2.6: *Esquema general.*

Para leer y crear PSFs en formato OIEA se requiere la compilación conjunta de las rutinas de C++ y las de Fortran. De forma genérica, las instrucciones de compilación son las siguientes:

1.- Compilación de las rutinas de C++:

```
g++ -c -o iaea_header.o iaea_header.cpp
g++ -c -o iaea_record.o iaea_record.cpp
g++ -c -o iaea_phsp.o iaea_phsp.cpp
g++ -c -o utilities.o utilities.cpp
```

2.- Compilación de penEasy con un compilador Fortran:

```
compilador -c -o penEasy.o penEasy.f
```

3.- Link de las rutinas de C++ y las de Fortran:

```
compilador -o penEasyIAEA penEasy.o iaea_header.o iaea_record.o iaea_phsp.o
utilities.o -lm -lstdc++
```

El link precisa de las librerías estándar de C++ (-lstdc++) y las librerías matemáticas (-lm). Las rutinas desarrolladas han sido probadas con éxito con los compiladores g95, gfortran e Intel Fortran sobre GNU/Linux. En principio también deberían funcionar correctamente en cualquier otro sistema operativo que soporte los dos lenguajes. El uso de GNU Compiler Collection facilita su implementación ya que se distribuye libremente [42] e incluye el compilador de C++ (g++) y el compilador de Fortran (gfortran) en el mismo paquete. En este caso, la compilación puede realizarse también de forma compacta:

```
gfortran penEasy.f iaea_header.cpp iaea_phsp.cpp iaea_record.cpp utilities.cpp
-lm -lstdc++ -o penEasy_IAEA
```

La compilación conjunta de C++ y Fortran es más complicada en el caso de g95. Debido al tiempo dedicado para su funcionamiento en Linux, considero importante dar los pasos a seguir para su correcta instalación (especifico la versión de los programas porque no todas las versiones funcionan para compilación mixta Fortran-C++).

1. Descargar la versión 4.0.3 de GCC [42] y descomprimir el paquete.
2. Crear un directorio llamado g95 dentro del directorio fuente de GCC.
3. Entrar en la carpeta g95.
4. Instalar GCC:
   - ../configure –enable-languages=c
   - make (después de este paso, tenemos que tener las librerías libbackend.a y libgcc.a)
5. Descargar la versión 0.92 de g95 [43] y descomprimir el paquete en otra carpeta.
6. Entrar en el directorio fuente de g95.
7. Instalar g95:
   - ./configure –prefix=<directorio instalación g95>–with-gcc-dir=<path del directorio de g95 dentro del GCC>
   - make

- make install

8. Descomprimir la librería con nombre libf95.a-0.92.tgz presente en el directorio fuente de g95 y instalarla de la misma forma que el programa g95.

Una vez nos hemos asegurado que funciona, es posible que cuando compilemos tengamos que especificar los directorios donde se encuentran todas las librerías necesarias. Por ejemplo, si hemos instalado todos los programas en el directorio /home/imma, el último paso de la compilación se hará de la siguiente forma:

```
g95 -L/home/imma/g95-0.92/libf95.a-0.92 -L/home/imma/gcc-4.0.3/g95/gcc -o
penEasyIAEA penEasy.o iaea_header.o iaea_record.o iaea_phsp.o utilities.o
-lm -lstdc++
```

Una vez se haya creado el ejecutable (de nombre *penEasyIAEA* en el ejemplo de compilación), el usuario podrá fácilmente generar o leer PSFs en formato OIEA seleccionando la opción deseada en el archivo de entrada de penEasy.

Aunque las rutinas han sido diseñadas para leer PSFs simulados con PENELOPE permiten también la lectura de aquellos que hayan sido generados (en formato OIEA) con EGS Ref. [44, 45] uno de los sistemas de simulación MC más utilizados en la resolución de problemas relacionados con la física médica, u otros programas de simulación MC.

Conclusión

Capítulo 1

En el Capículo 1 hemos podido ver que es posible tener una geometría del sistema de modificadores del haz que haga que el microtrón de pista diseñado por la UPC sea apto para radioterapia intraoperatoria. Después de un análisis exhaustivo, se ha visto que el problema básico está en el blindaje ya que la normativa exige dosis muy bajas fuera de la zona a irradiar.

El diseño final propuesto, que consiste en una combinación de un aplicador de lucita más un aplicador externo deslizante de latón, hace el sistema adecuado para nuestro objetivo. Este sistema permite tener un aplicador transparente cuando el médico tiene que ver el tumor y un aplicador blindado durante la irradiación. Además, cumple con todas las normativas referentes a radioterapia intraoperatoria. También se han optimizado las láminas dispersoras usadas en el diseño de tal forma que la dosis en la zona a tratar sea uniforme y la eficiencia sea lo mayor posible.

Para estudios posteriores más detallados deberemos de tener en cuenta el efecto de un detector de radiación (cámara de ionización) para monitorizar la radiación que sale del microtrón y estudiar el cumplimiento de la normativa para el resto de energías, para aplicadores oblicuos y para aplicadores con radios diferentes. Aún así, se prevé su cumplimiento ya que el caso de 12 MeV y 15 cm de diámetro del aplicador es el más complejo y el que tiene más problemas de contaminación externa de radiación.

Capítulo 2

Las rutinas desarrolladas para su utilización con PENELOPE y penEasy permiten generar y leer ficheros de espacio de fases en el formato estandarizado de la base de datos del Organismo Internacional de la Energía Atómica sobre aceleradores lineales y unidades de cobaltoterapia. El acceso de los usuarios de PENELOPE a dicha base de datos permitirá evitar la repetición de simulaciones que ya hayan sido realizadas por otros miembros de la comunidad científica.

Además, el usuario puede utilizar el programa para simulaciones de cualquier tipo y mantener un tamaño de los ficheros de salida razonables, ya que el formato es en binario.

APÉNDICES

APÉNDICE A

Manual breve de PENELOPE y penEasy

A.1. PENELOPE

PENELOPE (PENetration and Energy Loss of Positrons and Electrons) es un paquete de subrutinas escrito en Fortran 77 que simula el transporte acoplado de fotones, electrones y positrones utilizando técnicas Monte Carlo [15, 46].

La complejidad de la interacción radiación-materia para una geometría y material arbitrarios implica que tengamos que utilizar técnicas numéricas ya que es imposible encontrar una expresión analítica para describir el transporte de partículas en un medio. En la actualidad, la simulación Monte Carlo es la mejor alternativa para tratar este tipo de problemas.

En el ámbito de la física médica, el número de trabajos publicados en los que la simulación Monte Carlo juega un papel fundamental no ha parado de aumentar en las últimas décadas [47, 48]. Junto con la creciente disponibilidad de potencia de cálculo, el desarrollo de códigos Monte Carlo cada vez más robustos y flexibles como PENELOPE está permitiendo avances notables en muchas aplicaciones relacionadas con la física médica.

PENELOPE, de código abierto y libre, está disponible a través de la *Nuclear Energy Agency (NEA)* [40].

A.1.1. La física de PENELOPE: interacción radiación-materia

a. Definiciones: sección eficaz y recorrido libre medio

Suponemos un flujo incidente de partículas J_{inc} (número de partículas por unidad de area perpendicular al haz y de tiempo) que incide sobre un cierto blanco. Si a la salida tenemos un detector que analiza el número de partículas dispersadas por unidad de tiempo $\dot{N}$ que han experimentado una pérdida de energía entre W y $W + \mathrm{W}$ (respecto la energía inicial E) dentro de un cierto ángulo sólido $\mathrm{d}\Omega$ (ver Figura A.1), la sección eficaz total y diferencial se define como

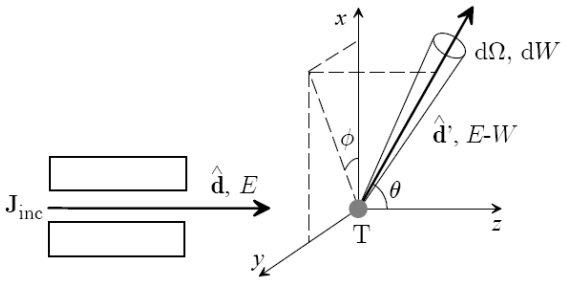

Figura A.1: *Concepto de sección eficaz (extraído de Ref. [15]).*

$$\sigma = \int \mathrm{d}W \int \mathrm{d}\Omega \frac{\mathrm{d}^2\sigma}{\mathrm{d}W\mathrm{d}\Omega} \qquad \frac{\mathrm{d}^2\sigma}{\mathrm{d}W\mathrm{d}\Omega} \equiv \frac{\dot{N}}{J_{inc}\mathrm{d}W\mathrm{d}\Omega} \tag{A.1}$$

El recorrido libre medio λ^{-1} se define como el recorrido medio que recorre una partícula entre dos interacciones consecutivas y mantiene la relación $\lambda^{-1} = \mathcal{N}\sigma$, siendo $\mathcal{N}$ el número de partículas por unidad de volumen. En el caso de fotones se define el coeficiente de atenuación μ como $\mu \equiv \lambda^{-1}$.

PENELOPE permite simular en un rango de energías de las partículas incidentes entre 50 eV y 1 GeV. En este rango tenemos diferentes interacciones de los fotones y los electrones/positrones con la materia.

b. Interacción de los fotones con la materia

Los fotones son partículas sin carga y sin masa en reposo y por tanto, su interacción es inferior a la de las partículas cargadas. En el rango de energía comentado, los principales mecanismos de interacción de los fotones con la materia son los que se muestran en la Figura A.2.

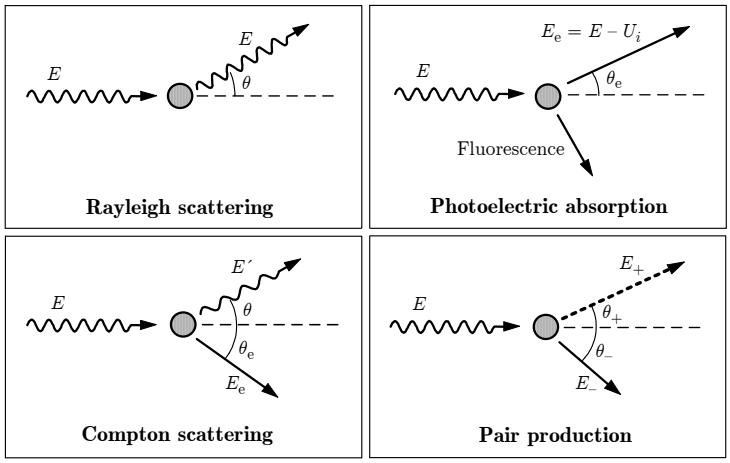

Figura A.2: *Mecanismos de interacción de los fotones (extraído de Ref. [15]).*

- **Dispersión elástica o Rayleigh**. Consiste en la dispersión de un fotón de energía E por uno de los electrones de un átomo (con número atómico Z) sin pérdida de energía del fotón y por tanto, excitación del átomo.
- **Efecto fotoeléctrico**. El fotón (E) es absorbido por el átomo (Z) y un electron de la capa i-ésima es emitido con energía $E_e = E - U_i$, siendo U_i la energía de ionización de la capa. Cuando el átomo se desexcita emite radiación de flurescencia.
- **Efecto Compton**. Un fotón (E) interacciona con un electron que lo absorbe y reemite un nuevo fotón con energía $E' < E$. Después de la interacción, el electron retrocede con una cierta energía E_e.
- **Producción de pares**. Un fotón ($E > 2m_e c^2$) desaparece y la energía es invertida en generar un electrón y un positrón.

En la Figura A.3 se muestra el coeficiente de atenuación másico del agua y del plomo para diferentes energías del fotón incidente E.

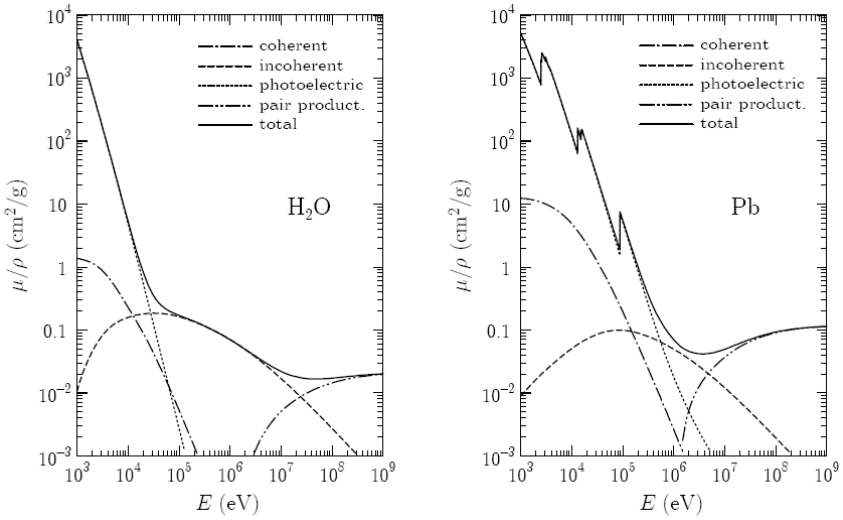

Figura A.3: *Coeficiente de atenuación másico para el agua y el plomo(extraído de Ref. [15]).*

c. Interacción de los electrones y positrones con la materia

La Figura A.4 muestra los mecanismos de interacción de las partículas cargadas con la materia.

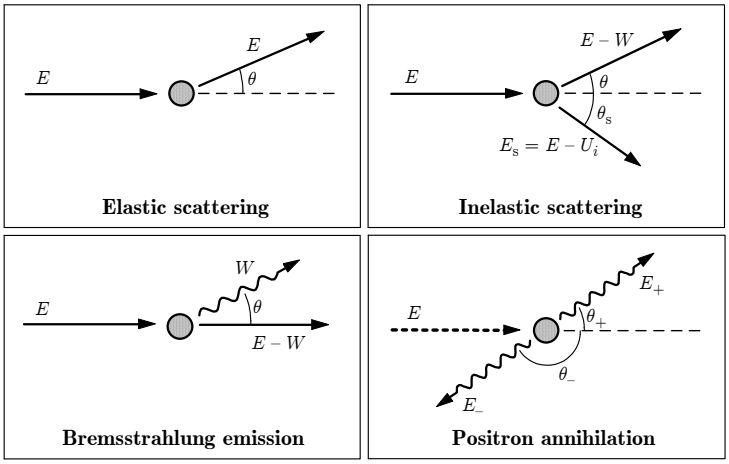

Figura A.4: *Mecanismos de interacción de los electrones (extraído de Ref. [15]).*

- **Dispersión elástica**. Consiste en la dispersión de una partícula cargada por los átomos del material sin pérdida de energía.

- **Dispersión inelástica**. La partícula cargada padece pérdida de energía que se invierte en excitar o ionizar el átomo blanco.

- **Emisión de bremsstrahlung**. Cuando las partículas cargadas se ven sometidas a cambios en la aceleración, emiten radiación. Esta radiación se llama de frenado o bremsstrahlung.

- **Aniquilación positrónica**. El positrón es inestable cuando viaja por la materia; cuando interacciona con un electrón se aniquila dando lugar a dos fotones de energía $m_e c^2$ en igual dirección y sentidos opuestos.

A.1.2. Los fundamentos de PENELOPE: el método Monte Carlo

a. Idea general

El método Monte Carlo (MC) es una técnica numérica no determinista basada en el muestreo aleatorio repetido para obtener el resultado. Es un método extensamente usado para simular sistemas físicos cuando es imposible tener una solución exacta con un algoritmo determinista. A diferencia de otras técnicas numéricas, que producen soluciones aproximadas, la ventaja del método MC es que tiene un error absoluto de la estimación que decrece como $\sim 1/\sqrt{N}$, con N el número de historias simuladas, independientemente de la dimensión del problema a estudiar.

b. Probabilidad y sorteo de números aleatorios

La probabilidad que una variable aleatoria esté entre un valor x_1 y $x_1 + \mathrm{d}x$ viene determinada por

$$\mathrm{P}\{x_1 < x < x_1 + \mathrm{d}x\} = p(x_1)\mathrm{d}x \tag{A.2}$$

donde $p(x)$ es la función de distribución de probabilidad. La función de distribución de probabilidad acumulada se define como

$$\mathcal{P}(x) \equiv \int_{x_{\min}}^{x} p(x')\mathrm{d}x' \tag{A.3}$$

donde $\mathcal{P}(x)$ es una función monótona creciente que varia de 0 a 1. Por lo tanto, si hacemos la transformación $\xi = \mathcal{P}(x)$, la nueva variable aleatoria ξ toma valores en el intervalo (0,1). La función de distribución de probabilidad de la variable ξ está relacionada con la de x de la siguiente forma

$$p_\xi(\xi) = p(x)\left(\frac{\mathrm{d}\xi}{\mathrm{d}x}\right)^{-1} = p(x)\left(\frac{\mathrm{d}\mathcal{P}}{\mathrm{d}x}\right) = 1 \tag{A.4}$$

de donde se deduce que ξ está distribuida uniformemente en el intervalo (0,1). Así pues, sorteando valores uniformemente distribuidos entre 0 y 1, podemos obtener x que siguen la distribución de probabilidad $p(x)$ invirtiendo la transformación $x = \mathcal{P}^{-1}(\xi)$.

Ésto se llama método de la transformada inversa, pero existen otros métodos para obtener variables aleatorias que sigan cierta distribución de probabilidad a partir de la generación de números aleatorios.

c. Generadores de números aleatorios

Existen diferentes generadores de números aleatorios; el utilizado en PENELOPE es una adaptación de la subrutina RANECU [49], basada en el método congruencial. El método congruencial genera números aleatorios u_i (en realidad son pseudo-aleatorios) uniformemente distribuídos en el intervalo (0,1) de la siguiente forma

$$S_{i+1} = (aS_i)\mathrm{mod}(m) \qquad u_i = \frac{S_i}{m} \tag{A.5}$$

donde se fijan a y m, enteros positivos con ciertas propiedades para asegurar un buen generador de números aleatorios. S_0 es lo que se llama semilla inicial.

En el caso de RANECU se combinan dos generadores de números aleatorios congruenciales

para conseguir números aleatorios con un período de repetición inagotable en las simulaciones prácticas (T $\sim 2.10^{18}$) de la siguiente forma:

$$S_i = (S_i^1 - S_i^2)\mathrm{mod}(m^1 - 1) \tag{A.6}$$

con los siguientes parámetros: $m^1 = 2147483563$, $m^2 = 2147483399$, $a^1 = 40014$ y $a^2 = 40692$.

A continuación vemos la programación de la función *rand*, la encargada de generar los números aleatorios en PENELOPE.

```
C  ***********************************************************************
C                          FUNCTION RAND
C  ***********************************************************************
      FUNCTION RAND(DUMMY)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z), INTEGER*4 (I-N)
      PARAMETER (USCALE=1.0D0/2.147483563D9)
      COMMON/RSEED/ISEED1,ISEED2
C
      I1=ISEED1/53668
      ISEED1=40014*(ISEED1-I1*53668)-I1*12211
      IF(ISEED1.LT.0) ISEED1=ISEED1+2147483563
      I2=ISEED2/52774
      ISEED2=40692*(ISEED2-I2*52774)-I2*3791
      IF(ISEED2.LT.0) ISEED2=ISEED2+2147483399
      IZ=ISEED1-ISEED2
      IF(IZ.LT.1) IZ=IZ+2147483562
      RAND=IZ*USCALE
      RETURN
      END
```

d. Simulación del transporte de radiación

El método MC nos reproduce la propagación de la radiación en la materia sorteando numéricamente:

- Distancia entre interacciones
- Tipo de interacción
- Deflexión angular y/o pérdida de energía en la interacción
- Generación de radiación secundaria

Evaluemos con más detalle la simulación del transporte de radiación mediante técnicas MC (véase Figura A.5). Suponemos que en nuestro problema hay dos tipos de interacciones: A y B, con sus secciones eficaces diferenciales $\frac{\mathrm{d}^2\sigma_A}{\mathrm{d}W\,\mathrm{d}\Omega}(E;W,\theta,\phi)$ y $\frac{\mathrm{d}^2\sigma_B}{\mathrm{d}W\,\mathrm{d}\Omega}(E;W,\theta,\phi)$. Las secciones eficaces totales de A y B son σ_A y σ_B; así pues, la sección eficaz total de interacción será $\sigma = \sigma_A + \sigma_B$.

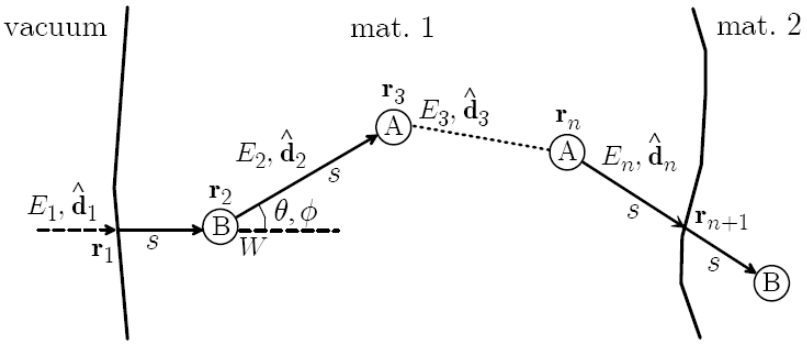

Figura A.5: *Transporte de radiación mediante técnicas MC (extraído de Ref. [15]).*

1. Inicialmente tenemos una partícula en un cierto estado inicial, definido por la posición $\vec{r} = (x, y, z)$, energía E y dirección $\vec{d} = (u, v, w)$.
2. Sorteamos la longitud s hasta la siguiente interacción.
3. Movemos la partícula $\vec{r} \rightarrow \vec{r} + s\vec{d}$.

4. El tipo de interacción lo seleccionaremos según las probabilidades $p_A = \sigma_A/\sigma$ y $p_B = \sigma_B/\sigma$.

5. El ángulo de dispersión y energía perdida se tendrán que mostrear según la sección eficaz diferencial.

6. La nueva energía y dirección de movimiento serán $E \rightarrow E - W$ y $\vec{d} \rightarrow R(\theta, \phi)\vec{d}$.

7. La simulación consistirá en repetir los pasos anteriores hasta que la partícula deja el material o bien su energía es inferior a una cierta energía de absorción (EABS).

8. Después de simular la vida de una partícula, se tiene que hacer lo mismo para las partículas secundarias a las que ha dado lugar.

En la Figura A.6 podemos ver la complejidad de la interacción de fotón de 10 MeV en una lámina de plomo.

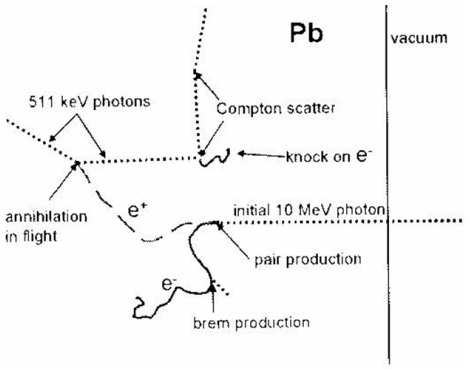

Figura A.6: *Historia de un fotón.*

Como comentario, se debe diferenciar entre historia (número de partículas iniciales) y partícula. Las partículas generadas por una misma partícula inicial se incluyen en la misma historia.

e. Simulación condensada

En la práctica, hacer una simulación como la anterior requiere largos tiempos de cálculo, sobretodo si intervienen electrones de alta energía (un electrón de 1 MeV puede interaccionar

un milión de veces antes de ser absorbido). Para solucionar este problema se usa la simulación condensada.

La simulación condensada consiste en simular el efecto general de muchas interacciones mediante una única interacción. Concretamente, PENELOPE utiliza la simulación mixta, que combina la simulación detallada para interacciones *hard* y la simulación condensada para interacciones *soft*.

Así pues, consideraremos que una interacción es hard cuando el ángulo de dispersión es mayor que un cierto ángulo definido por el usuario $\theta \geq \theta_C$ o bien cuando la pérdida de energía es mayor que una cierta pérdida de energía crítica $W \geq W_C$.

A.1.3. La estructura de PENELOPE

a. Estructura general

PENELOPE es un paquete de subrutinas; por lo tanto, requiere de un programa principal para funcionar, además de ciertos datos de entrada. En la Figura A.7 se ilustra la estructura general de PENELOPE.

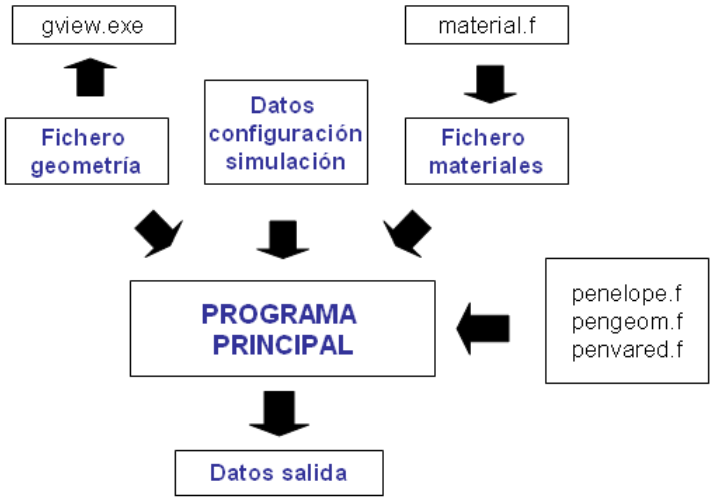

Figura A.7: *Estructura general de PENELOPE.*

Las fuentes básicas de PENELOPE son las siguientes:

- **penelope.f**. Paquete de subrutinas para la simulación MC del transporte acoplado de fotones y electrones en un medio homogeneo. Está compuesto de las siguientes rutinas:
  - **PEINIT**. Inicializa el paquete de simulación.
  - **CLEANS**. Inicia una pila donde se guarda el estado de las partículas generadas por una misma historia.
  - **START**. Inicia el transporte de una partícula.
  - **SECPAR**. Lee el estado de una nueva partícula de la pila. Nos retorna LEFT, el número de partículas secundarias generadas que todavía se guardan en la pila.
  - **JUMP**. Determina la distancia hasta la siguiente interacción.
  - **KNOCK**. Simula una interacción.
  - **RAND**. Genera números aleatorios uniformemente distribuidos entre 0 y 1.
- **pengeom.f**. Permite la simulación en cualquier sistema de materiales formado por diferentes cuerpos homogéneos limitados por superficies cuádricas.
  - **GEOMIN**. Lee las variables de entrada del archivo de geometría e inicializa el paquete de geometría.
  - **LOCATE**. Determina el cuerpo (IBODY) y material (MAT) donde se mueve la partícula en cada momento. Si MAT=0, la partícula ha escapado del sistema.
  - **STEP**. Mueve la partícula una cierta distancia DS. Si al mover la partícula una cierta distancia se encuentra una interfaz (NCROSS), se para la partícula en la interfaz y se vuelve a empezar la simulación de la partícula con las condiciones de antes de cruzar la frontera.
- **material.f**. Permite crear el archivo de materiales que contiene la información física de cada material que usaremos en la simulación.
- **penvared.f**. Permite aplicar técnicas de reducción de varianza (técnicas matemáticas para reducir el tiempo en conseguir una cierta incertidumbre en la simulación). En el presente trabajo no se han usado este tipo de técnicas y no vamos a detallar su funcionamiento (ver Ref. [15] para más información).

PENELOPE también dispone de otras subrutinas/programas:

- **GVIEW**. Programa para visualizar al geometría.
- **SHOWER**. Programa para visualizar la trayectoria de las partículas.
- **EMFIELDS**. Paquete de subrutinas para simular el transporte de partículas en campos electromagnéticos estáticos.
- **TABLES**. Genera tablas con los datos físicos del transporte de radiación.

b. El archivo de geometría

Para crear el archivo de geometría, uno de los conceptos básicos a tener en cuenta es el que una superficie se define por una función $\Phi(x,y,z)=0$ ($\Phi(x,y,z)>0$ para los puntos de fuera de la superficie y $\Phi(x,y,z)<0$ para los de dentro). Generaremos nuestros objetos a partir de cuádricas (planos, esferas, cilindros, paraboloides) que se pueden expresar muy fácilmente en su forma reducida:

$$\Phi(x,y,z)=I_1x^2+I_2y^2+I_3z^2+I_4z+I_5=0 \tag{A.7}$$

donde los coeficientes I_i solo pueden tomar valores -1, 0 y 1. Aplicando transformaciones de escalado (X-SCALE, Y-SCALE, Z-SCALE), de rotación (OMEGA, THETA) y de traslación (X-SHIFT, Y-SHIFT, Z-SHIFT) podemos obtener cualquier cuadrica en cualquier posición del espacio.

Los archivos de geometría de nuestro trabajo estarán compuestos básicamente por planos y cilindros. Vemos como se escribiría un plano infinito en z=1 cm, uno en z=5 cm y un cilindro infinito de radio 5 cm

```
SURFACE (   1)
INDICES=( 0, 0, 0, 1, 0)
Z-SHIFT=(+1.00000000000000E+00,   0)
```

```
SURFACE (   2)
INDICES=( 0, 0, 0, 1, 0)
Z-SHIFT=(+5.00000000000000E+00,   0)

SURFACE (   3)
INDICES=( 1, 1, 0, 0,-1)
X-SCALE=(+5.00000000000000E+00,   0)
Y-SCALE=(+5.00000000000000E+00,   0)
```

A posteriori, tenemos que definir los cuerpos que están limitados por diferentes superficies. Por ejemplo, si queremos el cilindro de radio 5 cm entre los planos z=1 cm y z=5 cm (substrayendo un cuerpo 1 previamente definido):

```
BODY    (   2)
MATERIAL(   1)
SURFACE (   1), SIDE POINTER=(+1)
SURFACE (   2), SIDE POINTER=(-1)
SURFACE (   3), SIDE POINTER=(-1)
BODY    (   1)
```

La geometría tiene muchas más posibilidades, pero para entender el presente trabajo es suficiente la visión general que se da en este apartado (para más información, ver Ref. [15]).

A.2. El programa principal para PENELOPE: penEasy

Hemos comentado que PENELOPE es un paquete de subrutinas y que por tanto, necesitamos un programa principal que controle la geometría, la evolución de las partículas y calcule ciertas cantidades relevantes con sus incertidumbres. Durante todo el trabajo se ha usado el programa principal para PENELOPE llamado penEasy [16].

penEasy es un programa principal genérico desarrollado para PENELOPE que provee modelos de fuentes de radiación y permite calcular diversas magnitudes de interés. El usuario de

penEasy debe modificar los ficheros de entrada de datos para definir la fuente de radiación, la geometría del problema y la obtención de, por ejemplo, la distribución 3D/cilíndrica/radial de dosis absorbida, el espectro de fluencia de partículas en un determinado material, etc. Además, su estructura modular facilita la adaptación del código si el problema a resolver así lo requiere.

a. Esquema general

El esquema de un programa principal para PENELOPE se ilustra en la Figura A.8. La conexión de PENELOPE con el programa principal se hace via el *COMMON*:

```
COMMON/TRACK/E,X,Y,Z,U,V,W,WGHT,KPAR,IBODY,MAT,ILB(5)
```

que contiene el estado de las partículas:

- **KPAR**. Tipo de partícula (1 = electrón, 2 = fotón, 3 = positrón).
- **E**. Energía de la partícula [eV] (para los electrones y positrones se refiere a la energía cinética).
- **X,Y,Z**. Coordenadas de posición de la partícula [cm].
- **U,V,W**. Cosenos directores de la dirección de movimiento.
- **WGHT**. Peso estadístico de la partícula (usado en técnicas de reducción de varianza).
- **IBODY**. Cuerpo donde se encuentra la partícula.
- **MAT**. Material del cuerpo donde se encuentra la partícula.
- **ILB(5)**. **ILB(1)** = 1 para partículas primarias y >1 para los descendientes. **ILB(2)** nos indica el tipo de partícula de la que desciende una partícula secundaria. **ILB(3)** describe el mecanismo de interacción que ha originado la partícula secundaria. **ILB(4)**: si la partícula se ha producido en un proceso de relajación de los átomos, indica que tipo de transición se ha producido. **ILB(5)** viene definido por el usuario.

b. *Sources* y *tallies* de penEasy

penEasy consta de fuentes previamente diseñadas (*sources*) y de contadores de magnitudes en interés en física mèdica (**tallies**):

- ***Source***. Fuente de partículas con ciertas características. Hay dos tipos de fuente:
  - **SECTION SOURCE BOX ISOTROPIC GAUSS SPECTRUM**. Fuente que permite la generación de partículas con una gran variedad de formas y espectros de energía.
  - **SECTION SOURCE PHASE SPACE FILE**. Permite la generación de partículas a partir de la lectura de un fichero de espacio de fase externo.
- ***Tally***. Lleva la "cuenta" de ciertas magnitudes de interés durante la simulación. Los *tallies* implementados en penEasy son:
  - **SECTION TALLY VOXEL DOSE**. Calcula la dosis absorbida por historia simulada en una región voxelizada.
  - **SECTION TALLY SPATIAL DOSE DISTRIB**. Calcula la dosis absorbida por historia simulada en voxels definidos por el usuario.
  - **SECTION TALLY CYLINDRICAL DOSE DISTRIB**. Calcula la dosis absorbida por historia simulada en elementos de volumen cilíndricos definidos por el usuario.
  - **SECTION TALLY SPHERICAL DOSE DISTRIB**. Calcula la dosis absorbida por historia simulada en elementos de volumen esféricos definidos por el usuario.
  - **SECTION TALLY ENERGY DEPOSITION PULSE SPECTRUM**. Nos proporciona la energía total depositada en el material de detección por historia simulada.
  - **SECTION TALLY FLUENCE TRACK LENGTH**. Nos determina el espectro de fluencia por unidad de historia simulada integrada en el material de detección.
  - **SECTION TALLY PHASE SPACE FILE**. Genera ficheros de espacio de fase.
  - **SECTION TALLY PARTICLE CURRENT SPECTRUM**. Nos proporciona el tipo y espectro de energía de las partículas que entran en el material de detección por unidad de historia simulada.

- **SECTION TALLY PARTICLE TRACK STRUCTURE**. Permite la representación gráfica de la trayectoria de las partículas.
- **SECTION INTERACTION FORCING**. Relacionado con las técnicas de reducción de varianza.

penEasy también permite el uso de geometrías voxelizadas junto con cuádricas, aunque no lo comentamos en detalle ya que no se usa en el presente trabajo.

c. Archivo de configuración

A parte de proporcionar el archivo de geometría, el archivo de materiales, las propiedades de la fuente de partículas y las características de los datos que queremos obtener, tenemos que proporcionar también otro tipo de datos de entrada:

- **Configuración general**: número de historias a simular, tiempo máximo de simulación, tiempo de *refresh* del archivo de salida y los valores de las semillas iniciales para el generador de números aleatorios.

```
[SECTION CONFIG v.2006-08-01]
 1.0e15                          NO. OF HISTORIES (<1e15)
 300.0                           ALLOTTED TIME (s)
 100.0                           UPDATE INTERVAL
 1  1                            INITIAL RANDOM SEEDS
[END OF CONFIG SECTION]
```

- **Datos del transporte de partículas**
  - **EABS**(e^-, e^+, γ). Energías de absorción; si una partícula tiene una energía inferior a la de absorción, se deposita toda su energía cinética en ese punto y deja de transportarse.
  - Parámetros relacionados con la simulación mixta:
    - **WCC** y **WCR**. Energías de corte para la producción de interacciones *hard* inelásticas y bremsstrahlung.

- **C2** y **C1**. Deflexión angular máxima entre dos colisiones *hard* y energía fraccional máxima perdida entre dos interacciones *hard*. Con estos parámetros se dividen las interacciones *soft* de las *hard*.
- **DSMAX**. Máxima longitud sin interaccionar.

La parte correspondiente del archivo de entrada de penEasy es la siguiente:

```
[SECTION PENELOPE v.2008-02-20]
(water.mat             ) MATERIAL FILE NAME (*** 30 characters ***)
 1                       No. OF MATERIALS EXPECTED IN MAT FILE (0 for AUTO)
 MAT EABS(e-) EABS(ph) EABS(e+) C1   C2   WCC     WCR     DSMAX  DESCRIPTION
 1   50.00e3  5.000e3  50.00e3  0.1  0.1  50.00e3 5.000e3 1.0e30 water
[END OF PEN SECTION]
```

A lo largo del trabajo se especificaran la *source*, *tallies* y parámetros de entrada usados en las simulaciones.

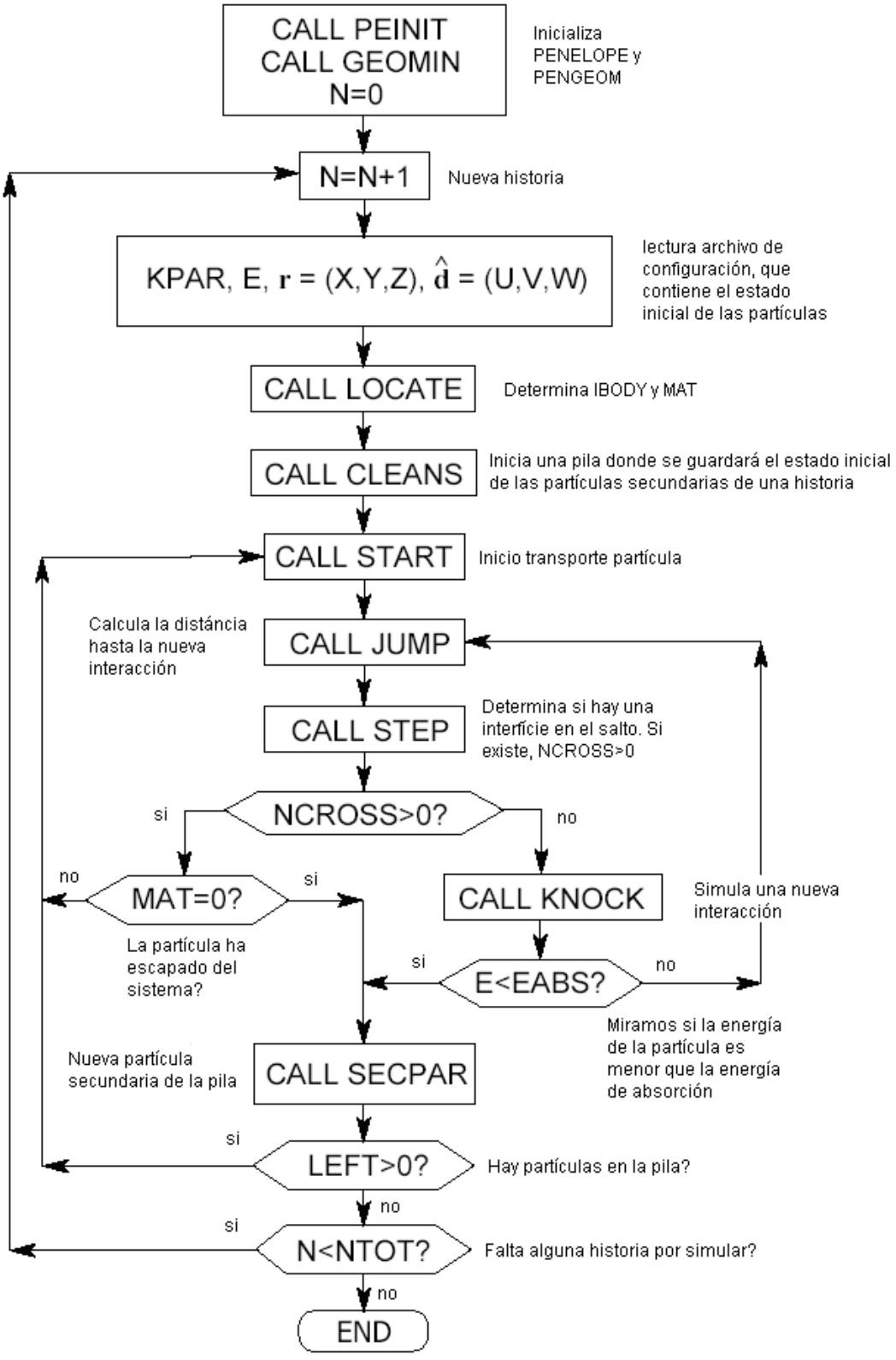

Figura A.8: *Esquema del programa principal para PENELOPE (adaptado de Ref. [15]).*

APÉNDICE B

Fuente del RTM para penEasy

```
!*****************************************************************
!*                          SOURCE                               *
!*                       RTM SOURCE                              *
!*                                                               *
!* Short description:                                            *
!*   Race-track microtron source                                 *
!*                                                               *
!* Dependencies:                                                 *
!*   from PENELOPE:                                              *
!*   -> common /TRACK/                                           *
!*   -> routines STORES,RAND                                     *
!*   from PENGEOM:                                               *
!*   -> routines LOCATE,STEP                                     *
!*   from other penEasy libraries:                               *
!*   -> routines GETLINE,TALLY                                   *
!*                                                               *
!* Compatible with PENELOPE versions:                            *
!*   2005,2006                                                   *
!*****************************************************************

      subroutine BIGSsource(n)
!*****************************************************************
!*    Input:                                                     *
!*      n -> History no.                                         *
!*    Output:                                                    *
!*      through /track/ and sec stack                            *
!*****************************************************************
      implicit none
      real*8 n
```

```
      integer*4 kpar,ibody,mat,ilb
      real*8 e,x,y,z,u,v,w,wght
      common/track/e,x,y,z,u,v,w,wght,kpar,ibody,mat,ilb(5)
      logical warned,srcpoint,active
      integer parsrc,matsrc,nspc,dim
      parameter (dim=1000)
      real*8 shots,usrc,vsrc,wsrc,cossrc,espc,pspc,despc,rot
      real*8 xsrc,ysrc,zsrc,dxsrc,dysrc,dzsrc
      common /srcbig/ rot(3,3),espc(dim),pspc(dim),despc(dim),shots,
     &                cossrc,usrc,vsrc,wsrc,xsrc,ysrc,zsrc,dxsrc,
     &                dysrc,dzsrc,parsrc,matsrc,nspc,warned,srcpoint,
     &                active
      integer ntrial,i,seeki
      integer*4 ncross
      real*8 costhe,phi,rand,infty,dsef,xrot,yrot,zrot
      real*8 pi,dospi,one3rd,randno
      parameter (pi=3.1415926535897932d0,dospi=2.0d0*pi,infty=1.0d30)
      parameter (one3rd=1.0d0/3.0d0)
      external rand

      real*8 mx1,sx1,mx2,sx2,my1,sy1,my2,sy2,rotax,rotay
      common /gauss/ mx1,sx1,mx2,sx2,my1,sy1,my2,sy2,rotax,rotay
      real*8 norm
      integer sign
      common /sign/sign

      if (.not.active) return

      ntrial = 0
      kpar = parsrc    ! Particle type

      direction: do
        call gauss2(mx1,sx1,mx2,sx2,rotax,xsrc,usrc)
        call gauss2(my1,sy1,my2,sy2,rotay,ysrc,vsrc)
    norm=usrc*usrc+vsrc*vsrc
        if (norm.gt.1.0d0) then
          write(*,*) 'ERROR: u**2+v**2 is greater than 1'
        else
        wsrc=sqrt(abs(1.0d0-usrc*usrc-vsrc*vsrc))
        endif
          u = usrc
          v = vsrc
          if (sign>0) then
            w = wsrc
          Else
            w= -wsrc
          endif
```

```
   position: do
     ntrial = ntrial+1
     if (n.lt.1.01d0.and.ntrial.gt.100) then
       write(*,*)
&    'BIGSsource:ERROR: could not generate a valid particle '//
&    'after 100 trials;'
       write(*,*)
&    '  box enclosure is not in declared '//
&    'material or particle direction is not correctly aimed'
       stop
     endif
   ! Sample coordinates relative to the box center:
     if (srcpoint) then  ! No need to sample for a point source
       x = mx1
       y = my1
       z = zsrc
     else
       x = xsrc
       y = ysrc
       z = zsrc
     endif
     call locate  ! Finds body and material (needs U,V,W)

     ! Accept or reject?
     if (matsrc.eq.0) then
       if (mat.eq.0) call step(infty,dsef,ncross)
       exit direction
     else if (srcpoint) then
       if (mat.eq.matsrc) exit direction
       do
         call step(infty,dsef,ncross)
         if (mat.eq.0.or.ncross.eq.0) cycle direction
         if (mat.eq.matsrc) exit direction
       enddo
     else if (mat.eq.matsrc) then
       exit direction
     endif
   enddo position
 enddo direction

 ! Kinetic energy:
 if (nspc.eq.2.and.pspc(2).eq.0.0d0) then
   do
     e = espc(2)+espc(1)*
&         sqrt(-2.0d0*log(rand(1.8d1)))*sin(dospi*rand(1.9d1))
     if (e.gt.espc(3).and.e.lt.espc(4)) exit
   enddo
```

```
      else
        randno = rand(2.0d1)
        i = seeki(pspc,randno,nspc)
        e = espc(i)+(randno-pspc(i))*despc(i)
      endif

      ! Average no. of shots per call:
      shots = shots+ntrial
      if (.not.warned.and.shots.gt.n*10.0d0) then
        write(*,*) ' '
        write(*,'(a)')
     &    '***************'
        write(*,'(a)')
     &    'BIGSsource:WARNING: source effectiveness is too low !'
        write(*,'(a)')
     &    '  Redefine source parameters appropiately.'
        write(*,'(a)')
     &    '  Histories and shots per source call so far:'
        write(*,'(2x,f18.0,1x,1pe12.5)') n,shots/n
        write(*,'(a)')
     &    '***************'
        write(*,*) ' '
        warned = .true.
      endif

      wght = 1.0d0
      ilb(1) = 1
      ilb(5) = 0

      call stores(e,x,y,z,u,v,w,wght,kpar,ilb)
      call tally(0,e)
      end


      subroutine BIGSinisrc(activated,emax)
!*****************************************************************
!*    Initializes the source.                                    *
!*                                                               *
!*    Output:                                                    *
!*      activated -> TRUE if the source is active.               *
!*      emax -> max source energy (eV)                           *
!*****************************************************************
      implicit none
      logical activated
      real*8 emax

      logical warned,srcpoint,active
```

```
 integer parsrc,matsrc,nspc,dim
 parameter (dim=1000)
 real*8 shots,usrc,vsrc,wsrc,cossrc,espc,pspc,despc,rot
 real*8 xsrc,ysrc,zsrc,dxsrc,dysrc,dzsrc
 common /srcbig/ rot(3,3),espc(dim),pspc(dim),despc(dim),shots,
&                cossrc,usrc,vsrc,wsrc,xsrc,ysrc,zsrc,dxsrc,
&                dysrc,dzsrc,parsrc,matsrc,nspc,warned,srcpoint,
&                active
 character*80 buffer
 character*(*) secid,eos
 parameter (secid='[RTM SOURCE]')
 parameter (eos='[END OF RTM SOURCE]')
 integer j,error
 real*8 prob,pi,norm,fwhm2sig,emean,sigma,deg2rad,emin,kfact
 real*8 zero
 parameter (pi=3.1415926535897932d0,deg2rad=pi/180.0d0)
 parameter (fwhm2sig=4.2466090014400952d-1,kfact=6.0d0)
 parameter (zero=1.0d-30)

 real*8 mx1,sx1,mx2,sx2,my1,sy1,my2,sy2,rotax,rotay
 common /gauss/ mx1,sx1,mx2,sx2,my1,sy1,my2,sy2,rotax,rotay
 real*8 pespc
 real*8 meeV
 parameter (meeV=0.510998918d6)
 integer sign
 common /sign/ sign

 write(*,*) ' '
 write(*,'(a)')
& '>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'
 call getline(buffer)
 if (index(buffer,secid).eq.0) then
   write(*,'(a)') 'BIGSinisrc:ERROR: incorrect section header;'
   write(*,'(a,a)') '  expecting to find: ',secid
   write(*,'(a,a)') '  found instead:     ',buffer
   stop
 endif
 write(*,'(a)') secid

 read(*,'(1x,a3)') buffer
 if (buffer(1:2).eq.'ON') then
   active = .true.
   activated = active
 else if (buffer(1:3).eq.'OFF') then
   active = .false.
   activated = active
   write(*, '(a)')
```

```
&    '>>>> RTM source is OFF >>>>'
   do
     read(*,'(a80)',iostat=error) buffer
     if (error.ne.0) then
       write(*,'(a,a,a)') 'BIGSinisrc:ERROR: ',
&       'Unable to find End-Of-Section mark: ',eos
       stop
     endif
     if (index(buffer,eos).ne.0) return
   enddo
 else
   write(*,'(a)')
&    'BIGSinisrc:ERROR: expecting to find ON or OFF'
   write(*,'(a)') 'found instead:'
   write(*,'(a)') buffer(1:3)
   stop
 endif

 ! Type of particle:
 write(*,'(a)') 'Particle type:'
 read(*,*) parsrc
 write(*,'(i1)') parsrc
 if (parsrc.ne.1.and.parsrc.ne.2.and.parsrc.ne.3) then
   write(*,*) 'BIGSinisrc:ERROR: invalid particle type'
   stop
 endif

 ! Energy:
 write(*,'(a)') 'Reading energy spectrum'
 read(*,'(a80)') buffer
 write(*,'(a)') '  Energy(eV)  Relat.Probability   Bin#'
 nspc = 0
 pspc(1) = 0.0d0
 do
   nspc = nspc+1
   read(*,*) pespc,prob
   espc(nspc)=sqrt(meeV*meeV+pespc*pespc)-meeV
   write(*,'(2(1x,1pe12.5),1x,i5)') espc(nspc),prob,nspc
   if (espc(nspc).lt.0.0d0) then
     write(*,*) 'BIGSinisrc:ERROR: negative energy'
     stop
   else if (espc(nspc).lt.espc(max(nspc-1,1))) then
     write(*,*) 'BIGSinisrc:ERROR: decreasing energy'
     stop
   endif
   if (prob.lt.0.0d0) exit
   if (nspc.ge.dim) then
```

```
      write(*,*) 'BIGSinisrc:ERROR: too many bins in spectrum;'
      write(*,*) '              enlarge DIM'
      stop
    endif
    pspc(nspc+1) = pspc(nspc)+prob
  enddo
  write(*,'(a)') 'No. of bins read:'
  write(*,'(i5)') nspc-1
  if (nspc.lt.2) then
    write(*,*)
&     'BIGSinisrc:ERROR: at least 1 bin must be defined'
    stop
  endif
  if (pspc(nspc).gt.0.0d0) then
    write(*,'(a)')
&     'Sum of relative probabilities before normalization:'
    write(*,'(1pe12.5)') pspc(nspc)
    do j=1,nspc
      pspc(j) = pspc(j)/pspc(nspc)
    enddo
    do j=1,nspc-1
      despc(j) = 0.0d0
      if (pspc(j+1).gt.pspc(j))
&       despc(j) = (espc(j+1)-espc(j))/(pspc(j+1)-pspc(j))
    enddo
    emax = espc(nspc)
  else
    write(*,'(a)')
&     'Null probability; assuming Gaussian spectrum'
    if (nspc.gt.2) then
      write(*,*)
&       'BIGSinisrc:ERROR: Gaussian requires only 1 bin'
      stop
    endif
    emean = 0.5d0*(espc(1)+espc(2))
    sigma = fwhm2sig*(espc(2)-espc(1))
    emin = max(0.0d0,emean-kfact*sigma)
    emax = emean+kfact*sigma
    write(*,'(a)')
&     'Mean energy, FWHM, sigma, Emin, Emax (eV) of Gaussian:'
    write(*,'(5(1x,1pe12.5))')
&     emean,espc(2)-espc(1),sigma,emin,emax
    if (emean.lt.0.0d0) then
      write(*,*) 'BIGSinisrc:ERROR: negative mean energy.'
      stop
    endif
    espc(4) = emax
```

```
      espc(3) = emin
      espc(2) = emean
      espc(1) = sigma
    endif

    ! Position and Direction data:
    write(*,'(a)') 'Mean and sigma x'
    read(*,*) mx1,mx2
    read(*,*) sx1,sx2
    write(*,'(4(1x,1pe12.5))') mx1,mx2,sx1,sx2
    write(*,'(a)') 'Mean and sigma y'
    read(*,*) my1,my2
    read(*,*) sy1,sy2
    write(*,'(4(1x,1pe12.5))') my1,my2,sy1,sy2

    ! Z source
    write(*,'(a)') 'Z source'
    read(*,*) zsrc
    write(*,'(1pe12.5)') zsrc
    write(*,'(a)') 'Z direction'
    read(*,*) sign
    write(*,'(i2)') sign

    ! Rotation angle
    write(*,'(a)') 'Rotation angle x'
    read(*,*) rotax
    write(*,'(1pe12.5)') rotax
    write(*,'(a)') 'Rotation angle y'
    read(*,*) rotay
    write(*,'(1pe12.5)') rotay

    ! Material:
    if (sx1.eq.0.and.sy1.eq.0) srcpoint = .true.
    write(*,'(a)') 'Source material:'
    read(*,*) matsrc
    write(*,'(i2)') matsrc
    if (srcpoint.and.matsrc.ne.0) write(*,'(a)')
   &  '  (interpreted as the material particles must be aiming at)'
    if (matsrc.lt.0) then
      write(*,*) 'BIGSinisrc:ERROR: negative materials are invalid'
      stop
    endif

    ! Init performance vars:
    shots = 0.0d0
    warned = .false.
```

```
      read(*,'(a80)') buffer
      if (index(buffer,eos).eq.0) then
        write(*,*) 'BIGSinisrc:ERROR: End-Of-Section mark not found'
        write(*,'(a,a)') '  expecting to find: ',eos
        write(*,'(a,a)') '  found instead:     ',buffer
        stop
      endif
      write(*,'(a)') '>>>> RTM source initialization finished >>>>'
      end


      integer function seeki(x,xc,n)
!*****************************************************************
!*    Finds the interval (x(i),x(i+1)] containing the value xc.    *
!*                                                                 *
!*    Input:                                                       *
!*      x(1..n) -> data array                                      *
!*      xc -> point to be located                                  *
!*      n -> no. of data points                                    *
!*    Output:                                                      *
!*      index i of the semiopen interval where xc lies             *
!*    Comments:                                                    *
!*      -> If xc=x(1) then i=1 is returned.                        *
!*      -> If xc is outside the closed interval [x(1),x(n)]  the   *
!*         execution is aborted.                                   *
!*****************************************************************
      implicit none
      integer n
      real*8 xc,x(n)

      integer itop,imid

      if(xc.gt.x(n)) then
        write(*,*) 'seeki error: value outside range, xc>x(n):'
        write(*,*) xc,x(n)
        stop
      endif
      if(xc.lt.x(1)) then
        write(*,*) 'seeki error: value outside range, xc<x(1):'
        write(*,*) xc,x(1)
        stop
      endif

      seeki = 1
      itop = n
      do
        imid = (seeki+itop)/2
```

```
        if(xc.gt.x(imid)) then
          seeki = imid
        else
          itop = imid
        endif
      if(itop-seeki.le.1) exit
      enddo
      end


      subroutine gauss2(m1,s1,m2,s2,rota,r1r,r2r)
!*****************************************************************
!*    I.Martínez-Rovira                                          *
!*    Generation of numbers following a 2D normal distribution   *
!*                                                               *
!*    Input:                                                     *
!*      m1,m2 -> mean x and y                                    *
!*      s1,s2 -> sigma x and y                                   *
!*      rota -> rotation angle                                   *
!*    Output:                                                    *
!*      r1r,r2r: Generation of numbers following                 *
!*               a 2D normal distribution                        *
!*****************************************************************
      real*8 pi,dospi
      parameter (pi=3.1415926535897932d0,dospi=2.0d0*pi)
      real*8 m1,m2,s1,s2,rota
      real*8 r1r,r2r
      real*8 ri1,ri2,auxvar1,auxvar2,r1,r2
      real*8 rand
      external rand
      real*8 sig
      parameter (sig=1.644855d0)

      ri1=rand(1.d0)
      ri2=rand(2.d0)
      auxvar1=sqrt(-2.d0*log(ri1))
      auxvar2=dospi*ri2
      r1=auxvar1*cos(auxvar2)
      r2=auxvar1*sin(auxvar2)
      r1=m1+s1*r1
      r2=m2+s2*r2
      r1r=1.0d-1*(r1*cos(rota)-r2*sin(rota))  !cm
      r2r=1.0d-3*(r1*sin(rota)+r2*cos(rota))  ! "rad"
      return
      end

!>>>> End Of File >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

APÉNDICE C

Fichero de geometría del RTM

```
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (   1)   titanium window: upper plane
INDICES=( 0, 0, 0, 1, 0)
Z-SHIFT=(+60.50000000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (   2)   titanium window: lower plane
INDICES=( 0, 0, 0, 1, 0)
Z-SHIFT=(+60.49800000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (   3)   scattering foil: upper plane (Al)
INDICES=( 0, 0, 0, 1, 0)
Z-SHIFT=(+60.08800000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (   4)   scattering foil: middle plane
INDICES=( 0, 0, 0, 1, 0)
Z-SHIFT=(+60.00800000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (   5)   scattering foil: lower plane (Au)
INDICES=( 0, 0, 0, 1, 0)
Z-SHIFT=(+60.00000000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (   6)   water phantom: upper plane
INDICES=( 0, 0, 0, 1, 0)
Z-SHIFT=(+0.000000000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000000000
SURFACE (   7)   water phantom: lower plane
INDICES=( 0, 0, 0, 1, 0)
Z-SHIFT=(-20.00000000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000000000
```

```
SURFACE (   8)   water phantom: cylindrical ext
INDICES=( 1, 1, 0, 0,-1)
X-SCALE=(+200.0000000000000E+00,   0)              (DEFAULT=1.0)
Y-SCALE=(+200.0000000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000
SURFACE (   9)   lucite applicator: cylindrical ext
INDICES=( 1, 1, 0, 0,-1)
X-SCALE=(+7.900000000000000E+00,   0)              (DEFAULT=1.0)
Y-SCALE=(+7.900000000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000
SURFACE (  10)   lucite applicator: cylindrical int
INDICES=( 1, 1, 0, 0,-1)
X-SCALE=(+7.500000000000000E+00,   0)              (DEFAULT=1.0)
Y-SCALE=(+7.500000000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000
SURFACE (  11)   scattering foils: cylindrical ext (brass int)
INDICES=( 1, 1, 0, 0,-1)
X-SCALE=(+2.000000000000000E+00,   0)              (DEFAULT=1.0)
Y-SCALE=(+2.000000000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000
SURFACE (  12)  top
INDICES=( 0, 0, 0, 1, 0)
Z-SHIFT=(+65.00000000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000
SURFACE (  13)  brass top
INDICES=( 0, 0, 0, 1, 0)
Z-SHIFT=(+55.00000000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000
SURFACE (  14)  brass: cylindrical ext
INDICES=( 1, 1, 0, 0,-1)
X-SCALE=(+8.200000000000000E+00,   0)              (DEFAULT=1.0)
Y-SCALE=(+8.200000000000000E+00,   0)              (DEFAULT=1.0)
0000000000000000000000000000000000000000000000000000000000000000
BODY    (   1)   titanium foil
MATERIAL(   1)   titanium
SURFACE (   1), SIDE POINTER=(-1)
SURFACE (   2), SIDE POINTER=(+1)
SURFACE (  11), SIDE POINTER=(-1)
0000000000000000000000000000000000000000000000000000000000000000
BODY    (   2)   aluminum foil
MATERIAL(   2)   aluminum
SURFACE (   3), SIDE POINTER=(-1)
SURFACE (   4), SIDE POINTER=(+1)
SURFACE (  11), SIDE POINTER=(-1)
0000000000000000000000000000000000000000000000000000000000000000
BODY    (   3)   gold foil
MATERIAL(   3)   gold
```

```
SURFACE (   4), SIDE POINTER=(-1)
SURFACE (   5), SIDE POINTER=(+1)
SURFACE (  11), SIDE POINTER=(-1)
0000000000000000000000000000000000000000000000000000000000000000
BODY    (   4)   water phantom
MATERIAL(   4)   water
SURFACE (   6), SIDE POINTER=(-1)
SURFACE (   7), SIDE POINTER=(+1)
SURFACE (   8), SIDE POINTER=(-1)
0000000000000000000000000000000000000000000000000000000000000000
BODY    (   5)   lucite
MATERIAL(   5)   lucite
SURFACE (   5), SIDE POINTER=(-1)
SURFACE (   6), SIDE POINTER=(+1)
SURFACE (   9), SIDE POINTER=(-1)
SURFACE (  10), SIDE POINTER=(+1)
0000000000000000000000000000000000000000000000000000000000000000
BODY    (   6)   brass
MATERIAL(   6)   brass
SURFACE (  13), SIDE POINTER=(-1)
SURFACE (   6), SIDE POINTER=(+1)
SURFACE (   9), SIDE POINTER=(+1)
SURFACE (  14), SIDE POINTER=(-1)
0000000000000000000000000000000000000000000000000000000000000000
BODY    (   7)   air
MATERIAL(   7)   air
SURFACE (  12), SIDE POINTER=(-1)
SURFACE (   6), SIDE POINTER=(+1)
SURFACE (   8), SIDE POINTER=(-1)
BODY    (   1)
BODY    (   2)
BODY    (   3)
BODY    (   4)
BODY    (   5)
BODY    (   6)
0000000000000000000000000000000000000000000000000000000000000000
END      0000000000000000000000000000000000000000000000000000000000
```

APÉNDICE D

Ejemplo de fichero de cabecera de un PSF en formato OIEA

```
$IAEA_INDEX:
0001   // IAEA website: www-nds.iaea.org/phsp/photon/

$TITLE:
Siemens PRIMUS 6MV photon beam 0.5 x 0.5 cm2 field

$FILE_TYPE:
0

$CHECKSUM:
2197591

$RECORD_CONTENTS:
    1     // X is stored ?
    1     // Y is stored ?
    0     // Z is stored ?
    1     // U is stored ?
    1     // V is stored ?
    1     // W is stored ?
    1     // Weight is stored ?
    0     // Extra floats stored ?
    1     // Extra longs stored ?
    2     // LATCH EGS variable stored in the extralong array [ 0]

$RECORD_CONSTANT:
    95.0000     // Constant Z
```

```
$RECORD_LENGTH:
29

$BYTE_ORDER:
1234

$ORIG_HISTORIES:
15000000

$PARTICLES:
75779

$PHOTONS:
75656

$ELECTRONS:
123

$TRANSPORT_PARAMETERS:
Photon transport cutoff(MeV)->0.010
Pair angular sampling->KM
Pair cross sections->BH
Triplet production->Off
Bound Compton scattering->ON
Radiative Compton corrections->Off
Rayleigh scattering->ON
Atomic relaxations->ON
Photoelectron angular sampling->ON
Electron transport cutoff(MeV)->0.001
Bremsstrahlung cross sections->NIST
Bremsstrahlung angular sampling->KM
Spin effects->On
Electron Impact Ionization->Off
Maxium electron step in cm (SMAX)->5.000
Maximum fractional energy loss/step (ESTEPE)->0.2500
Maximum 1st elastic moment/step (XIMAX)->0.5000
Boundary crossing algorithm->EXACT
Skin-depth for boundary crossing (MFP)->3.000
Electron-step algorithm->PRESTA-II

$MACHINE_TYPE:
SIEMENS PRIMUS 6MV

$MONTE_CARLO_CODE_VERSION:
version V1 of BEAMnrc (Rev 1.78 last edited 2004-01-12 11:44:06-05)

$GLOBAL_PHOTON_ENERGY_CUTOFF:
```

```
  0.010

$GLOBAL_PARTICLE_ENERGY_CUTOFF:
  0.001

$COORDINATE_SYSTEM_DESCRIPTION:
Cartesian Right-Handed, origin: z=0->bottom of the target,
x=y=0->central position of the target, z-axis parallel to beam direction,
incresing towards the patient, x-axis parallel to MLC leaf movement

//  OPTIONAL INFORMATION

$BEAM_NAME:
6MV photon beam

$FIELD_SIZE:
0.5 cm x 0.5 cm

$NOMINAL_SSD:
100 cm

$MC_INPUT_FILENAME:

$VARIANCE_REDUCTION_TECHNIQUES:
Bremsstrahlung splitting->DIRECTIONAL
splitting field radius->4.000 cm
splitting field SSD->100.000 cm
splitting no. in field->1500
Photon force interaction switch->OFF
Range rejection switch->ON
(IREJCT_GLOBAL = 2 => ECUTRR = ECUT(region) , ESAVE = 2.0 MeV)

$INITIAL_SOURCE_DESCRIPTION:
INITIAL PARTICLES->Electrons
PARALLEL BEAM WITH 2-D GAUSSIAN X-Y DISTRIBUTION ON FRONT FACE at Z->0.0000 cm
BEAM FWHM->0.10 cm
INCIDENT ENERGY SPECTRUM READ FROM .specdata FILE
================================================================================
Energy spectrum: gaussian centred at 6 MeV, FWHM =0.84 MeV
30, 4.90, 1
4.98, 1.4181119E-02
5.05, 2.5347747E-02
5.12, 4.3465299E-02
5.20, 7.1502697E-02
5.27, 1.1284451E-01
5.34, 1.7085137E-01
5.41, 2.4816358E-01
```

```
5.49, 3.4581247E-01
5.56, 4.6230390E-01
5.63, 5.9292519E-01
5.71, 7.2955606E-01
5.78, 8.6120131E-01
5.85, 9.7530087E-01
5.93, 1.0596464E+00
6.00, 1.1045161E+00
6.07, 1.1045161E+00
6.15, 1.0596464E+00
6.22, 9.7530087E-01
6.29, 8.6120131E-01
6.37, 7.2955606E-01
6.44, 5.9292519E-01
6.51, 4.6230390E-01
6.59, 3.4581247E-01
6.66, 2.4816358E-01
6.73, 1.7085137E-01
6.80, 1.1284451E-01
6.88, 7.1502697E-02
6.95, 4.3465299E-02
7.02, 2.5347747E-02
7.10, 1.4181119E-02
================================================================================
MEAN ENERGY->6.0 MeV
MINIMUM KINETIC ENERGY OF SPECTRUM->4.900 MeV
MAXIMUM KINETIC ENERGY OF SPECTRUM->7.100 MeV
NUMBER OF BINS IN SPECTRUM->30

$PUBLISHED_REFERENCE:
1->Automatic determination of primary electron beam parameters in Monte Carlo
simulation. 2007, Med. Phys. 34, 1076-1084
2->A new method for output factor determination in MLC shaped narrow beams.
Physica Medica 23 (2007) 58-66
3->Monte Carlo correction factors in small field absolute dosimetry:
a feasibility study (Unpublished, available from the phsp web site)

$AUTHORS:
J PENA, D M GONZALEZ-CASTANO, F SANCHEZ-DOBLADO, G HARTMANN

$INSTITUTION:
UNIVERSIDAD DE SANTIAGO DE COMPOSTELA (SPAIN)

$LINK_VALIDATION:

$ADDITIONAL_NOTES:
This is IAEA header as defined in the technical
```

```
report IAEA(NDS)-0484, Vienna, 2006

$STATISTICAL_INFORMATION_PARTICLES:
// Weight        Wmin       Wmax       <E>       Emin       Emax    Particle
    61.43   0.0006667          1     1.931     0.1006      6.869     PHOTONS
    0.082   0.0006667  0.0006667     1.143    0.01723      4.621   ELECTRONS

$STATISTICAL_INFORMATION_GEOMETRY:
 -14.6058  14.6422
 -13.2471  14.1754
```

APÉNDICE E

Subrutina SourcePhaseSpaceFile

```
!*****************************************************************
!*                          SOURCE                               *
!*                   PHASE SPACE FILE                            *
!*                                                               *
!* Short description:                                            *
!*   Generation of primary particle states for radiation transport *
!*   calculations with PENELOPE.                                 *
!*                                                               *
!*   Initial particle states are read from an external phase space *
!*   file (PSF). Notice that, due to the modification of history   *
!*   numbers according to the information in the PSF, this source  *
!*   is *incompatible* with other source models.                 *
!*                                                               *
!* Dependencies:                                                 *
!*   from PENELOPE:                                              *
!*   -> common /TRACK/                                           *
!*   -> routine STORES                                           *
!*   from PENAUX.F                                               *
!*   -> routine DOSTOP                                           *
!*   -> routine FINDUF                                           *
!*   from PENVOX.F                                               *
!*   -> routine LOCATEX                                          *
!*   -> routine STEPX                                            *
!*   from other penEasy libraries:                               *
!*   -> routine GETLINE,TALLY                                    *
!*                                                               *
!* Compatible with PENELOPE versions:                            *
!*   2005,2006                                                   *
!*****************************************************************
```

```
      subroutine PSFsource(n)
!*****************************************************************
!*    Input:                                                     *
!*      n -> top history counter                                 *
!*    Output:                                                    *
!*      through /track/ and sec stack                            *
!*      n -> top history counter                                 *
!*****************************************************************
      implicit none
      real*8 n

      integer*4 kpar,ibody,mat,ilb
      real*8 e,x,y,z,u,v,w,wght
      common/track/e,x,y,z,u,v,w,wght,kpar,ibody,mat,ilb(5)
      integer*4 kpars,ilbs,dns
      real*8 es,xs,ys,zs,us,vs,ws,wghts
      common /srcpsf/ es,xs,ys,zs,us,vs,ws,wghts,kpars,ilbs(5),dns
      logical active
      integer in,split,formatiaea
      real*8 rot,xshift,yshift,zshift,nlin
      common /srcps1/ rot(3,3),xshift,yshift,zshift,nlin,split,in,
     &                formatiaea,active
      logical getpar
      integer i
      integer*4 ncross
      real*8 infty,dsef,norm
      parameter (infty=1.0d30)

!IAEAini
      logical getparIAEA
      logical ilbstored,dnstored,ivarcnt(7)
      integer*4 source_read
      integer(selected_int_kind(R=18)) :: npcurriaea,npmaxiaea,ntopiaea
      real*4 varcnt(7)
      common /srciaeaps/ npcurriaea,npmaxiaea,ntopiaea,
     &       varcnt,source_read,ilbstored,dnstored,ivarcnt
!IAEAend

      if (.not.active) return

      if (formatiaea.eq.1.and.(.not.dnstored)) then  ! IAEA
        n = anint(ntopiaea*npcurriaea/dble(npmaxiaea))  ! IAEA
      else  ! IAEA
        n = n+dble(dns-1)
      endif  ! IAEA
```

```
 call tally(2,n)
 do
   ! Load previously stored particle into active common:
   kpar = kpars
   wght = wghts/split
   e = es
   ! Rotate and translate position:
   x = xshift+rot(1,1)*xs+rot(1,2)*ys+rot(1,3)*zs
   y = yshift+rot(2,1)*xs+rot(2,2)*ys+rot(2,3)*zs
   z = zshift+rot(3,1)*xs+rot(3,2)*ys+rot(3,3)*zs
   ! Rotate direction and renormalize to double precision:
   u = rot(1,1)*us+rot(1,2)*vs+rot(1,3)*ws
   v = rot(2,1)*us+rot(2,2)*vs+rot(2,3)*ws
   w = rot(3,1)*us+rot(3,2)*vs+rot(3,3)*ws
   norm = 1.0d0/sqrt(u**2+v**2+w**2)
   u = u*norm
   v = v*norm
   w = w*norm
   ilb(1) = ilbs(1)
   ilb(2) = ilbs(2)
   ilb(3) = ilbs(3)
   ilb(4) = ilbs(4)
   ilb(5) = ilbs(5)
   call locatex
   if (mat.eq.0) call stepx(infty,dsef,ncross)
   do i=1,split
     call stores(e,x,y,z,u,v,w,wght,kpar,ilb)
     call tally(0,e)
   enddo

   ! Read a new particle and store for later calls:
   if (formatiaea.eq.0) then  !IAEA
     if (.not.getpar()) then
       call dostop
       write(*,*) ''
       write(*,'(a)')
&        'PSFsource: PSF exhausted; simulation stopped forcefully'
       exit
     endif
   else  !IAEA
     if (.not.getparIAEA()) then  !IAEA
       n = anint(dble(ntopiaea))  !IAEA
       call dostop                !IAEA
       write(*,*) ''
       write(*,'(a)') 'PSFsource:WARNING: '//  !IAEA
&         'IAEA PSF exhausted; stopped forcefully'  !IAEA
       exit  !IAEA
```

```
          endif  !IAEA
        endif  !IAEA

        if (dns.ne.0) exit
      enddo
      end


      subroutine PSFinisrc(activated,emax)
!*****************************************************************
!*    Initializes. To be called before SOURCE.                   *
!*                                                               *
!*    Output:                                                    *
!*      activated -> TRUE if the source is active.               *
!*      emax -> max source energy (eV)                           *
!*****************************************************************
      implicit none
      logical activated
      real*8 emax

      integer*4 kpars,ilbs,dns
      real*8 es,xs,ys,zs,us,vs,ws,wghts
      common /srcpsf/ es,xs,ys,zs,us,vs,ws,wghts,kpars,ilbs(5),dns
      logical active
      integer in,split,formatiaea
      real*8 rot,xshift,yshift,zshift,nlin
      common /srcps1/ rot(3,3),xshift,yshift,zshift,nlin,split,in,
     &                formatiaea,active
      logical getpar,checkedFormat
      character*80 psfnam,buffer
      character*(*) secid,eos
      parameter (secid=
     &'[SECTION SOURCE PHASE SPACE FILE v.2008-06-01]')
      parameter (eos='[END OF SPSF SECTION]')
      integer finduf,error,validate
      real*8 ntop,nele,npho,npos
      real*8 omega,theta,phi,comega,ctheta,cphi,somega,stheta,sphi
      real*8 pi,deg2rad,mc2,twomc2
      parameter (pi=3.1415926535897932d0,deg2rad=pi/180.0d0)
      parameter (mc2=5.10998918d5,twomc2=2.0d0*mc2)

!IAEAini
      logical getparIAEA
      integer i
!IAEAend

      write(*,*) ''
```

```
 write(*,'(a)')
& '>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'
 call getline(buffer)
 if (index(buffer,secid).eq.0) then
   write(*,'(a)') 'PSFinisrc:ERROR: incorrect section header;'
   write(*,'(a,a)') '  expecting to find: ',secid
   write(*,'(a,a)') '  found instead:     ',buffer
   stop
 endif
 write(*,'(a)') secid

 read(*,'(1x,a3)') buffer
 if (adjustl(buffer(1:3)).eq.'ON') then
   active = .true.
   activated = active
 else if (buffer(1:3).eq.'OFF') then
   active = .false.
   activated = active
   write(*, '(a)')
&    '>>>> Source Phase Space File is OFF >>>>'
   do
     read(*,'(a80)',iostat=error) buffer
     if (error.ne.0) then
       write(*,'(a,a,a)') 'PSFinisrc:ERROR: ',
&       'Unable to find End-Of-Section mark: ',eos
       stop
     endif
     if (index(buffer,eos).ne.0) return
   enddo
 else
   write(*,'(a)')
&    'PSFinisrc:ERROR: expecting to find ON or OFF'
   write(*,'(a)') 'found instead:'
   write(*,'(a)') buffer(1:3)
   stop
 endif

 read(*,*) formatiaea
 if (formatiaea.eq.0) then
   write(*,'(a)') 'PSF format: standard penEasy in ASCII.'
 else if (formatiaea.eq.1) then
   write(*,'(a)') 'PSF format: IAEA, in binary.'
 else
   write(*,'(a)') 'PSFinisrc:ERROR: PSF format must be 0 or 1.'
   stop
 endif
```

```
 write(*,'(a)') 'PSF filename:'
 read(*,'(1x,a30)') psfnam
 write(*,'(1x,a)') psfnam

 write(*,'(a)') 'Splitting factor:'
 read(*,*) split
 write(*,'(1x,i0)') split
 if (split.lt.1) then
   write(*,'(a)') 'PSFinisrc:ERROR: split < 1'
   stop
 endif

 write(*,'(a)')
& 'Euler angles (deg) to rotate position and direction vectors:'
 read(*,*) omega,theta,phi
 write(*,'(3(1x,1pe12.5))') omega,theta,phi
 omega = omega*deg2rad
 theta = theta*deg2rad
 phi = phi*deg2rad
 ! Calculate rotation matrix:
 sphi   = sin(phi)
 cphi   = cos(phi)
 stheta = sin(theta)
 ctheta = cos(theta)
 somega = sin(omega)
 comega = cos(omega)
 rot(1,1) = cphi*ctheta*comega-sphi*somega
 rot(1,2) = -cphi*ctheta*somega-sphi*comega
 rot(1,3) = cphi*stheta
 rot(2,1) = sphi*ctheta*comega+cphi*somega
 rot(2,2) = -sphi*ctheta*somega+cphi*comega
 rot(2,3) = sphi*stheta
 rot(3,1) = -stheta*comega
 rot(3,2) = stheta*somega
 rot(3,3) = ctheta

 write(*,'(a)') 'Cartesian components of position shift (cm):'
 read(*,*) xshift,yshift,zshift
 write(*,'(3(1x,1pe12.5))') xshift,yshift,zshift

 read(*,*) validate
 if (formatiaea.eq.0) then  !IAEA
   if (validate.eq.1) then
     ! Pre-read PSF, validate and statistics:
     write(*,'(a)') 'Starting PSF validation'
     in = finduf()
     open(in,file=psfnam,status='old',iostat=error)
```

```
      if (error.ne.0) then
        write(*,'(a)') 'PSFinisrc:ERROR: cannot open the PSF'
        stop
      endif
      nlin = 0
      call checkFormat  ! Find if it is 2008-compliant
      checkedFormat = .true.

      emax = 0.0d0
      ntop = 0
      nele = 0
      npho = 0
      npos = 0
      do
        if (.not.getpar()) exit
        ! Count particles in PSF:
        ntop = ntop+dns
        if (kpars.eq.1) then
          nele = nele+1
        else if (kpars.eq.2) then
          npho = npho+1
        else if (kpars.eq.3) then
          npos = npos+1
        else
          write(*,'(a)') 'PSFinisrc:ERROR: invalid KPAR: in line:'
          write(*,'(1x,i10,1x,f18.0)') kpars,nlin
          stop
        endif
        if (es.lt.0.0d0.or.es.gt.1.0d9) then
          write(*,'(a)')
     &      'PSFinisrc:ERROR: invalid energy(eV): in line:'
          write(*,'(1pe12.5,1x,f18.0)') es,nlin
          stop
        endif
        emax = max(emax,es)
        if (us**2+vs**2+ws**2.lt.1.0d-30) then
          write(*,'(a)')
     &      'PSFinisrc:ERROR: null vector direction found in line:'
          write(*,'(f18.0)') nlin
          stop
        endif
      enddo
      close(in)

      write(*,'(a)') 'PSF statistics:'
      write(*,'(a)') '  No. electrons:'
      write(*,'(2x,f18.0)') nele
```

```
      write(*,'(a)') '  No. photons:'
      write(*,'(2x,f18.0)') npho
      write(*,'(a)') '  No. positrons:'
      write(*,'(2x,f18.0)') npos
      write(*,'(a)') '  No. particles, total:'
      write(*,'(2x,f18.0)') nele+npho+npos
      write(*,'(a)') '  No. top primary histories:'
      write(*,'(a)') '  (may be less than actual number if '//
     &   'last histories did not contribute to PSF)'
      write(*,'(2x,f18.0)') ntop
      write(*,'(a)') '  Max energy(eV):'
      write(*,'(2x,1pe12.5)') emax
      write(*,'(a)') ' (max energy declared in input file ignored)'
      if (npos.gt.0) emax = emax+twomc2
      read(*,'(a80)') buffer  ! Dummy line

    else  ! Do not validate the PSF
      write(*,'(a)') '** User opted not to pre-validate PSF **'
      write(*,'(a)') 'Max energy (eV) taken from input file:'
      read(*,*) emax
      write(*,'(1pe12.5)') emax
      checkedFormat = .false.
    endif

    ! Prepare for 1st call to SOURCE:
    in = finduf()
    open(in,file=psfnam,status='old',iostat=error)
    if (error.ne.0) then
      write(*,'(a)') 'PSFinisrc:ERROR: cannot open the PSF'
      stop
    endif
    nlin = 0
    if (.not.checkedFormat) call checkFormat
    if (.not.getpar()) then
      write(*,'(a)') 'PSFinisrc:ERROR: PSF is empty'
      stop
    endif

  else  !IAEA
    i = len_trim(psfnam)+1  !IAEA
    psfnam(i:i) = char(0)  !IAEA

    call IAEAread_ini(psfnam,validate,emax)  !IAEA
  endif

  read(*,'(a80)') buffer
  if (index(buffer,eos).eq.0) then
```

```
        write(*,'(a)')
     &    'PSFinisrc:ERROR: End-Of-Section mark not found'
        write(*,'(a,a)') '  expecting to find: ',eos
        write(*,'(a,a)') '  found instead:     ',buffer
        stop
       endif
       write(*,'(a)') '>>>> PSF source initialization finished >>>>'
       end


       logical function getpar()
!*****************************************************************
!*    Reads a new particle from the PSF.                         *
!*                                                               *
!*    Output:                                                    *
!*      -> returns .false. if EOF has been reached, else .true.  *
!*      -> particle state in /srcpsf/                            *
!*****************************************************************
       implicit none
       integer*4 kpars,ilbs,dns
       real*8 es,xs,ys,zs,us,vs,ws,wghts
       common /srcpsf/ es,xs,ys,zs,us,vs,ws,wghts,kpars,ilbs(5),dns
       logical active
       integer in,split,formatiaea
       real*8 rot,xshift,yshift,zshift,nlin
       common /srcps1/ rot(3,3),xshift,yshift,zshift,nlin,split,in,
     &                 formatiaea,active
       logical format2008
       common /formatVer/ format2008
       character*256 buffer
       integer error
       real*8 maxn
       parameter (maxn=1.0d15)

       do
         read(in,'(a256)',end=10,iostat=error) buffer
         if (error.ne.0) then
           write(*,'(a)')
     &      'getpar:ERROR: unable to read PSF line; last line read:'
           write(*,'(f18.0)') nlin
           stop
         endif
         if (nlin.gt.-0.5d0) nlin = nlin+1
         if (nlin.gt.maxn) then
           write(*,*) ''
           write(*,'(a)') '****************************************'
           write(*,'(a,f18.0)')
```

```
     &    'getpar:WARNING: No. of lines in PSF exceeds ',maxn
          write(*,'(a)')
     &    '  This is too large for penEasy real*8 counters to handle'
          write(*,'(a)')
     &    '  so, continuing without counting lines.'
          write(*,'(a)') '****************************************'
          write(*,*) ''
          nlin = -1
        endif
        if (buffer(1:1).ne.'#') exit
      enddo

      if (format2008) then  ! 2008-compliant format
        read(buffer,*,iostat=error)
     &    kpars,es,xs,ys,zs,us,vs,ws,wghts,dns,
     &    ilbs(1),ilbs(2),ilbs(3),ilbs(4),ilbs(5)
      else               ! Pre-2008 format
        read(buffer,*,iostat=error)
     &    kpars,es,xs,ys,zs,us,vs,ws,wghts,ilbs(5),dns
        ilbs(1) = 1      ! Assume all particles are primaries
        ilbs(2) = 0
        ilbs(3) = 0
        ilbs(4) = 0
      endif
      if (error.ne.0) then
        write(*,'(a)')
     &    'getpar:ERROR: invalid or missing datum in PSF line:'
        write(*,'(f20.0)') nlin
        write(*,'(a)') '  line contents:'
        write(*,'(a)') buffer
        stop
      endif
      getpar = .true.
      return

 10   getpar = .false.  ! EOF
      end


      subroutine checkFormat
!*********************************************************************
!*    Checks if PSF data format is 2008-compliant.                   *
!*********************************************************************
      implicit none
      logical active
      integer in,split,formatiaea
      real*8 rot,xshift,yshift,zshift,nlin
```

```
 common /srcps1/ rot(3,3),xshift,yshift,zshift,nlin,split,in,
&                formatiaea,active
 logical format2008
 common /formatVer/ format2008
 character*256 buffer,id2008,id2006
 parameter (id2008=
& '# [PHASE SPACE FILE FORMAT penEasy v.2008-05-15]')
 parameter (id2006=
& '# kpar : e : x : y : z : u : v : w : wght : ilb(5) : DeltaN')
 integer error

 read(in,'(a256)',iostat=error) buffer
 if (error.ne.0) then
   write(*,'(a)')
&    'ckeckFormat:ERROR: unable to read first PSF line.'
   stop
 endif
 nlin = nlin+1

 if (buffer.eq.id2008) then
   format2008 = .true.
 else if (buffer.eq.id2006) then
   format2008 = .false.
   write(*,*) ''
   write(*,'(a)')
&    '********************'
   write(*,'(a)')
&    'checkFormat:WARNING: header for 2008-compliant format '//
&    'not found;'
   write(*,'(a)')
&    '  assuming pre-2008 data format, which consists of:'
   write(*,'(a)')
&    '  KPAR:E:X:Y:Z:U:V:W:WGHT:ILB(5):DeltaN'
   write(*,'(a)')
&    '********************'
   write(*,*) ''
 else
   format2008 = .false.
   write(*,*) ''
   write(*,'(a)')
&  'checkFormat:ERROR: unable to identify PSF format;'
   write(*,'(a)')
&  '  expecting one of these two headers in first line of PSF:'
   write(*,'(a)') id2008
   write(*,'(a)') id2006
   write(*,'(a)')
&    '  (for 2008 and 2006 formats, respectively) '//
```

```
     &    'but neither was found.'
          stop
      endif
      end

!IAEAini

      logical function getparIAEA()
!*****************************************************************
!*    Reads a new particle from the IAEA PSF.                    *
!*                                                               *
!*    Output:                                                    *
!*      -> returns .false. if EOF has been reached, else .true.  *
!*      -> particle state in /srciaeapsf/                        *
!*****************************************************************
      implicit none

      integer*4 kpars,ilbs,dns
      real*8 es,xs,ys,zs,us,vs,ws,wghts
      common /srcpsf/ es,xs,ys,zs,us,vs,ws,wghts,kpars,
     &                   ilbs(5),dns

      logical ilbstored,dnstored,ivarcnt(7)
      integer*4 psfformat,source_read,n_stat,kpariaea,extra_ints(3)
      integer(kind=8) :: npcurriaea,npmaxiaea,ntopiaea
      real*4 extra_floats(3),varcnt(7)
      real*4 eiaea,xiaea,yiaea,ziaea,uiaea,viaea,wiaea,wghtiaea
      common /srciaeaps/ npcurriaea,npmaxiaea,ntopiaea,
     &       varcnt,source_read,ilbstored,dnstored,ivarcnt

      npcurriaea = npcurriaea+1

      if (npcurriaea.le.npmaxiaea) then
        call iaea_get_particle(source_read,n_stat,kpariaea,eiaea,
     &  wghtiaea,xiaea,yiaea,ziaea,uiaea,viaea,wiaea,extra_floats,
     &  extra_ints)

        if (n_stat.eq.-2) then
          write(*,'(a)')
     &    'getparIAEA:ERROR: EOF reached before reading all particles'
          stop
        else if (n_stat.eq.-1) then
          write(*,'(a)') 'getparIAEA:ERROR: Reading particle error'
          stop
        else

          if (kpariaea.eq.1) then
```

```
  kpars = 2
else if (kpariaea.eq.2) then
  kpars = 1
else if (kpariaea.eq.3) then
  kpars = 3
else
  write(*,'(a)') 'getparIAEA:ERROR: Invalid KPAR'
  stop
endif

es = dble(eiaea*1.0e6)

if (ivarcnt(1)) then
  xs = dble(varcnt(1))
else
  xs = dble(xiaea)
endif
if (ivarcnt(2)) then
  ys = dble(varcnt(2))
else
  ys = dble(yiaea)
endif
if (ivarcnt(3)) then
  zs = dble(varcnt(3))
else
  zs = dble(ziaea)
endif
if (ivarcnt(4)) then
  us = dble(varcnt(4))
else
  us = dble(uiaea)
endif
if (ivarcnt(5)) then
  vs = dble(varcnt(5))
else
  vs = dble(viaea)
endif
if (ivarcnt(6)) then
  ws = dble(varcnt(6))
else
  ws = dble(wiaea)
endif
if (ivarcnt(7)) then
  wghts = dble(varcnt(7))
else
  wghts = dble(wghtiaea)
endif
```

```
          ilbs(1) = 1
          ilbs(2) = 0
          ilbs(3) = 0
          ilbs(4) = 0

          if (ilbstored) then
            ilbs(5) = extra_ints(2)
          else
            ilbs(5) = 0
          endif

          if (dnstored) then
            dns = extra_ints(1)
          else
            dns = 1
          endif

          getparIAEA = .true.
          return
        endif
      else
        getparIAEA = .false.  ! EOF
      endif
      end


      subroutine IAEAread_ini(psfnam,validate,emax)
!*********************************************************************
!*    Initializes                                                    *
!*    Input:                                                         *
!*      psfnam ->  PSF filename                                      *
!*      validate ->  Validation before simulation                    *
!*    Output:                                                        *
!*      emax ->  Maximum energy of the PSF                           *
!*********************************************************************

      logical getparIAEA
      logical ilbstored,dnstored,ivarcnt(7)
      character*80 psfnam
      integer i,validate
      integer*4 source_read,access_read,result,n_stat
      integer*4 kpariaea,extrafloat,extralong
      integer*4 extrafloattypes(3),extralongtypes(3),extra_ints(3)
      integer(selected_int_kind(R=18)) :: npcurriaea,npmaxiaea,ntopiaea
      real*4 eiaea,xiaea,yiaea,ziaea,uiaea,viaea,wiaea,wghtiaea
      real*4 emaxiaea,extra_floats(3),constant,varcnt(7)
```

```
 real*8 emax,emaxcheck
 real*8 mc2,twomc2
 parameter (access_read=1)
 parameter (mc2=5.10998918d5,twomc2=2.0d0*mc2)
 common /srciaeaps/ npcurriaea,npmaxiaea,ntopiaea,
&       varcnt,source_read,ilbstored,dnstored,ivarcnt
 real*8 nele,npho,npos
 real*8 es,xs,ys,zs,us,vs,ws,wghts


 !*** Init vars:
 ilbstored = .false.
 dnstored = .false.

 call iaea_new_source(source_read,psfnam,access_read,result)
 if (result.lt.0) then
   write(*,'(a)') 'IAEAread_ini:ERROR: '//
&  'Error creating new source for reading particles'
   stop
 endif
 call iaea_get_max_particles(source_read,-1,npmaxiaea)
 if (npmaxiaea.lt.0) then
   write(*,'(a)') 'IAEAread_ini:ERROR: '//
&  'Unable to get the total number of particles'
   stop
 endif
 call iaea_get_total_original_particles(source_read,ntopiaea)
 if (ntopiaea.lt.0) then
   write(*,'(a)') 'IAEAread_ini:ERROR: '//
&  'Unable to get the total number of histories'
   stop
 endif

 do i = 1,7
   call iaea_get_constant_variable(source_read,i-1,constant,
&  result)
   if (result.eq.-1) then
     write(*,'(a)')
&    'IAEAread_ini:ERROR: Unable to get constant variables'
     stop
   else if (result.eq.-2) then
     write(*,'(a)')
&    'IAEAread_ini:ERROR: Variable constant index is out of range'
     stop
   else if (result.eq.-3) then
     ivarcnt(i) = .false.
     varcnt(i) = 0.0e0
```

```
        else
          ivarcnt(i) = .true.
          varcnt(i) = constant
        endif
      enddo

      if (.not.(ivarcnt(1).or.ivarcnt(2).or.ivarcnt(3).or.
     & ivarcnt(4).or.ivarcnt(5).or.ivarcnt(6).or.ivarcnt(7))) then
        write(*,'(a)') 'No variables stored as a constant'
      else
        write(*,'(a)') 'Variables stored as a constant: '
      endif

      if (ivarcnt(1)) then
        write(*,'(2x,a,1pe12.5,a)') 'x = ',varcnt(1),' cm'
      else if (ivarcnt(2)) then
write(*,'(2x,a,1pe12.5,a)') 'y = ',varcnt(2),' cm'
      else if (ivarcnt(3)) then
        write(*,'(2x,a,1pe12.5,a)') 'z = ',varcnt(3),' cm'
      else if (ivarcnt(4)) then
        write(*,'(2x,a,1pe12.5)') 'u = ',varcnt(4)
      else if (ivarcnt(5)) then
        write(*,'(2x,a,1pe12.5)') 'v = ',varcnt(5)
      else if (ivarcnt(6)) then
        write(*,'(2x,a,1pe12.5)') 'w = ',varcnt(6)
      else if (ivarcnt(7)) then
        write(*,'(2x,a,1pe12.5)') 'weight = ',varcnt(7)
      endif

      call iaea_get_extra_numbers(source_read,extrafloat,extralong)

      call iaea_get_type_extra_variables(source_read,result,
     &  extralongtypes,extrafloattypes)

      write(*,'(a,i2)') 'No. extra floats stored: ',extrafloat
      if (extrafloat.ne.0) then
        do i=1,extrafloat
          if (extrafloattypes(i).eq.1) then
            write(*,'(2x,a)') 'XLAST (EGS) variable stored'
          else if (extrafloattypes(i).eq.2) then
            write(*,'(2x,a)') 'YLAST (EGS) variable stored'
          else if (extrafloattypes(i).eq.3) then
            write(*,'(2x,a)') 'ZLAST (EGS) variable stored'
          else
            write(*,'(2x,a)') 'User defined variable stored'
          endif
        enddo
```

```
 endif

 write(*,'(a,i2)') 'No. extra longs stored: ',extralong
 if (extralong.ne.0) then
   do i=1,extralong
     if (extralongtypes(i).eq.1) then
       write(*,'(2x,a)') 'Incremental history number stored'
       dnstored = .true.
     else if (extralongtypes(i).eq.2) then
       write(*,'(2x,a)') 'LATCH (EGS) stored'
     else if (extralongtypes(i).eq.3) then
       write(*,'(2x,a)') 'ILB(5) (PENELOPE) stored'
       ilbstored = .true.
     else
       write(*,'(2x,a)') 'User defined variable stored'
     endif
   enddo
 endif
 if (.not.dnstored) then
   write(*,'(2x,a)')
&  'Incremental history number not stored. '//
&  'The number of histories is read from the PSF header file'
 endif

 if (validate.eq.1) then  !IAEA

   !*** Init vars:
   nele = 0
   npho = 0
   npos = 0
   npcurriaea = 0
   ntopiaeacheck = 0
   es = 0.

   ! Pre-read IAEA PSF, validate and statistics:
   write(*,'(a)') 'Starting IAEA PSF validation'

   call iaea_check_file_size_byte_order(source_read,result)
   if (result.eq.-1) then
     write(*,'(a)') 'IAEAread_validation:ERROR: '//
&    'Unable to check file size and byte order of the PSF'
   stop
   else if (result.eq.-2) then
     write(*,'(a)') 'IAEAread_validation:ERROR: '//
&    'The function fseek fails'
     stop
   else if (result.eq.-3) then
```

```
      write(*,'(a)') 'IAEAread_validation:ERROR: File size',
&     'does not equals the value of checksum in the header'
      stop
    else if (result.eq.-4) then
      write(*,'(a)') 'IAEAread_validation:ERROR: '//
&     'There is a byte order mismatch'
      stop
    else if (result.eq.-5) then
      write(*,'(a)') 'IAEAread_validation:ERROR: '//
&     'There is a file size and byte order mismatch'
      stop
    endif

    do j=1,npmaxiaea
       call iaea_get_particle(source_read,n_stat,kpariaea,eiaea,
&      wghtiaea,xiaea,yiaea,ziaea,uiaea,viaea,wiaea,extra_floats,
&      extra_ints)
    if (n_stat.eq.-1) then
      write(*,'(a)')
&     'IAEAread_validation:ERROR: Reading particle error'
      stop
    endif
    if (n_stat.eq.-2) then
      write(*,'(a)') 'IAEAread_validation:ERROR: '//
&     'EOF reached before reading all particles'
       stop
    endif

    if (kpariaea.eq.2) then
      nele = nele+1
    else if (kpariaea.eq.1) then
      npho = npho+1
    else if (kpariaea.eq.3) then
      npos = npos+1
    else
      write(*,'(a)') 'IAEAread_validation:ERROR: invalid KPAR'
      stop
    endif

    es = dble(eiaea*1.0e6)
    if (kpariaea.eq.3) then
      es = es+twomc2
    endif
    if (es.gt.emaxcheck) then
      emaxcheck = es
    endif
```

```
      if (ivarcnt(4)) then
        us = dble(varcnt(4))
      else
        us = dble(uiaea)
      endif
      if (ivarcnt(5)) then
        vs = dble(varcnt(5))
      else
        vs = dble(viaea)
      endif
      if (ivarcnt(6)) then
        ws = dble(varcnt(6))
      else
        ws = dble(wiaea)
      endif

      if (dnstored) then
        ntopiaeacheck = ntopiaeacheck+extra_ints(1)
      else
        ntopiaeacheck = ntopiaeacheck+1
      endif

      if (es.lt.0.0d0.or.es.gt.1.0d9) then
        write(*,*) 'IAEAread_validation:ERROR: invalid energy(eV)'
      stop
      endif
      if (us**2+vs**2+ws**2.lt.1.0d-30) then
        write(*,'(a)')
     &   'IAEAread_validation:ERROR: null vector direction found'
        stop
      endif
      enddo

      if (ntopiaeacheck.gt.ntopiaea) then
        write(*,'(a)')
     &  'IAEAread_validation:ERROR: Calculated number of histories '//
     &  'is greater than the number of histories in header file'
          stop
      endif

      write(*,'(a)') 'IAEA PSF statistics:'
      write(*,'(a)') '  No. electrons:'
      write(*,'(2x,f18.0)') nele
      write(*,'(a)') '  No. photons:'
      write(*,'(2x,f18.0)') npho
      write(*,'(a)') '  No. positrons:'
      write(*,'(2x,f18.0)') npos
```

```
        write(*,'(a)') '  No. particles, total:'
        write(*,'(2x,f18.0)') nele+npho+npos
        write(*,'(a)') '  No. top primary histories:'
        write(*,'(2x,i20)') ntopiaea
        read(*,*) buffer  ! Dummy line

      else  ! Do not validate the IAEA PSF
        read(*,*) buffer  ! Dummy line  !IAEA
      endif  !IAEA

      call iaea_get_maximum_energy(source_read,emaxiaea)
      if (emaxiaea.lt.0.0e0) then
        write(*,'(a)')
     &  'IAEAread_ini:ERROR: Unable to get the maximum energy'
        stop
      endif
      emax = dble(emaxiaea)
      emaxcheck = emaxcheck/1.0d6
      if (emaxcheck.gt.emax) then
        emax = emaxcheck
      endif
      write(*,'(a)') 'Maximum energy (MeV):'
      write(*,'(2x,1pe12.5)') emax
      emax = emax*1.0d6

      if (validate.eq.0) then  !IAEA
        write(*,'(a)') '** User refused to pre-validate IAEA PSF **'  !IAEA
      endif  !IAEA

      call iaea_new_source(source_read,psfnam,access_read,result)  !IAEA
      if (result.lt.0) then  !IAEA
        write(*,'(a)') 'PSFinisrc:ERROR: '//  !IAEA
     &  'Error creating new source for reading particles'  !IAEA
        stop  !IAEA
      endif  !IAEA
      if (.not.getparIAEA()) then  !IAEA
        write(*,'(a)') 'PSFinisrc:ERROR: PSF is empty'  !IAEA
        stop  !IAEA
      endif  !IAEA
      end

!IAEAend

!>>>> End Of File >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

APÉNDICE F

Subrutina TallyPhaseSpaceFile

```
!*******************************************************************
!*                          TALLY                                  *
!*                   PHASE SPACE FILE                              *
!*                                                                 *
!* Short description:                                              *
!*   Tally routines for radiation transport calculations with      *
!*   PENELOPE.                                                     *
!*                                                                 *
!*   Writes to a file the state of all particles that reach        *
!*   a given material, considered as a particle sink.              *
!*                                                                 *
!* Dependencies:                                                   *
!*   from PENELOPE:                                                *
!*   -> common /TRACK/                                             *
!*   -> common /RSEED/                                             *
!*   from other penEasy files:                                     *
!*   -> routine GETLINE,FINDUF                                     *
!*                                                                 *
!* Compatible with PENELOPE versions:                              *
!*   2005,2006                                                     *
!*******************************************************************

      subroutine PSFtally(mode,arg)
!*******************************************************************
!*    Input:                                                       *
!*      mode -> Identifies the state of the calling procedure      *
!*      arg -> current history number (when mode=1)                *
!*******************************************************************
      implicit none
```

```
 integer mode
 real*8 arg

 integer*4 kpar,ibody,mat,ilb
 real*8 e,x,y,z,u,v,w,wght
 common/track/e,x,y,z,u,v,w,wght,kpar,ibody,mat,ilb(5)
 logical active
 integer detmat,psfunit,formatiaea
 real*8 nhist,nhlast,npar,nele,npos
 common /scopsf/ nhist,nhlast,npar,nele,npos,detmat,psfunit,
&                formatiaea,active
 character*80 fmtstr
 parameter (fmtstr='(i0,8(1x,es12.5),6(1x,i0))')
 integer*4 dn

 integer*4 source_write  !IAEA
 common /scoiaeapsf/ source_write,dn  !IAEA

 if (.not.active) return

 if (mode.eq.4.or.mode.eq.-99) then
   if (mat.ne.detmat) return
   dn = nhist-nhlast+0.5d0

   if (formatiaea.eq.0) then  !IAEA
     write(psfunit,fmtstr) kpar,e,x,y,z,u,v,w,wght,dn,
&                          ilb(1),ilb(2),ilb(3),ilb(4),ilb(5)
   else  !IAEA
     call IAEAwrite  !IAEA
   endif  !IAEA

   nhlast = nhist
   npar = npar+1
   if (kpar.eq.1) then
     nele = nele+1
   else if (kpar.eq.3) then
     npos = npos+1
   endif
   ! mat = 0
   ! Particle is stopped by setting Eabs=+infty in the PSF detector

 else if (mode.eq.1.or.mode.eq.2) then
   nhist = arg

 endif
 end
```

```
      subroutine PSFreport(n,cputim)
!*****************************************************************
!*    Input:                                                     *
!*      n -> no. of histories simulated                          *
!*      cputim -> elapsed CPU time                               *
!*    Comments:                                                  *
!*      -> 'cputim' should not include initialization procedures;  *
!*         enter 0 or neg. if not available.                     *
!*****************************************************************
      implicit none
      real*8 n,cputim

      integer*4 seed1,seed2
      common/rseed/seed1,seed2
      logical active
      integer detmat,psfunit,formatiaea
      real*8 nhist,nhlast,npar,nele,npos
      common /scopsf/ nhist,nhlast,npar,nele,npos,detmat,psfunit,
     &                formatiaea,active
      integer out,finduf,error

      if (.not.active) return

      ! Prepare output files:
      out = finduf()
      open(out,file='tallyPhaseSpaceFile.dat',iostat=error)
      if (error.ne.0) then
        write(*,*)
        write(*,'(a)')
     &    '*****************************************'
        write(*,'(a)')
     &    'PSFreport:ERROR: cannot open output data file'
        write(*,'(a)')
     &    '*****************************************'
        close(out)  ! Just in case
        return
      endif

      write(out,'(a)')
     &'#>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'
      write(out,'(a)') '# [SECTION REPORT PSF]'
      write(out,'(a)')
     &'# No. of electrons/photons/positrons/total written to PSF:'
      write(out,'(f18.0)') nele
      write(out,'(f18.0)') npar-nele-npos
      write(out,'(f18.0)') npos
```

```
      write(out,'(f18.0)') npar

      ! Generic report:
      write(out,*)
      write(out,'(a)') '# Performance report'
      write(out,'(a)') '#   Random seeds:'
      write(out,'(a,i10)') '#   ',seed1
      write(out,'(a,i10)') '#   ',seed2
      write(out,'(a)') '#   No. of histories simulated [N]:'
      write(out,'(a,f18.0)') '#   ',n
      write(out,'(a)') '#   CPU time [t] (s):'
      write(out,'(a,1pe12.5)') '#   ',cputim
      if (cputim.gt.0.0d0) then
        write(out,'(a)') '#   Speed (histories/s):'
        write(out,'(a,1pe12.5)') '#   ',n/cputim
      endif
      write(out,'(a)') '#'
      write(out,'(a)') '# Have a nice day.'
      close(out)

      if (formatiaea.eq.1) then  !IAEA
        call IAEAwrite_report  !IAEA
      endif !IAEA
      end


      subroutine PSFinitally
!*********************************************************************
!*    Initializes. To be called before TALLY                         *
!*********************************************************************
      implicit none
      logical active
      integer detmat,psfunit,formatiaea
      real*8 nhist,nhlast,npar,nele,npos
      common /scopsf/ nhist,nhlast,npar,nele,npos,detmat,psfunit,
     &                formatiaea,active
      character*(*) secid,eos
      parameter (secid=
     &'[SECTION TALLY PHASE SPACE FILE v.2008-06-01]')
      parameter (eos='[END OF PSF SECTION]')
      character*80 buffer
      integer finduf,error
      integer i  !IAEA

      write(*,*) ' '
      write(*,'(a)')
     & '>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>'
```

```
 call getline(buffer)
 if (index(buffer,secid).eq.0) then
   write(*,'(a)') 'PSFinitally:ERROR: incorrect section header;'
   write(*,'(a,a)') '  expecting to find: ',secid
   write(*,'(a,a)') '  found instead:     ',buffer
   stop
 endif
 write(*,'(a)') secid

 read(*,'(1x,a3)') buffer
 if (adjustl(buffer(1:3)).eq.'ON') then
   active = .true.
 else if (buffer(1:3).eq.'OFF') then
   active = .false.
   write(*, '(a)')
&    '>>>> Tally Phase Space File is OFF >>>>'
   do
     read(*,'(a80)',iostat=error) buffer
     if (error.ne.0) then
       write(*,'(a,a,a)') 'PSFinitally:ERROR: ',
&       'Unable to find End-Of-Section mark: ',eos
       stop
     endif
     if (index(buffer,eos).ne.0) return
   enddo
 else
   write(*,'(a)')
&    'PSFinitally:ERROR: expecting to find ON or OFF'
   write(*,'(a)') 'found instead:'
   write(*,'(a)') buffer(1:3)
   stop
 endif

 read(*,*) formatiaea
 if (formatiaea.eq.0) then
   write(*,'(a)') 'PSF format: standard penEasy in ASCII.'
 else if (formatiaea.eq.1) then
   write(*,'(a)') 'PSF format: IAEA, in binary.'
 else
   write(*,'(a)') 'PSFinitally:ERROR: PSF format must be 0 or 1.'
   stop
 endif

 write(*,'(a)') 'Detection material set to:'
 read(*,*) detmat
 write(*,'(i3)') detmat
 if (detmat.le.0) then
```

```
      write(*,*) 'PSFinitally:ERROR: detection material must be >0'
      stop
    endif
    write(*,'(a)') 'PSF filename:'
    read(*,'(1x,a30)') buffer
    write(*,'(a)') buffer
    psfunit = finduf()
    write(*,'(a)') 'Opening PSF as unit:'
    write(*,'(1x,i0)') psfunit

    if (formatiaea.eq.0) then  !IAEA
      open(psfunit,file=buffer)
      write(psfunit,'(a)')
&     '# [PHASE SPACE FILE FORMAT penEasy v.2008-05-15]'
      write(psfunit,'(a)')
&     '# KPAR : E : X : Y : Z : U : V : W : WGHT : '//
&     'DeltaN : ILB(1..5)'

    else  !IAEA
      i = len_trim(buffer)+1  !IAEA
      buffer(i:i) = char(0)  !IAEA

      call IAEAwrite_ini(buffer)  !IAEA
    endif  !IAEA

    !*** Init vars:
    nhist  = 0.0d0
    nhlast = 0.0d0
    npar   = 0.0d0
    nele   = 0.0d0
    npos   = 0.0d0

    read(*,'(a80)') buffer
    if (index(buffer,eos).eq.0) then
      write(*,*) 'PSFinitally:ERROR: End-Of-Section mark not found'
      write(*,'(a,a)') '  expecting to find: ',eos
      write(*,'(a,a)') '  found instead:     ',buffer
      stop
    endif
    write(*,'(a)') '>>>> PSF tally initialization finished >>>>'
    end
```

```
!IAEAini

      subroutine IAEAwrite_ini(buffer)
!*****************************************************************
!*    Initializes                                                *
!*    Input:                                                     *
!*      buffer ->  PSF filename                                  *
!*****************************************************************

      implicit none
      character*80 buffer
      integer*4 source_write,dn,access_write,result
      integer*4 extrafloat,extralong,ind,typ,extralong_type
      integer*4 extra_ints(3)
      real*4 extra_floats(3)
      parameter (access_write=2)
      parameter (extrafloat=0,extralong=2)
      common /scoiaeapsf/ source_write,dn


      call iaea_new_source(source_write,buffer,access_write,result)
      if (result.lt.0) then
        write(*,'(a)') 'IAEAwrite_ini:ERROR: '//
     &  'Error creating new source for writing particles'
        stop
      endif

      call iaea_set_extra_numbers(source_write,extrafloat,extralong)

      ind = 0             ! First extra long-int number
      extralong_type = 1  ! deltaN
      call iaea_set_type_extralong_variable(source_write,ind,
     &  extralong_type)
      if (extralong_type.eq.-1) then
        write(*,'(a)') 'IAEAwrite_ini:ERROR: '//
     &  'Unable to set extralong variable'
        stop
      else if (extralong_type.eq.-2) then
        write(*,'(a)') 'IAEAwrite_ini:ERROR: '//
     &  'The extralong variable index is out of range'
        stop
      else if (extralong_type.eq.-3) then
        write(*,'(a)') 'IAEAwrite_ini:ERROR: '//
     &  'The extralong variable type is out of range'
        stop
      endif
```

```
      ind = 1             ! Second extra long-int number
      extralong_type = 3  ! ilb(5)
      call iaea_set_type_extralong_variable(source_write,ind,
     &  extralong_type)
      if (extralong_type.eq.-1) then
        write(*,'(a)') 'IAEAwrite_ini:ERROR: '//
     &  'Unable to set extralong variable'
        stop
      else if (extralong_type.eq.-2) then
        write(*,'(a)') 'IAEAwrite_ini:ERROR: '//
     &  'The extralong variable index is out of range'
        stop
      else if (extralong_type.eq.-3) then
        write(*,'(a)') 'IAEAwrite_ini:ERROR: '//
     &  'The extralong variable type is out of range'
        stop
      endif
      end


      subroutine IAEAwrite
!*********************************************************************
!*    Write a particle to the PSF in IAEA format                     *
!*********************************************************************

      implicit none

      integer*4 kpar,ibody,mat,ilb
      real*8 e,x,y,z,u,v,w,wght
      common/track/e,x,y,z,u,v,w,wght,kpar,ibody,mat,ilb(5)
      integer*4 source_write,dn,n_stat,kpariaea
      integer*4 extra_ints(3)
      real*4 extra_floats(3)
      real*4 eiaea,xiaea,yiaea,ziaea,uiaea,viaea,wiaea,wghtiaea
      common /scoiaeapsf/ source_write,dn

      if (kpar.eq.1) then
        kpariaea = 2
      else if (kpar.eq.2) then
        kpariaea = 1
      else if (kpar.eq.3) then
        kpariaea = 3
      else
        write(*,'(a)') 'IAEAwrite:ERROR: Invalid KPAR'
        stop
      endif
```

```
      eiaea = real(e/1.0d6)  ! E in MeV
      xiaea = real(x)
      yiaea = real(y)
      ziaea = real(z)
      uiaea = real(u)
      viaea = real(v)
      wiaea = real(w)
      wghtiaea = real(wght)

      extra_ints(1) = dn
      extra_ints(2) = ilb(5)
      extra_ints(3) = 0
      extra_floats(1) = 0.
      extra_floats(2) = 0.
      extra_floats(3) = 0.

      n_stat = dn

      call iaea_write_particle(source_write,n_stat,kpariaea,eiaea,
     &  wghtiaea,xiaea,yiaea,ziaea,uiaea,viaea,wiaea,extra_floats,
     &  extra_ints)

      if (n_stat.eq.-1) then
        write(*,'(a)') 'IAEAwrite:ERROR: Writing particle error'
        stop
      endif
      end


      subroutine IAEAwrite_report
!*********************************************************************
!*    Uptate IAEA header                                             *
!*********************************************************************

      implicit none
      logical active
      integer detmat,psfunit,formatiaea
      real*8 nhist,nhlast,npar,nele,npos
      common /scopsf/ nhist,nhlast,npar,nele,npos,detmat,psfunit,
     &                formatiaea,active
      integer*4 source_write,dn,result
      integer(selected_int_kind(R=18)) :: ntopiaea
      common /scoiaeapsf/ source_write,dn

      call flush(6)
      ntopiaea = nhist+0.5d0
      call iaea_set_total_original_particles(source_write,ntopiaea)
```

```
      if (ntopiaea.lt.0) then
        write(*,'(a)') 'IAEAwrite_report:ERROR: '//
     &  'Unable to write the number of histories to the header file'
        stop
      endif
      call iaea_update_header(source_write,result)
      if (result.lt.0) then
        write(*,'(a)') 'IAEAwrite_report:ERROR: '//
     &  'Updating header file error'
        stop
      endif
      end

!IAEAend

!>>>> End Of File >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Agradecimientos

Quiero agradecer a mis supervisores, el Dr. Josep Sempau y el Dr. José María Fernández-Varea, por su soporte científico y gran apoyo en mi etapa de formación como investigadora.

También agradezco la ayuda del Dr. Yuri Kubyshin, de Francesc Verdera y de Juan Pablo Rigla en la parte de diseño y optimización del aplicador de un microtrón de pista para radioterapia intraoperatoria y la ayuda de la Dra. María Amor Duch en la parte de adaptación a penEasy para el uso de una base de datos del OIEA en radioterapia externa.

Finalmente, agradezco las ayudas económicas recibidas por parte del *Departament d'Educació i Universitats de la Generalitat de Catalunya* y del Fondo Social Europeo y de l'*Institut de Tècniques Energètiques* (*Universitat Politècnica de Catalunya*), sin los cuales no hubiera sido posible el desarrollo del presente proyecto.

Bibliografía

[1] Ministerio de Sanidad y Consumo. Radioterapia intraoperatoria. Informe de Evaluación de Tecnologías Sanitarias, 1999.

[2] Gunderson L. [et al.]. *Intraoperative Irradiation: Techniques and Results*. Humana Press, 1999.

[3] Calvo F.A., Meirino R.M., and Orecchia R. Intraoperative radiation therapy. First part: rationale and techniques. *Critical Reviews in Oncology/Hematology*, 59:106–115, 2006.

[4] Calvo F.A., Meirino R.M., and Orecchia R. Intraoperative radiation therapy. Part 2: clinical results. *Critical Reviews in Oncology/Hematology*, 59:116–127, 2006.

[5] XIX Russian Particle Accelerator Conference. *Conceptual Design of the Miniature Electron Accelerator Dedicated to IORT*, Dubna, Russia, 2004.

[6] Wikipedia, la enciclopedia libre. `http://es.wikipedia.org/wiki/Portada`, Marzo 2008.

[7] Díaz J. La física nuclear experimental en el mundo: instalaciones, aceleradores y temas de investigación. `http://www.uv.es/diazj/fna_tema1.pdf`, Marzo 2008.

[8] Ferrer M. [et al.]. Equipo para radioterapia intraoperatoria basado en un microtrón de pista de 12 MeV. `www.juntadeandalucia.es/servicioandaluzdesalud/hsc/hospital/FMGR07/01014.pdf`, Marzo 2008.

[9] Rossbach J. and Schmüser P. Basic course on accelerator optics. Technical Report 94-01, CERN Accelerator School, 1994.

[10] Courant E.D. and Snyder H.S. Theory of the alternating-gradient synchrotron. *Annals of Physics*, (3):1–48, 1958.

[11] Theuws W.H.C. *Design of the Beam-Control System of the Eindhoven Racetrack Microtron.* PhD thesis, Technische Universiteit Eindhoven, 2000.

[12] Shvedunov V.I. [et al.]. A 70 MeV racetrack microtron. *Nucl. Instr. Meth. A*, 550:39–53, 2005.

[13] Gevorkyan V.G. [et al.]. Computer codes for simulation of beam dynamics in recirculating accelerators: RTMTRACE code. VINITI deposit number 183-B89, 1989.

[14] Spiegel M.R., Liu J., and Abellanas L. *Fórmulas y Tablas de Matemática Aplicada.* Mc Graw Hill, 2000.

[15] Salvat F., Fernández-Varea J.M., and Sempau J. *PENELOPE-2006, A Code System for Monte Carlo Simulation of Electron and Photon Transport.* OECD-NEA 2006, Issy-les-Moulineaux, France.

[16] Sempau J. and Badal A. PenEasy, a modular main program and voxelised geometry package for PENELOPE. `http://www.upc.es/inte/downloads/penEasy.htm`, Febrero 2008.

[17] Tabulation of error function values. `http://www.atilim.edu.tr/`, Marzo 2008.

[18] Fernández-Varea J.M. [et al.]. Curso de fundamentos de física médica (módulo 1), 2008.

[19] McCullough E.C. and Anderson J.A. The dosimetric properties of an applicator system for intraoperative electron-beam therapy utilizing a Clinac-18 accelerator. *Med. Phys.*, 9(2):261–68, 1982.

[20] Ronsivallea C. [et al.]. Technical features and experimental characterization of the IORT-1 system, a new IORT dedicated accelerator. *Nucl. Instr. Meth. A*, 526:1042–1045, 2006.

[21] Björk P. [et al.]. Design and dosimetry characteristics of a soft-docking system for intraoperative radiation therapy. *Int. J. Radiat. Oncol. Biol. Phys.*, 47(2):527–33, 2000.

[22] Nelson C.E., Cook R., and Rakfal S. The dosimetric properties of an intraoperative radiation therapy applicator system for a Mevatron-80. *Med. Phys.*, 16(5):794–99, 1989.

[23] Janssen R.W.J., Faddegon B.A., and Dries W.J.F. Prototyping a large field size IORT applicator for a mobile linear accelerator. *Phys. Med. Biol.*, 53:2089–102, 2008.

[24] Beddar A.S. [et al.]. Intraoperative radiation therapy using mobile electron linear accelerators: Report of AAPM radiation therapy committee task group no. 72. *Med. Phys.*, 33(5):1476–89, 2006.

[25] International Electrotechnical Commission. IEC 60601-2-1. Part 2-1: Particular requirements for safety of electron accelerators in the range 1 MeV to 50 MeV, 1998.

[26] International Commission on Radiation Units and Measurements. Special techniques. *Journal of the ICRU*, 4(1):69–73, 2004.

[27] Webelements Periodic Table. `http://www.webelements.com/gold/orbital_properties.html`, Marzo 2008.

[28] Lewis H.W. Multiple scattering in an infinite medium. *Phys. Rev.*, 78(526-29), 1950.

[29] Sempau J. and Andreo P. Configuration of the electron transport algorithm of PENELOPE to simulate ion chambers. *Phys. Med. Biol.*, 51:3533–48, 2006.

[30] National Institute of Standards and Technology. ESTAR, stopping-power and range tables for electrons. `http://physics.nist.gov/PhysRefData/Star/Text/ESTAR.html`, month = Marzo 2008.

[31] Fernández-Varea J.M. Apuntes de física médica. 2008.

[32] Duch M. [et al.]. Dose evaluation in lung-equivalent media in high-energy photon external radiotherapy. *Radiat. Prot. Dosim.*, 120:43–7, 2006.

[33] Ma C-M. and Jiang S.B. Monte Carlo modelling of electron beams from medical accelerators. *Phys. Med. Biol.*, 44:R157–89, 1999.

[34] Verhaegen F. and Seuntjens J. Monte Carlo modelling of external radiotherapy photon beams. *Phys. Med. Biol.*, 48:R107–64, 2003.

[35] Capote R. [et al.]. Phase-space database for external beam radiotherapy. Technical Report INDC NDS-0484, IAEA Nuclear Data Section, 2006.

[36] IAEA Phase-space database for external beam radiotherapy. `http://www-nds.iaea.org/phsp/phsp.htmlx`, Febrero 2008.

[37] Mazurier J. [et al.]. Simulation of photon beams from a Satume 43 accelerator using the code PENELOPE. *Physica Medica*, XV:101–10, 1999.

[38] Sempau J. [et al.]. Monte Carlo simulation of electron beams from an accelerator head using PENELOPE. *Phys. Med. Biol.*, 46:1163–86, 2001.

[39] Panettieri V. [et al.]. Monte Carlo simulation of MOSFET detectors for high-energy photon beams using the PENELOPE code. *Phys Med Biol*, 52:303–16, 2007.

[40] Nuclear Energy Agency. `http://www.nea.fr/`, Febrero 2008.

[41] Institut de Tècniques Energètiques. `http://www.upc.edu/inte/downloads/`, Febrero 2008.

[42] GNU Compiler Collection. `http://gcc.gnu.org/`, Abril 2008.

[43] The G95 Project. `http://www.g95.org/`, Mayo 2008.

[44] Nelson W.R., Hirayama H., and Rogers D.W.O. The EGS4 code system. Technical report, Stanford Linear Accelerator Center, 1985.

[45] Kawrakow I. Accurate condensed history Monte Carlo simulation of electron transport. I. EGSnrc, the new EGS4 version. *Med. Phys.*, 27(485-98), 2000.

[46] Sempau J. [et al.]. An algorithm for Monte Carlo simulation of coupled electron-photon transport. *Nucl. Instrum. Meth. B*, 132:377–90, 1997.

[47] Andreo P. Monte Carlo techniques in medical radiation physics. *Phys. Med. Biol.*, 36(861-920), 1991.

[48] Rogers D.W.O. Monte carlo techniques in radiotherapy. *Physics in Canada, Medical Physics Special Issue*, 58(2):63–70, 2002.

[49] James F. A review of pseudorandom number generators. *Comput. Phys. Commun.*, 60:329–44, 1990.

UNIVERSITAT POLITÈCNICA DE CATALUNYA

UNIVERSITAT DE BARCELONA

Projecte Final d'Estudis
MÀSTER EN ENGINYERIA BIOMÈDICA

Projecte Final d'Estudis presentat el dia de de 200 amb la següent Comissió Avaluadora:

Dr. President

Dr. Vocal 1

Dr. Vocal 2

Dr. Secretari

Amb la qualificació de: