UNIVERSIDAD DE ALCALÁ

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

Contribución al Desarrollo de una Interfaz Gráfica de Usuario para el Diseño, Optimización y Análisis de Antenas mediante el Método de los Momentos

D. Abdelhamid Tayebi Tayebi

TESIS DOCTORAL

Alcalá de Henares 2011

UNIVERSIDAD DE ALCALÁ

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

Tesis Doctoral

CONTRIBUCIÓN AL DESARROLLO DE UNA INTERFAZ GRÁFICA DE USUARIO PARA EL DISEÑO, OPTIMIZACIÓN Y ANÁLISIS DE ANTENAS MEDIANTE EL MÉTODO DE LOS MOMENTOS

Autor:

D. Abdelhamid Tayebi Tayebi

Directores:

Dr. D. Manuel Felipe Cátedra Pérez

Dr. D. Iván González Diego

2011

Universidad de Alcalá

DEPARTAMENTO DE
CIENCIAS DE LA COMPUTACIÓN
Campus Universitario. Edificio Politécnico
28805 Alcalá de Henares (Madrid)
Teléfonos: 91 885 66 51
Fax: 91 885 66 46

Dr. D. José Javier Martínez Herráiz, Profesor Titular de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial, en calidad de Director del Departamento de Ciencias de la Computación de la Universidad de Alcalá,

HACE CONSTAR:

Que la tesis titulada **"Contribución al Desarrollo de una Interfaz Gráfica de Usuario para el Diseño, Optimización y Análisis de Antenas mediante el Método de los Momentos"**, presentada por D. Abdelhamid Tayebi Tayebi para optar al título de Doctor Ingeniero de Telecomunicación y dirigida por Dr. D. Manuel Felipe Cátedra Pérez y co-dirigida por Dr. D. Iván González Diego, reúne los requisitos para su presentación y defensa pública.

Y para que así conste, firma la presente en Alcalá de Henares, a 23 de Mayo de 2011.

El Director del Departamento de Ciencias de la Computación

Dr. D. José Javier Martínez Herráiz

Universidad de Alcalá

DEPARTAMENTO DE
CIENCIAS DE LA COMPUTACIÓN
Campus Universitario. Edificio Politécnico
28805 Alcalá de Henares (Madrid)
Teléfonos: 91 885 66 51
Fax: 91 885 66 46

Dr. D. Manuel Felipe Cátedra Pérez, Catedrático de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial del Departamento de Ciencias de la Computación de la Universidad de Alcalá.

Dr. D. Iván González Diego, Titular de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial del Departamento de Ciencias de la Computación de la Universidad de Alcalá.

HACEN CONSTAR:

Que, una vez concluido el trabajo de tesis doctoral titulado: "**Contribución al Desarrollo de una Interfaz Gráfica de Usuario para el Diseño, Optimización y Análisis de Antenas mediante el Método de los Momentos**" realizado por D. Abdelhamid Tayebi Tayebi, dicho trabajo tiene suficientes méritos teóricos, que se han contrastado adecuadamente mediante validaciones experimentales y que son altamente novedosos. Por todo ello consideran que procede su defensa pública.

Y para que así conste, firman la presente en Alcalá de Henares, a 23 de mayo de 2011.

El Director de la Tesis

Dr. Manuel Felipe Cátedra Pérez

El Codirector de la Tesis

Dr. Iván González Diego

A mi familia.

Agradecimientos.

En esta página quisiera recordar y expresar mi agradecimiento a todas aquellas personas que, de alguna forma u otra, han hecho posible la realización de este trabajo.

En primer lugar me gustaría expresar mi gratitud a mis directores de tesis Dr. Manuel Felipe Cátedra y Dr. Iván González por su guía y apoyo a lo largo de este trabajo. Así mismo, quisiera extender dicho reconocimiento a todo el Departamento de Ciencias de la Computación de la Universidad de Alcalá.

También quiero dar las gracias a todos los compañeros del grupo de investigación al que pertenezco, tanto los que están ahora como los que han formado parte de él en algún momento durante la realización de esta tesis.

Agradecer también al Dr. Weng Cho Chew su buena acogida durante la estancia en el *Department of Electrical and Electronic Engineering* de la universidad de Hong Kong y a todos sus investigadores, con los que he colaborado.

Tampoco quiero olvidarme de aquellos que me apoyaron y me aconsejaron en momentos claves de mi vida y en especial a mis profesores del IES San Juan Bosco de Lorca: Damián, Juan Francisco, Lola, etc.

A todos,

GRACIAS.

Resumen.

En esta tesis se presenta el desarrollo de una herramienta gráfica avanzada de resolución de problemas de radiación y dispersión. El núcleo electromagnético que incluye la aplicación se basa en el Método de los Momentos y es capaz de analizar estructuras con formas arbitrarias eléctricamente muy grandes. El sistema implementado facilita el proceso de diseño de la estructura bajo análisis, pues incorpora un potente módulo de geometría. Además de incluir las opciones básicas presentes en cualquier programa de diseño gráfico, tales como la creación de primitivas y la posibilidad de realizar transformaciones geométricas, la herramienta también dispone de un sofisticado módulo mediante el cual se pueden crear y optimizar los principales tipos de antenas. El módulo de optimización desarrollado permite calcular las dimensiones óptimas de la estructura bajo análisis, dependiendo de una función de coste que define su comportamiento electromagnético ideal. Todas las entidades geométricas generadas por la herramienta son parametrizables, lo que permite la identificación de los parámetros a optimizar, así como el rango de variación de dichos parámetros. Además, la función de coste global puede incluir especificaciones en distintas zonas del diagrama de radiación de la estructura.

El lenguaje de programación empleado para implementar la herramienta ha sido Java y la parte de la visualización, tanto de las geometrías diseñadas como de los resultados obtenidos, se ha implementado utilizando las librerías de Java 3D. Dichas librerías proporcionan muchos beneficios a la herramienta, ya que incluyen una gran variedad de funcionalidades y opciones de visualización tales como iluminaciones, colores, texturas, animaciones, etc.

La configuración de los parámetros de simulación, así como el proceso de simulación numérica y la posterior visualización de los resultados obtenidos se han integrado en el sistema, de modo que el proceso de análisis sea sencillo e intuitivo. Por otro lado, la herramienta desarrollada incluye un gestor de simulaciones que ofrece la

posibilidad de llevar a cabo los análisis de forma local o de forma remota. También es posible ejecutar las simulaciones en paralelo, utilizando más de un procesador de forma simultánea. Otro aspecto destacable del módulo de ejecución es la opción de realizar los análisis en segundo plano, de modo que sea posible cerrar la sesión de trabajo sin perder los datos de la simulación.

Finalmente, se han llevado a cabo una serie de simulaciones utilizando la herramienta, para validar el trabajo realizado. Los resultados obtenidos han sido satisfactorios, y además el funcionamiento del sistema ha resultado ser robusto, estable y eficiente.

Abstract.

This thesis presents the development of an advanced graphical tool for solving radiation and scattering problems. The electromagnetic kernel included in the application is based on the Method of Moments and is capable of analyzing electrically large structures with arbitrary shapes. The implemented system facilitates the process of designing the structure under analysis, since it incorporates a powerful geometry module. Besides including the basic options present in any graphics program, such as the creation of primitives and the possibility of performing geometric transformations, the tool also has a sophisticated module that can create and optimize the main types of antennas. The optimization module allows for the calculation of the optimal dimensions of the structure under analysis, depending on a cost function that defines the ideal electromagnetic behavior. All geometric entities generated by the tool are parametrizable, allowing the identification of parameters to be optimized, as well as the variation range of these parameters. In addition, the overall cost function may include specifications in different areas of the radiation pattern of the structure.

The programming language used to implement the tool is Java and the visualization of both the geometric design and the obtained results, has been implemented using the Java 3D libraries. These libraries provide many benefits for the tool as they include a variety of features and visualization options such as lighting, colors, textures, animations, etc.

The configuration of the simulation parameters, the numerical simulation process and the subsequent visualization of the results, have been integrated into the system, so that the analysis process becomes simple and intuitive. On the other hand, the developed tool includes a simulation manager which offers the possibility of conducting analysis locally or remotely. Parallel simulations using multiple processors simultaneously can be also performed. Another notable feature of the execution module

is the option of performing analysis in the background, so that it is possible to close the work session without losing data in the simulation.

Finally, a series of simulations using the tool to validate their work have been carried out. The results have been satisfactory, and also the functioning of the system has proven to be robust, stable and efficient.

INDICE

1.- Introducción.

1.1.- MOTIVACIÓN DE LA TESIS.

En la actualidad, la gran demanda de herramientas de simulación numérica, es un hecho destacable no solo en centros de investigación y en universidades, sino también en muchos sectores industriales. Los importantes avances tecnológicos, tanto a nivel de hardware como a nivel de software, acometidos en las últimas décadas hacen que este tipo de herramientas sean de gran utilidad, pues facilitan la creación de los diseños y aceleran el proceso de análisis.

En primer lugar, los avances conseguidos a nivel de hardware permiten a cualquier usuario disponer de ordenadores personales muy potentes, con excelentes sistemas de memoria para el almacenamiento de datos y veloces procesadores capaces de realizar cálculos en paralelo muy rápidamente. Este aspecto es fundamental hoy en día, ya que las elevadas exigencias de rendimiento tanto en empresas como en centros de investigación obligan a que los análisis se lleven a cabo de forma paralela en múltiples procesadores. De esta forma el tiempo de simulación se reduce considerablemente y la obtención de resultados es mucho más rápida.

Por otro lado, a nivel teórico también se ha avanzado mucho. En los últimos años se ha contribuido de forma decisiva al desarrollo de métodos muy sofisticados para resolver problemas de radiación y dispersión considerando estructuras muy grandes eléctricamente. Por ejemplo, el método rápido de los multipolos multinivel ha permitido el análisis de estructuras enormes en términos de longitud de onda, como aeronaves en la banda X [1, 2]. Sin duda, llevar a cabo estos análisis hace unos años era algo impensable. El desarrollo de nuevos métodos iterativos en electromagnetismo computacional y las importantes contribuciones para la mejora del método de los momentos [3, 4] son una muestra clara de los avances alcanzados a día de hoy.

La complejidad interna de las herramientas de simulación numérica implica una etapa de configuración de parámetros previa a la ejecución. Para facilitar la interacción entre los usuarios y este tipo de herramientas, éstas incluyen interfaces gráficas visualmente muy atractivas que simplifican tanto el pre-proceso requerido antes de realizar la simulación, como el post-proceso necesario para visualizar los resultados obtenidos.

Los programas comerciales que permiten realizar simulaciones numéricas han sufrido un crecimiento notable en los últimos años. Algunos como FEKO, CST o HFSS proporcionan entornos gráficos muy completos que permiten llevar a cabo diversas simulaciones aplicando distintos métodos de resolución dependiendo del caso bajo estudio. Su gran éxito se debe, en gran medida, a que sus aplicaciones superan cualquier expectativa prevista. Además de utilizarse extensamente en el campo del electromagnetismo computacional, también se empiezan a usar en otras ramas de la ingeniería como en mecánica, automoción o ingeniería civil, naval, aeroespacial, etc., e incluso están ampliamente extendidas en el ámbito de la medicina [5]. Otra disciplina en auge en los últimos años que está empezando a utilizar programas de simulación numérica es la de energías renovables, debido a las posibles interferencias que pueden causar los enormes aerogeneradores en los sistemas radioeléctricos presentes en las cercanías de las centrales eólicas.

Como se puede ver, las ventajas de este tipo de herramientas son muy numerosas. Cuando se analiza un fenómeno científico desde un punto de vista experimental, es bastante común que resulte muy caro el disponer de un completo sistema de medida que permita analizar dicho fenómeno. En las etapas de diseño, las simulaciones por ordenador evitan muchos gastos debidos a la realización de medidas en cámaras anecoicas, normalmente sobre modelos escalados. El método más eficiente a seguir es realizar análisis mediante herramientas de software, y dejar las medidas como validación final del estudio realizado.

1.2.- OBJETIVOS DE LA TESIS.

El primero de los objetivos establecidos es realizar un estudio sobre las distintas alternativas disponibles para desarrollar una interfaz gráfica de usuario (GUI, *Graphical User Interface*) [6] avanzada para la herramienta de simulación numérica NewFasant. Para ello, se deben considerar los siguientes aspectos: elección del lenguaje de programación más idóneo, determinar cuál es el entorno de desarrollo más apropiado y decidir si es adecuado el uso de librerías comerciales para llevar a cabo la implementación del sistema.

El objetivo central de la tesis es implementar un sistema gráfico avanzado capaz de gestionar los procesos previo y posterior a la simulación, integrando el núcleo electromagnético en un sistema visualmente atractivo de modo que se favorezca la interactividad con los usuarios. El desarrollo del sistema debe tener en cuenta las especificaciones enumeradas a continuación:

- Visualmente, el sistema debe proporcionar un entorno amigable que facilite la interacción con los usuarios.

- A pesar de la complejidad de los cálculos llevados a cabo para proporcionar los resultados, la interfaz debe ser fácil de usar, incluso para usuarios inexpertos en el campo del electromagnetismo.

- El sistema debe ser estable, considerando todas las posibles acciones del usuario para evitar fallos en el sistema.

- Se debe mantener una estandarización en el aspecto visual de los elementos que constituyen la interfaz (menús, cuadros de diálogo, ventanas de información, etc.).

- Se deben implementar las funcionalidades adecuadas para poder visualizar el modelo geométrico sin problemas de ralentización realizando zooms y

transformaciones geométricas ágilmente sobre las entidades mediante el uso del ratón.

- Debe incluir un módulo avanzado de geometría capaz de importar varios tipos de formatos CAD (*Computer Aided Design*) como *.dxf, .iges* y *.sat.* Dicho módulo también debe disponer de una amplia variedad de primitivas (curvas, superficies y sólidos) y de operaciones de transformación como rotación, translación, simetría, copia, etc.

- Se debe incluir también un módulo de optimización que permita establecer unos requerimientos mediante una función de coste y unos parámetros geométricos que varíen hasta conseguir los objetivos establecidos, siempre que sea posible. La función de coste proporcionará distintas facilidades de configuración, de modo que sea posible minimizar lóbulos secundarios del diagrama de radiación, maximizar la ganancia, mejorar la pureza de polarización, etc.

- La herramienta dispondrá de varias opciones de visualización de los resultados obtenidos, en función del tipo de simulación ejecutada.

- Incluirá un gestor de proyectos que ofrecerá la posibilidad de guardar los resultados obtenidos en una simulación, así como la opción de abrir un proyecto existente.

El sistema a desarrollar proporcionará dos modalidades distintas de ejecución: local y remota. Con el modo local, el análisis y la optimización se llevarán a cabo en la máquina local donde está instalado el software. Las simulaciones ejecutadas de forma remota seguirán el modelo cliente-servidor y permitirán analizar y optimizar estructuras complejas en un *cluster* de ordenadores utilizando el paradigma MPI. Tanto las simulaciones locales como las remotas se podrán llevar a cabo en modo *background*, es decir, en segundo plano. De este modo, el sistema mantendrá un proceso activo que controle la simulación ejecutada en *background*, al mismo tiempo que se permita seguir trabajando en otros diseños. También se permitirá cerrar una sesión de trabajo con una simulación en curso, de forma que al iniciar la herramienta se informe del estado de

dicha simulación y se ofrezca la opción de visualizar los resultados en caso de que haya finalizado.

Otro objetivo prioritario será desarrollar un módulo específico de diseño de antenas en el que se ofrezcan modelos parametrizados que permitan la creación directa de los principales tipos de antenas. Para verificar el correcto funcionamiento de la herramienta, también se llevará a cabo un intensivo proceso de diseño, análisis y optimización para estudiar el comportamiento electromagnético de distintos tipos de antenas, como sondas, reflectores, radomos, bocinas, lentes, circuitos microstrip, antenas de hilo, etc.

Por último, y con vistas a una futura continuación del trabajo presentado en la memoria, la implementación de la herramienta se hará de tal forma que resulte fácil de mantener y esté bien documentada, para que posteriormente pueda ser modificada o extendida sin dificultad.

1.3.- ANTECEDENTES HISTÓRICOS.

El grupo de investigación en el que se ha realizado la presente tesis lleva trabajando desde los años 90 en el desarrollo de métodos de análisis electromagnético. Su experiencia se centra en la aplicación de métodos numéricos para la resolución de diversos problemas de propagación, radiación y dispersión [7-13]. Esta tesis presenta la contribución al desarrollo de una herramienta gráfica que de soporte a dichos métodos de simulación numérica abarcando las etapas previa y posterior a la ejecución de uno de los núcleos desarrollados basado en el Método de los Momentos, el cual forma parte de la herramienta NewFasant.

El Método de los Momentos es una de las técnicas más conocidas y utilizadas dentro del electromagnetismo computacional, sobre todo en ingeniería de antenas. Sus orígenes se remontan a los años 60, considerándose el trabajo presentado en [14] uno de los pioneros en su desarrollo. Su popularidad se incrementó tras darse a conocer los trabajos de Richmond y Harrington [15, 16] a mediados de la década de los 60. La información detallada sobre los orígenes y el desarrollo del Método de los Momentos se

puede encontrar en [17]. Su aplicación sobre diversos problemas de radiación y dispersión resultó satisfactoria, prediciendo el comportamiento electromagnético de varias estructuras como antenas de hilo, arrays de antenas, análisis de circuitos microstrip, etc.

Existe un gran número de herramientas software gratuitas y comerciales disponibles en la actualidad basadas en el Método de los Momentos. Por ejemplo, el "Numerical Electromagnetic Code (NEC)" que se conoce popularmente como NEC2 es una de las herramientas libres más usadas. El estado del arte de software basado en el Método de los Momentos también incluye programas como FEKO, CST, HFFS, Wipl-d, IE3D, AXIEM, Momentum, Puma-EM, SONNET, superNEC, GEMACS, etc. Por otro lado, a nivel académico, son muchos los simuladores basados en técnicas de análisis electromagnético como el Método de los Momentos, desarrollados en universidades y en centros de investigación.

Respecto al diseño gráfico de antenas, existen varios módulos comerciales que permiten generar una gran variedad de modelos de antenas de forma inmediata. Uno de los más conocidos y completos es *Antenna Magus* [18], incluido tanto en FEKO como en CST. Este paquete software dispone de una amplia base de datos en la que se puede encontrar una gran variedad de antenas. Otros sistemas especializados en el diseño de antenas son SatSoft [19], que proporciona herramientas para el diseño de antenas empleadas en las comunicaciones por satélite, y PCAAD [20], EZNEC [21] o RF TooBox [22], que permiten diseñar y modelar los tipos de antenas más comúnmente utilizados.

1.4.- SISTEMAS GRÁFICOS POR ORDENADOR.

Para facilitar el desarrollo de cualquier sistema gráfico avanzado, existen ciertos paquetes de programación, que proporcionan bibliotecas de funciones gráficas que se pueden utilizar normalmente en programas de Java, C, C++ o Fortran. Entre las funciones básicas proporcionadas por una biblioteca gráfica típica se incluyen aquellas que sirven para especificar componentes de la imagen (líneas, rectángulos, esferas, etc.), establecer el color de dichos componentes, seleccionar las vistas de la escena o aplicar

transformaciones geométricas. Algunos ejemplos de paquetes de programación gráfica disponibles como software libre son GL (*Graphics Library*), OpenGL [23], Direct3D [24], VRML (*Virtual Reality Modelling Language*) [25], Java 2D y Java 3D [26].

El paquete *Open Inventor* proporciona un conjunto de funciones orientadas a objetos para describir una escena visualizada mediante llamadas a OpenGL. Tanto OpenGL, propiedad de *Silicon Graphics*, como Direct3D, propiedad de *Microsoft*, constituyen una API multiplataforma bastante completa para la programación de gráficos 3D. También existe una especificación de OpenGL para Linux llamada Mesa 3D [27] muy útil para el renderizado interactivo de gráficos tridimensionales. Por otro lado, el lenguaje VRML, que se creó como un subconjunto de *Open Inventor*, permite la creación de modelos tridimensionales virtuales en Internet. También es posible visualizar imágenes en Internet utilizando bibliotecas gráficas desarrollas a través del lenguaje Java. Por ejemplo, con Java 2D se pueden crear escenas bidimensionales, mientras que Java 3D permite generar imágenes tridimensionales complejas dentro de applets. Con Renderman Interface [28] de *Pixar Corporation*, se pueden generar escenas empleando una gran variedad de modelos de iluminación. Finalmente, las bibliotecas gráficas se proporcionan a menudo en otros tipos de sistemas, tales como Mathematica y MatLab.

Por otra parte, también existen paquetes de software especializados en modelado de geometrías tridimensionales. En los últimos años, el número de empresas en este sector se ha visto incrementado notablemente. Las ventajas que ofrecen son obvias: calidad de los diseños y de la visualización, disponibilidad de un gran abanico de primitivas geométricas, amplia variedad de operaciones CAD, etc. Sin embargo, también existen algunos inconvenientes que se deben tener en cuenta como por ejemplo: gran desembolso económico, incompatibilidad de formato a la hora de integrar el módulo geométrico con el núcleo electromagnético, formación de los desarrolladores para usar las librerías adquiridas, etc.

Dos de los núcleos de modelado gráfico más utilizados en herramientas de análisis electromagnético son Acis [29], desarrollado por Spatial y Parasolid [30], propiedad de Siemens PLM Software. Ambos proporcionan un conjunto de librerías en C++ que permiten implementar prácticamente cualquier opción de diseño CAD.

También existen plataformas de desarrollo de aplicaciones gráficas, como HOOPS [31], que facilita la implementación de toda la parte de visualización de la interfaz. HOOPS ofrece buen soporte y buena documentación, pero no incluye librerías dedicadas al núcleo geométrico. Es decir, proporciona muy buenas prestaciones de visualización, post-procesado, creación de menús, etc., pero hay que adquirir el núcleo geométrico por otros medios, normalmente a través de empresas especializadas. Para solucionar este inconveniente, HOOPS ha establecido convenios con las empresas Siemens (Parasolid) y Spatial (ACIS) para ofrecer una suite de librerías más completa.

1.5.- TRABAJOS RELACIONADOS.

Este apartado presenta algunas aplicaciones prácticas en las que se ha incluido el diseño de una GUI para facilitar el manejo de los sistemas para algunas aplicaciones particulares.

Hay algunos ejemplos de GUIs cuyo objetivo es social, es decir, se diseñan con el fin de obtener beneficios para la ciudadanía. Este es el caso del sistema descrito en [32], donde se presenta una GUI multiplataforma destinada al ámbito sanitario, con el fin de intermediar entre los dispositivos que almacenan el historial clínico de cada usuario y un sistema que sea capaz de mostrar la información almacenada al personal sanitario en caso de emergencia. Otro ejemplo destinado a mejorar la calidad de vida de usuarios invidentes es el presentado en [33]. Dicho trabajo describe un sistema que permite a usuarios ciegos utilizar una GUI de la misma forma que lo haría un usuario que pudiera ver. El usuario controla un puntero mediante un ratón y, cuando el puntero pasa por encima de ventanas, botones, y otros elementos gráficos, se generan una serie de sonidos. Los distintos tonos permiten al usuario crear un mapa mental del entorno gráfico del mismo modo que usando un sistema táctil.

Dentro del ámbito tecnológico también se pueden encontrar muchísimos proyectos implementados. Por ejemplo, en [34] los autores describen una GUI desarrollada en Matlab para proporcionar un entorno amigable para el cálculo y visualización de los resultados de campo dispersado de objetos canónicos 2D o 3D.

Cada objeto se programa en módulos separados, y el paquete se puede ampliar fácilmente para incluir nuevos objetos. Otro sistema muy parecido es el que se propone en [35], donde se presenta una interfaz gráfica interactiva diseñada para intermediar entre los usuarios y un simulador que realiza análisis mediante el método de los elementos finitos.

Por otro lado, en [36] se describen las recomendaciones para el proceso de diseño y el método para desarrollar una GUI amigable. Dicho proceso es muy metódico y está basado en conceptos teóricos. El objetivo es facilitar el uso y la comprensión de los equipos desarrollando una herramienta sencilla, intuitiva y fácil de usar. Las pautas de diseño propuestas en [36] pueden ser muy útiles también para proyectos que requieran el control remoto de algunos componentes del sistema. Este es el caso que se presenta en [37], donde se implementa una GUI eficiente para desarrollar sistemas que controlen robots en la universidad de Florida. El software se ha diseñado para poder manejar a los robots de forma remota, pero también se puede usar para controlar plataformas móviles o cualquier otro sistema remoto.

Por otro lado, hoy en día las GUIs están implantadas en cualquier tipo de dispositivo y prácticamente todos los ordenadores personales del mundo tienen algún tipo de interfaz instalada. Sin lugar a dudas, las más usadas son las de Windows, seguidas de KDE, Aqua y Gnome.

Parece ser que el futuro y la evolución de las interfaces gráficas pasa por la adopción de las tres dimensiones y de nuevos paradigmas, alejados ya de los tradicionales WYSIWYG *(What You See Is What You Get)* y WIMP *(Windows Icons Menus Pointers)*, que sin duda son buenas soluciones, pero no las mejores, y se podrían superar ampliamente. Sería muy interesante que la nueva generación de GUIs permitiese manipular la información de forma no ligada a la realidad cotidiana, sino adaptada a lo que es, algo mucho más abstracto. Así se podría pasar a un nuevo nivel de uso de las computadoras, haciendo previsiblemente más sencilla la manipulación de grandes cantidades de información y aumentando la productividad.

A continuación se describen brevemente las tres versiones más recientes de interfaces gráficas que se han desarrollado para los sistemas operativos con mayor éxito comercial.

Microsoft AERO (*Authentic, Energetic, Reflective and Open*) [38] es el nombre de la interfaz gráfica que incluyen los sistemas operativos Windows Vista y Windows 7. Está pensada para ser más clara, eficiente y potente que las anteriores, a la vez que más agradable a la vista. Ejecutada en su totalidad, Microsoft AERO incluye un aspecto totalmente renovado en lo que se refiere a la interfaz gráfica de las ventanas, mensajes de error, y algunos otros aspectos del sistema, proporcionándoles un efecto de cristalizado transparente con la opción de cambiar el color completamente transparente por algún color de la colección de colores para Aero.

Enlightenment [39] es el gestor de ventanas de Unix y GNU/Linux y una de las alternativas de código abierto a las GUIs comerciales de Microsoft o Macintosh. Es configurable y muy atractivo visualmente y se puede usar tanto en máquinas de última generación como en máquinas más antiguas sin perder funcionalidades.

Según sus creadores, se trata de una interfaz gráfica funcional a la vez que bonita. Enlightenment no sólo permite personalizar todos los elementos gráficos de la interfaz (su apariencia), como la mayoría del resto de GUIs, sino que también permite configurar elementos básicos como los menús, la barra de inicio e incluso el menú emergente obtenido al pulsar el segundo botón del ratón, obteniéndose así una funcionalidad y eficiencia únicas.

Por último, Plasma [40] se sitúa en tercer lugar de popularidad. Se trata de la nueva interfaz gráfica de KDE que funciona bajo la mayoría de sistemas operativos basados en Unix como Linux, BSD, etc. Plasma es un proyecto aún en fase de desarrollo, altamente configurable, en el que se han rediseñado las aplicaciones y el entorno desde el nivel más básico. Esta nueva GUI aporta un nuevo 'look' a la mayoría de elementos tradicionales, además de un sistema de extensiones para escritorios.

Una de sus características es que permite escribir pequeñas aplicaciones o *widgets* llamadas plasmoides, los cuales se pueden situar en el escritorio o en sus

paneles. Plasma separa los componentes en la parte visual e interactiva, de la parte de obtención de datos. Los motores de datos o data engines, son los encargados de obtener información de fuentes determinadas y ofrecerla a los plasmoides. De este modo se trata de no repetir trabajo, ya que cada motor de datos puede ser usado por uno o más plasmoides.

1.6.- CONTENIDO DE LA TESIS.

Esta tesis se divide en ocho capítulos, los cuales se resumen a continuación:

En primer lugar, en el capítulo 1 se expone la motivación de esta tesis, es decir, por qué es necesaria su realización, los objetivos propuestos inicialmente, los antecedentes históricos y un breve resumen de los trabajos relacionados.

El capítulo 2 intenta mostrar un enfoque genérico sobre la Ingeniería del Software aplicada al desarrollo de interfaces gráficas. En este capítulo se tratan los modelos de desarrollo de software, y algunas nociones básicas sobre programación orientada a objetos, lenguaje de programación, librerías utilizadas, diagramas UML, etc. Además, incluye los aspectos fundamentales del diseño de sistemas centrados en el usuario.

El capítulo 3 está dedicado a la creación y visualización de modelos geométricos. Ambos procesos se llevan a cabo gracias a librerías de Java 3D, las cuales han sido básicas para la implementación de la herramienta. Su uso ha sido imprescindible tanto para la creación y manipulación de geometrías como para la visualización del escenario y de los resultados.

En el capítulo 4 se describen las características principales del módulo dedicado al diseño de antenas. Se presentan los modelos geométricos implementados de las antenas más comunes, junto con un breve resumen de los fundamentos teóricos de cada tipo de antena. La novedad de este módulo estriba en que todas las dimensiones que definen las antenas están parametrizadas, lo que facilita el proceso de creación, edición y optimización. Este capítulo también aborda el tema de la alimentación de las antenas, dependiendo del tipo de simulación establecida.

El capítulo 5 describe el tratamiento que se realiza sobre la geometría cuando se lleva a cabo una simulación. Se presenta la descripción matemática de las superficies paramétricas que describen la geometría bajo estudio y se expone el mecanismo de interpolación necesario para adaptar la malla de puntos proporcionada por el fichero geométrico, a los parámetros que definen las superficies paramétricas. Por último se describe el proceso de análisis electromagnético mediante la aplicación del Método de los Momentos sobre la versión discretizada de la geometría.

El capítulo 6 presenta los procesos de ejecución y optimización, así como la configuración de los parámetros de simulación y la representación de los resultados obtenidos. Además se describen los dos modos de operación disponibles, modo local o modo remoto, detallando el funcionamiento y la implementación de cada uno de ellos.

Por último, el capítulo 7 se ha dedicado a la presentación de los resultados que muestran la validez de la herramienta desarrollada. La memoria finaliza con las conclusiones y las futuras líneas de trabajo expuestas en el capítulo 8.

1.7.- REFERENCIAS.

[1].- Song JM, Lu CC, Chew WC, "Multilevel Fast Multipole Algorithm for Electromagnetic Scattering by Large Complex Objects", *IEEE Transactions on Antennas and Propagation* 45 (10): 1488-1493 Oct 1997.

[2].- Song JM, Chew WC, "Multilevel Fast-Multipole Algorithm for Solving Combined Field Integral-Equations of Electromagnetic Scattering", *Microwave and Optical Technology Letters* 10 (1): 14-19 Sep 1995.

[3].- Delgado, C.; Garcia, E.; Cátedra, F.; Mittra, R.; , "Generation of Characteristic Basis Functions Defined Over Large Surfaces by Using a Multilevel Approach," *Antennas and Propagation, IEEE Transactions on* , vol.57, no.4, pp.1299-1301, April 2009.

[4].- Delgado, C.; Catedra, F.; Mittra, R.; , "A numerically efficient technique for orthogonalizing the basis functions arising in the solution of electromagnetic scattering problems using the CBFM," *Antennas and Propagation Society International Symposium, 2007 IEEE* , vol., no., pp.3608-3611, 9-15 June 2007.

[5].- Ghista, D.N.; , "Biomedical engineering: yesterday, today, and tomorrow," *Engineering in Medicine and Biology Magazine, IEEE* , vol.19, no.6, pp.23-28, Nov.-Dec. 2000.

[6].- Object-Oriented Interface Design. IBM Common User Access Guidelines December 1992.

[7].- M. Domingo, F. Rivas, J. Pérez, R. P. Torres, M.F. Cátedra, "Computation of the RCS of complex bodies modeled using NURBS surfaces*", IEEE Antennas and Propagation Magazine*, vol. 37, no. 6, pp. 36-47, December 1995.

[8].- M.F. Cátedra, J. Pérez, F. Sáez de Adana, O. Gutiérrez, "Efficient ray-tracing techniques for three-dimensional analyses of propagation in mobile communications: Application to picocell and microcell scenarios", *IEEE Antennas and Propagation Magazine,* vol. 40, no. 2, pp. 15-28, April 1998.

[9].- J. Pérez, F. Sáez de Adana, O. Gutiérrez, I. González, M.F. Cátedra, I. Montiel, J. Guzmán, "Fasant: Fast computer tool for the analysis of on-board antennas", *IEEE Antennas and Propagation Magazine*, vol. 49, no.2, pp. 94-98, April 1999.

[10].- F. Sáez de Adana, O. Gutiérrez, I. González, M.F. Cátedra, "Faspro: Fast computer tool for the analysis of propagation in mobile communications", in *IEEE International Conference on Industrial Informatics*, August 2005, pp. 257-261.

[11].- L. Valle, F. Rivas, M.F. Cátedra, "Combining the momento method with geometrical modelling by NURBS surfaces and Bezier patches", *IEEE Transactions on Antennas and Propagation*, vol. 42, no. 3, pp. 373-381, March 1994.

[12].- M.F. Cátedra, E. Gago, L. Nuño, "A numerical scheme to obtain the RCS of three-dimensional bodies of resonant size using the conjugate gradient method and the fast Fourier transform", *IEEE Transactions on Antennas and Propagation*, vol. 37, no. 5, pp. 528-537, May 1989.

[13].- M.F. Cátedra, E. Gago, "Spectral domain analysis of conducting patches of arbitrary geometry in multilayer media using the cg-fft method*", IEEE Transactions on Antennas and Propagation*, vol. 38, no. 10, pp. 1530-1536, October 1990.

[14].- L.V. Kantorovich and V.I. Krylov. "Approximate Methods of Higher Analysis". John Wiley and Sons, New York, 1964.

[15].- J.H. Richmond. "Digital Computer Solutions of the Rigorous Equations for Scattering Problems" *Proceedings of the IEEE*, 53: 796-804, August 1965.

[16].- R.F. Harrington. "Matrix Methods for Field Problems". *Proc. IEEE*, 55(2): 136-149, February 1967.

[17].- R.F. Harrington. "Origin and Development of the Method of Moments for Field Computation*". IEEE Antenna and Propagation Magazine*, 32: 31-35, June 1990.

[18].- http://www.antennamagus.com/

[19].- http://www.satcom.co.uk/article.asp?article=27

[20].- http://in.scientech.bz/web/antennadesign.jsp

[21].- http://www.eznec.com/

[22].- http://www.blackcatsystems.com/software/electronics-antenna-design-software.html

[23].- D. Hearn, M.P.Baker, “Gráficos por computadora con OpenGL”, ed. Pearson Prentice Hall, 2006.

[24].- http://www.gamesforwindows.com/en-US/directx/

[25].- http://www.vrmlsite.com/

[26].- D. Selman, “Java 3D Programming”. Independent Publishers Group, 2002.

[27].- http://www.mesa3d.org/

[28].- https://renderman.pixar.com/products/rispec/index.htm

[29].- http://www.spatial.com

[30].- http://www.siemens.com/plm/parasolid

[31].- http://www.techsoft3d.com/

[32].- Rybynok, V.O.; Kyriacou, P.A.; Binnersley, J.; Woodcock, A.; , "MyCare Card Development: Portable GUI Framework for the Personal Electronic Health Record Device," *Information Technology in Biomedicine, IEEE Transactions on* , vol.15, no.1, pp.66-73, Jan. 2011.

[33].- McKiel, F., Jr.; , "Audio-enabled graphical user interface for the blind or visually impaired," Computing Applications to Assist Persons with Disabilities, 1992*., Proceedings of the Johns Hopkins National Search for* , vol., no., pp.185-187, 1-5 Feb 1992.

[34].- Sharkawy, M.A.; Demir, V.; Elsherbeni, A.; Mahfza, B.; , "A graphical user interface (GUI) for electromagnetic scattering from two- and three-dimensional canonical and non-canonical objects [EM Programmer's Notebook]," *Antennas and Propagation Magazine, IEEE* , vol.48, no.6, pp.135-141, Dec. 2006.

[35].- Coco, S.; Emma, F.; Laudani, A.; Pulvirenti, S.; Sergi, M.; , "A new interactive graphic user interface and mesh generator for 3D TWT collector analysis," *Vacuum Electronics Conference*, 2000. Abstracts. International, vol., no., pp.2 pp., 2000.

[36].- Lavergne, M.; , "Graphical user interface for next generation power systems," *Telecommunications Energy Conference*, 2000. *INTELEC. Twenty-second International,* vol., no., pp.109-112, 2000.

[37].- De Rossi, V.; Batsomboon, P.; Tosunoglu, S.; Repperger, D.W.; , "Interactive modular graphical user interface development for telesensation systems," Systems, Man, and Cybernetics, 1997. *'Computational Cybernetics and Simulation*'., 1997 *IEEE International Conference on* , vol.2, no., pp.1604-1608 vol.2, 12-15 Oct 1997.

[38].- http://windows.microsoft.com/es-ES/windows7/products/features/aero

[39].- http://www.enlightenment.org/

[40].- lhttp://www.kde.org/announcements/4.0/desktop.php

2.- Metodología y Desarrollo de Interfaces Gráficas.

2.1.- INTRODUCCIÓN.

En este capítulo se exponen las metodologías y técnicas utilizadas para llevar a cabo el diseño y la implementación de productos software, orientado al desarrollo de interfaces gráficas. En primer lugar se hace referencia a la Ingeniería del Software para enumerar los factores de calidad y los modelos de desarrollo de software existentes. A continuación se especifican las etapas del diseño de GUIs y se describen las características principales de Java y de las librerías de Java 3D, las cuales han permitido implementar la parte gráfica de la herramienta, especialmente la parte de visualización del modelo geométrico en el escenario. Por último, se presentan tres técnicas de modelado UML (*Unified Modelling Language*): diagramas de clases, diagramas de casos de uso y diagramas de secuencia.

2.2.- FACTORES DE CALIDAD DEL SOFTWARE.

Para desarrollar software de calidad, es conveniente seguir las directrices que aconseja la Ingeniería del Software [1-3]. Esta disciplina se encarga de proporcionar los métodos y técnicas necesarios para desarrollar y mantener software, intentando automatizar el proceso de creación con el objetivo de minimizar riesgos y costes. Se considera una ingeniería moderna, que trata con varias áreas de la informática y de las ciencias de la computación, y que aborda todas las fases del ciclo del desarrollo de cualquier producto software. Entre sus objetivos se encuentran los siguientes: mejorar la calidad y eficiencia del proceso de desarrollo del software, aumentar la productividad de los ingenieros y desarrollar software fácil de mantener y bien documentado en el plazo establecido y dentro del presupuesto estimado.

Para poder evaluar la calidad de un producto software una vez finalizado y para comprobar que cumple la normativa de calidad [4, 5], se deben tener en cuenta una serie de consideraciones. El software desarrollado debe ser:

- Correcto y funcional: si se ajusta a las especificaciones dadas por el cliente.
- Fácil de usar: si su aprendizaje y manejo son sencillos.
- Fiable: si es capaz de ofrecer los mismos resultados bajo las mismas condiciones.
- Portable: si es capaz de integrarse en entornos distintos con el mínimo esfuerzo.
- Adaptable: si puede soportar modificaciones en alguna parte del código sin que afecten a su funcionalidad.
- No erróneo: si no existen diferencias entre los valores reales y los calculados.
- Confiable: si puede estar disponible para su uso a lo largo del tiempo. Tiene que ver con la madurez y la facilidad de recuperación.
- Eficiente: si lleva a cabo una utilización óptima de los recursos de la máquina.
- Robusto: si no tiene un comportamiento catastrófico ante situaciones excepcionales. En otras palabras, si es tolerante a fallos.
- Inteligible: si tiene un diseño claro, bien estructurado y documentado.
- Reutilizable: si puede ser usado con facilidad en nuevos desarrollos.
- Fácil de mantener: si las modificaciones se llevan a cabo sin problemas. Está relacionado con la estabilidad del sistema.

Todos estos factores son deseables para que el sistema desarrollado presente un rendimiento óptimo. Sin embargo, a la hora de desarrollar y evaluar el funcionamiento de un producto software, lo común es encontrar varios de los puntos de la lista anterior sin cumplir. La pregunta que cabe hacerse es: ¿por qué hay tantos problemas cuando se desarrolla software? La solución a esta pregunta no es única ni simple. De hecho, existen varios motivos, dentro de los cuales destacan los siguientes:

- Naturaleza abstracta de la programación. El software es un concepto intangible, y por tanto difícil de diseñar y desarrollar.
- Problemas de comunicación con los clientes. Es uno de los problemas más comunes, y a la vez más fáciles de resolver. Antes de empezar a planificar la estructura del sistema es imprescindible disponer de todas las especificaciones y

requerimientos. Sin esta información, el desarrollador es incapaz de hacer un buen diseño. De hecho, muchos proyectos se convierten en inviables cuando las especificaciones iniciales se modifican a lo largo del proceso de desarrollo. Por tanto, es muy importante que haya una comunicación fluida entre cliente y desarrollador, y que las especificaciones iniciales del proyecto varíen lo menos posible.

- El software es la parte más cambiante del sistema. Al contrario que el hardware, el software es más flexible y adaptable, puesto que sufre cambios constantemente durante todo el proceso de desarrollo.
- Difusión limitada de las nuevas técnicas, métodos y herramientas. A veces ocurre que, por desconocimiento, los desarrolladores no utilizan recursos disponibles que les facilitarían su trabajo (nuevas librerías, *plugins*, entornos de desarrollo, etc.). Por eso es muy importante llevar a cabo un fase previa de documentación para decidir qué técnicas, métodos y herramientas son las más apropiadas para desarrollar el sistema.
- Poco esfuerzo en análisis y diseño. Es muy común entre los programadores escribir cientos de líneas de código sin diseñar un esquema previo. Esta mala praxis conlleva a la obtención de códigos enrevesados y poco modulares.
- Problemas de gestión: planificaciones optimistas, plantillas poco cualificadas, vencimiento de plazos de ejecución, etc.
- Problemas derivados de la intervención de grupos. En este apartado se engloban problemas relacionados con la falta de comunicación entre los miembros del grupo.

2.3.- MODELOS DE DESARROLLO.

En los últimos años se ha realizado un gran esfuerzo en la investigación de métodos que permitan la inclusión del diseño de una GUI dentro de un proceso de desarrollo basado en modelos [6, 7]. Los beneficios de esta técnica son los siguientes: la automatización del proceso de creación, la generación de interfaces para distintos dispositivos o lenguajes y la mejora de las propiedades de usabilidad del sistema. Respecto al diseño de interfaces de usuario, aunque modelar sus componentes habituales puede resultar sencillo, el desarrollo a nivel de diseño visual, asistencia, manejo de errores, facilidades de ayuda, etc. no es tan directo. A continuación se

muestra un breve resumen de las distintas metodologías para el desarrollo de GUIs basadas en modelos.

2.3.1.- Modelo de Cascada.

Es el modelo más conocido y utilizado, ya que sigue el proceso tradicional de desarrollo. Está basado en el ciclo de vida del software, de forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Las etapas típicas de este modelo son: análisis de requisitos, diseño, implementación, pruebas, implantación y mantenimiento.

La ventaja de este método es su sencillez, pues sigue los pasos convencionales necesarios a la hora de desarrollar el software. Sin embargo, cualquier error detectado en la fase de pruebas implica el rediseño y la re-implementación del código afectado, lo que repercute drásticamente en los costes del proyecto.

2.3.2.- Modelo de Desarrollo Evolutivo.

Este modelo es más flexible que el anterior debido a que las especificaciones del sistema se intercalan con su desarrollo. Esto implica que si, por alguna razón, las especificaciones cambian, los cambios se podrán llevar a cabo mucho más rápidamente. El proceso de desarrollo comienza considerando unos requisitos iniciales y en las últimas versiones se prevén los requisitos que faltan. De esta forma, aunque el sistema esté incompleto, es posible proporcionar una versión utilizable que satisfaga los requisitos básicos iniciales.

2.3.3.- Modelo de Prototipado.

Este modelo se basa en la construcción de un prototipo del sistema final. El prototipado es muy útil cuando los requisitos del sistema no están bien detallados, o cuando no se está seguro de la eficiencia de un algoritmo. En estos casos, lo normal es construir prototipos que sirvan para identificar los requisitos del software. La participación del usuario es clave para este modelo, ya que se exige un alto grado de

interacción para mejorar la efectividad del proceso de desarrollo. El inconveniente del prototipado es la lentitud.

2.3.4.- Modelo de Desarrollo Basado en Reutilización.

Mediante este modelo, el sistema se desarrolla a partir de componentes existentes. Este modelo consta de cuatro fases: análisis de componentes, modificación de requisitos, diseño del sistema y desarrollo e integración. Gracias a esta tecnología se simplifican las pruebas y el mantenimiento del sistema, por lo que el ciclo de desarrollo es más corto. También mejora la calidad, ya que los componentes se mejoran continuamente después de ser construidos. Por otro lado, si un componente no existe hay que programarlo, lo que implica un retraso en el proceso.

2.4.- DISEÑO CENTRADO EN EL USUARIO.

El diseño centrado en el usuario se caracteriza por asumir que todo el proceso de diseño y desarrollo de la interfaz debe estar dirigido por el usuario, sus necesidades, características y objetivos [8, 9]. Centrar el diseño en sus usuarios implica involucrar a los usuarios en todo el proceso de desarrollo. Es deseable que la interfaz sea probada por ellos mismos, así como conocer cuáles son sus reacciones ante el diseño, cómo es su experiencia de uso, etc. En la figura 2.1 se muestra el proceso de diseño de forma esquemática.

La descripción de cada una de las etapas que se muestran en el esquema se especifica con detalle en los siguientes apartados.

2.4.1.- Planificación.

En la primera etapa de cualquier proyecto se debe realizar una buena planificación donde se identifiquen los objetivos que se quieren alcanzar. Para llevar a cabo una planificación correcta, es necesario obtener previamente la máxima cantidad de información posible.

Figura 2.1.- Esquema del proceso de diseño de un producto software.

2.4.2.- Diseño.

De acuerdo a la información adquirida en la etapa de planificación, en esta fase se toman las decisiones que afectan al diseño o a los problemas de usabilidad descubiertos en etapas de prototipado y evaluación. El proceso de diseño consta de varias fases:

- Fase de diseño conceptual: El objetivo de esta fase es definir el esquema de organización y funcionamiento de la interfaz. Esta etapa no se centra en el aspecto de la aplicación, sino en la arquitectura de información.

- Fase de diseño visual: Esta fase se ocupa de la apariencia de la interfaz. Es necesario definir un estilo común entre todos los componentes visuales del sistema, manteniendo la coherencia.

- Fase de diseño de contenidos: Para llevar a cabo un buen diseño de los contenidos se deben tener en cuenta los siguientes aspectos:

- El usuario debe tener el control del sistema, no se puede limitar su actuación.
- La interfaz debe seguir estándares de diseño utilizados ampliamente. Cuanto más se parezca al resto de interfaces gráficas, más fácil de usar resultará para los usuarios.
- La GUI debe ser fácil de usar para usuarios nóveles, pero también proporcionar opciones avanzadas para usuarios expertos.
- Cualquier tipo de información que no sea relevante para el usuario y que sobrecargue la interfaz debe ser eliminada.
- Lo ideal es que la herramienta se pueda utilizar sin necesidad de ayuda o documentación. Sin embargo, el usuario siempre debe tener acceso a ella.
- Los mensajes de ayuda deben ser sencillos y proveer respuestas a los problemas. Los menús y etiquetas de botones deben incluir las palabras claves del proceso.
- Se debe permitir que el usuario pueda salir ágilmente de la GUI, dejando una marca del estado de su trabajo, para que pueda continuarlo en otro momento.
- Se debe asegurar que el usuario nunca pierda su trabajo, ya sea por error de su parte, problemas de transmisión de datos, de energía, o alguna otra razón inevitable.
- Para que la GUI favorezca la usabilidad del sistema de software, la información que se exhiba en ella debe ser fácil de ubicar y leer.

2.4.3.- Prototipado.

En esta etapa se elaboran modelos o prototipos de la interfaz de usuario. Su aspecto no tiene por qué ser el mismo que tendrá la interfaz una vez finalizada, pero puede servir para evaluar la usabilidad de la interfaz sin necesidad de esperar al final de su implementación.

2.4.4.- Evaluación.

Esta fase del proceso se lleva a cabo sobre varias representaciones de la interfaz, como por ejemplo: prototipos en papel, prototipos software, interfaz implementada, etc.

Los factores que se evalúan son los enumerados en el apartado de fase de diseño de contenidos.

Como indica el esquema de la figura 2.1, las tres fases anteriores: "diseño", "prototipado" y "evaluación" son cíclicas e iterativas. Esto quiere decir que todo lo que se diseñe debe ser constantemente evaluado a través de su prototipado, para así poder corregir errores de usabilidad desde los primeros momentos del desarrollo.

2.4.5.- Implementación y Lanzamiento.

En la fase de implementación se debe mantener un control de calidad, supervisando que todo funcione tal y como había sido planificado, puesto que la usabilidad de la interfaz depende directamente de la funcionalidad. Una vez implementada y comprobada su funcionalidad, se procede al lanzamiento de la GUI.

2.4.6.- Mantenimiento.

La fase de mantenimiento no tiene una duración prefijada, sino que se sucede a lo largo del tiempo, porque una GUI no es una entidad estática. Normalmente, sus contenidos cambian, y por tanto requiere mejoras continuamente.

2.5.- PROGRAMACIÓN ORIENTADA A OBJETOS.

La Programación Orientada a Objetos (POO) [10] se creó para conseguir que el código sea reutilizable, modular y fácil de mantener. Para utilizar este paradigma correctamente es necesario pensar las cosas de una manera distinta, e implementar los programas en términos de objetos, propiedades y métodos.

La POO difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, puesto que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar en términos de métodos o funciones, y en segundo lugar en las estructuras de datos que procesan esos métodos. En cambio, en la

POO, primero se definen los objetos para luego enviarles mensajes solicitándoles que realicen sus métodos.

El diseño modular de la POO reduce la complejidad, facilita los cambios y produce como resultado una implementación más sencilla, permitiendo el desarrollo paralelo de las diferentes partes del sistema. El paradigma de la POO es complicado de entender debido a la naturaleza abstracta de los conceptos. A continuación se exponen brevemente las características que definen este tipo de programación:

- Abstracción: consiste en obtener una descripción simplificada del sistema que resalta los detalles más importantes y suprime los irrelevantes. Cada objeto se considera un ente abstracto que puede modificar su estado, variando el valor de sus atributos, y comunicarse con otros objetos, ejecutando sus métodos.

- Encapsulamiento: consiste en ocultar los detalles de un objeto que no contribuyen a sus características esenciales. Para ello, se intentan definir interfaces estrictos escondiendo detalles que no afecten a otros objetos.

- Modularidad: permite descomponer el sistema en módulos independientes fuertemente cohesionados, de forma que incluso puedan compilarse de forma independiente.

- Herencia: es un mecanismo muy versátil que permite crear objetos que incorporen propiedades y métodos de otros objetos sin necesidad de reescribir todo el código. Los objetos pueden heredar los atributos y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo la creación de objetos especializados a partir de objetos preexistentes más genéricos. De esta forma, los objetos de las clases hijas pueden compartir y extender su comportamiento sin tener que volver a implementarlo.

- Polimorfismo: sirve para definir un código que sea compatible con objetos de varios tipos. Esto se consigue haciendo que comportamientos diferentes, asociados a objetos distintos, compartan el mismo nombre. De este modo, al

llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté referenciando.

- Recogida de basura: el *garbage collector* es un mecanismo de destrucción de objetos proporcionado por la máquina virtual de Java. Básicamente consiste en liberar memoria automáticamente cuando se detecta que hay objetos sin referenciar.

2.6.- JAVA.

El lenguaje de programación escogido para implementar la interfaz gráfica de NewFasant ha sido Java. En esta sección se exponen sus características principales y sus ventajas respecto a otros lenguajes similares.

Java es un lenguaje de POO desarrollado por la compañía *Sun Microsystems* [11] en 1990, siendo actualmente propiedad de *Oracle* [12]. Está construido a partir de lenguajes orientados a objetos anteriores como C++, pero no pretende ser compatible con ellos sino añadir nuevas funcionalidades como programación multi-hilo y manejo de memoria a cargo del lenguaje. Inicialmente, Java se diseñó para que la ejecución de código a través de la red fuera segura, para lo cual fue necesario eliminar algunas herramientas de C como el uso de punteros. También se eliminaron aspectos que demostraron ser mejores en la teoría que en la práctica, como la sobrecarga de operadores y la herencia múltiple.

La portabilidad fue otra de las claves para el desarrollo de Java, para lograr que las aplicaciones se escriban una sola vez sin la necesidad de modificarlas para que se puedan ejecutar en plataformas diferentes. Esta independencia se consigue compilando el código fuente para generar un código intermedio llamado *bytecodes,* similar para cualquier plataforma. Posteriormente, la máquina virtual interpreta dicho código. Esto significa que Java es capaz de crear programas multiplataforma, por lo que evita tener que generar distintas versiones dependiendo del sistema operativo que se utilice.

Otro punto positivo de Java es su facilidad para crear interfaces gráficas, ya que proporciona librerías gráficas muy útiles, como se verá en el siguiente apartado. Es

importante señalar que aparte de estas librerías específicas para GUI, Java permite el uso de múltiples librerías para realizar todo tipo de funcionalidades. También permite trabajar en grupo posibilitando la colaboración entre los desarrolladores. Por último, destacar que Java dispone de mecanismos para integrar sus aplicaciones en la web, a través de *web services*.

La plataforma de Java se compone de tres módulos:

- El API (*Application Programming Interface*) de Java, que contiene las clases básicas que utiliza el lenguaje. Las clases se organizan en paquetes o librerías dependiendo de su funcionalidad.

- El JRE (Java Runtime Environment) o entorno de ejecución, el cual es necesario para poder ejecutar las aplicaciones.

- El JDK (Java Development Kit), que incluye un conjunto de herramientas para desarrollar las aplicaciones. Entre ellas están el compilador de Java a código bytes, un generador de documentación, el depurador de programas, etc.

2.6.1.- Bibliotecas Gráficas de Java.

Java proporciona dos paquetes especializados en el diseño de aplicaciones gráficas: AWT (*Abstract Windowing Toolkit*) y SWING. AWT es una librería formada por un conjunto de clases Java, que permite crear elementos básicos para la interfaz gráfica. Posteriormente, AWT ha sido mejorada dando lugar al paquete *javax.swing*. La estructura básica del AWT se basa en Componentes (*Button, Label, Panel*, etc.) y Contenedores (*Panel, Frame, Applet*, etc.). Estos últimos contienen Componentes, de forma que los Eventos pueden tratarse tanto en Contenedores como en Componentes. Los Eventos son los encargados de modelar las acciones del usuario y se manejan mediante interfaces definidas en el paquete *java.awt.event*. Los tipos de Eventos físicos más comunes son: *ComponentEvent, ContainerEvent, FocusEvent, KeyEvent, MouseEvent, WindowEvent*, etc.

Por otro lado, los Gestores de Posición se utilizan para posicionar los Componentes dentro de los Contenedores. Dentro de la librería AWT existen varios Gestores de Posición, aunque los más utilizados son los siguientes:

- *FlowLayout*: es el que tienen establecido por defecto todos los Contenedores. Los Componentes se van colocando por filas en el mismo orden que se van añadiendo al Contenedor. Cuando una fila se llena, los nuevos elementos se colocan en la siguiente.

- *GridLayout*: los componentes se van colocando sobre una malla de izquierda a derecha y de arriba abajo. Todos los elementos de la malla son del mismo tamaño y ocupan todo el área del contenedor.

- *BorderLayout*: este gestor divide el área del Contenedor en cinco zonas: norte, sur, este, oeste y centro.

Respecto a SWING, sus componentes forman una jerarquía cuya cabeza es *JComponent* y ésta, a su vez, tiene su base en AWT, por lo que *JComponent* hereda de *Container* (AWT), *Container* hereda de *Component* (AWT) y *Component* herada de *Object*. Los miembros de esta jerarquía actúan como depósitos de otros componentes, es decir, contienen una lista de componentes, tales como botones, campos de texto, etiquetas, etc. Por ejemplo, *JComponent* tiene numerosas subclases que dan lugar a los componentes que forman la interfaz. Entre ellos están elementos como *JmenuBar, JFrame, JPanel, JButton*, etc. Todos ellos muy utilizados a lo largo del proceso de implementación de la herramienta. La siguiente figura muestra la relación entre ambos paquetes. Las clases en color azul forman parte de AWT, mientras que las clases en color verdoso forman parte del paquete de SWING.

2.7.- JAVA 3D.

En este apartado se presentan las librerías que se han utilizado para desarrollar la parte de visualización gráfica 3D de la herramienta. Se trata de unas librerías gratuitas disponibles en la página web de *Sun Microsystems* llamadas Java 3D [13-15], que componen un API (*Application Programming Interface*) para la representación de

gráficos interactivos utilizando Java como lenguaje de programación. Java 3D extiende de una versión estándar del JDK 2 de Java y proporciona a los desarrolladores un conjunto de clases, interfaces y librerías para crear, manipular y visualizar geometrías en tres dimensiones.

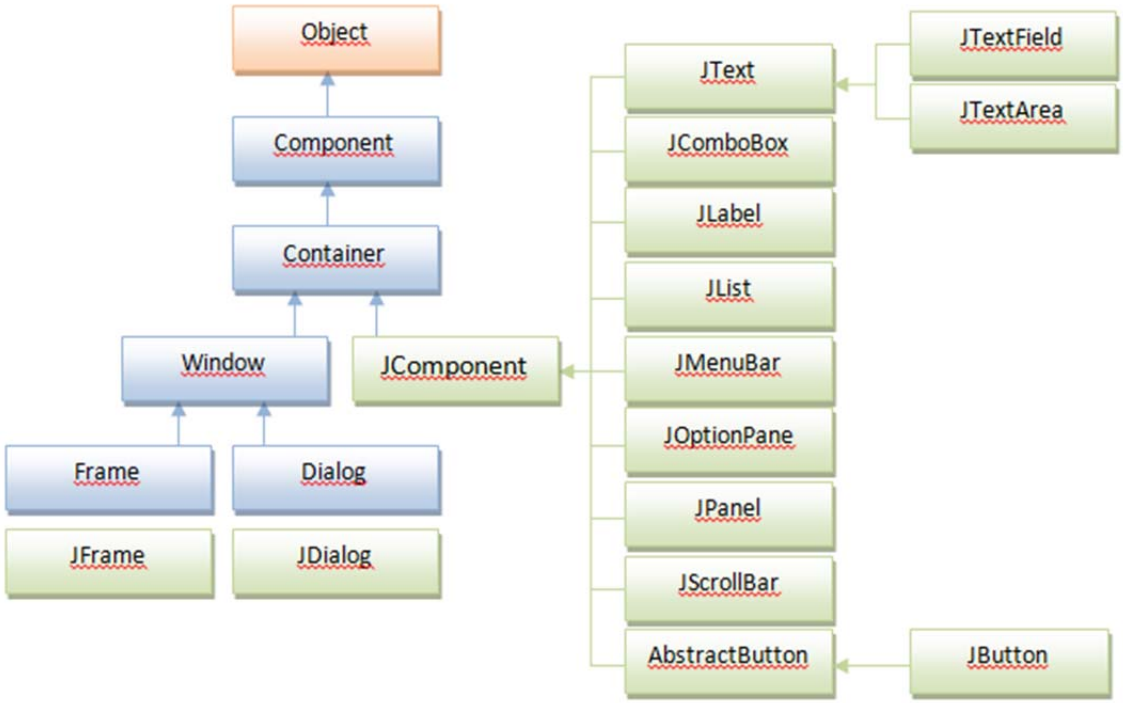


Figura 2.2.- Esquema de librerías gráficas de Java: AWT y SWING.

La creación de objetos y elementos tridimensionales requiere no sólo la generación de los elementos 3D, sino también la definición de todos los aspectos relacionados con la visualización. La principal ventaja que presenta este API frente a otros entornos de programación 3D es que permite crear aplicaciones gráficas independientes del tipo de plataforma.

Java 3D también forma parte del API Java Media Framework o JMF (extensión de Java que permite la programación de tareas multimedia) y por tanto puede hacer uso de toda la versatilidad del lenguaje Java, así como soportar un gran número de formatos como VRML (*Virtual Reality Modeling Language*) [16], formatos de CAD (*Computer Aid Design*) como .DXF, .IGES, etc. A pesar de que tanto conceptualmente como oficialmente Java 3D forma parte del API JMF, son unas librerías que se instalan de forma independiente. Además, permiten aprovechar la aceleración gráfica por hardware

que incorporan muchas tarjetas gráficas, ya que las llamadas a los métodos de Java 3D son transformadas en llamadas a funciones de OpenGL (*Open Graphics Library*) [17] o Direct3D.

Por otro lado, aunque las librerías no soporten directamente cada posible elemento 3D, sí que proporcionan la capacidad de implementarlo a través de código. Java 3D dispone de cargadores que traducen ficheros de formatos VRML, X3D, etc., en objetos entendibles para el API. El hecho de proporcionar una interfaz de programación de alto nivel basado en el paradigma orientado a objetos, permite obtener todas las ventajas de éste: desarrollo simple y rápido de aplicaciones.

En el siguiente capítulo se explican en detalle las funcionalidades que se han utilizado de estas librerías, así como su aplicación al desarrollo de la parte gráfica de la GUI.

2.8.- DIAGRAMAS UML.

Los diagramas UML [18] son elementos muy recomendables para explicar el funcionamiento de cualquier herramienta software que se base en el paradigma de POO. El lenguaje de modelado UML surgió como respuesta a la creciente aparición de herramientas software implementadas usando el paradigma de orientación a objetos. Los diagramas UML se clasifican en dos tipos dependiendo del tipo de representación que se quiera hacer. Entre los diagramas que modelan las partes estáticas de un sistema se encuentran los diagramas de clases, de componentes, de estructura compuesta, de objetos, de despliegue y de artefactos. Por otro lado, para las partes dinámicas del sistema se usan los diagramas de casos de uso, de secuencia, de comunicación, de estados y de actividades. De entre todos ellos, los más empleados son los diagramas de clases, los diagramas de casos de uso y los diagramas de secuencia.

2.8.1.- Diagramas de Clases.

Los diagramas de clases son los más usados dentro de la POO. Contienen información sobre las clases que muestran el sistema, atributos, métodos, interfaces, relaciones de herencia, etc. Como ocurre con el resto de diagramas, se pueden usar para

modelar todo el sistema o sólo parte de él. En este caso, dada la gran envergadura del proyecto desarrollado, este tipo de representación no es aconsejable para describir todas las clases del sistema en un único diagrama, por lo que se suelen hacer varios diagramas de menor tamaño que incluyan unas pocas clases.

Los componentes básicos de un diagrama de clases son tres:

- Clases: son las entidades sobre las que se instancian los objetos y vienen representadas mediante cajas divididas en tres partes. La primera incluye el nombre de la clase, la segunda muestra sus atributos y la tercera sus métodos.

- Interfaces: definen un comportamiento sobre un determinado tipo de clases, pero no instancian objetos. Se representan mediante rectángulos divididos en dos partes: la primera para indicar el nombre de la interfaz y la segunda para mostrar los métodos.

- Relaciones entre clases: se representan mediante una línea y pueden hacer referencia a dos tipos de relación. Si la terminación de la línea en uno de los lados es un triángulo se trata de herencia y si la terminación es en forma de rombo, quiere decir que la clase es compuesta y el tipo de relación es de agregación.

La figura 2.3 muestra un ejemplo de diagrama de clases en el que se observan los dos tipos de relaciones. Según las indicaciones para interpretar el diagrama, se puede deducir que la clase *FasFrame* contiene una lista de objetos, la cual contiene objetos del tipo *Objeto3D*. Ésta, a su vez, es la clase padre de las clases *GeoRotmanLens, GeoCassegrainSystem, GeoPlanarAntenna* y *GeoWireAntenna*. Por otro lado, en la parte de la izquierda se especifica que las clases *RotmanLensMenuFrame, CassegrainSystemMenuFrame, PlanarAntennaMenuFrame* y *WireAntennaMenuFrame* contienen, cada una de ellas, a una de las clases hijas de *Objetos3D* indicadas anteriormente. Por la nomenclatura se entiende que las clases que acaban en "*MenuFrame"* representan los frames de creación de cada una de las geometrías, las cuales vienen indicadas por el prefijo "*Geo"*.

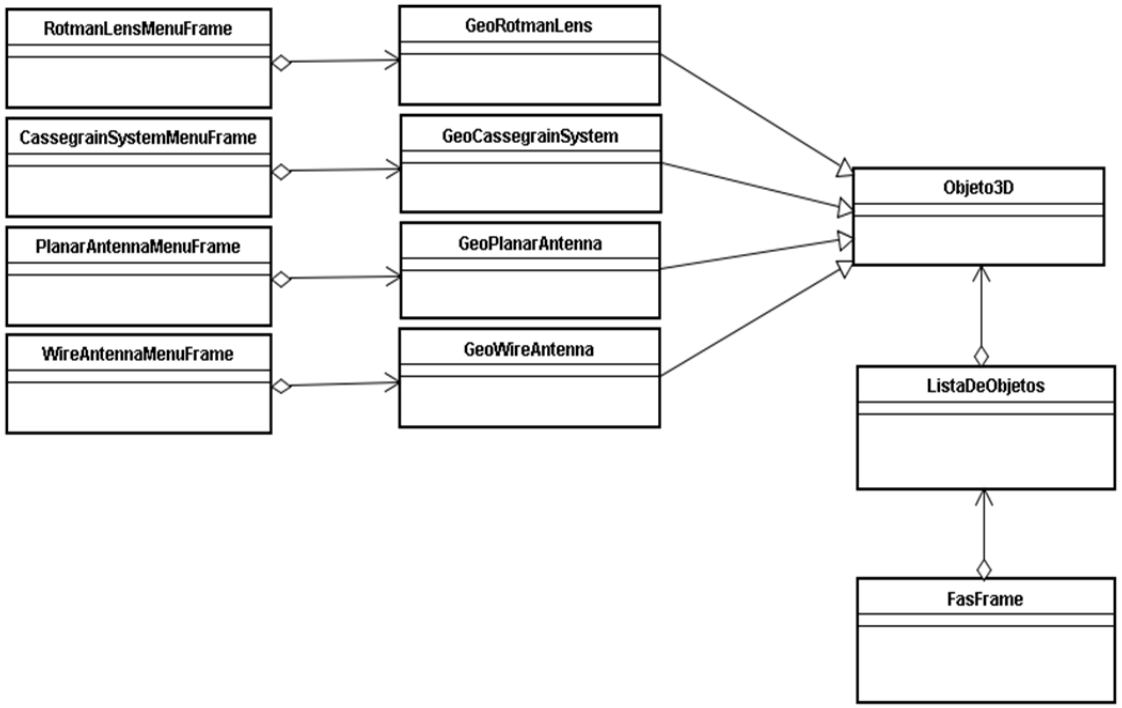


Figura 2.3.- Diagrama de clases de una pequeña parte de la GUI.

2.8.2.- Diagramas de Casos de Uso.

Otros tipos de diagramas UML son los llamados diagramas de casos de uso. Éstos hacen referencia a la parte dinámica de la herramienta e intentan modelar los aspectos dinámicos del sistema, especificando su comportamiento esencial pero sin especificar cómo se implementa.

En este tipo de diagramas se define un "actor" que simula a un usuario, el cual indica una serie de acciones y espera la respuesta del sistema a esas acciones. Los diagramas de casos de uso son muy útiles para analizar los sistemas de forma general y también para llevar a cabo la definición de requisitos. Los componentes básicos de un diagrama de casos de uso son los siguientes:

- Sujetos: es el nombre del sistema y se representa con un rectángulo que agrupa los casos de uso.

- Actores: son las entidades que actúan con el sistema y pueden representar a personas o a otros sistemas. Se identifican mediante iconos de personas en la parte exterior del sujeto.

- Casos de uso: se representan mediante elipses conectadas con los actores mediante líneas, y con otros casos de uso mediante su relación correspondiente.

- Relaciones entre casos: se indican mediante líneas de trazo discontinuo y su tipo viene dado por el sentido de las flechas. Las relaciones de extensión van al caso "anterior" y dan opciones sobre el caso de uso. Las relaciones de uso van hacia el caso "siguiente" y denota que es parte de lo que hará el caso de uso.

La figura 2.4 muestra un diagrama de casos de uso general que modela la interacción de un usuario con la interfaz, es decir, qué puede hacer un usuario con la interfaz.

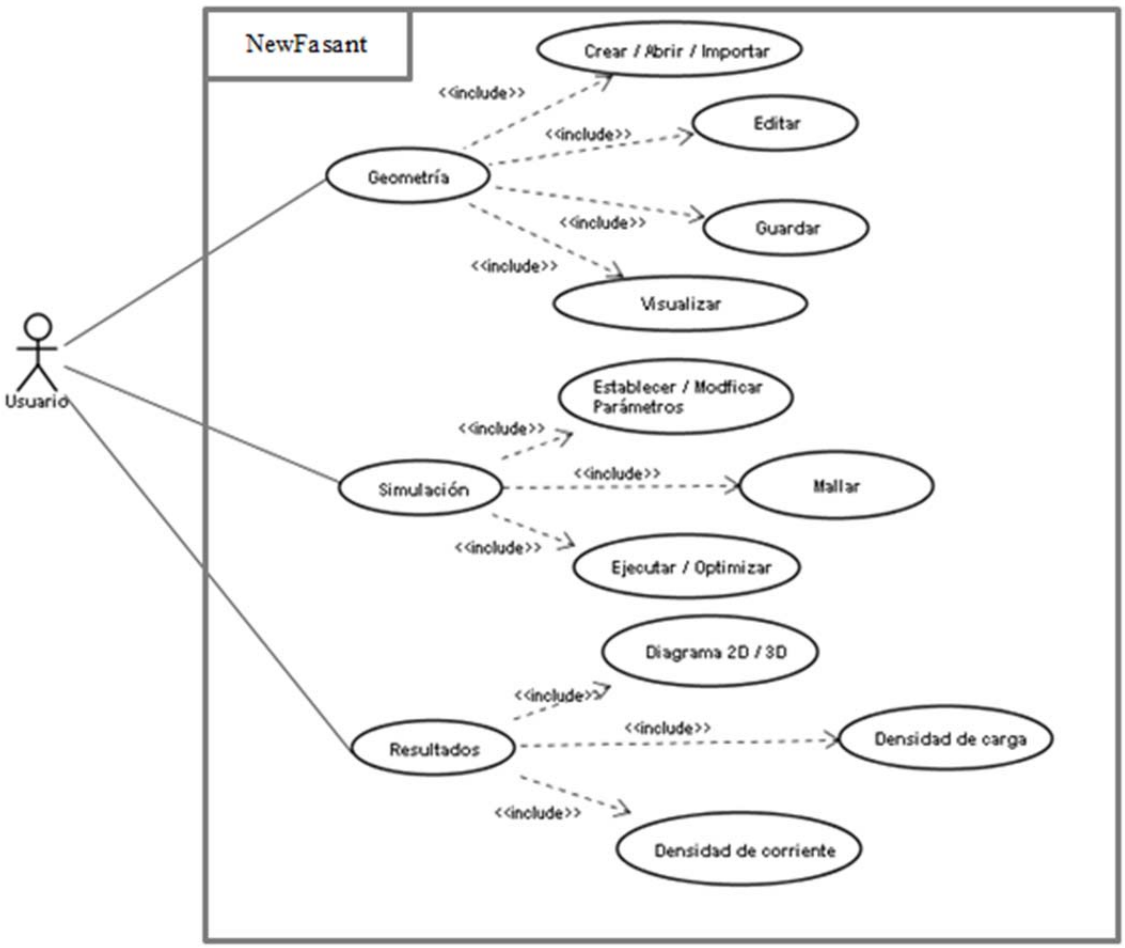


Figura 2.4.- Diagrama de casos de uso general.

2.8.3.- Diagramas de Secuencia.

En ocasiones, es necesario ver con más detalle cómo se realizan los procesos dentro de una ejecución, en otras palabras, ver cómo interactúan los objetos durante la ejecución. Para estos casos se usan los diagramas de secuencia, que también se encargan de modelar el aspecto dinámico de un sistema.

Gráficamente, un diagrama de secuencia es una tabla que representa los objetos a lo largo del eje X, y mensajes, ordenados según se suceden en el tiempo, a lo largo del eje Y. Los componentes básicos de un diagrama de secuencia son:

- Roles u objetos: son las entidades implicadas en la interacción durante el proceso que se está modelando. Los objetos se sitúan en la parte superior del diagrama y se identifican mediante rectángulos.

- Mensajes: es la información que intercambian los objetos y se representan mediante una flecha que conecta los focos de control de cada objeto en ese instante de tiempo. Sobre la flecha se indica el texto aclaratorio del mensaje.

- Líneas de vida: determinan el tiempo de existencia de un objeto durante la ejecución de un proceso. Se representan mediante líneas discontinuas en el eje X debajo del objeto al que se refieren.

- Foco de control: determina el tiempo de actuación de un rol y se representa mediante un rectángulo delgado y estrecho.

La figura 2.5 muestra un diagrama de secuencia donde se especifica en detalle el proceso de creación de una bocina piramidal. El diagrama corresponde al manejo del evento que se dispara cuando el usuario selecciona el menú para generar una bocina piramidal. Cuando se pincha en la opción *Crear Pyramidal Horn*, se ejecuta el evento correspondiente gestionado por el método *ActionPerformed*. Dicha función crea un cuadro de diálogo llamado *PyramidalHornMenuFrame* donde se introducen los parámetros de la bocina: posición, dimensiones, etc. Posteriormente se llama al método

que genera las polilíneas que definen la bocina dentro de la clase *GeoPyramidalHorn* para poder mostrarlas en el escenario.

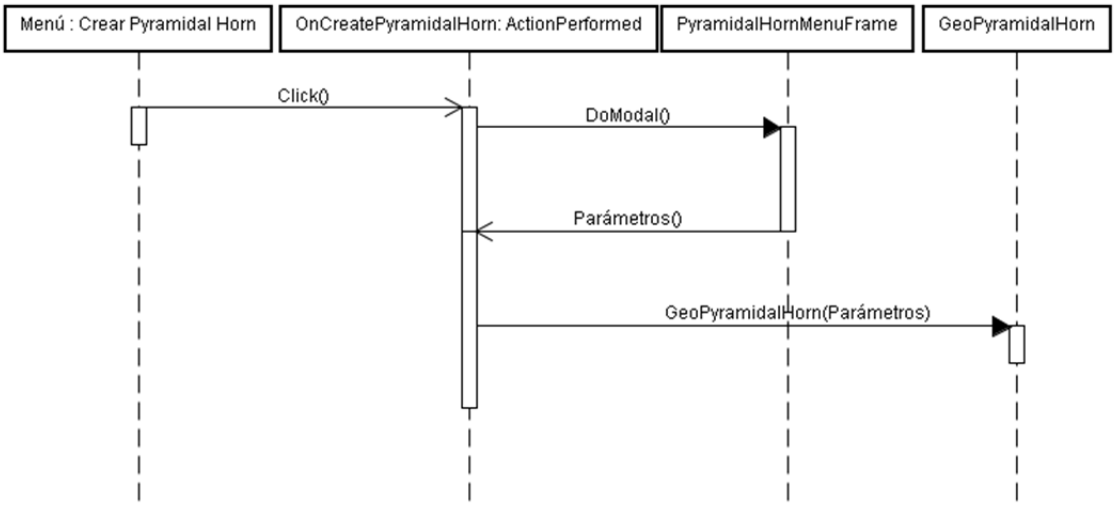


Figura 2.5.- Ejemplo de un diagrama de secuencia del proceso de creación de una bocina piramidal.

2.9.- REFERENCIAS.

[1].- "IEEE Standards Collection. Software Engineering". 1998 Edition. Published by the Institute of Electrical and Electronics Engineers, Inc.

[2].- Pressman, R.S., "Ingeniería del Software. Un enfoque práctico". 6ª ed. 2006: Mc Graw Hill, Interamericana de España S.A.U.

[3].- José R. Hilera, José A. Gutiérrez, J. Javier Martínez. "Estándares en la Ingeniería del Software". Novática. Nov./dic. 1997. Número 130.

[4].- Maguire, M. "A review of human factors guidelines and techniques for the design of graphical human-computer interfaces". Prentice-Hall, 1990, pp. 161-184.

[5].- Mayhew, D. "Principles and guidelines in software User interface design". Prentice Hall, 1992.

[6].- Nunes, D.N.J., "Object Modeling for User-Centered Development and User Interface Design: The Wisdom Approach", 2001, Universidade da Madeira: Funchal.

[7].- Giraldo, W.J., et al. "A Model Based Approach for GUI development in groupware systems", 14th International Workshop on Groupware (CRIWG'2008), 2008. Omaha (EEUU): Lecture Notes in Computer Science. Springer-Verlag.

[8].- International Standard ISO/IEC 12207. "Information technology-Software life cycle processes". 1995.

[9].- J. L. Esteban, M. Piattini. "Procesos del ciclo de vida del software". Novática, Nov./dic. 1995.

[10].- Larman, C. "UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado", Prentice-Hall, 2003.

[11].- Java Sun, "The Java Tutorial: A practical guide for programmers", http://Java.sun.com/docs/books/tutorial, Marzo de 2001.

[12].- http://www.oracle.com

[13].- http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138252.html

[14].- Joan J. Pratdepadua, "Programación en 3D con Java 3D", ed. Ra-Ma, 2003.

[15].- D. Selman, "Java 3D Programming". Independent Publishers Group, 2002.

[16].- Nathan Smith, Christopher Egert, Elisabeth Cuddihy, Deborah Walters: "Implementing Virtual Robots in Java3D using a Subsumption Architecture". Proceedings from the Association for the Advancement of Computing in Education, 1999.

[17].- D. Hearn, M.P.Baker, "Gráficos por computadora con OpenGL", ed. Pearson Prentice Hall, 2006.

[18].- Grady Booch, James Rumbaugh, Ivar Jacobson. "El Lenguaje Unificado de Modelado". Pearson Addison Wesley, 2006.

3.- Creación y Visualización de Modelos Geométricos.

3.1.- INTRODUCCIÓN.

La base de cualquier programa de simulación electromagnética es disponer de un módulo geométrico que permita la creación y modelado de cuerpos, que posteriormente serán analizados.

Este capítulo ofrece una visión general del proceso de creación/importación, manipulación y visualización de modelos geométricos, explicando todo el proceso considerando varios niveles de abstracción: desde el nivel más alto en la que el usuario interactúa con la aplicación, hasta el nivel más bajo en el que se describen las clases que hacen posible la visualización del fichero de geometría en la escena de la GUI.

La implementación de esta parte de la interfaz se ha llevado a cabo íntegramente en Java, sin dependencias externas de librerías comerciales. Las librerías que han permitido llevar a cabo la visualización y el tratamiento de entidades geométricas han sido las librerías de distribución libre de Java 3D [1, 2].

La figura 3.1 muestra un esquema general en el que se exponen las distintas fases del proceso de creación y visualización de un modelo geométrico. A lo largo del capítulo se irán describiendo dichas etapas detalladamente.

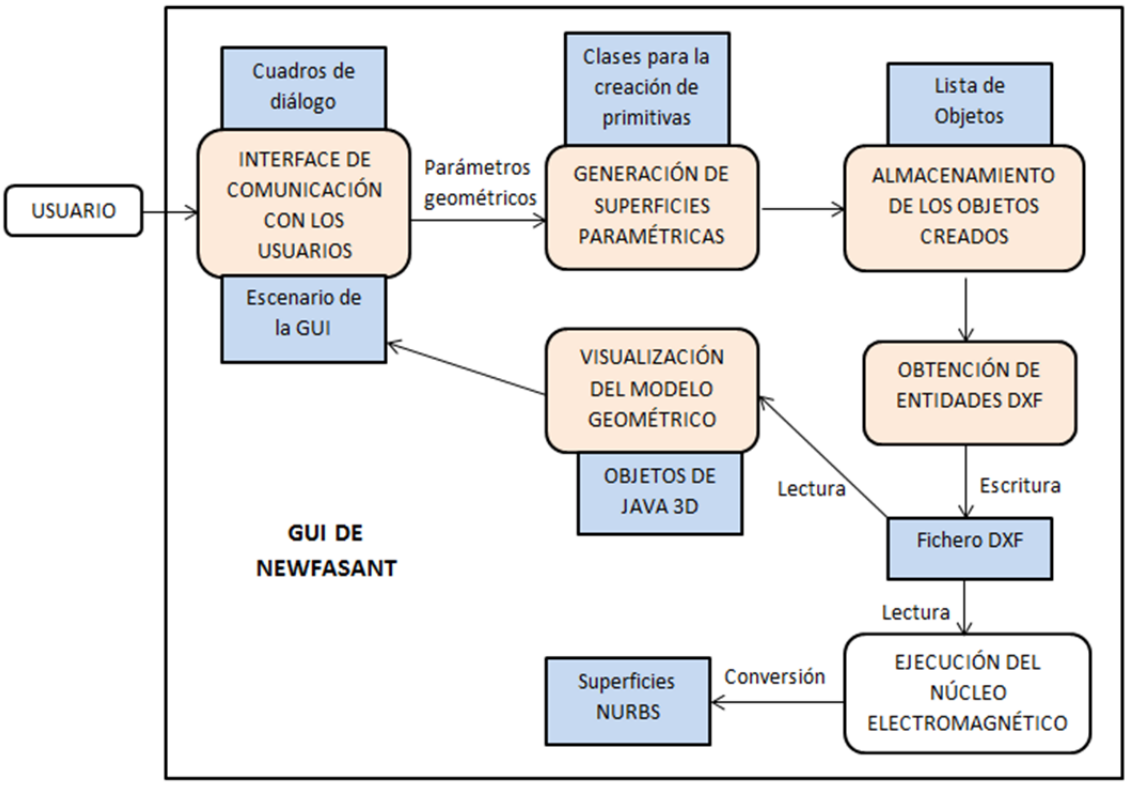


Figura 3.1.- Diagrama de bloques del módulo geométrico.

3.2.- INTERACCIÓN CON EL USUARIO.

En este apartado se describen los mecanismos de interacción entre el sistema y el usuario. El elemento de interacción principal es el propio escenario de la GUI, en el cual se van visualizando los objetos creados. Los cuadros de diálogo, menús, barras de herramientas, iconos, etc., también tienen un papel relevante en el intercambio de información, ya que facilitan la interactividad entre el usuario y la aplicación. Por último, se destaca el papel de los periféricos incidiendo en el ratón como elemento básico de interacción, ya que facilita la navegación por la interfaz, la modificación de las vistas, la selección de objetos, etc. Todos estos elementos se describen en los siguientes apartados, así como la relación existente con las clases de Java 3D que modelan sus funciones.

3.2.1.- Escenario Gráfico.

La zona central de la herramienta está destinada a la visualización de todas las entidades geométricas disponibles en la GUI: primitivas, ejes de coordenadas, puntos de

observación, antenas, etc. Para llevar a cabo la implementación de esta pantalla principal se han seguido las directrices especificadas por Java 3D. A continuación se explica el proceso de creación de un escenario gráfico, así como la estructura de cualquier aplicación cuya implementación se base en Java 3D.

- Creación del Escenario.

La programación de aplicaciones utilizando Java 3D se basa en el modelo de grafos de escena o *SceneGraph*. Dicho modelo conecta objetos separados en una estructura de árbol que incluye los datos geométricos, los atributos y la información de visualización, los cuales proporcionan una descripción completa de la escena y facilitan el control de las capacidades del escenario. El camino que lleva a cada nodo de la estructura arbórea define completamente la información del estado de la hoja y su representación gráfica sirve como herramienta de diseño y como documentación para programas Java 3D.

Para construir un escenario gráfico en Java 3D, se deben crear y manipular objetos geométricos tridimensionales que se encuentran en un universo virtual, el cual se renderiza automáticamente. Java 3D tiene predefinidos los objetos primitivos, pero también es capaz de construir cualquier otro elemento geométrico. Para construir elementos más complejos es necesario utilizar los grupos de transformación o *TransformGroup,* los cuales sirven para situar elementos en el espacio y para realizar rotaciones, translaciones, expansiones, deformaciones, etc.

Los grupos de transformación se pueden enlazar con otro grupo de transformación, siempre que el primero de ellos se una a un grupo de rama o *BranchGroup*. Éste, a su vez, se debe enlazar a un punto de referencia (*Locale*) que dependerá del universo virtual (*VirtualUniverse*). Por tanto, cualquier escenario gráfico debe disponer de un *VirtualUniverse*, que a su vez tiene un único objeto *Locale*, el cual proporciona referencias a un punto en el escenario virtual y sirve de raíz para varios sub-gráficos. Por último, destacar que los objetos *BranchGroup* son la raíz de los sub-gráficos y se dividen en dos categorías: por un lado está la rama de contenido gráfico, que especifica el contenido del universo virtual y por otro lado está la rama de vista

gráfica que especifica los parámetros de visualización. La figura 3.2 muestra el diagrama general de un programa de Java 3D.

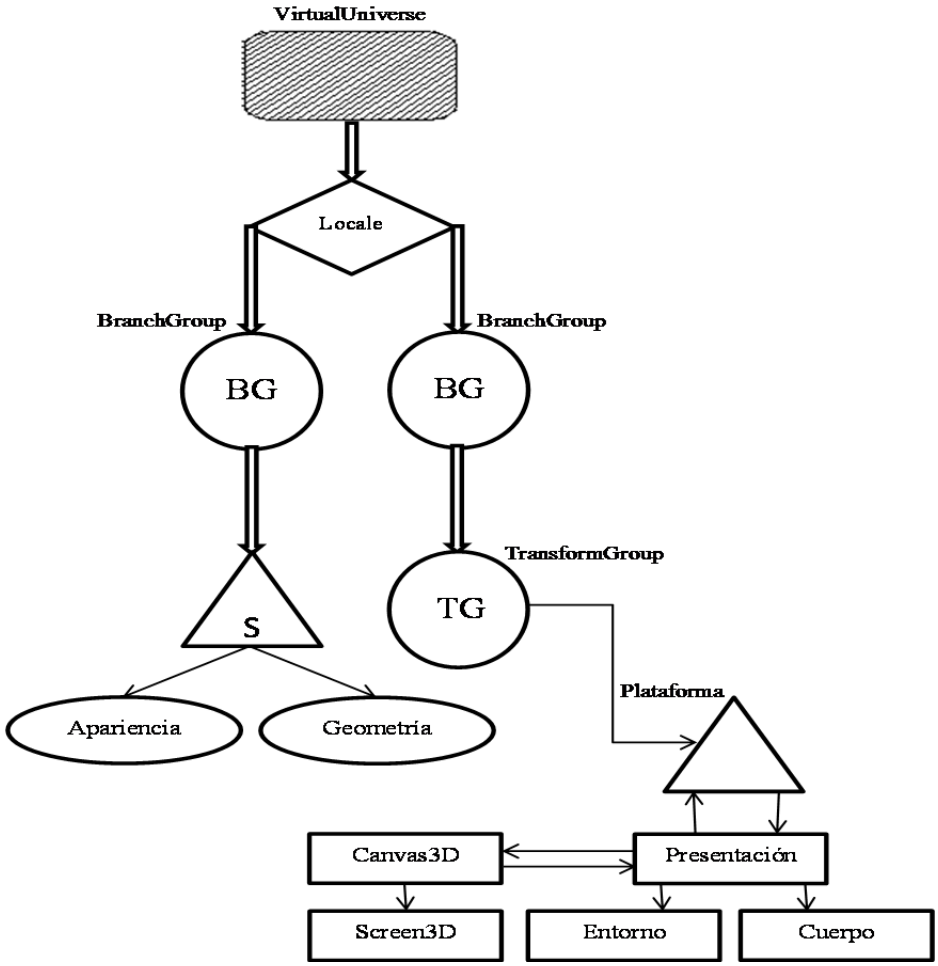


Figura 3.2.- Ejemplo de un único universo virtual con un único punto de referencia desde el cual salen dos ramas: una que generará un determinado objeto tridimensional S (*Shape*) con su geometría y apariencia, y otra que se encargará de definir el marco de presentación por pantalla.

- Jerarquía de las clases de Java 3D.

En este apartado se explica cómo están organizadas las clases de Java 3D, de mayor a menor importancia, estableciendo el mismo orden que sigue el compilador a la hora de crear el archivo ejecutable. El API de Java 3D define más de cien clases, denominadas clases núcleo, incluidas en el paquete *javax.media.j3d.* Los tres primeros niveles de jerarquía de este paquete se muestran en la figura 3.3.

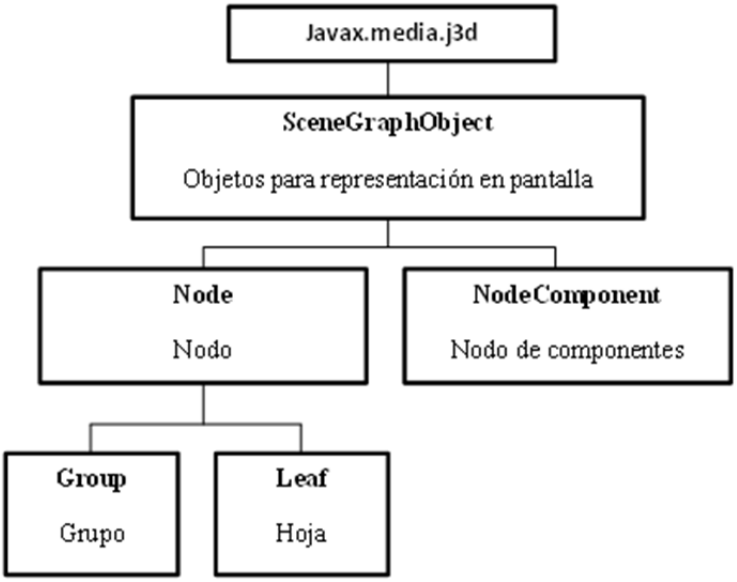


Figura 3.3.- Primeros niveles de la jerarquía de clases de Java 3D.

Del esquema de la figura 3.3 se deduce que *SceneGraphObject* es la clase padre de *Node* y *NodeComponent*. A su vez, la clase *Node* es la superclase de *Group* y de *Leaf* y se encarga de definir subclases para representar diagramas de escena. Por su parte, *Group* se utiliza para establecer las posiciones y orientaciones de los objetos visuales. Sus subclases más importantes son *BranchGroup* y *TransformGroup.* Para especificar la geometría, apariencia, textura y propiedades materiales de los objetos se usa *NodeComponent*. Por último, la clase *Leaf* sirve para especificar formas, sonidos, comportamientos, etc., de los objetos visuales. Los nodos hoja o *Leaf* forman una clase abstracta para todos los nodos del grafo de escena que no tienen hijos. Proporcionan también una plataforma de visión para colocar y orientar los objetos en un punto de vista dentro del mundo virtual. Sus subclases más importantes son:

- *Shape3D*: proporciona el soporte necesario para la creación de objetos geométricos. Contiene dos referencias: una apunta a un objeto de geometría que determina los datos geométricos y otra apunta al objeto de apariencia que especifica los atributos referentes al aspecto del objeto como pueden ser color, material, textura, etc.

- *ViewPlatform*: define una plataforma de visualización que se referencia mediante un objeto *View*. La posición, orientación y escala de las

transformaciones, desde el grafo de escena hasta el nodo *ViewPlatform,* especifican la localización del punto de vista y hacia qué dirección está orientado.

- *Behavior*: proporciona dos tipos de funcionalidades. Por un lado, permite definir las acciones que puede realizar el usuario sobre la escena y por otro lado permite especificar animaciones con objetos visuales. La diferencia entre interacción y animación es que la primera responde a las acciones del usuario, mientras que la segunda responde a eventos que se llevan a cabo a lo largo del tiempo. El objeto *Behavior* puede cambiar cualquier atributo de un objeto visual (posición, orientación, color, etc.). Esta es una de clases que más se utiliza para modelar la interacción entre el sistema y el usuario.

Como se ha mencionado, la interacción con el usuario se lleva a cabo mediante objetos *Behavior*. Se trata de una clase abstracta que proporciona el mecanismo para incluir código que modifique el escenario gráfico. Es decir, el propósito del objeto *Behavior* en un escenario gráfico es modificar el propio escenario gráfico, o los objetos que hay dentro de él, en respuesta a algunos estímulos.

Un estímulo puede ser una pulsación de tecla, un movimiento del ratón, la colisión de objetos, el paso del tiempo, algún otro evento, o una combinación de estos. Los cambios producidos incluyen la adicción de objetos al escenario gráfico, la eliminación o selección de objetos, cambio de atributos de los objetos del escenario gráfico, reordenación de los objetos del escenario gráfico, o una combinación de estos. Las posibilidades sólo están limitadas por las capacidades de los objetos del escenario gráfico.

Como se ha visto hasta ahora, el paquete núcleo *javax.media.j3d* incluye sólo las clases de nivel más bajo necesarias para programar en Java 3D. A la hora de desarrollar programas en Java 3D se utilizan otros paquetes, como por ejemplo el paquete de utilidades (*com.sun.j3d.utils*). Las clases de este paquete son extensiones muy prácticas y potentes y se dividen en cuatro categorías: cargadores de contenido, ayuda a la construcción del escenario gráfico, geometrías y utilidades de conveniencia. La utilización de clases de utilidad reduce significativamente el número de líneas de código

a la hora de programar. Respecto al tratamiento matemático, existe un paquete llamado *javax.vecmath* que define clases matemáticas de puntos, vectores, matrices y otros objetos.

- Etapas de un Programa con Java 3D.

En las aplicaciones que usan Java 3D hay que programar por etapas cada pieza del diagrama de la escena y después conectar cada etapa entre sí para formar el programa final. Si el orden no se lleva a cabo de forma correcta, la visualización no es posible, a pesar de que puede ser que el código compile. El orden de las etapas es el siguiente:

1- Crear el objeto Soporte con la clase *Canvas3D*, la cual simula una ventana donde se van colocando los objetos que se van creando.
2- Crear el objeto Universo Virtual.
3- Crear el punto de referencia.
4- Construir la Rama de Representación
5- Construir la Rama de Contenido.
6- Compilar el diagrama de la escena.
7- Insertar sub-grafos (otras ramas) dentro del punto de referencia.

Sin embargo, existe otra posibilidad que requiere menos líneas de código, en la que se incluye la clase *SimpleUniverse.* Al usar esta clase se evita tener que crear la Rama de Representación, por lo que el diagrama de la escena general queda reducido al esquema mostrado en la figura 3.4.

Los nuevos pasos a seguir con esta alternativa son:

1- Crear objeto *SimpleUniverse*.
2- Crear el objeto Soporte con la clase *Canvas3D*.
3- Construir la Rama de Contenido.
4- Compilar el diagrama de la escena.
5- Insertar la Rama de Contenido dentro del punto de referencia del objeto *SimpleUniverse*.

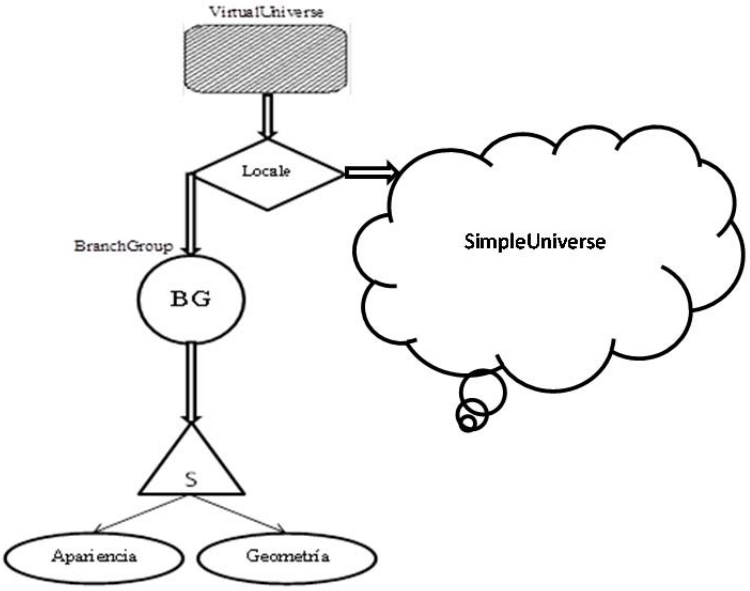


Figura 3.4.- Esquema simplificado.

Cuando se habla de universo virtual se hace referencia al espacio que hay detrás de la pantalla. Al programar en tres dimensiones, el objetivo es que el usuario aprecie un objeto en 3D cuando realmente el objeto tiene únicamente dos dimensiones. Este efecto se consigue proyectando haces de luz desde el objeto al ojo del usuario. En un plano imaginario, entre el objeto y el ojo, se van dibujando estas proyecciones de manera que el resultado final será la imagen que apreciará el usuario como tridimensional.

Por defecto, el plano imaginario se encuentra en el centro de *SimpleUniverse,* también llamado origen de coordenadas. El sistema de coordenadas del universo virtual Java 3D proporciona todas las unidades en metros, y los valores de todos los ejes comprendidos entre -1 y 1. El eje X es positivo hacia la derecha, el eje Y es positivo hacia arriba y el eje Z es positivo hacia el usuario. La Figura 3.5 muestra la orientación con respecto al espectador en un *SimpleUniverse*. De hecho, esa es la orientación convencional para los ejes de coordenadas en un sistema de referencia cartesiano tridimensional. Este tipo de sistema cumple la "regla de la mano derecha", porque el pulgar de dicha mano apunta en la dirección *z* positiva mientras que el resto de los dedos giran desde el eje x positivo hacia el eje y positivo (abarcando 90º). En la

mayoría de los programas, las descripciones de los objetos y otros tipos de coordenadas se especifican mediante coordenadas cartesianas que cumplen esta regla.

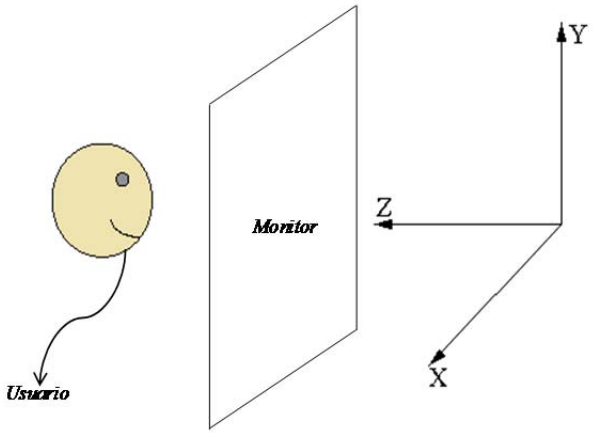


Figura 3.5.- Sistema de coordenadas Java 3D.

El paquete *com.sun.j3d.utils.universe* configura el entorno mínimo para obtener rápida y fácilmente un universo virtual, base de cualquier programa de Java 3D. Este paquete crea todos los objetos necesarios para la rama de vista gráfica. Específicamente crea los objetos *Locale, VirtualUniverse, ViewingPlatform,* y *Viewer* (todos con sus valores por defecto).

En los programas Java 3D, una instancia de *VirtualUniverse* sirve como raíz del escenario gráfico. El término universo virtual se refiere al espacio virtual de tres dimensiones donde se sitúan los objetos 3D. El objeto *Locale* del *SimpleUniverse* establece un sistema de coordenadas cartesianas en el universo virtual, es decir, sirve como punto de referencia para los objetos visuales en el universo virtual.

3.2.2.- Elementos Visuales.

Los elementos visuales representan la forma de comunicación más directa con la interfaz. Aparte del escenario gráfico, existen otros elementos tales como ventanas, menús, barras de herramientas, iconos, botones, etc. a través de los cuales el usuario puede introducir los datos de entrada necesarios para generar las entidades geométricas.

Todos los elementos gráficos mencionados forman parte de las librerías gráficas de Java AWT y SWING, tal y como se comentó en el capítulo anterior. La figura 3.6 muestra uno de los muchos cuadros de diálogo implementados en la GUI, el de la lente de Rotman. Los cuadros de diálogo de creación de primitivas o antenas tienen la misma estructura, de forma que se cumpla el principio de diseño de estandarización. En la parte superior se especifica el nombre de la entidad geométrica y a continuación se especifican las dimensiones que definen la forma del objeto. En la zona central se muestra un esquema donde se puede ver el significado de cada parámetro. Por último, en la parte inferior se establecen los botones *OK* y *Cancel.*

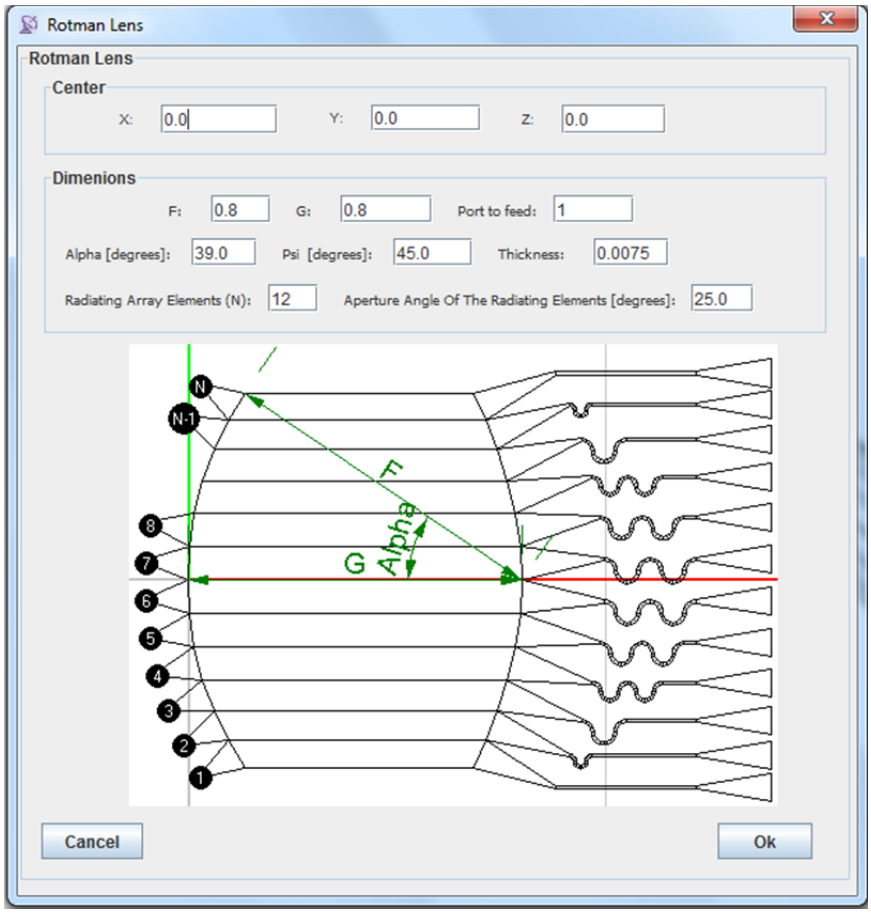


Figura 3.6.- Ejemplo de cuadro de diálogo.

La implementación de los cuadros de diálogo conlleva la comprobación de todos los campos de texto disponibles de modo que el usuario no pueda introducir letras cuando se esperan números, o valores incongruentes como una frecuencia negativa. Internamente, cada cuadro de diálogo incluye un objeto del tipo de entidad que se haya creado. Por ejemplo, en este caso se crearía un objeto *LenteRotmanMenuFrame*, para

modelar el cuadro de diálogo, que contendría un objeto *GeoLenteRotman*, para modelar la lente creada, el cual almacena los parámetros introducidos por el usuario.

3.2.3.- Eventos de Ratón.

El paquete de comportamientos de ratón *com.sun.j3d.utils.behaviors.mouse* contiene las clases del comportamiento en las cuales el ratón se utiliza como entrada de información para la interacción con los objetos visuales. A continuación se resumen las cuatro clases específicas del comportamiento del ratón incluidas en el paquete.

- *MouseRotate*: ejecuta la acción de rotar el objeto sin moverlo como consecuencia de pulsar el botón izquierdo y mover el ratón simultáneamente.

- *MouseTranslate*: ejecuta la acción de mover el objeto en un plano paralelo al escenario como consecuencia de pulsar el botón derecho y mover el ratón simultáneamente.

- *MouseZoom*: ejecuta la acción de mover el objeto en un plano ortogonal al escenario como consecuencia de pulsar la rueda central y mover el ratón simultáneamente.

- *MouseWheelZoom*: ejecuta la acción de mover el objeto en un plano ortogonal al escenario como consecuencia de girar la rueda central del ratón hacia delate y hacia atrás (*zoom in / zoom out* respectivamente).

Además de estas cuatro clases, hay otras dos importantes: la clase abstracta *MouseBehavior,* que es la superclase de estas cuatro, y la interface *MouseCallback,* la cual está implementada en cada una de ellas. Ambas clases se pueden utilizar en la creación de nuevas clases específicas para crear comportamientos personalizados del ratón. Para usar las clases de comportamientos específicos del ratón hay que seguir los siguientes pasos:

1) Proporcionar capacidades de lectura y escritura para el *TransformGroup* fuente.
2) Crear un objeto *MouseBehavior.*

3) Seleccionar el *TransformGroup* fuente.
4) Proporcionar unos límites (o *BoundingLeaf*) para el objeto *MouseBehavior.*
5) Añadir el objeto *MouseBehavior* al escenario gráfico.

Dentro de las interacciones del usuario por medio del ratón se incluye otro comportamiento muy importante. Se trata del *picking* o elección, que permite al usuario interactuar con objetos visuales de la escena por medio de los botones del ratón. En la selección de un objeto visual, el usuario sitúa el puntero del ratón sobre el objeto elegido y pulsa el botón del ratón. En ese instante, el objeto *Behavior* se activa por la pulsación de este botón y empieza la operación de selección:

- Primero se proyecta un rayo dentro del mundo virtual desde la posición del puntero del ratón paralela con la proyección.
- Se calcula la intersección de este rayo con los objetos del mundo virtual.
- El objeto visual seleccionado será el que quede más próximo a dicho rayo.

3.3.- GENERACIÓN DE SUPERFICIES PARAMÉTRICAS.

Una vez que el usuario ha introducido los datos de entrada para generar una primitiva, los datos son parametrizados y almacenados en distintas clases, dependiendo del tipo de entidad geométrica.

Tal y como se mencionó en el capítulo anterior, el código implementado sigue las directrices de la POO, por lo que durante su desarrollo se han aprovechado sus beneficios a través de las propiedades que ofrecen los paradigmas de herencia y polimorfismo. En el proceso de generación de objetos se aplica la herencia, ya que existe una clase base llamada *Objeto3D* a partir de la cual se genera cualquier primitiva. En el siguiente apartado se resumen sus características principales.

3.3.1.- Clase Objeto 3D.

En este apartado se presenta la clase que se encarga de modelar una entidad genérica, ya sea tridimensional o bidimensional, a partir de la cual se puedan obtener

primitivas específicas mediante el mecanismo de la herencia. Para diseñar esta clase hay que determinar cuáles son los atributos y los métodos comunes a cualquier entidad geométrica.

Los tres atributos principales de la clase *Objeto3D* son:

1.- La lista de componentes que contiene todas las entidades que modelan un objeto. Dicha lista contiene objetos del tipo *Polyline* y/o *Facet*, los cuales heredan de la clase *Componente*. La figura 3.7 muestra el diseño de los atributos de la clase *Objeto3D* para visualizar mejor su arquitectura.

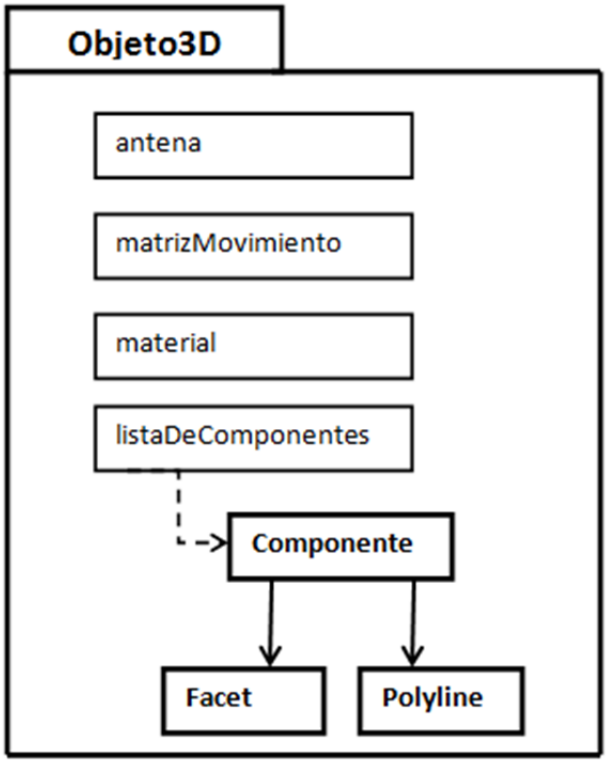
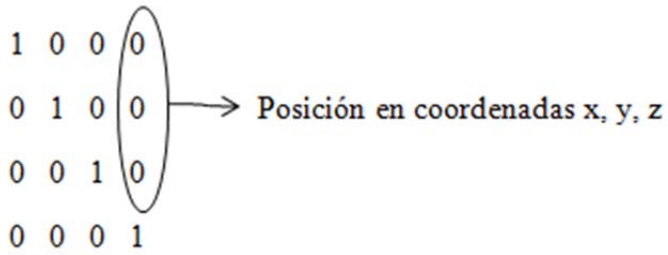


Figura 3.7.- Estructura de los atributos de Objeto3D.

2.- La matriz de movimientos o matriz de transformación sirve para indicar la posición y orientación del objeto en cualquier momento. Consiste en una matriz 4x4 que se va actualizando cada vez que se produce una alteración en la posición (ej: operación mover) o en la orientación (ej: operación rotar) de un elemento geométrico. Cuando se crea un objeto de

la clase *Objeto3D*, dicha matriz se inicializa con la matriz identidad, del siguiente modo:



3.- El tipo de material asociado a las superficies que componen la primitiva. El material que asigna por defecto es el denominado PEC (*Perfect Electric Conductor*).

4.- Si el objeto que hereda de Objeto3D es una antena (bocina, reflector, etc.) esta propiedad incluirá la definición del sistema antena: el tipo de alimentación, las coordenadas del punto de alimentación, la orientación, el tipo de polarización, etc.

Uno de los métodos que incluye la clase *Objeto3D* se encarga de multiplicar cada punto que define el objeto por su matriz de movimiento. Así, cada vez que se actualice la matriz, también se actualizará la posición y orientación del objeto. Sin embargo, el de mayor relevancia es el método que devuelve la lista de componentes, pues está presente en todas las subclases. Este método se redefine en cada primitiva y su implementación es más o menos compleja dependiendo de las características de la clase a la que haga referencia.

3.3.2.- Creación de Geometrías con Java 3D.

Hay tres formas principales de crear contenidos geométricos [3, 4]. Una forma es usar las clases de utilidades geométricas para *box, cone, cylinder,* y *sphere*. Otra forma es especificar coordenadas de vértices para puntos, segmentos de líneas y/o superficies poligonales y una tercera forma es usar un cargador geométrico. A continuación se exponen las tres alternativas con más detalle.

- Geometrías primitivas.

  El paquete *com.sun.j3d.utils.geometry* contiene las clases para crear figuras geométricas básicas. Dichas geometrías son *Box*, *Cone*, *Cylinder* y *Sphere*. Todos estos objetos tienen en común que se generan por defecto en color blanco. Sin embargo, su aspecto se puede modificar con el método *setApperarance (Appearance ap).*

- Geometrías complejas.

  Mediante esta segunda opción, el desarrollador puede crear cualquier tipo de geometría arbitraria, dado que los objetos se generan a partir de puntos distribuidos en el espacio. Éste ha sido el método utilizado dentro de la interfaz gráfica de NewFasant para crear tanto las geometrías sencillas (esferas, cubos, elipses, etc.) como las complejas (bocinas, reflectores, lentes, etc.).

Por otro lado, la clase *GeometryArray* hereda de *Geometry* y permite crear objetos complejos a partir de sus vértices. El único constructor que proporciona esta clase necesita el número y el formato de los vértices como parámetros de entrada. Los tipos de formato disponibles para cada vértice son:

- *Coordinates*: especifica que el vértice contiene las coordenadas de posición.
- *Normals*: especifica que el vértice contiene la dirección del vector normal a la superficie en esa posición.
- *Color_3:* especifica el color de cada vértice sin transparencia.
- *Color_4:* especifica el color y grado de transparencia de cada vértice.
- *Texture_coordinate_2:* especifica las dos coordenadas necesarias para definir la textura en cada vértice.
- *Texture_coordinate_3:* especifica las tres coordenadas de la textura tridimensional para vértice.

A su vez, *GeometryArray* es superclase de otras clases geométricas muy útiles para generar objetos complejos: *PointArray, LineArray, TriangleArray* y *QuadArray*. Estas clases han sido muy útiles para llevar a cabo la visualización de los resultados, sobre todo la clase *QuadArray*, que aporta mucha flexibilidad

a la hora de representar los diagramas de radiación en tres dimensiones, como puede observarse en la figura 3.8.

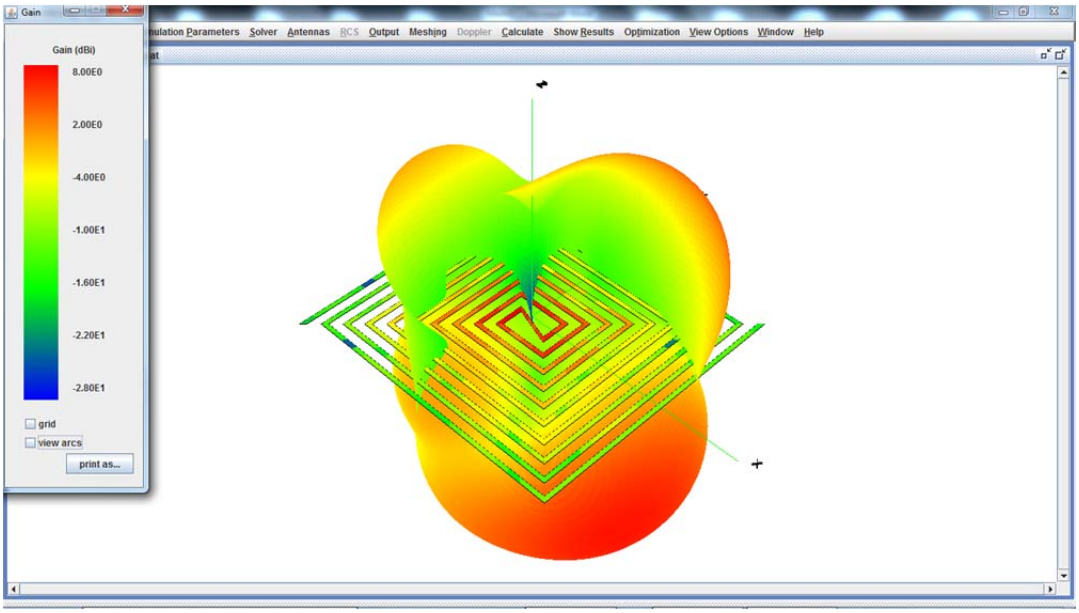


Figura 3.8.- Diagrama de radiación tridimensional implementado utilizando la clase QuadArray.

- Cargador de geometrías.

Para conseguir un grado de complejidad mayor a la hora de generar un modelo geométrico, se usan los cargadores de geometría o *Loaders*, los cuales permiten incluir cualquier tipo de figura tridimensional en el diagrama de escena que se haya creado con una aplicación externa. Una vez introducida en el diagrama de escena se podrá tratar como cualquier otra geometría, con la ventaja de que se evita crearla desde cero.

Básicamente, el procedimiento de carga necesita dos clases: *Loader* y *Scene*. La primera sirve para localizar y capturar las características del fichero donde está la imagen y determinar cómo insertarla, mientras que la segunda contiene los datos del diagrama de la escena, como pueden ser el número de nodos, los elementos geométricos que la componen, la iluminación, etc.

Es importante destacar en este punto que Java 3D no proporciona todos los cargadores de todas las herramientas CAD que existen en el mercado, ya que éstas son muy numerosas y se van renovando continuamente. Sin embargo,

proporciona las directrices que hay que seguir para que los desarrolladores sean capaces de implementar sus propios cargadores. Actualmente, los cargadores disponibles en Java 3D son para aplicaciones como: 3D-Studio, Solid Works, Imagine, AutoCad Drawing Interchange File, WaveFront, Imagine, etc.

El objeto devuelto por un *Loader* se llama *Scene* y contiene toda la información necesaria para poder representar la imagen en el universo virtual (puntos, valores para la iluminación, para el sonido, perspectiva, etc.). La información más importante que contiene es la del *BranchGroup*, ya que en ella están contenidos todos los objetos que forman el escenario. Algunos de los métodos básicos incluidos en cualquier cargador son:

- *getSceneGroup(),* para obtener la rama principal del escenario.
- *load(),* para extraer la información del escenario.
- *setBasePath(),* para establecer un vínculo con archivos asociados al archivo de imagen anteriormente cargado.
- *setFlags(),* para establecer la información que se quiere extraer de la imagen a partir de unas constantes.

En el caso que nos ocupa, se ha partido de un cargador de archivos con formato *.dxf* previamente desarrollado. Dicho cargador ha sido mejorado, ya que sólo disponía de opciones muy básicas. A la clase inicial se le han añadido funcionalidades, como por ejemplo el método que genera y visualiza los ejes de coordenadas que aparecen en el centro de la pantalla cuando se inicia el programa. También se añadió la opción de visualizar algunos elementos básicos en cualquier simulación electromagnética como los puntos de observación y el sistema de referencia absoluto de una antena. La figura 3.9 muestra el aspecto de dichos elementos.

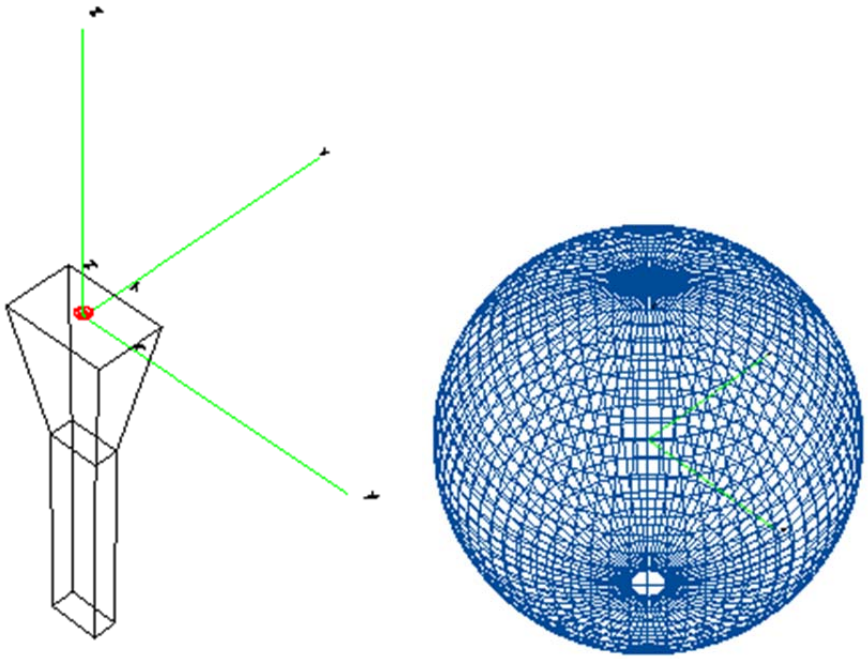


Figura 3.9.- Funcionalidades añadidas al cargador básico de ficheros .dxf. Ejes de coordenadas y sistema de referencia absoluto de la antena (izquierda). Puntos de observación definidos sobre una esfera (derecha).

Otras funcionalidades incluidas en el cargador son las relacionadas con las opciones de visualización de la escena, como por ejemplo:

- *Zooms (in/out)* para alejar o acercar la cámara. Esta opción se efectúa moviendo la ruleta del ratón (girándola hacia delante o hacia atrás).

- *Orbit* para girar la perspectiva y visualizar la escena desde cualquier punto de vista. Su funcionamiento se maneja manteniendo pulsado el botón izquierdo del ratón a la vez que se mueve.

- *Pan* para mover la perspectiva sin giros, sólo con desplazamientos. Esta opción se ejecuta igual que la anterior, pero utilizando el botón derecho del ratón.

La descripción detallada del cargador se encuentra en la clase *LoadDXF*, la cual se describe en el apartado 3.6.1.

3.3.3.- Apariencia y Atributos.

Además de generar las geometrías desde un punto de vista puramente formal, Java 3D también proporciona soporte para modificar el aspecto visual de los objetos creados. Por ejemplo, cualquier objeto *Shape3D* tiene definidos dos campos: la geometría y la apariencia. Un atributo es cada una de las características que Java 3D permite definir de la apariencia de un objeto. Los atributos son subclases del tipo *NodeComponent*. Algunos de ellos son:

- *ColoringAttributes*: determina los colores que se usarán en el objeto y en su sombra. La degradación del color entre los vértices de un objeto se especifica a través del parámetro *Shading*. Si está desactivado, no existirá cambio de tonalidad y cada cara o vértice se verá de un color.

  Según la convención de colores empleada en la GUI, todos los objetos se introducen en el escenario sin color, definidos mediante líneas sencillas de color negro (figura 3.10a). Sin embargo, cuando un objeto es seleccionado, su color cambia utilizando el parámetro *Shading*, para obtener el resultado mostrado en la figura 3.10b.

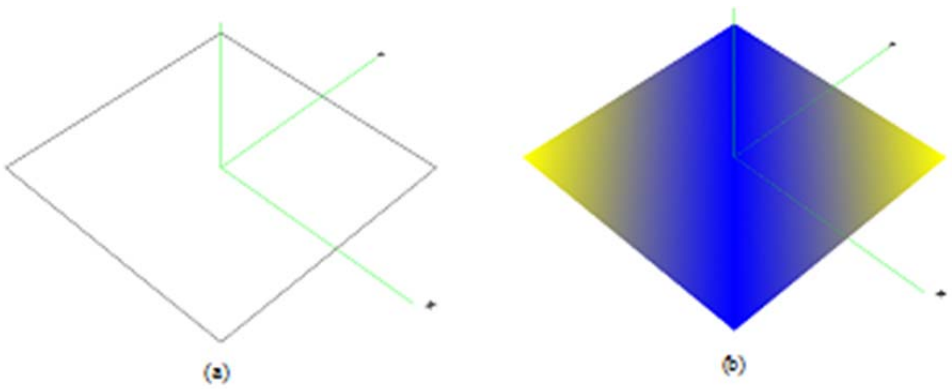


Figura 3.10.- Aplicación del atributo ColoringAttributes en la GUI. (a) Faceta creada en color blanco por defecto. (b) Modificación del aspecto de la faceta al ser seleccionada.

- *LineAttributes*: define el estilo de línea (*pattern_solid, pattern_dot, pattern_dash_dot* y *pattern_dash*) y su grosor.

- *PointAttributes*: indica las características del punto. Por defecto, un punto ocupa un pixel de la pantalla. Si el tamaño de un punto es muy grande, se puede usar la función de 'Antialiasing' para aproximar el formato del punto a una forma más redondeada.

- *PolygonAttributes*: para definir cómo se representa un polígono. Se puede controlar su apariencia compacta, dibujando las caras de los polígonos o sólo dibujando los vértices. También se pueden ocultar las caras para facilitar la interpretación. Si se activa *cull_front* se ocultan las caras visibles, si se activa *cull_back* se ocultan las caras no visibles y si se activa *cull_none* no se oculta ninguna cara.

Dentro de la herramienta NewFasant existe una opción de visualización llamada *View Hidden Axonometric* que utiliza este atributo para obtener una imagen renderizada de los objetos. Por ejemplo, a la izquierda de la figura 3.11 se muestra el diseño de una bocina circular corrugada creada por defecto, mientras que a la derecha de la figura 3.11 se muestra el mismo objeto visualizado activando la opción *View Hidden Axonometric*. La implementación de este mecanismo de visualización consiste en activar las opciones de *cull_front* y *polygon_fill*.

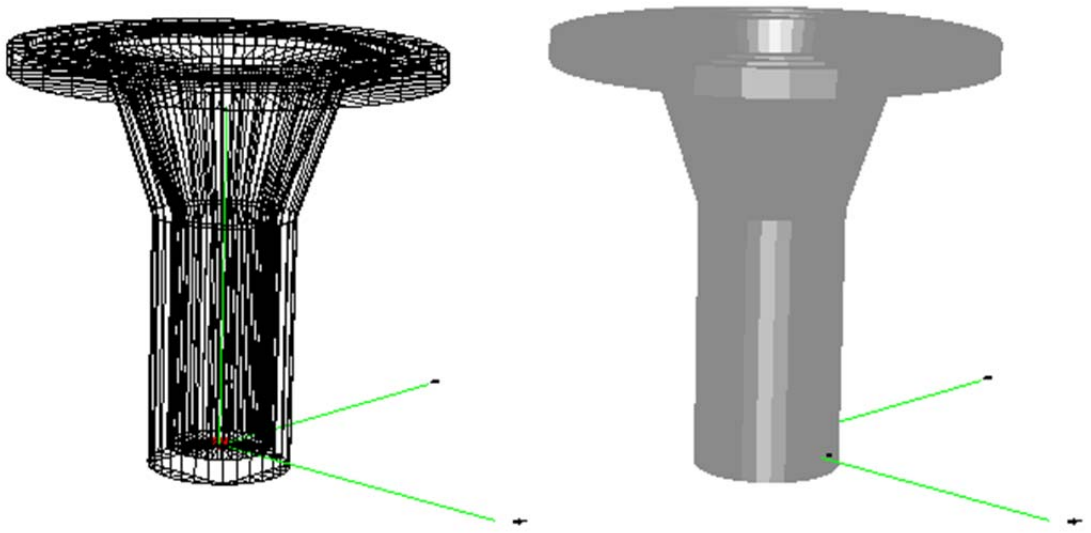


Figura 3.11.- Aplicación del atributo PolygonAttributes en la GUI. Bocina corrugada creada en color blanco por defecto (izquierda). Modificación del aspecto de la bocina al utilizar la opción de visualización View Hidden Axonometric (derecha).

- *RenderingAttributes*: para indicar las características comunes que comparten los objetos de las clases primitivas *Box, Cylinder, Cone* y *Sphere* como el tipo de renderizado o el color de los vértices.

- *TransparencyAttributes*: define el tipo y grado de transparencia de un objeto. Los modos posibles son *none, blended, fastest, nicest* y *screen_door*. El grado de transparencia varía entre 0.0 y 1.0. Si se le asigna un valor 0.0 se obtiene un objeto totalmente opaco, por el contrario, un valor de 1.0 indica que se tratará de un objeto completamente transparente.

- *Material*: para definir el comportamiento de un objeto al ser iluminado por un haz de luz.

El procedimiento para asignar un determinado aspecto a un objeto es el siguiente: primero se crea uno de los objetos anteriores dependiendo de la característica que se quiera modificar, después se definen las propiedades deseadas y por último se añaden a un objeto *Appearance*.

3.3.4.- Transformaciones Básicas.

Las matrices de transformación se utilizan para realizar las operaciones geométricas básicas como traslación, rotación o escalado. Con pequeños cambios y combinaciones en estas matrices, se puede conseguir que una sola matriz resultante sirva para varias de estas transformaciones. Para llevar a cabo una rotación, una translación o un escalado en el espacio son necesarias tres clases de Java 3D: *Transform3D, TransformGroup* y *Vector3f*. A continuación se muestran sus principales características.

- *Transform3D*: los objetos de esta clase pueden modelar transformaciones del tipo rotación, translación, compresión y expansión. Implícitamente, dichas transformaciones se definen mediante sumas y productos de matrices 4x4. Dentro de la clase Transform3D existen muchos métodos que permiten

modificar dicha matriz. Los más importantes son: *mul(), rotX(), rotY(), rotZ()* y *setScale()*.

- *TransformGroup:* esta clase hereda de la clase *Group* y sus objetos se usan para representar las transformaciones geométricas de los objetos. Para crear un objeto *TransformGroup* es necesario disponer previamente de un objeto *Transform3D*.

- *Vector3f:* esta clase se encuentra en el paquete *javax.vecmath* y sirve para definir un vector de tres dimensiones con una precisión de coma flotante. Su uso es común dentro de otros objetos, por ejemplo para definir translaciones o direcciones.

La utilización de estas clases dentro de NewFasant se lleva a cabo en varias opciones, como por ejemplo en la rotación de los ejes de coordenadas cuando se mantiene pulsado el botón izquierdo del ratón. La parte izquierda de la figura 3.12 muestra el modelo geométrico de un satélite junto con los ejes de coordenadas centrados en su posición original, mientras que en la parte derecha de la figura se muestra su aspecto después de haber aplicado varios movimientos y giros a través del ratón.

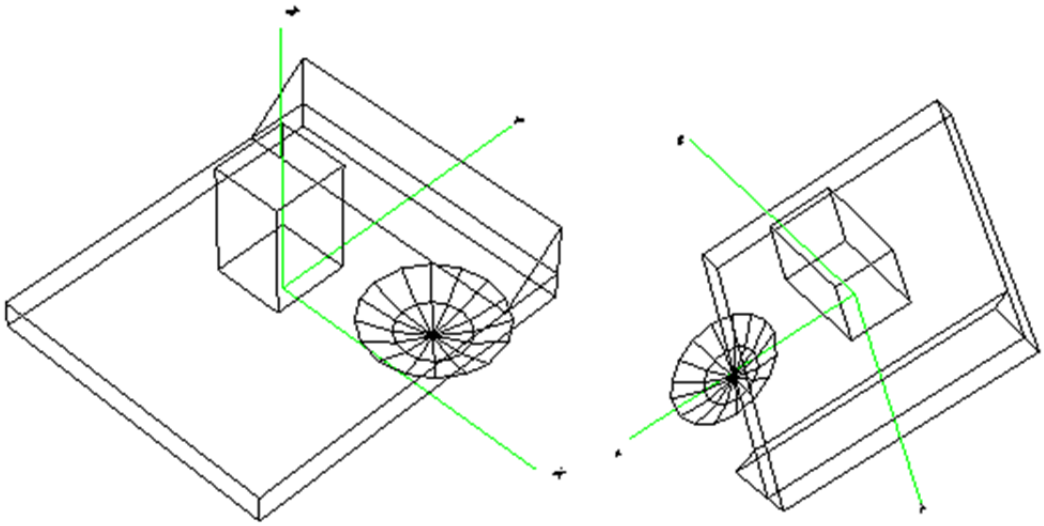


Figura 3.12.- Aplicación de transformaciones 3D sobre el escenario. Geometría centrada con la orientación por defecto (izquierda). Modificación de la orientación mediante eventos a través del ratón (derecha).

3.3.5.- Primitivas Desarrolladas.

Las primitivas disponibles en el módulo de geometría se clasifican en cuatro categorías:

1. Objetos abiertos: rectángulo, triángulo, disco, elipse, placa con agujero, paraboloide, hiperboloide, etc.
2. Objetos cerrados: cubo, esfera, cilindro, cono y elipsoide.
3. Estructuras periódicas: la celda unidad puede ser un anillo, una cruz metálica, una cruz cargada, una placa rectangular, una placa con agujero, etc.
4. Elementos de circuitos microstrip.

A partir de estas entidades básicas es posible generar cualquier tipo de estructura, ya que la interfaz dispone de una amplia variedad de operaciones geométricas que se pueden aplicar sobre los objetos del escenario.

Como ejemplo de entidad geométrica desarrollada, a continuación se describe el proceso de creación de un paraboloide, que se compone de 4 superficies degeneradas, definidas mediante cuatro polilíneas. Los datos necesarios para crear esta superficie se muestran en la figura 3.13.

La malla de puntos que componen la superficie del paraboloide siguen las ecuaciones (3.1)-(3.3):

$$x = radio\ \cos(\varphi + \alpha) \tag{3.1}$$

$$y = radio\ \sin(\varphi + \alpha) \tag{3.2}$$

$$z = h \tag{3.3}$$

donde α indica el ángulo de giro del paraboloide respecto al eje z. La figura 3.14 muestra la vista de planta de un paraboloide cuya con un valor de α establecido en 30º.
φ es la coordenada esférica que realiza un barrido desde φ=0º hasta φ=360º.
h viene dado por la ecuación (3.6), donde R es el radio de la apertura que se puede generar en el centro del paraboloide, f es la distancia focal y $incrZ$ es una variable dependiente de la altura máxima $Zmax$ (3.5), que se va incrementando en cada iteración.

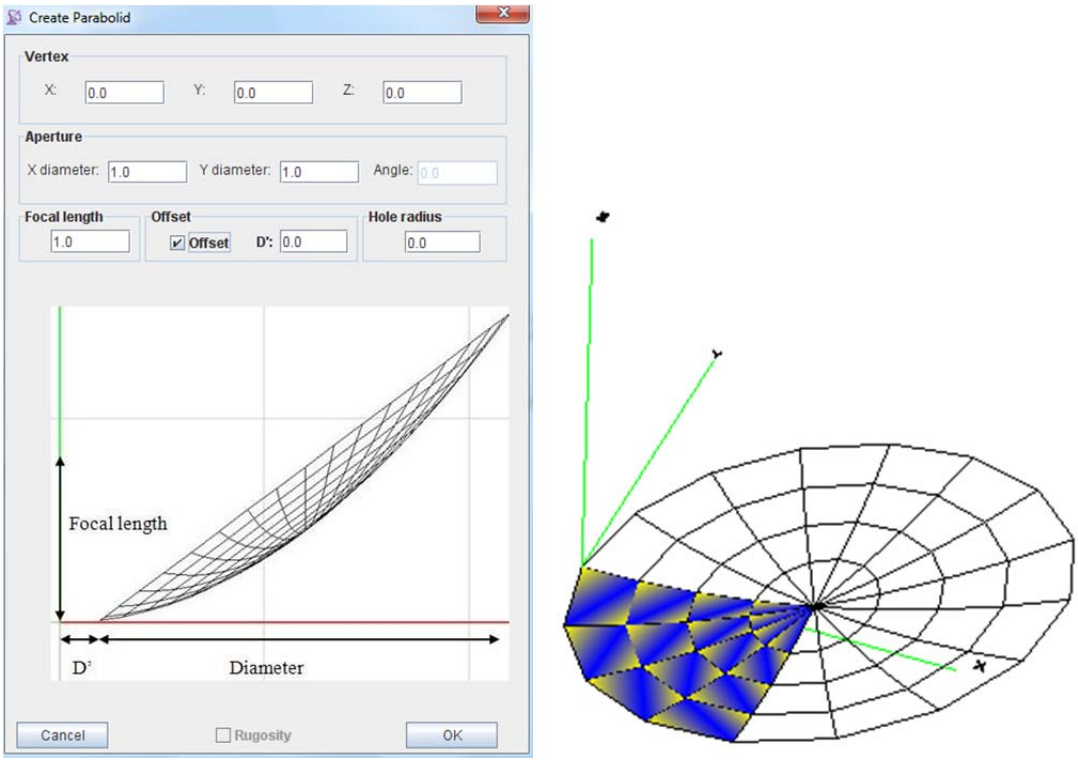


Figura 3.13.- Primitiva Paraboloid junto a su cuadro de diálogo.

En la figura 3.15 se muestra un paraboloide que incluye una apertura. Este inusual parámetro es muy útil a la hora de diseñar antenas reflectoras reales, ya que prácticamente todas ellas incorporan este mecanismo para facilitar su alimentación.

Por último, *radio* es una variable expresada en coordenadas esféricas dependiente de la relación entre los radios establecidos por el usuario en *x* y en *y*, tal y como se indica en (3.9). Estos radios vienen definidos por *A* y *B* respectivamente. Notar que los valores iniciales de *A* y *B* sirven para identificar la relación entre ambos ejes en (3.4). Sin embargo, en cada iteración, estos valores se actualizan en función del parámetro *h*, tal y como se observa en (3.7) y (3.8).

$$relacionEjes = B/A \tag{3.4}$$

$$Zmax = A^2/(4f) \tag{3.5}$$

$$h = incrZ + \frac{R^2}{4f} \tag{3.6}$$

$$A = \sqrt{4hf} \tag{3.7}$$

$$B = A\, relacionEjes \tag{3.8}$$

$$radio = \sqrt{\frac{A^2B^2}{(A\sin\varphi)^2 + (B\cos\varphi)^2}} \tag{3.9}$$

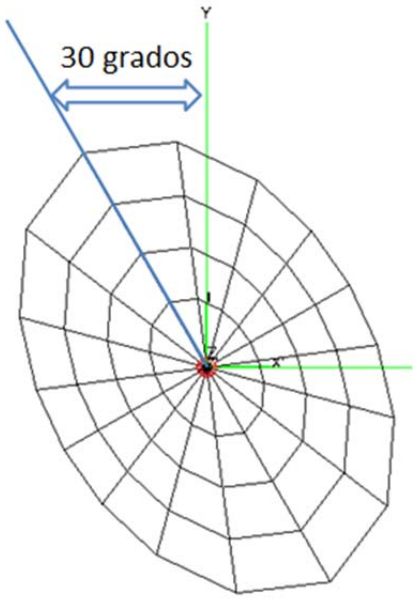


Figura 3.14.- Vista de planta de un paraboloide de diámetro en x igual a 4m, diámetro en y igual a 6m y ángulo de giro igual a 30º.

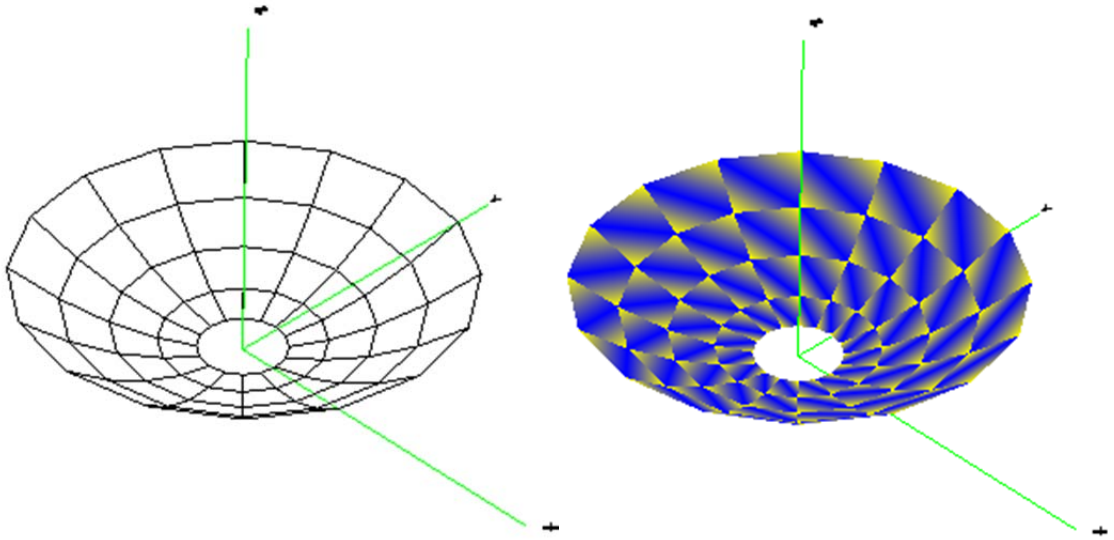


Figura 3.15.- Ejemplo de un paraboloide con apertura.

Esta primitiva también ofrece la posibilidad de establecer o no un offset, de modo que el objeto resultante se corresponda con la superficie parabólica determinada por un cilindro que corte a dicha superficie a una cierta distancia, dada por el valor del

parámetro *D'*. La figura 3.16 ilustra el esquema de obtención de un paraboloide con offset. El ejemplo mostrado en la figura 3.13 es un caso de paraboloide con offset cuyo parámetro *D'* es cero.

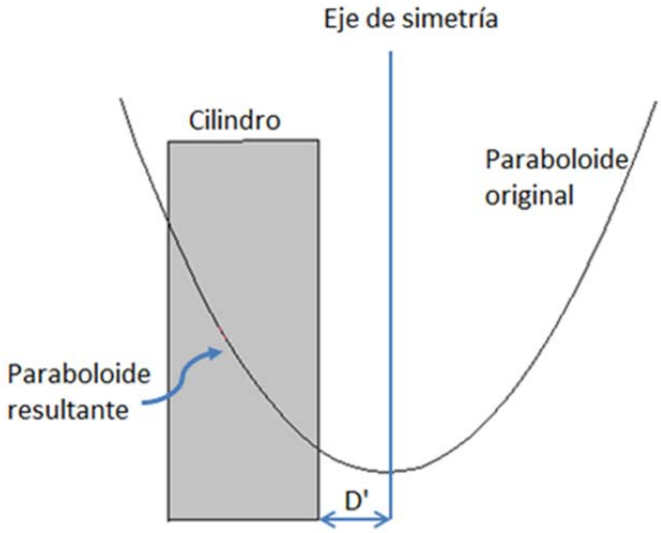


Figura 3.16.- Obtención de un paraboloide con offset.

Tenga o no tenga offset, el paraboloide generado habrá actualizado su lista de componentes, de tal forma que las cuatro polilíneas que lo definen estarán almacenadas en dicha lista en forma de cuatro objetos tipo *Polyline*. A su vez, los cuatro objetos *Polyline* habrán definido los valores *u* y *v* iguales a cinco y habrán establecido los veinticinco puntos en coordenadas *x, y, z* que se han calculado siguiendo las ecuaciones explicadas anteriormente.

3.4.- ALMACENAMIENTO Y TRATAMIENTO DE OBJETOS.

Tanto el almacenamiento como la gestión de toda la información que se maneja en la interfaz se lleva a cabo en la clase *FasFrame*. Se trata de una clase controladora y contenedora de prácticamente todas las demás, cuyos datos se almacenan en una estructura en forma de árbol.

Uno de los atributos de la clase *FasFrame* es una estructura que contiene referencias de cada uno de los objetos presentes en la escena. Se trata de una lista en la que se van insertando o eliminando posiciones en función de las acciones realizadas por

el usuario. Cada vez que se añade un nuevo objeto al modelo geométrico, se aumenta el tamaño de la lista de objetos en una posición. Cuando algún elemento es eliminado, su posición dentro de la lista también se elimina de modo que siempre se mantenga actualizada. Como cada elemento posee información sobre su tamaño, orientación y posición, cada vez que se produce una transformación geométrica sobre alguno de los objetos, éste se vuelve a cargar en la lista para actualizar su información. Por otro lado, cuando se edita un objeto, el mecanismo de selección devuelve su referencia. Por tanto, la búsqueda de objetos se realiza en función de dicha referencia, que actúa como identificador.

Como se puede ver, el almacenamiento y tratamiento de los objetos es sencillo, ya que ambos procesos se resumen en la gestión de un vector unidimensional. Como todas las clases que representan entidades geométricas heredan de *Objeto3D*, en realidad la lista de objetos contiene, estrictamente hablando, un conjunto de referencias del tipo *Objeto3D*.

En el siguiente apartado se explica la relación entre la lista de objetos y el fichero de geometría, que es un elemento básico para poder realizar la simulación electromagnética de la estructura creada.

3.5.- GENERACIÓN DEL FICHERO DE GEOMETRÍA.

Una vez que el diseño geométrico ha finalizado, el siguiente paso consiste en traspasar toda la información contenida en la lista de objetos a un fichero de extensión .DXF. Según se explica en el capítulo 5, el núcleo electromagnético realizará posteriormente la lectura de dicho fichero y llevará a cabo un proceso de interpolación para transformar la malla de puntos dada por el fichero .DXF en superficies NURBS.

En el proceso de creación del fichero de geometría intervienen por un lado la lista de objetos de la clase *FasFrame*, y por otro lado la clase *FileDXF*, cuya descripción se realiza en el siguiente sub-apartado.

3.5.1.- Clase FileDXF.

Esta clase modela un fichero .DXF y se usa para leer y escribir los ficheros de geometrías en formato .DXF. Sus atributos son un búfer de escritura, otro de lectura y un objeto tipo *File*. Entre sus métodos se encuentran dos principales: *loadFile()* y *saveFile()*. El primero se encarga de leer el fichero físico y de devolver un vector con todos los componentes (polilíneas o facetas) leídos. El segundo método es el encargado de escribir el fichero en sí.

La información de la malla de puntos que hay que escribir en el fichero se obtiene a partir de la lista de componentes que tienen los objetos almacenados en la lista de objetos. Básicamente, el proceso de escritura consiste en hacer una transferencia de información para expresar los mismos datos de forma distinta. Para poder generar el fichero .DXF hay que acceder a cada posición de la lista de objetos y hacer un volcado de la información de cada objeto diferenciando entre componentes tipo *Facet* o tipo *Polyline*.

3.6.- VISUALIZACIÓN DEL MODELO GEOMÉTRICO.

Las clases involucradas en el proceso de visualización del modelo geométrico en el escenario son tres: *LoadDXF, FileDXF* y *FasFrame*. La clase *FasFrame* sirve de puente de comunicación entre *FileDXF* y *LoadDXF*. Como se ha mencionado anteriormente, *FileDXF* se usa para leer los ficheros de geometrías, cargar su contenido en la interfaz y transferir todas las entidades geométricas leídas a la clase *FasFrame*. Una vez que dicha información se encuentra disponible, la clase *FasFrame* transfiere todos los datos a la clase *LoadDXF* para que ésta inicie el proceso de visualización de la escena.

3.6.1.- Clase LoadDXF.

Esta clase controla la parte de geometrías 3D que se muestran en la escena, almacenando toda la información que se muestra en pantalla una vez que se ha cargado.

En ella se pueden encontrar los cargadores de geometría, así como los datos de la escala y de la traslación con respecto al origen necesarios para ajustar y centrar la geometría dentro del campo de visión. También contiene elementos comunes a todas las geometrías en general, como los ejes de coordenadas, el contenedor *canvas3D* o la dirección específica de cada geometría que esté en uso. Como atributos destacados están los siguientes:

- Almacenes temporales de la geometría, o de parte de ella. Atributos tales como *tmpSwitch* o *tmpSimpleUniverse*. El primero incluye el nodo que agrupa a la geometría cargada desde el fichero .DXF, y el segundo es el contenedor general de toda la escena. También se encuentran en esta clase atributos como *currentShapes*, en el que se almacenan las superficies seleccionadas.

- Datos sobre la escala y traslación de una figura cargada. Cuando se carga una geometría, el cargador proporciona una escala con la que ha sido cargada y un vector de traslación, debido a que, por defecto, el cargador centra la figura en el escenario.

- Definición de atributos de modo. Dependiendo del modo que se elija en el atributo modificación se podrán seleccionar figuras, superficies, puntos, bloques, etc. Los diferentes modos se pueden encontrar en la clase *DefConf*.

  Como operaciones destacadas se enumeran las siguientes:

- Capacidades y permisos de geometría. La geometría tiene una serie de comportamientos que hay que definir. Dependiendo de los parámetros establecidos sobre una misma geometría se podrán seleccionar, o no, sus facetas, se podrá visualizar como opaca, se podrá representar mediante líneas, se podrán modificar sus colores, luminosidades, etc.

- Cambios del punto de vista de la geometría. Se pueden configurar puntos de vista muy comunes en herramientas CAD, tales como la vista centrada, las plantas, perfiles, etc.

- Métodos de selección: para interactuar con parte de la geometría es necesario poder seleccionar dicha parte. La clase también incluye los métodos que se encargan de gestionar los índices y devolver las partes seleccionadas. Se pueden encontrar métodos de selección tales como *getGeometryShape(), getCurrentShapes(),* o *getGeometryShapeIndex()*.

- Métodos de adición de elementos comunes. Hay ciertos tipos de entidades que se repiten en todas las geometrías como por ejemplo los ejes de coordenadas, las antenas, los puntos de observación, etc. Para tratar estos elementos se han implementado los métodos *cargarAntenas(), pintaPuntosObservacion(),* etc. que se apoyan en métodos de más bajo nivel como *addCone(), addPoint(),* o *addBox()*.

Una vez que se han expuesto las características principales de la clase *LoadDXF*, se puede continuar con la descripción del proceso de visualización de geometrías.

En primer lugar hay que definir el entorno estableciendo variables como *canvas3D* y *simpleUniverse*. Para que el cargador pueda ejecutar su función es necesario pasarle el fichero de geometría, que se modela mediante un objeto *FileDXF*. El cargador de geometría ejecuta el método de carga del fichero y devuelve un objeto tipo *switch* que contiene toda la escena correspondiente al fichero especificado. En este punto hay que tener en cuenta que el cargador tiende a colocar la figura en el centro de los ejes de coordenadas y sin escala, por lo que el modelo cargado no se corresponderá exactamente al incluido en el fichero .DXF. Por esta razón, el siguiente paso consiste en obtener las traslaciones y la escala correspondiente para poder tratar la geometría cargada y hacer que diferentes elementos que se introduzcan posteriormente con una medida real, se ajusten a la representación de la geometría.

Una vez se ha tratado el punto de la geometría del fichero se definen las capacidades y apariencia de la geometría. También se definen los ejes de coordenadas y sus correspondientes letras *x, y, z*.

Para poder interactuar con la geometría, cambiando los puntos de vista en tiempo real mediante el uso de ratón y conseguir así los efectos de rotación, traslación y

zoom es necesario implementar los manejadores pertinentes y asociarlos a la rama principal de representación del grafo de escena.

Después de definir los manejadores, se da paso a la introducción de otros elementos comunes, como son las antenas y los puntos de observación. Una vez que todos los elementos están incluidos en el grafo de escena, se ejecuta la instrucción *compile()*. Esta función sirve para acelerar el tratamiento y la renderización de la representación de los elementos visuales.

La figura 3.17 muestra un ejemplo de una geometría visualizada en la en el escenario mediante la importación de un fichero .DXF creado con una herramienta externa.

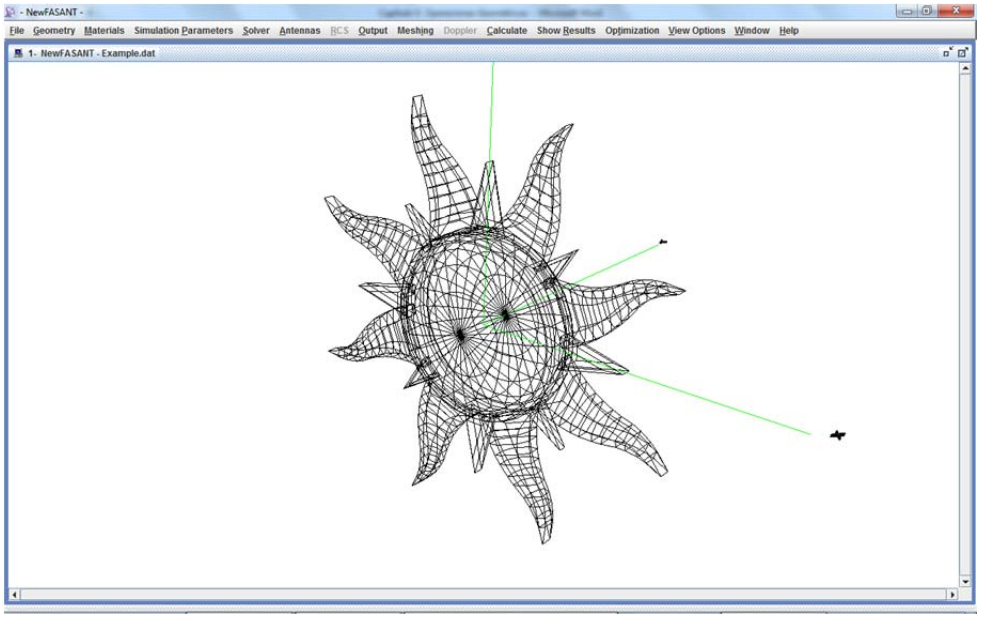


Figura 3.17.- Ejemplo de geometría importada.

3.7.- REFERENCIAS.

[1].- http://www.oracle.com/technetwork/java/javase/tech/index-jsp-138252.html

[2].- Joan J. Pratdepadua, "Programación en 3D con Java 3D", ed. *Ra-Ma*, 2003.

[3].- D. Selman, "Java 3D Programming". *Independent Publishers Group*, 2002.

[4].- Palmer, Ian, "Essential Java 3D fast developing 3D graphics applications in Java", Springer, 2001.

4.- Módulo de Antenas.

4.1.- INTRODUCCIÓN.

Tanto en el ámbito académico como en el industrial es muy común la realización de simulaciones de todo tipo de antenas. En la actualidad, existen muchas herramientas CAD como *AutoCAD* o *Rhinoceros,* que permiten la creación de modelos geométricos de estructuras complejas, antenas, etc. Por otro lado, es muy común que las propias herramientas de simulación incorporen facilidades para generar primitivas y aplicar transformaciones sobre ellas, con el objeto de modelar estructuras que posteriormente serán analizadas. Aparte de disponer de prácticamente las mismas posibilidades de tratamiento y manipulación de geometrías que los programas de CAD, es muy conveniente que las herramientas de simulación como NewFasant, ofrezcan la posibilidad de poder crear de forma rápida los modelos geométricos de las antenas más comunes, como bocinas, reflectores, etc. Como se comentó en el capítulo introductorio, tanto FEKO como CST incluyen el paquete comercial *Antenna Magus*, cuyo propósito es facilitar la creación de modelos de antenas, para poder ser analizadas posteriormente.

Debido a la importancia y utilidad de este tipo de facilidades gráficas dentro de las herramientas software de simulación, se ha incluido en la interfaz de NewFasant un módulo especializado en la creación, parametrización, simulación y optimización de antenas.

De este modo, cuando se necesite diseñar una antena relativamente compleja, en vez de descomponerla en entidades sencillas para construirla a partir de primitivas, se podrá generar automáticamente sin más dificultad que especificar sus dimensiones y la forma en la que es alimentada.

4.2.- BOCINAS.

La gran variedad de primitivas y operaciones geométricas proporcionadas por la herramienta, permite llevar a cabo el proceso de diseño de antenas de bocina sin dificultades. Sin embargo, existen algunos tipos de antenas de bocina que, debido a su uso tan ampliamente extendido, es conveniente poder generarlas de forma automática. Este es el caso, por ejemplo, de las bocinas piramidales o cónicas, cuyo diseño a partir de primitivas se convierte a veces en una tarea tediosa y repetitiva. En los siguientes apartados se describen brevemente los fundamentos teóricos de las antenas de bocina, los tipos de bocinas que se han implementado y cómo se ha llevado a cabo su proceso de diseño.

4.2.1.- Fundamentos Teóricos.

Las antenas de bocina se utilizan en muchas aplicaciones debido a sus múltiples ventajas: gran ancho de banda, facilidad de construcción, robustez mecánica y gran pureza de polarización. Entre sus usos convencionales destacan su aplicación como antena individual embarcada en satélites [1-4], por ejemplo, o formando parte de agrupaciones de antenas [5, 6]. Otra aplicación muy extendida es la de alimentar antenas reflectoras [6-8].

Proporcionan haces directivos con ganancias que oscilan entre 10 y 25 dBi y generalmente operan a frecuencias de microondas. Dependiendo de su forma, se clasifican en bocinas cónicas y rectangulares. Las primeras se componen de una apertura en forma cónica y de una guía circular, que propaga el modo fundamental TE_{11}. Las segundas se alimentan mediante una guía rectangular que propaga el modo fundamental TE_{10}, y a su vez se dividen en tres categorías en función de su apertura. Las bocinas de plano H se caracterizan por disponer su apertura en el plano horizontal, mientras que las bocinas de plano E establecen su apertura en el plano vertical. Por último las bocinas piramidales se distinguen porque la apertura se produce en ambos planos simultáneamente.

La distribución de campos en la apertura de las bocinas rectangulares viene dada por las expresiones (4.1)-(4.3):

Para la bocina de plano E: $$E_y = E_0 \cos\left(\frac{\pi x}{a}\right) e^{-j\frac{\beta y^2}{2L_E}} \quad (4.1)$$

Para la bocina de plano H: $$E_y = E_0 \cos\left(\frac{\pi x}{a}\right) e^{-j\frac{\beta x^2}{2L_H}} \quad (4.2)$$

Para la bocina piramidal: $$E_y = E_0 \cos\left(\frac{\pi x}{a}\right) e^{-j\frac{\beta x^2}{2L_H}} \cdot e^{-j\frac{\beta y^2}{2L_E}} \quad (4.3)$$

donde *a* es el tamaño de la apertura en el eje X, y L_H y L_E hacen referencia a las diferencias de fases que se producen como resultado de la apertura en los planos horizontal y vertical respectivamente.

La bocina piramidal incluye ambas diferencias de fase debido a que aumentan tanto las dimensiones horizontales como las verticales.

Respecto a las bocinas cónicas, la distribución de campos en la apertura se muestra en las expresiones (4.4) - (4.7).

$$E_\rho = E_0 e^{-j2\pi s\left(\frac{\rho}{a}\right)^2} \frac{1}{\rho} J_1\left(1.841\frac{\rho}{a}\right) \sin\phi \quad (4.4)$$

$$E_\phi = E_0 e^{-j2\pi s\left(\frac{\rho}{a}\right)^2} \frac{1}{\rho}\frac{\partial}{\partial\rho}\left(J_1\left(1.841\frac{\rho}{a}\right)\right) \cos\phi \quad (4.5)$$

$$s = \frac{d_m^2}{8\lambda L} \quad (4.6)$$

$$d_m = 2a \quad (4.7)$$

donde *a* es el radio de la apertura, *L* es su longitud y J_1 es la función de Bessel de primer orden.

El error cuadrático de fase que aparece en las ecuaciones (4.4) y (4.5) se debe, al igual que ocurre con la bocina piramidal, a la diferencia de caminos que recorren las ondas.

4.2.2.- Tipos de Bocinas Implementadas.

El bloque de antenas que se ha implementado permite crear los modelos geométricos de bocinas rectangulares (de plano E, de plano H y piramidal), cónicas, guías de onda rectangulares convencionales, con afilamiento y con apertura inclinada y bocinas tanto rectangulares como cónicas que incluyen corrugaciones en su zona de apertura. La figura 4.1 muestra un ejemplo de cada una de ellas.

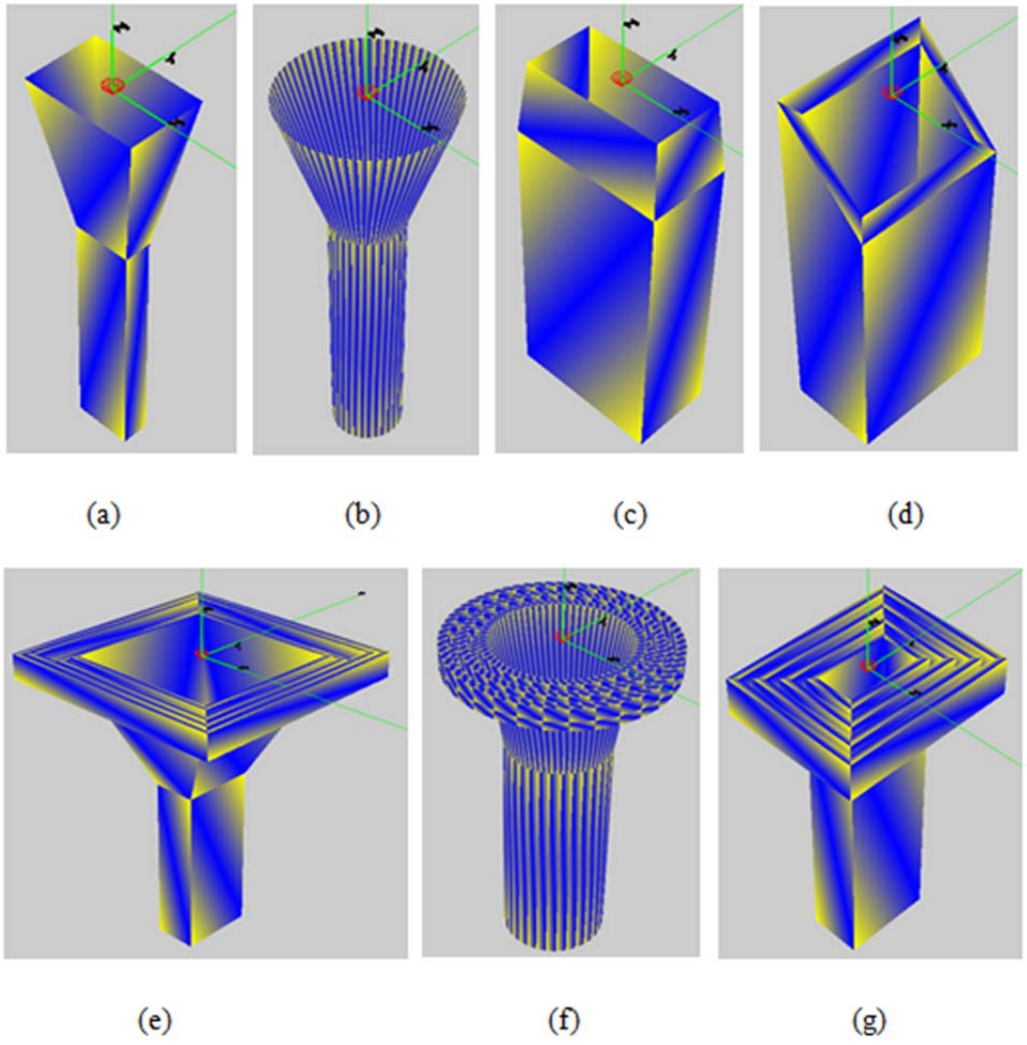


Figura 4.1.- Tipos de antenas bocina disponibles en el módulo de antenas. (a) Bocina rectangular. (b) Bocina cónica. (c) Guía de onda rectangular con afilamiento. (d) Guía de onda rectangular con apertura inclinada. (e) Bocina rectangular con corrugaciones. (f) Bocina cónica con corrugaciones. (g) Guía de onda rectangular con corrugaciones.

Todas las dimensiones físicas que definen la forma de estas estructuras se han parametrizado, lo que acelera el proceso de generación de las superficies que componen

las antenas, así como su visualización. Además, al disponer de una geometría parametrizada, se facilitan la edición y optimización, porque ambos procesos dependen directamente de los parámetros que caracterizan a las antenas (anchura, altura, número de corrugaciones, radio, abocinamiento, etc.).

Las bocinas mostradas en los apartados (e), (f) y (g) de la figura 4.1 se caracterizan por incluir corrugaciones alrededor de su apertura. Su inclusión en las bocinas convencionales implica una mejora de sus características de radiación. Por ejemplo, se ha demostrado que las sondas para realizar mediciones en campo cercano basadas en el diseño 4.1.g mejoran notablemente su pureza de polarización, tal y como puede comprobarse en [9]. Otra ventaja interesante para este tipo de aplicación es que gracias a las corrugaciones, los valores de RCS (*Radar Cross Section*) obtenidos son menores que los obtenidos en las bocinas tradicionales, lo que implica una disminución de errores de medición debido a la reducción del número de reflexiones múltiples y del acoplo entre la antena a medir y la propia sonda.

Diseñar una bocina corrugada, ya sea rectangular o cónica, no es un proceso complicado, puesto que todas las dimensiones que definen las corrugaciones también están parametrizadas. De esta forma, las corrugaciones son generadas a la vez que se crea el modelo de la bocina. Es posible establecer un número indeterminado de corrugaciones, cada una con una determinada profundidad y anchura. Por ejemplo, la figura 4.2 muestra una guía rectangular con corrugaciones de distinta altura y distinta anchura. Y en la figura 4.3 se pueden ver dos bocinas cónicas cuyas corrugaciones son iguales, pero varía su número.

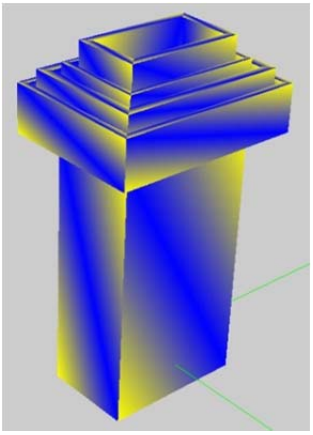


Figura 4.2.- Guía rectangular con corrugaciones de distintos tamaños.

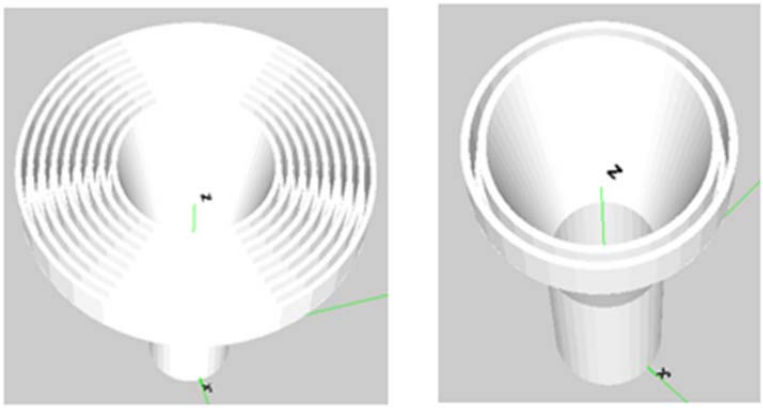


Figura 4.3.- Bocina cónica con distinto número de corrugaciones. (a) Nueve corrugaciones. (b) Dos corrugaciones.

En la literatura se pueden encontrar varios estudios del modelo de bocina cónica corrugada. Por ejemplo, el prototipo mostrado en 4.4.a se ha extraído del trabajo presentado en [10], donde se optimizan sus dimensiones para operar en la banda V, desde 50GHz hasta 75GHz. El modelo geométrico de dicha antena también se puede crear y optimizar con NewFasant, como se observa en la figura 4.4.b.

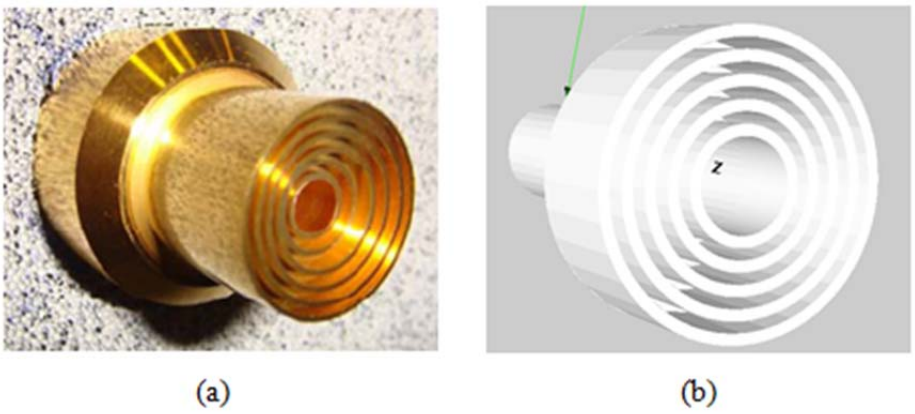


Figura 4.4.- Bocinas cónicas con corrugaciones. (a) Prototipo presentado en [10]. (b) Diseño de una bocina cónica corrugada similar al mostrado en (a) generado mediante el módulo de antenas proporcionado por la herramienta.

4.2.3.- Proceso de Diseño.

El diseño de las antenas de bocina se realiza en función de la frecuencia para conseguir que solo se propague el modo fundamental a lo largo de la guía. Para el caso

de las bocinas rectangulares, la frecuencia de corte del modo TE_{10} viene dada por la expresión (4.8).

$$fc(TE_{10}) = \frac{1}{2a\sqrt{\mu\varepsilon}} \tag{4.8}$$

donde *a* es el lado más largo de la apertura, μ es la permeabilidad magnética del vacío y ε es la permeabilidad eléctrica.

Teniendo en cuenta la expresión (4.8) y considerando una frecuencia de operación determinada, se pueden obtener las dimensiones de la guía que faciliten la propagación del modo fundamental. Una vez calculada la dimensión dada por el parámetro *a*, la dimensión del otro lado de la apertura se calcula sabiendo que la guía rectangular óptima tiene dimensiones a=2b, donde *b* hace referencia al tamaño del lado más corto.

Para el caso de las bocinas cónicas se tiene que la frecuencia de corte del modo TE_{11} es la siguiente:

$$fc(TE_{11}) = \frac{1.841}{2\pi a\sqrt{\mu\varepsilon}} \tag{4.9}$$

siendo *a* el radio de la guía.

Respecto al resto de las dimensiones, el tamaño longitudinal de las antenas de bocina, normalmente se establece con un número entero de longitudes de onda, mientras que el tamaño de la apertura, tanto cónica como rectangular, se determina en función de la directividad, dado que ésta aumenta conforme aumentan las dimensiones de la apertura equivalente.

La alimentación de las bocinas se realiza de forma automática cuando se genera el modelo geométrico, aunque posteriormente puede ser editada. Las bocinas rectangulares son adecuadas para sistemas con polarización lineal, mientras que las bocinas cónicas presentan un mejor comportamiento en sistemas con polarización

circular. Para conseguir que la polarización sea lineal, el punto de alimentación se modela mediante un dipolo eléctrico, orientado en el eje más corto de la guía y situado a una distancia de λ/4 de la parte inferior de la guía. Por otro lado, para proporcionar una polarización de tipo circular, se establecen dos dipolos cruzados perpendiculares situados a λ/4 del fondo de la guía.

La orientación de los dipolos se indica mediante una matriz de cosenos directores, la cual establece relaciones entre los ejes que definen la orientación del sistema antena y los ejes de los dipolos. La figura 4.5 muestra tres posibles orientaciones de un dipolo respecto al eje del sistema antena.

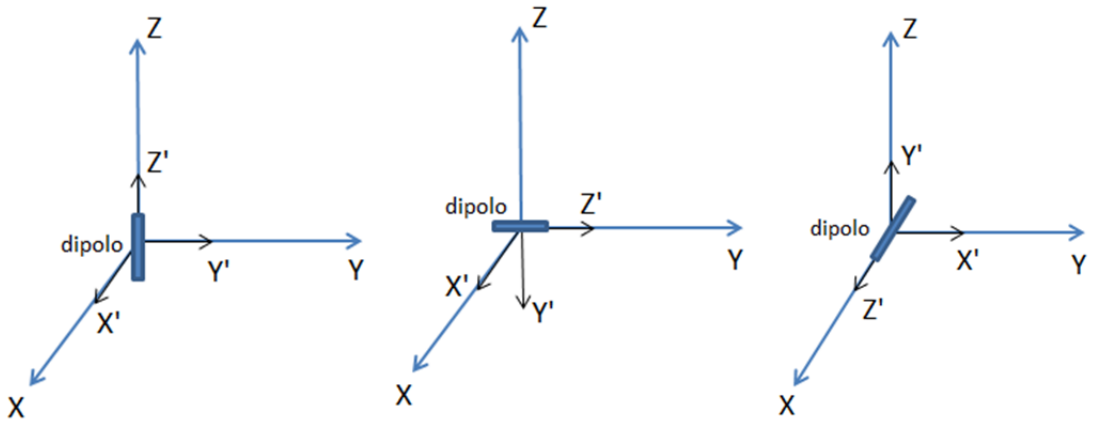


Figura 4.5.- Orientación del dipolo en el eje Z (izquierda). Orientación del dipolo en el eje Y (centro). Orientación en el eje X (derecha).

En la figura de la izquierda se muestra la orientación establecida por defecto, en el eje Z, en la que los ejes del dipolo coinciden con los de la antena (X'=X, Y'=Y, Z'=Z). Por tanto, la matriz de cosenos directores viene dada por la matriz unidad:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

En cuanto al esquema del centro, el dipolo se orienta según el eje Y, siguiendo las siguientes correspondencias: X'=X, Y'=-Z, Z'=Y. Su matriz de cosenos directores se muestra a continuación:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

Por último, la configuración de la derecha establece la orientación en el eje X. Se observa que cada uno de los ejes ha girado 90º, obteniéndose las relaciones X'=Y, Y'=Z, Z'=X. La matriz de cosenos directores queda del siguiente modo:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

A pesar de describir tres casos particulares en los que el dipolo se ha orientado según los tres ejes X, Y y Z, lo más común es que, tanto la orientación de los dipolos como la del sistema antena, sea totalmente arbitraria y no coincida con ningún eje de referencia.

La figura 4.6 muestra cómo se ha alimentado una bocina piramidal.

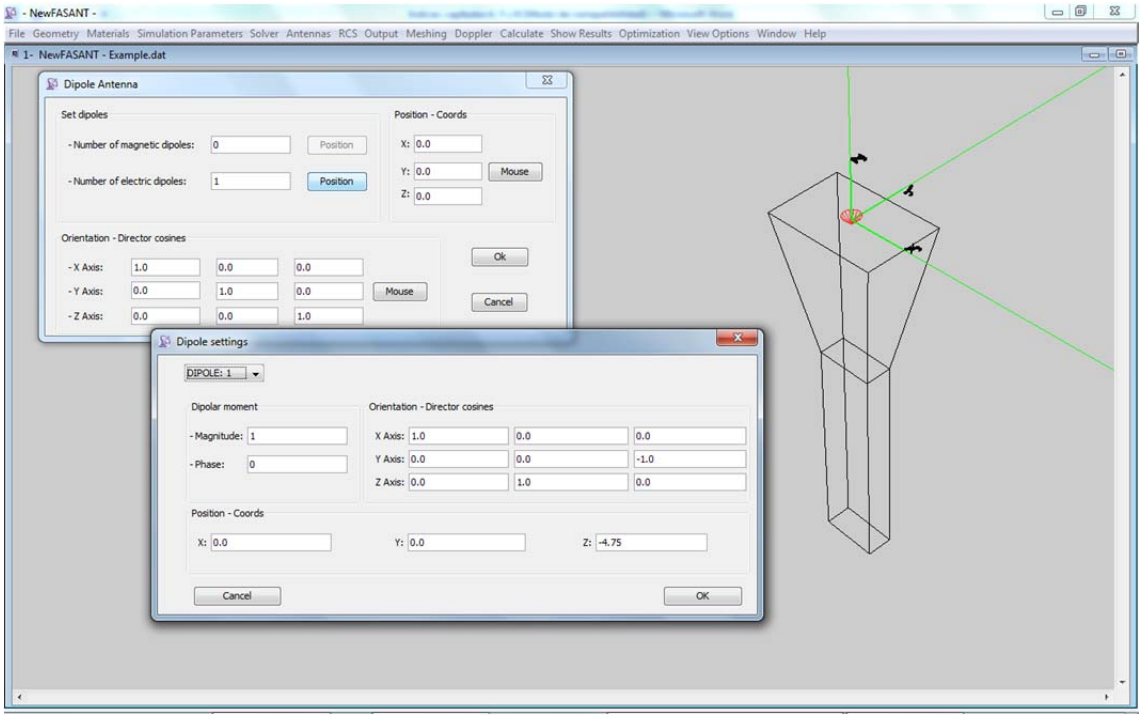


Figura 4.6.- Alimentación de las antenas mediante dipolos.

En dicha figura se puede ver que el sistema antena se coloca en el centro de fase, situado en la apertura de la bocina. Esta configuración es la que se establece por defecto, definiendo la posición del sistema antena en (0, 0, 0) con una matriz de orientación mediante cosenos directores que viene dada por la matriz identidad, indicando que los ejes del sistema antena coinciden con los ejes de coordenadas del sistema de referencia.

Por otro lado, el dipolo se configura estableciendo su momento dipolar con magnitud igual a uno y una fase nula. En la figura 4.6 se observa que la matriz de orientación del dipolo es la correspondiente a la dirección del eje Y.

Cuando el tipo de polarización es circular, se definen dos dipolos perpendiculares, como se puede ver en la figura 4.7. Cada uno de ellos se define con un momento dipolar de magnitud igual a 1. En cuanto a la fase, se establece una diferencia de 90º entre ambos.

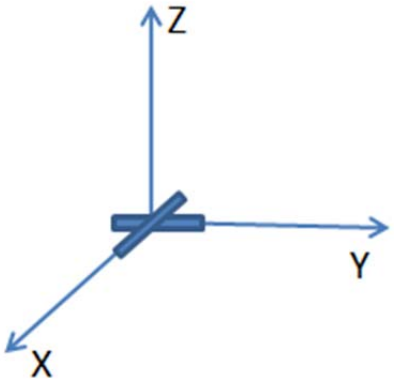


Figura 4.7.- Dos dipolos perpendiculares definidos cuando el tipo de polarización es de tipo circular.

4.3.- ANTENAS REFLECTORAS.

Las primeras aplicaciones de las antenas reflectoras se remontan a finales de los años 20, donde su uso era habitual en el campo de la astronomía. Sin embargo, su utilidad para transmitir y recibir información no se descubrió hasta la segunda guerra mundial, cuando empezaron a usarse para establecer comunicaciones a frecuencias de microondas y en sistemas radar.

Desde aquellos años hasta hoy, se han alcanzado numerosos avances tecnológicos en el ámbito de las comunicaciones por satélite [11, 12]. Algunos de ellos son el desarrollo de radiotelescopios [13, 14], de reflectores desplegables [15, 16], reconfigurables [17, 18], o incluso reflectores inflables [19, 20]. También se han usado extensamente en estaciones base para la comunicación con satélites geoestacionarios y en radioenlaces a frecuencias milimétricas.

Aparte de su sencillez de construcción, la gran ventaja de este tipo de antenas estriba en la concentración de radiación de un pequeño alimentador poco directivo, en un haz colimado de alta directividad tras reflejarse en la superficie de la antena. Tal y como se observa en la figura 4.8, la onda esférica generada por el alimentador se transforma en una onda plana, que da lugar a un diagrama de radiación muy directivo. Cuando la antena es receptora, se produce el efecto inverso, es decir, la onda plana entrante paralela al eje de la antena se refleja y concentra toda su potencia en el alimentador.

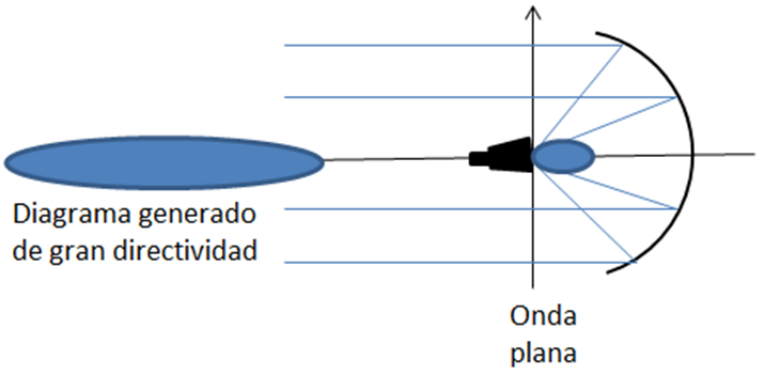


Figura 4.8.- Esquema de generación del diagrama de radiación de una antena reflectora.

El centro de fase del alimentador debe coincidir con el foco para que los campos presentes en la apertura estén en fase. Como elemento alimentador normalmente se usan bocinas o arrays de bocinas, pero también pueden utilizarse dipolos y guías abiertas, dependiendo del tipo de antena reflectora. Para conseguir la máxima eficiencia, la orientación del alimentador debe ser tal que su máximo de radiación apunte al vértice del reflector.

Las antenas reflectoras se clasifican en función del tipo de superficie que utilicen para reflejar las ondas recibidas o transmitidas. Éstas pueden ser planas, parabólicas, hiperbólicas y elípticas. Las más usadas son las parabólicas con simetría de revolución, que son las que se han incluido en el módulo de antenas para modelar los reflectores primarios. Por otro lado, las hiperbólicas y elípticas se utilizan como reflectores secundarios, debido a la propiedad que poseen de convertir las ondas esféricas incidentes en ondas esféricas salientes.

Uno de los inconvenientes de los reflectores con alimentador en el foco es que el alimentador obstruye parte de los rayos reflejados, produciendo una zona de sombra en el centro de la apertura. Este problema se soluciona con una configuración *offset*, en la que se desplaza la posición del alimentador, de forma que no interfiera en la trayectoria de los rayos reflejados. En este caso también hay que tener en cuenta que la orientación del alimentador debe establecerse de tal forma que su máximo de radiación siga apuntando al centro de la superficie reflectora, tal y como muestra el esquema de la figura 4.9.

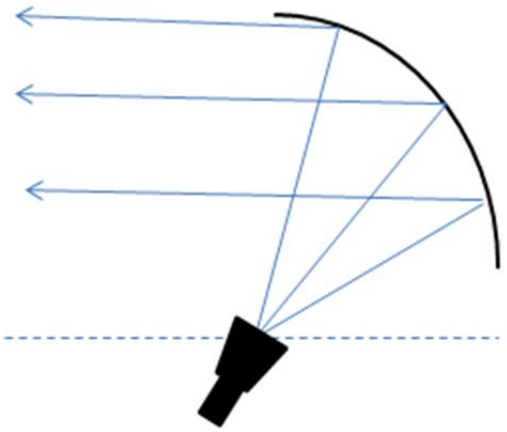


Figura 4.9.- Configuración *offset* de un reflector parabólico.

4.3.1.- Fundamentos Teóricos.

La superficie del reflector parabólico se puede obtener aplicando simetría de revolución a partir de la ecuación de la parábola. Las expresiones (4.10) y (4.11) muestran las ecuaciones que describen una curva parabólica, dadas en coordenadas cartesianas y en coordenadas esféricas, respectivamente.

$$x^2 + y^2 = 4fz \tag{4.10}$$

$$\rho = \frac{2f}{1 + \cos\theta} \tag{4.11}$$

En ambas ecuaciones se asume que el vértice del paraboloide se sitúa en el origen de coordenadas.

Respecto a la ganancia de la antena reflectora, teóricamente su valor viene dado por:

$$G = \left(\frac{4\pi D}{\lambda}\right)^2 \tag{4.12}$$

de donde se deduce que ésta es directamente proporcional a la frecuencia y al diámetro de la apertura. Sin embargo, las condiciones ideales que se aplican en la teoría no son válidas en la práctica, por lo que es necesario tener en cuenta el factor que representa la eficiencia de la antena, como se indica en (4.13):

$$G = \eta\left(\frac{4\pi D}{\lambda}\right)^2 \tag{4.13}$$

El valor típico de η se establece normalmente en 0.55 y depende, entre otros, de los siguientes factores:

- La eficiencia del alimentador. En la situación ideal, el alimentador emite un frente de onda esférico, algo que es bastante difícil de conseguir en la práctica.
- La obstrucción de radiación emitida debida al alimentador o a elementos estructurales de sujeción.
- La pérdida de potencia en los bordes del reflector, que se produce cuando el diagrama de radiación del alimentador presenta un lóbulo principal demasiado ancho.

- Rugosidad de la superficie reflectora. El material sobre el cual se produce la reflexión de las ondas, se considera totalmente liso, capaz de reflejar los rayos con una trayectoria paralela al eje del paraboloide.
- Existencia de polarización cruzada.

Este último factor está relacionado con el nivel de pureza de polarización que presente la bocina alimentadora. Lo ideal sería que las ondas emitidas por la bocina se reflejaran en la superficie reflectora de tal forma que todas tuvieran la misma polarización. Las ondas radiadas con polarización perpendicular a la deseada contribuirán a la existencia de lóbulos secundarios en el diagrama de radiación de la antena.

4.3.2.- Tipos de Antenas Reflectoras Implementadas.

Los tres tipos de antenas reflectoras incluidas en la herramienta son los reflectores primarios y los reflectores dobles Cassegrain y Gregorian. Como se mencionó anteriormente, su configuración puede ser centrada o descentrada (considerando un cierto *offset*), para evitar la obstrucción causada por el alimentador en las ondas reflejadas.

- Reflectores Simples.

La figura 4.10 muestra dos tipos de reflectores simples creados mediante el módulo de antenas.

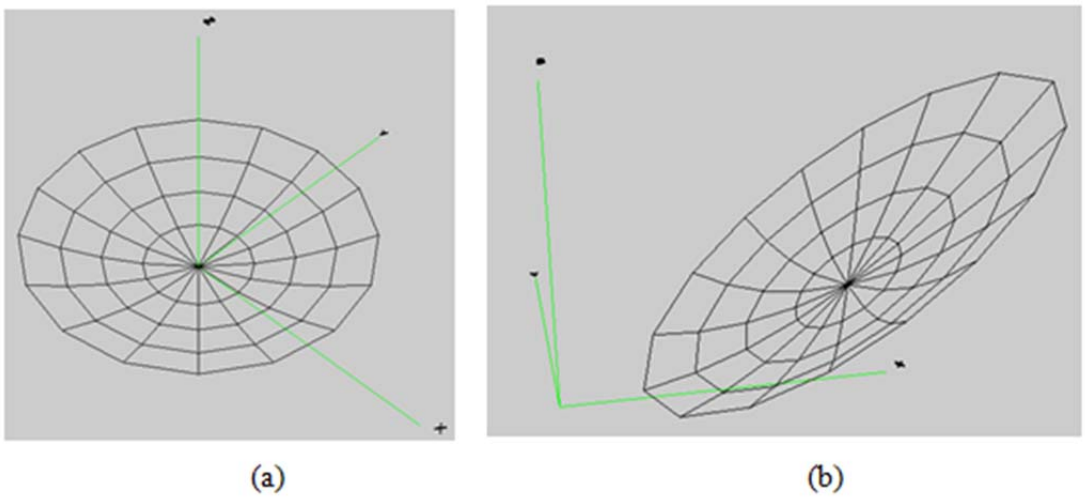


Figura 4.10.- Reflectores primarios. (a) Centrado. (b) Descentrado.

Para establecer la alimentación de ambas antenas, se ofrecen varias posibilidades:

- Incluir en el diseño una bocina rectangular o cónica.
- Utilizar un modelo eléctrico de bocina.
- Modelar el comportamiento de la bocina mediante un diagrama de radiación externo.

Si la alimentación se realiza mediante la incorporación al modelo geométrico de una bocina física, también se ofrece la opción de poder elegir el tipo de polarización. La figura 4.11 muestra dos ejemplos de reflectores alimentados mediante dicha opción. La bocina incluida en 4.11.a es de tipo piramidal, mientras que la mostrada en 4.11.b es cónica.

En ambos casos, reflector centrado o descentrado, los dipolos que propagan el modo fundamental dentro de cada bocina se definen de forma transparente, tal y como se indicó en la sección anterior. La configuración de los dipolos, así como la orientación de la bocina se fijan de tal forma que el máximo de radiación de las bocinas siempre apunte al centro de la superficie reflectora. En la figura 4.11 se aprecia que el centro de fase de la antena se sitúa en la apertura de las bocinas. Además, se puede comprobar que la orientación de los ejes del sistema antena se ajusta en función de la posición del reflector.

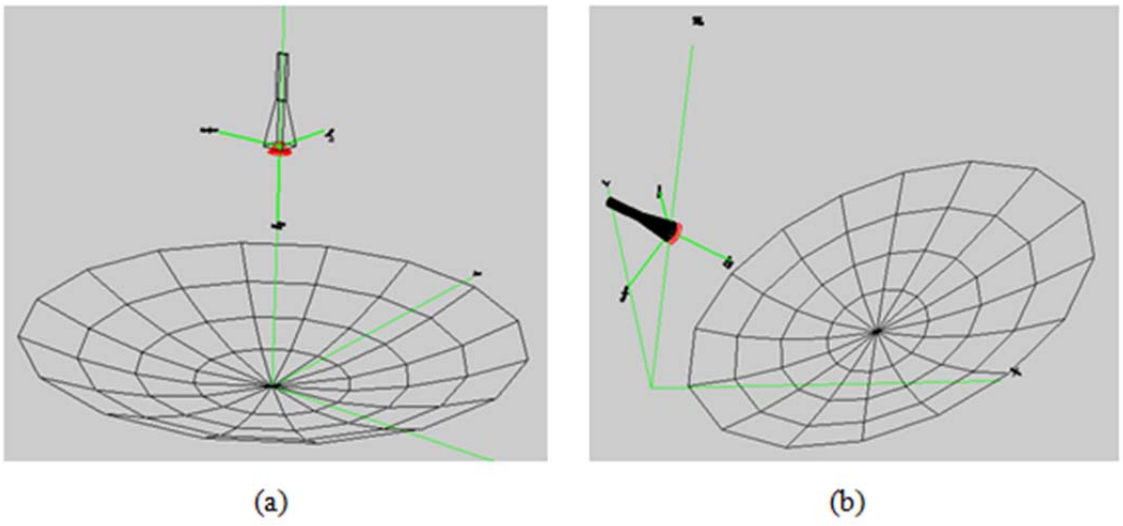


Figura 4.11.- Alimentación de un reflector parabólico en NewFasant.

Las dimensiones de las bocinas alimentadoras también se calculan en función de la frecuencia, de la distancia focal y del diámetro del reflector. El tamaño de las guías de onda (rectangular o cónica) se obtiene mediante las expresiones (4.8) y (4.9), respectivamente, considerando la frecuencia de corte que define el modo fundamental. La longitud de las guías se fija en tres longitudes de onda. Por último, las dimensiones que definen la apertura (rectangular o circular), se establecen a partir de la expresión del área efectiva y teniendo en cuenta la directividad de la bocina, su diagrama de radiación, el nivel de caída en los bordes del reflector (nivel de *taper)*, etc.

Para calcular el área efectiva de la bocina, se utiliza la siguiente expresión:

$$A_{ef} = \frac{D\lambda^2}{4\pi} \tag{4.14}$$

donde la longitud de onda es conocida y la directividad se puede obtener a partir de la ganancia, estableciendo un valor de eficiencia estimado para la bocina del 50%, como se indica en (4.15).

$$D = \frac{G}{\eta_b} = \frac{G}{0.5} \tag{4.15}$$

Por otro lado, el diagrama de radiación de potencia de la bocina se puede aproximar por:

$$t(\theta) = \cos(\theta)^n \tag{4.16}$$

Y el valor de *n* se puede obtener del siguiente modo:

$$n = \frac{10\log(t(\theta))}{10\log(\cos\theta)} = \frac{t(\theta)(dB)}{10\log(\cos\theta)} \tag{4.17}$$

Considerando un nivel de caída de -13dB en el borde del reflector, se tiene que el valor de *n* es el indicado en la ecuación (4.18).

$$n = \frac{-13}{10\log(\cos\theta)} \tag{4.18}$$

Sabiendo que el valor de la ganancia para alimentadores tipo $\cos(\theta)^n$ es:

$$G(\theta) = \begin{cases} 2(n+1)\cos(\theta)^n & si\ 0 < \theta < \pi/2 \\ 0 & si\ \theta > \pi/2 \end{cases} \tag{4.19}$$

De (4.15) se deduce que la directividad de la bocina es:

$$D = \frac{2(n+1)\cos(\theta)^n}{\eta_b} \tag{4.20}$$

Sustituyendo (4.20) en (4.14) se obtiene la expresión que indica el valor del área efectiva que tiene que tener la apertura de la bocina para alimentar correctamente al reflector.

$$A_{ef} = \frac{\lambda^2 2(n+1)\cos\theta}{4\pi\eta_b} \tag{4.21}$$

donde todos los parámetros son conocidos. El parámetro θ representa el ancho de haz del lóbulo principal del diagrama de radiación de la bocina.

Dependiendo del tipo de bocina seleccionada, se hallará el radio de la bocina cónica o el lado de la bocina piramidal a partir del valor del área efectiva.

Respecto a la generación de la superficie parabólica, existen dos posibilidades de diseño. Por un lado, los reflectores se pueden crear a partir de sus dimensiones físicas (diámetro *D* y distancia focal *f*), y por otro lado, se puede especificar el nivel de lóbulo principal a secundario, la directividad y la relación *f/D*. A partir de todos estos parámetros, el sistema proporciona las dimensiones que debe tener el reflector para cumplir con dichas especificaciones.

Para llevar a cabo los cálculos, se han utilizado dos gráficas proporcionadas por Ticra [21], en un curso impartido en el *APS/URSI Symposium* del 2009. De la figura

4.12 se obtiene la relación entre el nivel de lóbulo principal a secundario (*SLL*) y el nivel de caída en el borde del reflector (nivel de *taper*). Por otra parte, de la figura 4.13 se deduce el valor de eficiencia de la antena reflectora a partir del nivel de *taper* obtenido de la gráfica 4.12.

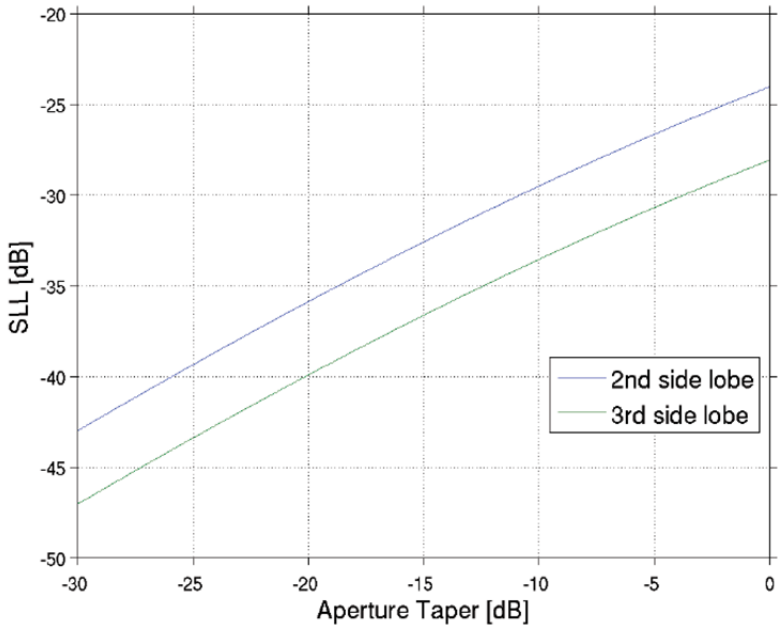


Figura 4.12.- Relación entre el nivel de lóbulo principal a secundario del diagrama de radiación del reflector en función del nivel de *taper*.

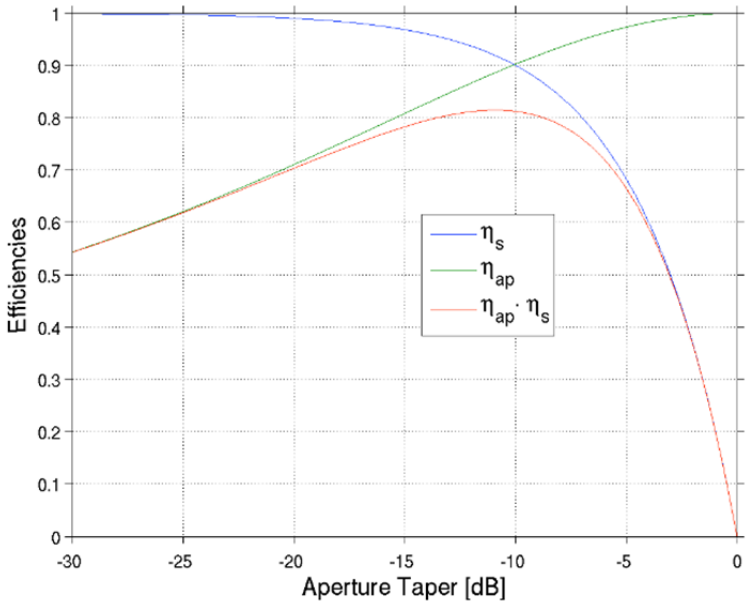


Figura 4.13.- Relación entre la eficiencia de la antena reflectora y el nivel de *taper*.

Teniendo en cuenta la gráfica mostrada en la figura 4.12, se puede obtener una expresión aproximada que permita obtener el nivel de *taper* a partir del nivel de lóbulo principal a secundario. Dicha expresión se indica en (4.22).

$$taper(dB) = (SLL + 43)\frac{10}{7} - 30 \tag{4.22}$$

Dependiendo del resultado obtenido en (4.22), si el nivel de *taper* es menor de -10dB, la eficiencia del reflector, según indica la gráfica de la figura 4.13, se aproxima mediante la recta dada por (4.23) y si es mayor o igual, se aproxima mediante la recta dada en (4.24).

$$\eta = 0.55 + (taper + 30)\frac{0.08}{5} \tag{4.23}$$

$$\eta = 0.82 + (taper + 10)\frac{0.42}{5} \tag{4.24}$$

Una vez obtenida la eficiencia de la antena, a partir de la expresión (4.13) se obtiene que el diámetro de la superficie reflectora viene dado por (4.25).

$$D = \frac{\lambda\sqrt{G/\eta}}{4\pi} \tag{4.25}$$

La distancia focal se obtiene a partir del valor del diámetro, puesto que uno de los parámetros iniciales era la relación *f/D*.

- Reflectores Dobles.

Las antenas de doble reflector están formadas por dos superficies reflectoras: una principal de forma parabólica y otra secundaria, que puede tener forma hiperbólica o elíptica, según se trate de reflectores Cassegrain o Gregorian respectivamente.

En el caso de los reflectores Cassegrain, uno de los focos del hiperboloide coincide con el del paraboloide, mientras que el segundo foco se utiliza para indicar la

posición del centro de fase del alimentador. El hiperboloide se orienta de tal forma que su parte convexa quede enfrentada al reflector primario. De este modo, la radiación emitida por el alimentador se refleja en la superficie convexa del subreflector, reflejándose de nuevo en el reflector principal para finalmente formar un frente de onda plano paralelo al eje de la antena. El esquema de la figura 4.14 resume la configuración descrita.

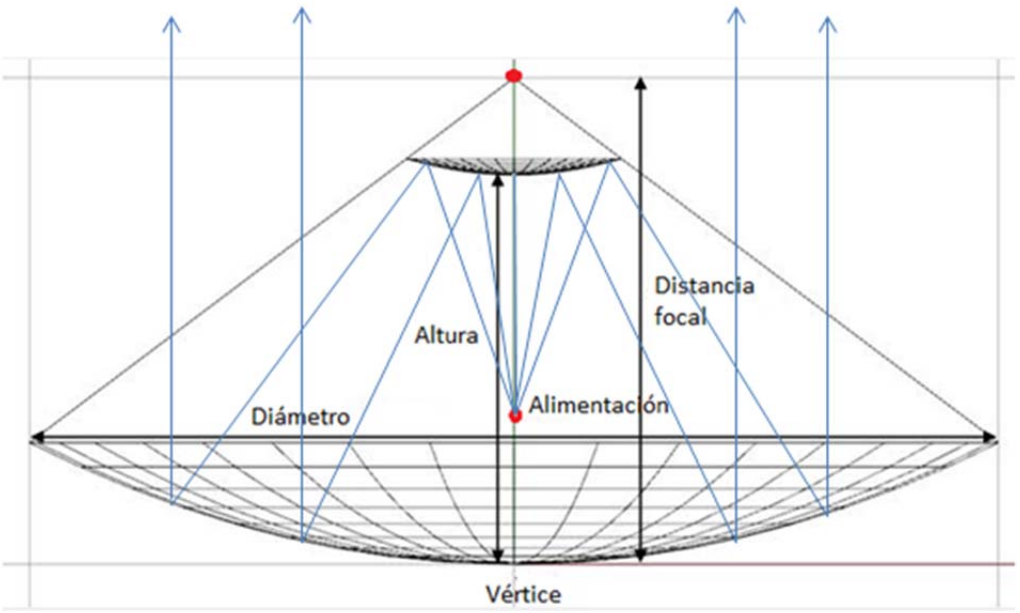


Figura 4.14.- Esquema del reflector Cassegrain.

Los dos focos del hiperboloide se han destacado con dos puntos en color rojo. El que está situado en la parte superior es común a ambos reflectores y representa la distancia focal. Según la figura 4.14, en el foco de la parte inferior se sitúa el alimentador, el cual se comporta de forma análoga al caso de los reflectores simples.

La superficie del hiperboloide viene dada por las siguientes expresiones:

$$z = a\left(\sqrt{1+\left(\frac{x}{b}\right)^2}-1\right) \tag{4.26}$$

$$b = \sqrt{f^2 - a^2} \tag{4.27}$$

donde *a* representa la distancia entre los vértices de ambos reflectores y *f* es la distancia entre el foco común y el vértice del reflector principal, indicados por los parámetros "altura" y "distancia focal" respectivamente, en la figura 4.14.

Respecto a las aplicaciones de los reflectores Cassegrain, se utilizan ampliamente en radioastronomía y para establecer comunicaciones espaciales. La figura 4.15 muestra dos fotografías del radiotelescopio del Centro Astronómico de Yebes [22], cuya configuración es de tipo Cassegrain. El reflector principal posee un diámetro de 40 metros.

Figura 4.15.- Radiotelescopio Aries XXI del Centro Astronómico de Yebes (Guadalajara).

Su ganancia se calcula del mismo modo que los reflectores simples, según la ecuación (4.13). Sin embargo, la eficiencia de este tipo de reflectores es mayor, alrededor del 70% [23].

Al igual que ocurría con los reflectores simples, el diseño de reflectores dobles también se puede llevar a cabo introduciendo un *offset* para evitar zonas de sombra causadas por el bloqueo del alimentador. Las figuras 4.16 y 4.17 muestran los modelos geométricos de las dos configuraciones posibles, con y sin *offset*, creados con el módulo de antenas que incluye el sistema.

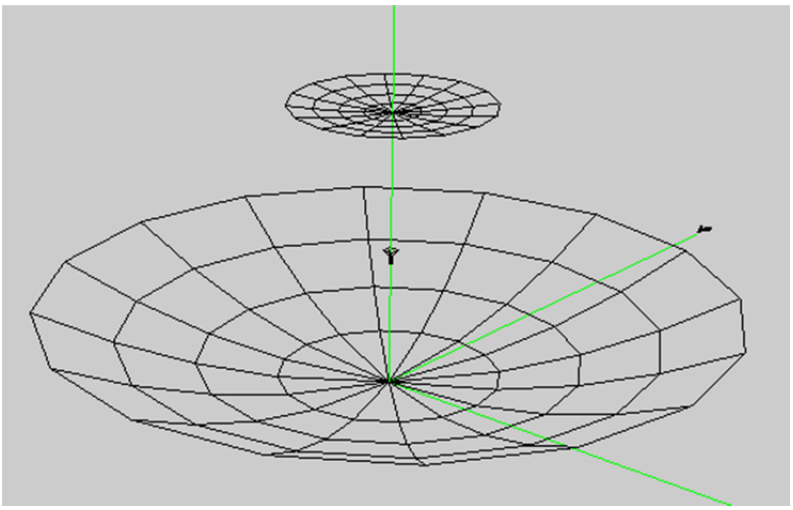


Figura 4.16.- Creación de un reflector Cassegrain centrado.

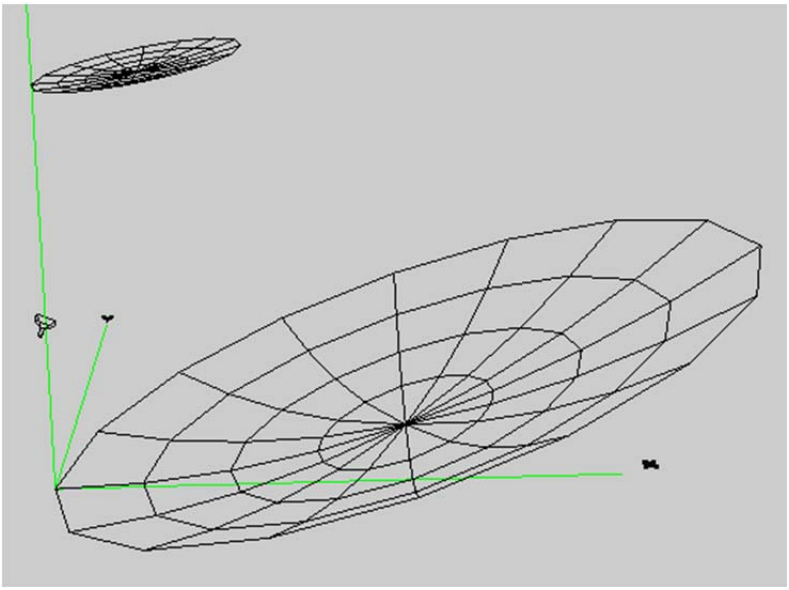


Figura 4.17.- Creación de un reflector Cassegrain con *offset*.

Si en vez de usar un subreflector hiperbólico se usa uno de tipo elíptico, se obtiene el diseño del reflector Gregorian, cuyo esquema se muestra en la figura 4.18. Este sistema es menos usado que el reflector Cassegrain porque necesita soportes más largos y porque produce mayor bloqueo de apertura. Los rayos emitidos desde el alimentador se reflejan en la parte cóncava del elipsoide para volver a reflejarse posteriormente en el reflector principal.

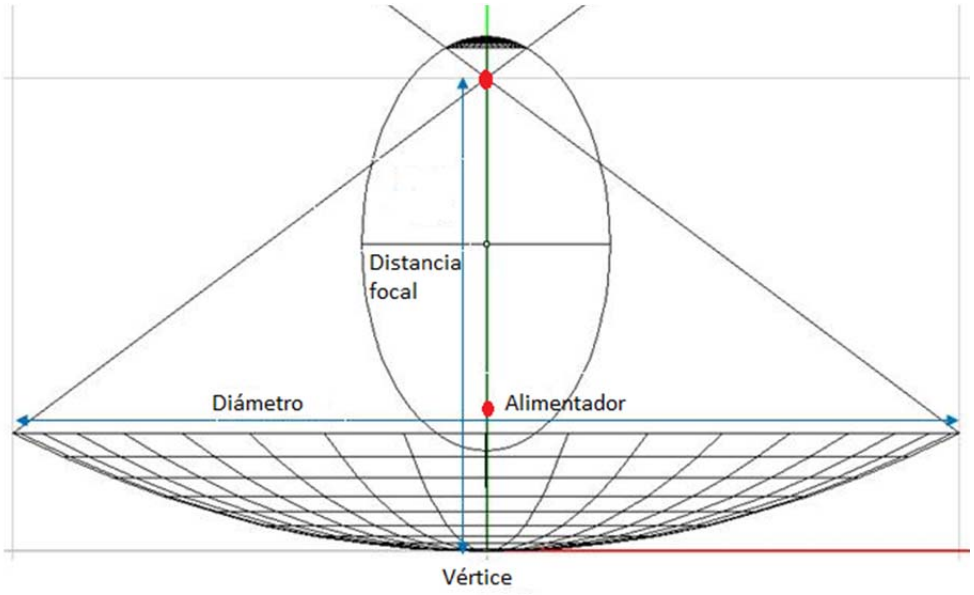


Figura 4.18.- Esquema del reflector Gregorian.

En este caso, el alimentador se sitúa en el foco inferior del elipsoide. El foco superior sirve para establecer el diámetro del subreflector, el cual se define a partir de las líneas que unen los extremos del reflector principal y el foco, según se indica en la figura 4.18.

De forma análoga al reflector simple y al Cassegrain, la herramienta permite establecer varios tipos de alimentación y, en caso de incorporar una bocina al modelo, también ofrece la posibilidad de definir el tipo de polarización. Las versiones con y sin *offset* de esta antena se muestran en las figuras 4.19 y 4.20 respectivamente.

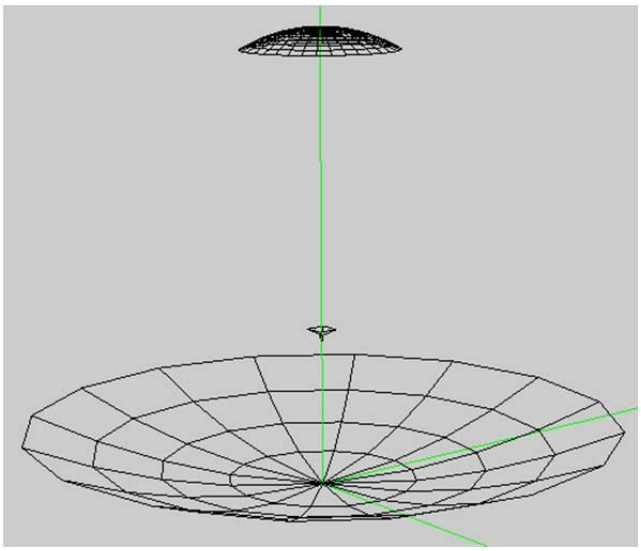


Figura 4.19.- Reflector Gregorian centrado.

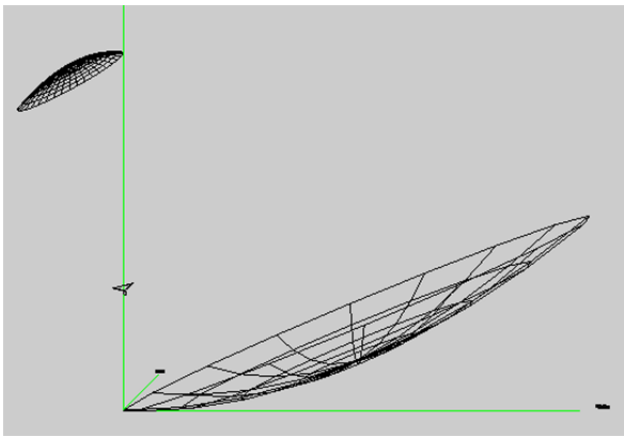


Figura 4.20.- Reflector Gregorian con *offset*.

La figura 4.21 muestra una fotografía del reflector Gregorian de Arecibo (Puerto Rico). Se trata de la antena reflectora más grande del planeta, con un diámetro de 305 metros, compuesto por 40.000 paneles de aluminio. La superficie reflectora no es parabólica, sino esférica. En la figura 4.22 se observa el subreflector con forma de elipsoide que refleja los rayos hacia el reflector principal.

Figura 4.21.- Vista aérea del reflector Gregorian de Arecibo.

Figura 4.22.- Detalle del elipsoide que actúa como subreflector.

4.4.- ANTENAS DE HILO.

Las antenas de hilo [24] se clasifican dentro del grupo de antenas de banda ancha, cuya característica principal es que son capaces de cubrir un gran rango de frecuencias manteniendo el valor de alguno de sus parámetros. El interés por este tipo de antenas se originó en los años 40, debido a la necesidad de minimizar el número de antenas embarcadas en aeronaves. Las primeras antenas de banda ancha fueron las antenas de hilo largo o antenas de onda progresiva, las cuales presentan una gran longitud eléctrica para aumentar su ganancia [25, 26]. Las antenas en "V" y en forma de rombo son dos ejemplos de antenas de onda progresiva. El uso de estas dos antenas está bastante extendido, sobre todo en frecuencias de HF y VHF (desde 3MHz hasta 300MHz).

Entre las ventajas de las antenas de hilo destacan su sencillez y facilidad de construcción. Además, son muy económicas y de bajo peso y presentan bajas pérdidas en la línea de alimentación. Se usan en aplicaciones de banda ancha y para realizar análisis de compatibilidad electromagnética [27]. Dentro de este grupo de antenas se encuentran las antenas de array de dipolos, las antenas de hélice, las antenas bicónicas y las antenas log-espiral y log-periódica.

4.4.1.- Fundamentos Teóricos.

Las antenas de hilo están formadas por elementos radiantes en forma de hilo cuya sección es despreciable respecto a la longitud de onda. La distribución de corriente sobre su superficie es de tipo filiforme con componente longitudinal a lo largo de los hilos, como se puede ver en la ecuación (4.28).

$$\vec{A} = \frac{\mu}{4\pi}\frac{e^{-jkr}}{r}\iint \overrightarrow{J_s}\, e^{jk\overrightarrow{r}\hat{r}}\, dS \approx \frac{\mu}{4\pi}\frac{e^{-jkr}}{r}\int I(u)\, e^{jk\overrightarrow{r_u}\hat{r}}\hat{u}\, du \qquad (4.28)$$

Estas antenas se caracterizan por mantener el valor de alguno de sus parámetros (impedancia, directividad, dirección del haz principal, etc.) constante o casi constantes en un rango amplio de frecuencias (alrededor de dos o más octavas). Por otro lado, dada

una longitud física determinada de la antena, cuanto mayor sea la frecuencia de trabajo, mayor será su ganancia.

4.4.2.- Tipos de antenas de hilo implementadas.

Dentro del módulo de antenas de NewFasant se han implementado cuatro tipos de antenas de hilo: antena de hélice, antena espiral (circular o rectangular), antena de hilo en zig-zag log-periódica y antena de hilo de forma arbitraria. Las figuras 4.23-4.26 muestran un ejemplo de cada antena. En los siguientes sub-apartados se exponen sus características más relevantes.

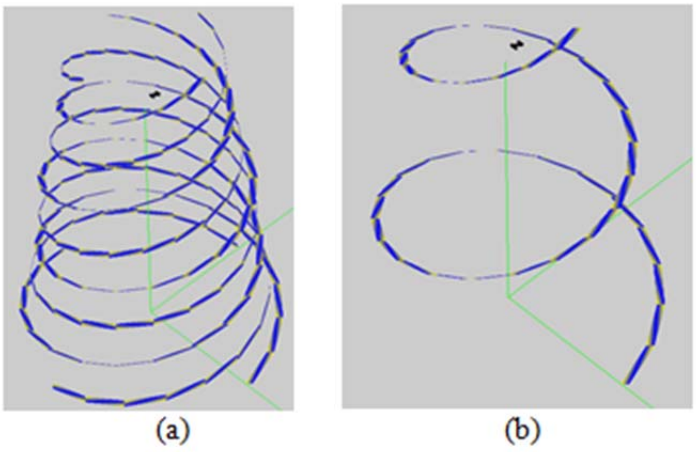


Figura 4.23.- Antena de hilo helicoidal. (a) Hélice cuadrifilar. (b) Hélice unifilar.

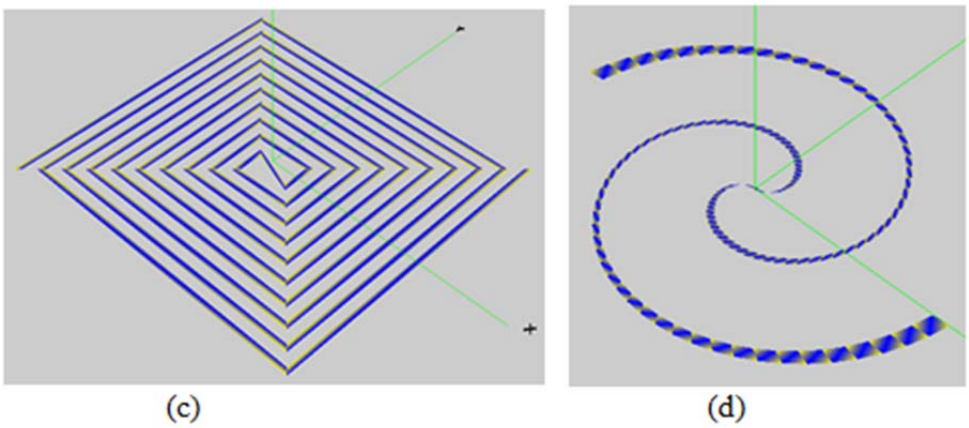


Figura 4.24.- Antenas de hilo espirales. (a) Espiral rectangular. (b) Espiral circular.

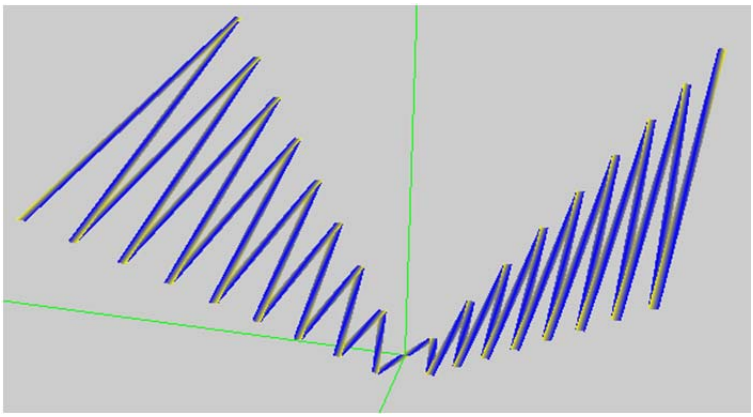


Figura 4.25.- Antena de hilo zig-zag log-periódica.

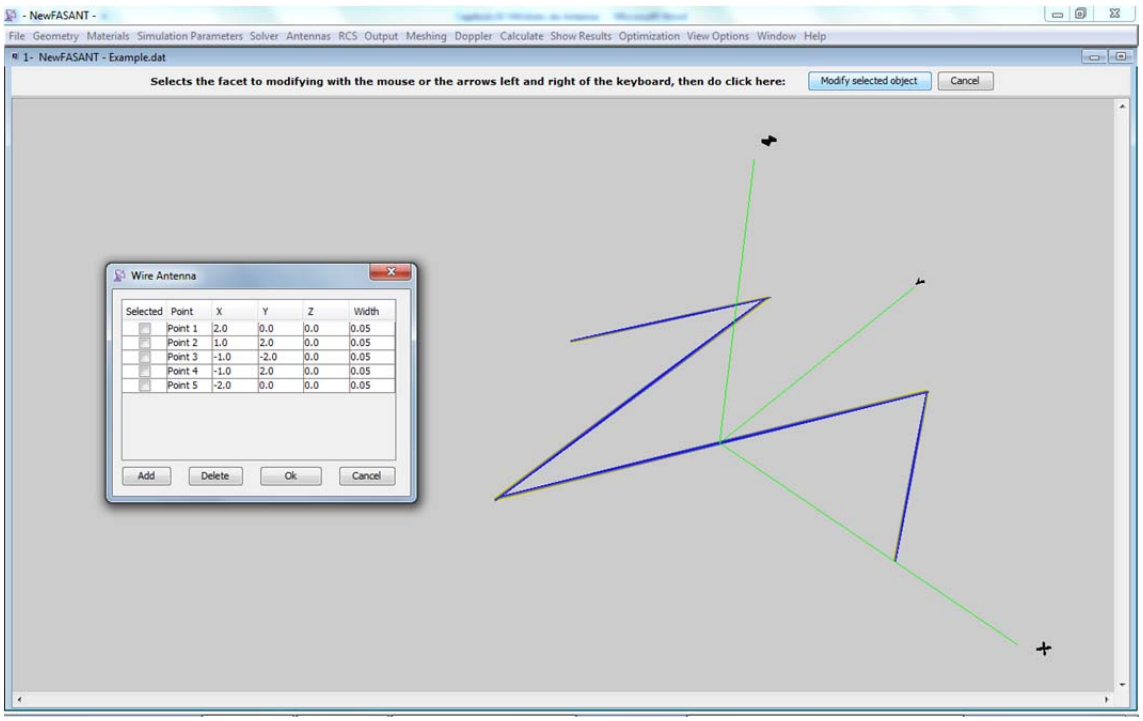


Figura 4.26.- Antena de hilo de forma arbitraria.

- Antena de Hélice.

Las antenas de hélice se utilizan en los sistemas de recepción de satélites debido a sus buenas propiedades de radiación y polarización circular [28, 29]. Son compactas y ligeras y suelen agruparse formando arrays circulares.

El tipo de hélice implementada en el módulo de antenas de la herramienta puede ser de tipo cilíndrica o cónica, dependiendo de los valores especificados en los radios superior e inferior que definen la hélice. También se puede especificar el número de

hélices que forman parte de la antena. De esta forma, se pueden crear hélices unifilares o arrays circulares de varias hélices.

Las coordenadas cartesianas de los puntos que definen la curvatura de la hélice se obtienen mediante las siguientes ecuaciones:

$$z = \varphi \frac{h}{2\pi n} \tag{4.29}$$

$$x = R_{inf} + \frac{z}{m}\cos\varphi \tag{4.30}$$

$$y = R_{inf} + \frac{z}{m}\sin\varphi \tag{4.31}$$

$$m = \frac{-h}{R_{inf} - R_{sup}} \tag{4.32}$$

donde φ hace un barrido desde 0º hasta 360º, R_{inf} es el radio inferior de la hélice, R_{sup} es el radio superior de la hélice, h es la altura y n es el número de vueltas.

Un tipo de hélice muy utilizado en los últimos años en aplicaciones de monitorización y control espacial desde estaciones terrestres es la hélice cuadrifilar. Sus propiedades principales son ganancia media-baja, cobertura hemisférica y polarización circular. Entre sus ventajas destacan su tamaño compacto, comportamiento resonante y capacidad de funcionamiento en varias frecuencias. En el capítulo 7 se describe en detalle el proceso de optimización y análisis de la hélice cuadrifilar.

- Antena Espiral.

Las antenas espirales pueden ser de dos tipos: circular o rectangular. Ambas se diseñan en tecnología impresa sobre un sustrato de dieléctrico y suelen radiar con polarización circular. El sentido de giro de la espiral indica si es a derechas o a

izquierdas. Las ecuaciones de la espiral circular bidimensional vienen dadas por las siguientes expresiones:

$$x = R\cos\varphi\frac{\varphi}{2\pi n} \tag{4.33}$$

$$y = R\sin\varphi\frac{\varphi}{2\pi n} \tag{4.34}$$

$$z = 0 \tag{4.35}$$

donde φ hace un barrido desde 0º hasta 360º, R es el radio de una vuelta y n es el número de vueltas de la espiral.

Por otro lado, la espiral rectangular se define en función de las dimensiones externas de la espiral, del número de vueltas y de la anchura de la tira. También se ofrece la posibilidad de generar esta antena en tres dimensiones estableciendo un parámetro adicional de altura, al igual que en el caso anterior.

Tanto la espiral circular como la rectangular se alimentan en el punto central donde se unen los brazos que forman la espiral. En la figura 4.27 se muestra cómo se crea una espiral rectangular tridimensional, en la que se ha establecido un punto de alimentación, según se indica en la parte inferior izquierda de la imagen. El voltaje especificado se aplica sobre la zona central, como se puede apreciar en la figura 4.28, en la que se muestran los valores establecidos al crear la antena. Dichos valores son editables, e incluso se pueden definir nuevos puntos de alimentación.

- Antena zig-zag log-periódica.

Se trata de una antena balanceada compuesta por dos brazos que describen una trayectoria en zig-zag. Su funcionamiento es independiente de la frecuencia y se utiliza para aplicaciones de vigilancia radar [30].

La alimentación de esta antena se realiza de forma similar a las anteriores, estableciendo un punto de alimentación entre las dos superficies centrales donde se unen las dos ramas.

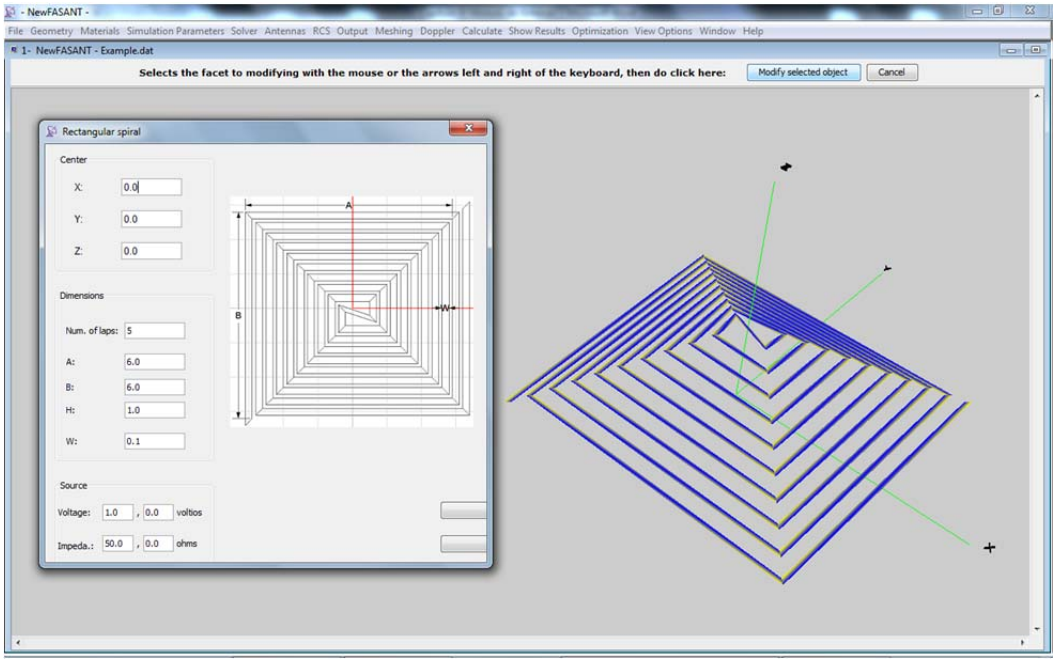


Figura 4.27.- Creación de una espiral rectangular.

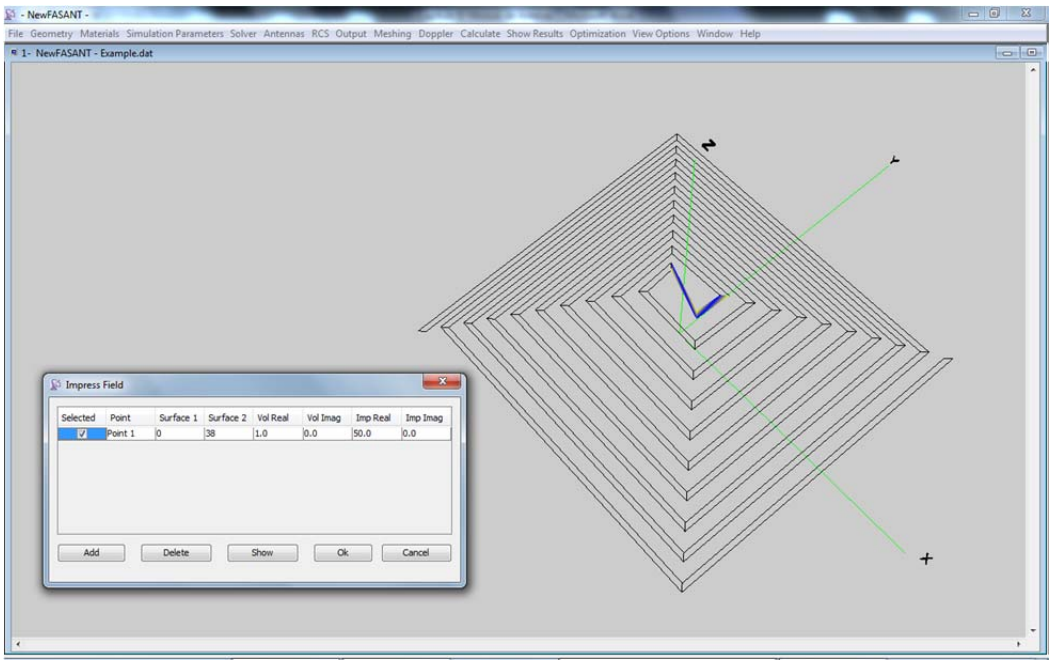


Figura 4.28.- Visualización de los parámetros del punto de alimentación de la antena espiral rectangular.

- Antena de Hilo de Forma Arbitraria.

Como se puede ver en la figura 4.26, esta antena se crea en función de una serie de puntos dados por sus coordenadas cartesianas *x, y, z* y por la anchura de la antena en cada punto. Proporciona mucha flexibilidad en la creación de modelos, pues permite realizar diseños tridimensionales con formas totalmente arbitrarias. La figura 4.29 muestra un ejemplo en el que se ha diseñado un elemento mediante esta opción y posteriormente se ha copiado varias veces hasta obtener el diseño mostrado en la imagen.

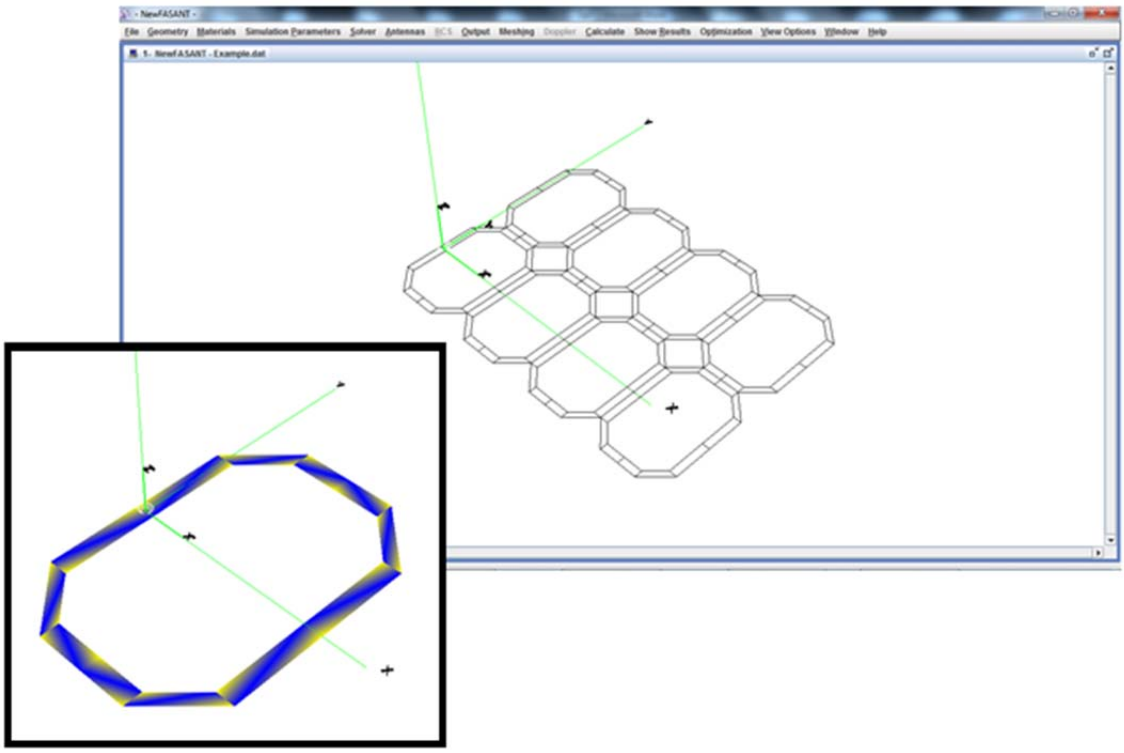


Figura 4.29.- Ejemplo de creación de una antena de hilo de forma arbitraria.

4.5.- LENTE DE ROTMAN.

Una lente de Rotman [31] es una antena construida mediante técnicas microstrip que alimenta a un array lineal de antenas y que ofrece una alternativa muy interesante para varias aplicaciones. Por ejemplo, es buena candidata para ser usada en redes de conformación de haces en aplicaciones espaciales. También puede operar a altas frecuencias, lo que la convierte en buena solución para aplicaciones milimétricas en el ámbito de la automoción. De hecho, los sistemas radar basados en ondas milimétricas se

están empezando a incluir en los sistemas de sensores de la próxima generación de automóviles. Estas aplicaciones incluyen control adaptativo de crucero, ayuda en el aparcamiento, y sensores en los puntos muertos, así como sistemas que ayudan a evitar colisiones. Otra de las aplicaciones de la lente es como receptor o detector de señales, de modo que comparando la potencia de la señal recibida en los puertos de entrada, se puede detectar la dirección de llegada de la señal. Entre sus ventajas, destacan las siguientes: proporciona una alta ganancia, amplios ángulos de escaneo, geometría conformada, fácil de fabricar, bajo peso y bajo coste. Además, la lente puede proporcionar muchos haces estrechos de gran ganancia y, combinando haces adyacentes, se pueden obtener haces más amplios con una ganancia un poco menor. Con un buen diseño se pueden obtener ángulos de escaneo de 60º y ganancias medias/altas.

Los diagramas de radiación de múltiples haces se obtienen debido al propio diseño de la lente. Dependiendo de por qué puerto de entrada se introduzca la señal, se obtendrá una respuesta diferente en los puertos de salida.

4.5.1.- Fundamentos Teóricos.

Una lente de Rotman se compone de una región plana con una serie de puertos de entrada y una serie de puertos de salida distribuidos en contornos opuestos. Los elementos radiantes se sitúan sobre un array lineal perpendicular al eje de la lente. El puerto del haz central proporciona trayectos iguales para todos los elementos del array, mientras que un puerto con offset produce una diferencia de caminos y, por tanto, una diferencia de fases a lo largo del array, dado por un haz de dirección.

El diseño de la lente se basa en la geometría mostrada en la figura 4.30, donde se identifican los siguientes parámetros: F_0, F_1 y F_2 son los tres focos que definen el arco circular de la parte izquierda de la lente, en los que no hay errores de fase, Ψ es el ángulo que indica la dirección de máxima radiación, y_3 es la posición de cada elemento radiante del array y ω es la longitud del cable TEM que conecta los elementos radiantes con los puertos del array. Los parámetros que definen la forma de la lente son el ángulo focal α, la relación dada por f_2/f_1 y el factor $\gamma = \sin\psi/\sin\alpha$.

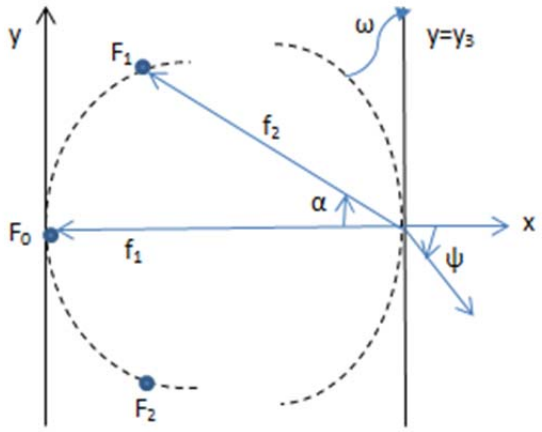


Figura 4.30.- Topología de la lente.

La longitud de cada cable, normalizada con respecto a la distancia focal $W = \omega/f_1$, que conecta a los elementos del array situados en $y=y_3$ satisface la siguiente ecuación:

$$AW^2 + BW + C = 0 \tag{4.36}$$

donde los coeficientes *A, B* y *C* vienen dados por:

$$A = 1 - \frac{(1-\beta)^2}{(1-\beta\cos\alpha)^2} - \frac{\xi^2}{\beta^2} \tag{4.37}$$

$$B = -2 + \frac{2\xi^2}{\beta} + \frac{2(1-\beta)}{1-\beta\cos\alpha} - \frac{\xi^2 \sin\alpha^2\,(1-\beta)}{(1-\beta\cos\alpha)^2} \tag{4.38}$$

$$C = -\xi^2 + \frac{\xi^2 \sin\alpha^2}{1-\beta\cos\alpha} - \frac{\xi^4 \sin\alpha^4}{4(1-\beta\cos\alpha)^2} \tag{4.39}$$

y donde $\xi = y_3\gamma/f_1$.

El centro de cada uno de los puertos que se conectan con los elementos del array se obtiene a partir de (4.40) y (4.41).

$$X_2 = \frac{x_2}{f_1} = 1 - \frac{\frac{1}{2}\xi^2 \sin\alpha^2 + (1-\beta)W}{1 - \beta\cos\alpha} \tag{4.40}$$

$$Y_2 = \frac{y_2}{f_1} = \xi\left(1 - \frac{W}{\beta}\right) \tag{4.41}$$

Por otro lado, las expresiones que determinan el centro de fase de los puertos de alimentación vienen dadas por:

$$X_1 = \frac{x_1}{f_1} = \rho[1 - \cos(\alpha' + \phi)] \tag{4.42}$$

$$Y_1 = \frac{y_1}{f_1} = \rho\sin(\alpha' + \phi) \tag{4.43}$$

donde

$$\rho = 1 - \frac{1-\beta^2}{2(1-\beta\cos\alpha)} \tag{4.44}$$

$$\alpha' = \sin^{-1}\left(\frac{\sin\theta}{\gamma}\right) \tag{4.45}$$

$$\phi = \sin^{-1}\left(\frac{1-\rho}{\rho}\sin\alpha'\right) \tag{4.46}$$

4.5.2.- Proceso de Diseño.

La figura 4.31 muestra el modelo geométrico de una lente de Rotman desarrollada mediante el módulo de antenas. La antena se compone de dos capas metálicas separadas por una pequeña distancia que determina el espesor de la línea microstrip.

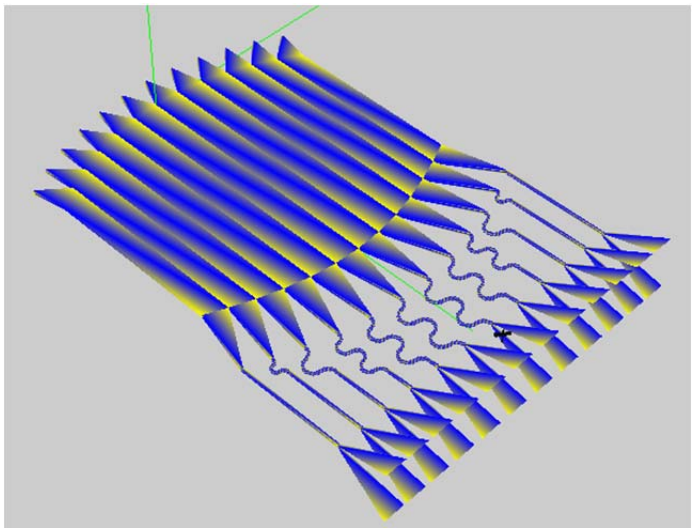


Figura 4.31.- Modelo geométrico de la lente de Rotman.

El diseño de los puertos de entrada y de salida puede afectar mucho al funcionamiento de la antena, especialmente en los niveles de lóbulo principal a secundario y al acoplo entre puertos adyacentes. En la implementación se ha intentado que el cambio de tamaño fuese lo más gradual posible, modelando todos los puertos con forma triangular. Por otro lado, los elementos radiantes se han modelado simulando la forma de pequeñas bocinas piramidales, cuya apertura es parametrizable.

La lente se puede alimentar por cualquiera de sus puertos de entrada, especificando un voltaje y una impedancia que viene dada por las ecuaciones (4.48)-(4.49).

$$\varepsilon_{eff} = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2}\left[\frac{1}{\sqrt{1 + 12b/w}} + 0.04\left(1 - \frac{w}{b}\right)^2\right] \tag{4.47}$$

$$Si\,\frac{w}{b} \leq 1, entonces\; Z_0 \approx \frac{\eta_0}{2\pi\sqrt{\varepsilon_{eff}}} ln\left(\frac{8b}{w} + \frac{w}{4b}\right) \tag{4.48}$$

$$Si\,\frac{w}{b} > 1, entonces\; Z_0 \approx \frac{\eta_0}{\sqrt{\varepsilon_{eff}}} \frac{1}{\frac{w}{b} + 1.393 + 0.667 \ln\left(\frac{w}{b} + 1.444\right)} \tag{4.49}$$

donde ε_r es la permitividad relativa del dieléctrico, que en este caso coincide con la del vacío dado que no hay dieléctrico entre las dos capas de conductor que componen la

lente, y *b* es su espesor (la distancia entre las capas). El valor de *w* hace referencia a la anchura del puerto que se está alimentando.

Estas expresiones fueron halladas por Wheeler [32] para obtener la impedancia característica de una línea microstrip. En ellas se desprecia el espesor de la tira conductora situada sobre el dieléctrico.

4.6.- ANTENAS MICROSTRIP.

Sus orígenes se remontan a los años 50, aunque no es hasta los años 70 cuando se empieza a trabajar con ellas de forma activa [33, 34]. En los últimos años están siendo muy utilizadas, sobre todo en aplicaciones en las que el tamaño de la antena debe ser muy reducido, como en el caso de comunicaciones inalámbricas, dispositivos móviles, sistemas GPS, aeronáutica, y, en general, para cualquier dispositivo que opere a frecuencias muy elevadas (microondas y ondas milimétricas).

Las antenas microstrip se pueden considerar extensiones de las líneas de transmisión. Se componen de un plano de masa en la parte inferior, un substrato dieléctrico en la zona central y un parche metálico situado en la parte superior. El espesor del dieléctrico debe ser lo suficientemente fino para evitar pérdidas por ondas superficiales. Sin embargo, se debe alcanzar un compromiso a la hora de establecer sus dimensiones, ya que un aumento en el espesor implica un aumento en la eficiencia de radiación de la antena. Respecto a la constante dieléctrica, interesa que sea lo menor posible, pues el dieléctrico presentará menos pérdidas y aumentará la eficiencia de radiación.

Entre sus ventajas destacan su bajo peso y pequeñas dimensiones, facilidad de integración prácticamente en cualquier tipo de superficie, fabricación sencilla, bajo coste, fácil de adaptar en circuitos integrados, etc. Por otro lado, sus inconvenientes son la baja potencia de radiación, pérdidas elevadas, ancho de banda reducido y pureza de polarización baja.

El patrón de radiación de las antenas microstrip suele ser omnidireccional, ya que la parte posterior está bloqueada por el plano de masa y la potencia es radiada en la dirección perpendicular al parche metálico. Es decir, el campo presenta el máximo de radiación en la dirección perpendicular al plano de la antena, siendo su polarización de tipo lineal.

Las configuraciones típicas de las antenas microstrip de un solo elemento se presentan normalmente en forma de anillo, cruz, círculo, rectángulo, triángulo, elipse, etc. La figura 4.32 muestra los elementos disponibles en la herramienta para el diseño de este tipo de antenas.

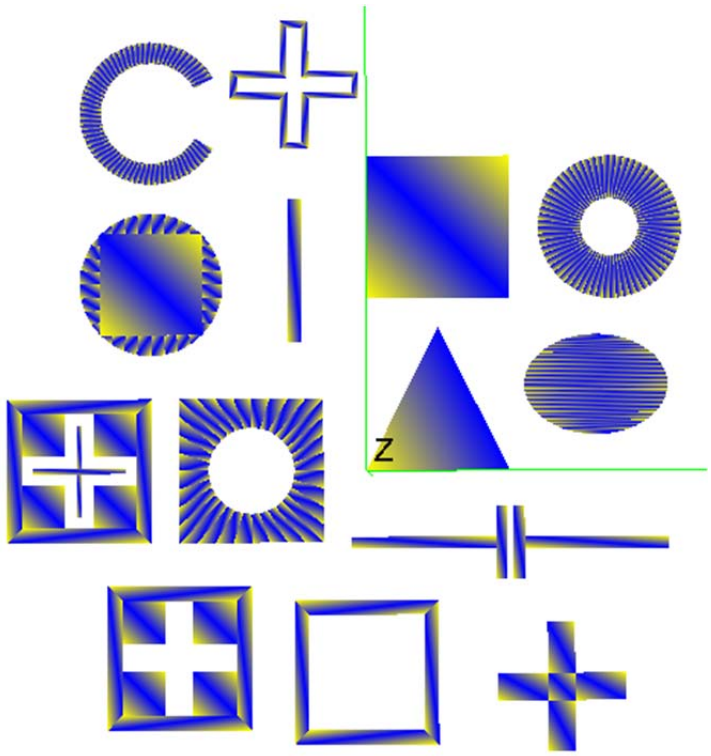


Figura 4.32.- Elementos disponibles en la herramienta para diseñar antenas microstrip.

También es posible diseñar antenas microstrip con configuraciones que requieren la repetición de los elementos mencionados anteriormente, dando como resultado un array de elementos microstrip. En estos casos, la antena resultante deja de presentar algunos de los inconvenientes mencionados anteriormente, como el de baja potencia de radiación [35, 36].

Respecto a la alimentación de este tipo de antenas, las formas más comunes son por campo impreso a través de un conector coaxial o mediante una línea microstrip. También existen otros métodos más complejos, como la alimentación por proximidad por apertura, en los que la alimentación no es directa, si no que el acoplamiento es electromagnético.

Una antena microstrip muy popular últimamente en el ámbito de las comunicaciones móviles es la antena PIFA (*Planar Inverted F Antenna*) [37]. Como se puede ver en la figura 4.33, está formada por un plano de masa y un elemento radiante situado en la parte superior. Ambos elementos se separan una cierta distancia y se unen mediante un hilo conductor que actúa como cortocircuito. La alimentación del elemento radiante se realiza también a través de un hilo conductor.

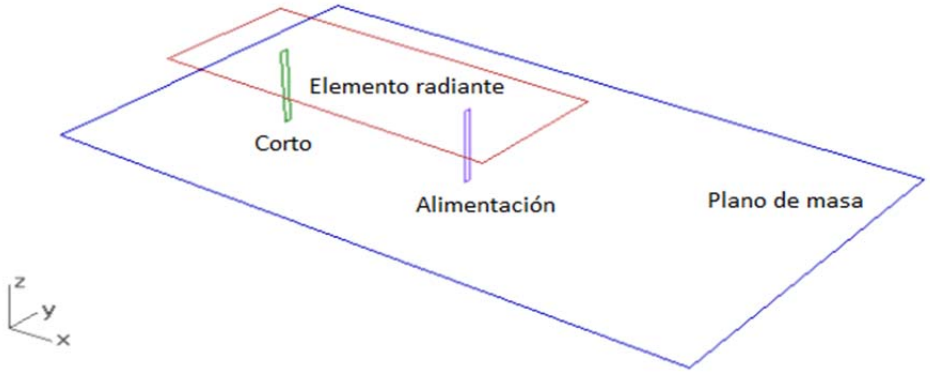


Figura 4.33.- Esquema genérico de una antena PIFA.

La figura 4.34 muestra las dimensiones de una antena PIFA que ha sido analizada con la herramienta para estudiar su comportamiento a una frecuencia de 1.800 MHz. El modelo geométrico generado por la herramienta, así como el mallado de la estructura se muestran en la figura 4.35.

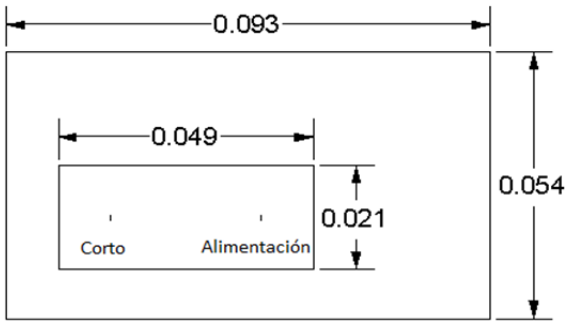


Figura 4.34.- Dimensiones de la antena PIFA. Unidades en metros.

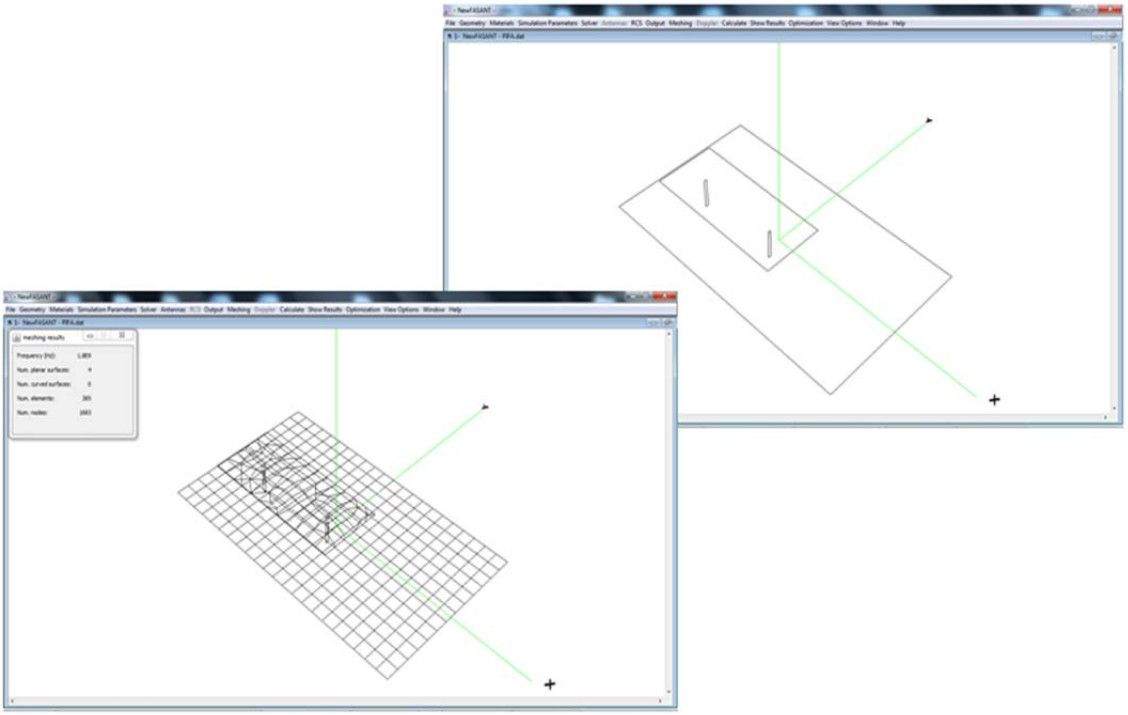


Figura 4.35.- Modelo geométrico (derecha) y mallado (izquierda) de la antena PIFA.

Los resultados de la simulación se pueden ver en las figuras 4.36-4.38.

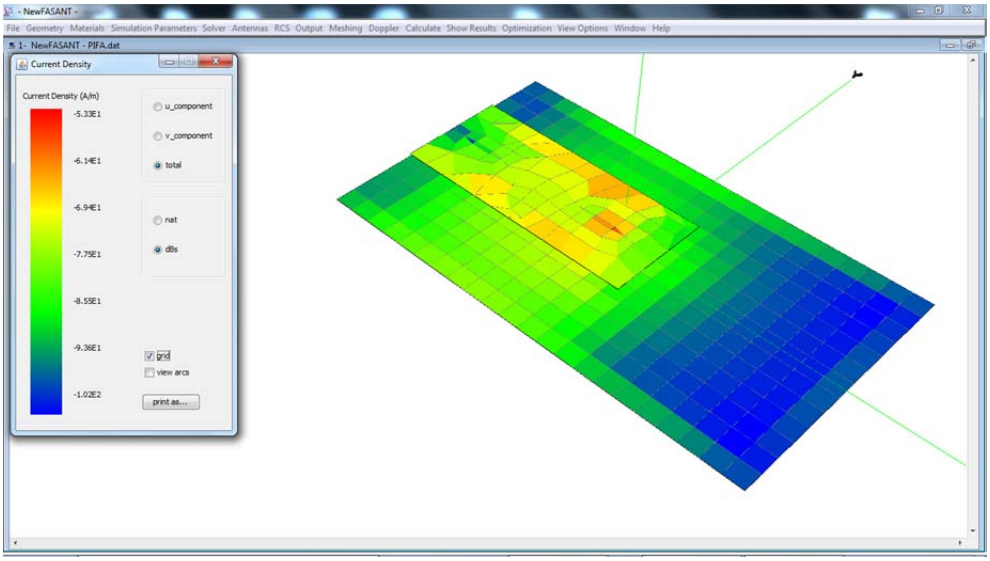


Figura 4.36.- Distribución de corrientes sobre la antena.

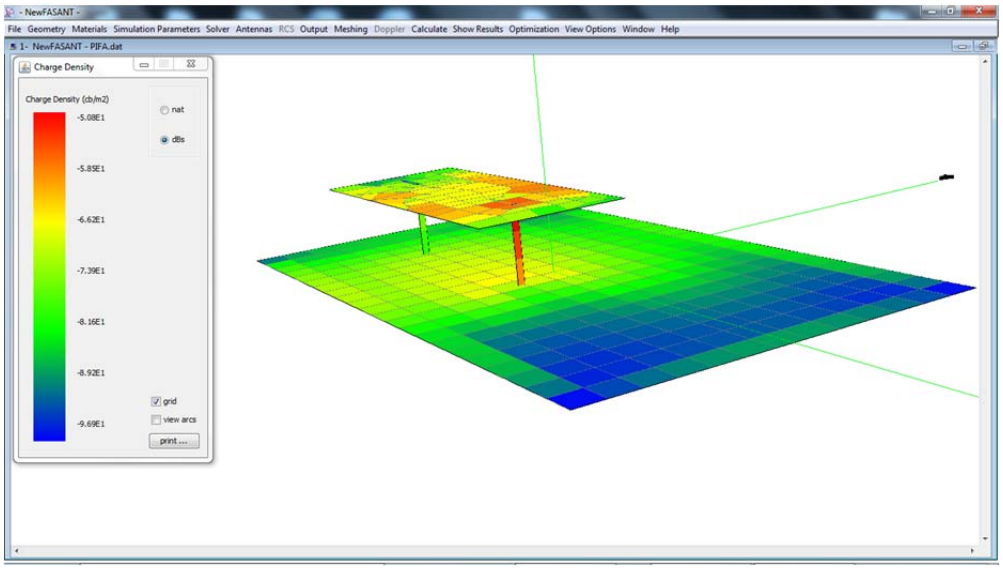


Figura 4.37.- Densidad de carga sobre la antena.

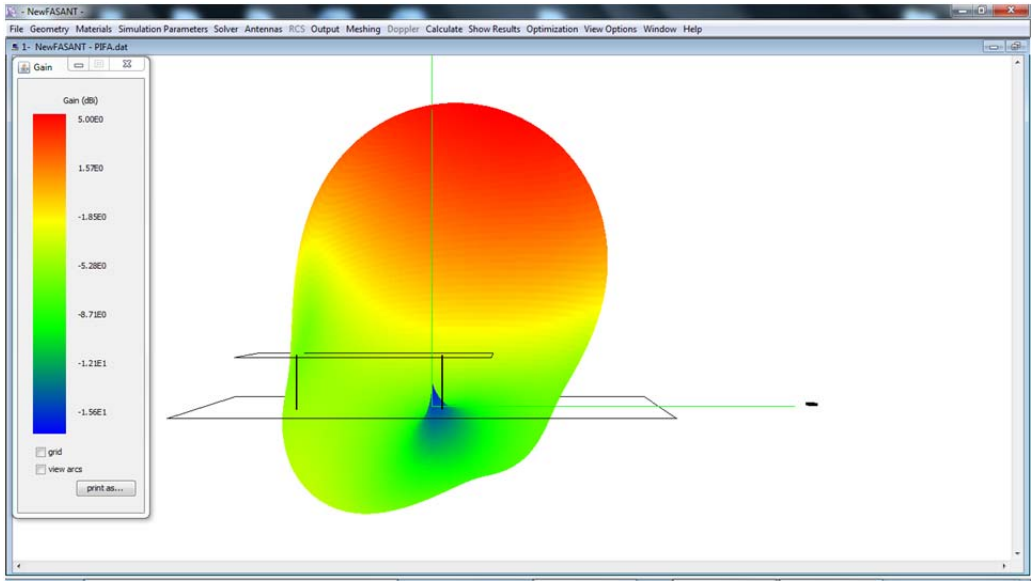


Figura 4.38.- Diagrama de radiación de la antena PIFA.

4.7.- SUPERFICIES SELECTIVAS EN FRECUENCIA.

Las superficies selectivas en frecuencia (FSS, *Frequency Selective Surfaces*) [38] son estructuras compuestas por elementos periódicos, ampliamente utilizadas para diseño de radomos y de estructuras EBG (*Electromagnetic BandGap*). Su diseño es inmediato, pues la herramienta dispone de un módulo específico para el diseño de estructuras periódicas. Además, mediante la opción de 'Array 3D' se puede generar un array rectangular a lo largo de una, dos o tres dimensiones considerando cualquier entidad geométrica como celda de repetición. La figura 4.39 muestra un ejemplo de la

opción 'Array 3D', en la que el elemento unidad es una bocina piramidal. Los únicos parámetros de configuración que hay que indicar para generar el array son el número de elementos y el periodo de repetición que se quieren obtener en cada dirección *x, y, z*, como se indica en la figura 4.40.

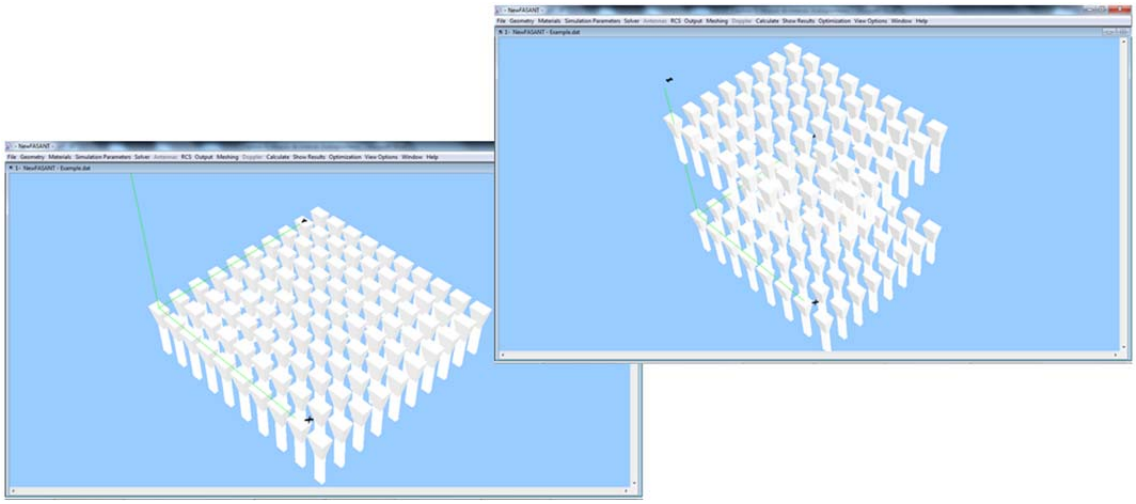


Figura 4.39.- Aplicación de la operación Array 3D sobre una bocina piramidal.

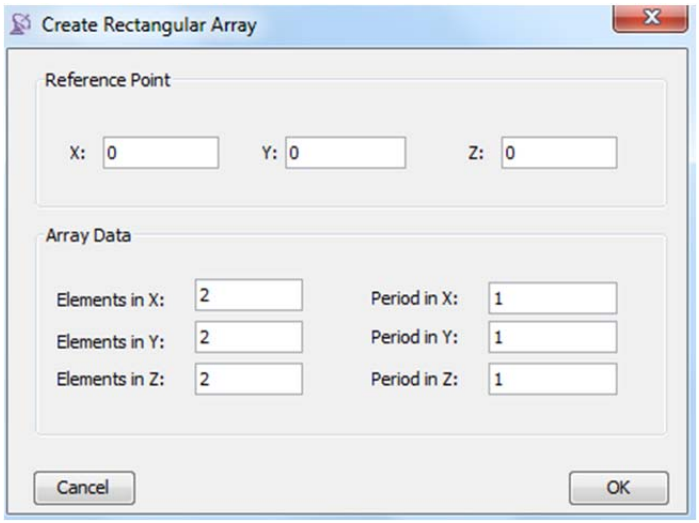


Figura 4.40.- Cuadro de diálogo de la operación Array 3D.

Los radomos paso banda son superficies selectivas en frecuencia que se usan para restringir el ancho de banda, limitando las señales que pueden penetrar y alcanzar la antena. Normalmente, la estructura selectiva en frecuencia forma parte de una especie de carcasa que protege a la antena. Las características de los radomo son su transparencia en la banda de operación y su rechazo de las señales fuera de dicha banda. Su comportamiento depende de los patrones geométricos para definir las bandas

de paso o de rechazo, los cuáles pueden presentar distintas formas, al igual que ocurría con los elementos microstrip.

A continuación se presenta el análisis de un radomo modelado mediante una estructura periódica cuya celda unidad es un parche cuadrado que contiene una ranura en forma de cruz. El objetivo del análisis es obtener los coeficientes de transmisión a lo largo de un rango de frecuencias amplio en el que se pueda apreciar la banda de paso y la banda de rechazo. La figura 7.41 muestra el modelo geométrico del radomo, diseñado mediante el módulo de estructuras periódicas disponible en la herramienta. También se muestra la forma de una celda unidad, cuyo periodo de repetición coincide con su anchura. El número de celdas en cada dirección es de 10.

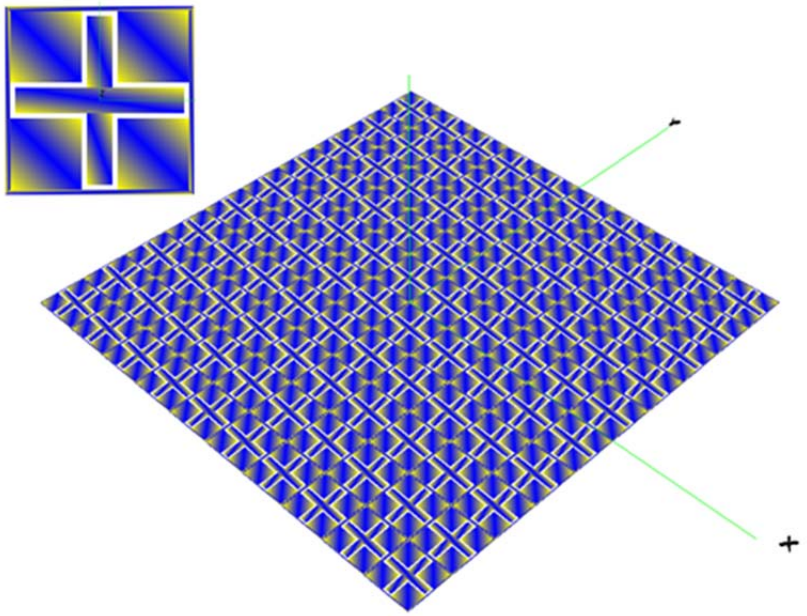


Figura 4.41.- Modelo geométrico del radomo diseñado mediante el módulo especializado en estructuras periódicas.

Respecto al resto de parámetros de simulación, se considera que una onda plana con polarización según $\hat{\theta}$ ilumina al radomo considerando cinco direcciones de incidencia en un plano paralelo a uno de los lados de las cruces, desde un ángulo de incidencia de 0º (incidencia normal) hasta 70º. El rango de frecuencias se establece desde 1GHz hasta 20 GHz con incrementos de 1GHz.

Las figuras 7.42 y 7.43 muestran los coeficientes de transmisión para polarización HH (campo eléctrico paralelo al plano de la rejilla) y VV (campo magnético paralelo a la rejilla). Observando dichas figuras se deduce que la estructura tiene un buen comportamiento en la banda de 10 a 13GHz. La transmisión presenta unas perdidas inferiores a 0.5 dB dentro de la banda de paso, mientras que fuera de ella se tienen atenuaciones que crecen al alejarse de la frecuencia central, hasta llegar a alcanzar unas pérdidas de unos 20 dB.

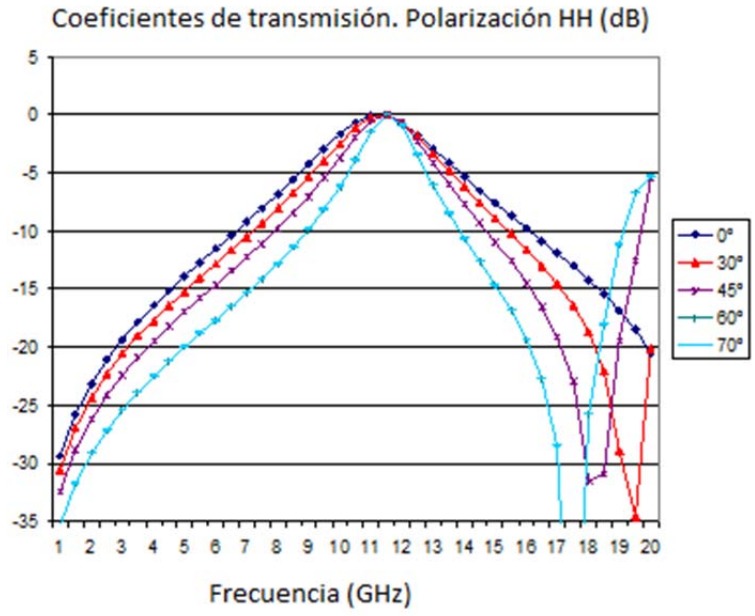


Figura 4.42.- Coeficientes de transmisión para polarización HH.

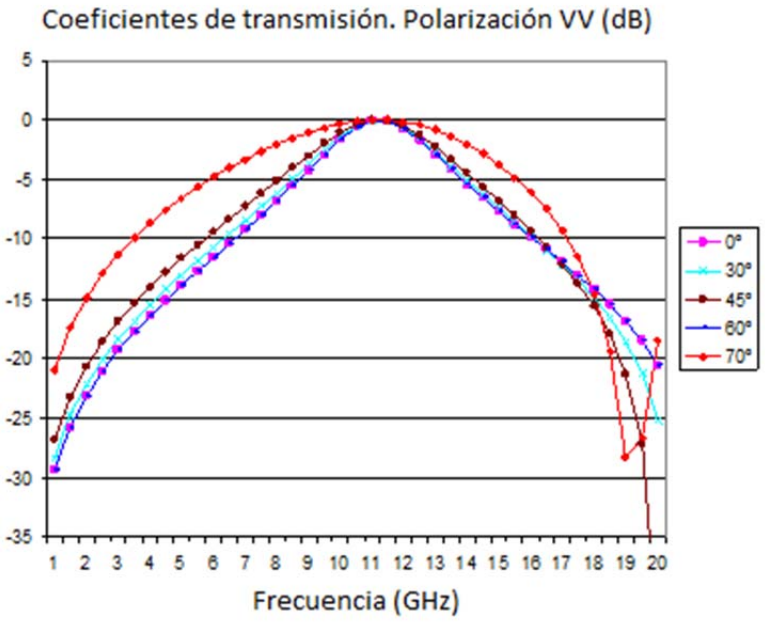


Figura 4.43.- Coeficientes de transmisión para polarización VV.

4.8.- REFERENCIAS.

[1].- Hasan, S.A.; "Design & measurements techniques for circularly polarized, dual fed, high gain, lightweight, wideband conical horn antenna with suppressed side lobes & high performance radome for space applications," *Antennas Propagation and EM Theory (ISAPE),* 2010 9th International Symposium on , vol., no., pp.26-29, Nov. 29 2010-Dec. 2 2010.

[2].- Jung, Y.B.; Eom, S.Y.; Yun, J.S.; Jeon, S.I.; Kim, C.J.; , "A Horn Antenna Design with Novel Waveguide Polarizer for Mobile Satellite Communications Applications," *Vehicular Technology Conference Fall (VTC 2009-Fall),* 2009 IEEE 70th , vol., no., pp.1-3, 20-23 Sept. 2009.

[3].- Nair, R.A.; Shafiei, R.;, "A multimode pyramidal horn with symmetrical dielectric loading-a high efficiency feed for reflector antennas in satellite communications" *Antennas and Propagation Society International Symposium*, 1990. AP-S. Merging Technologies for the 90's. Digest. , vol., no., pp.1522-1525 vol.4, 7-11 May 1990.

[4].- Gupta, R.C.; Saxena, S.; Mahajan, M.B.; Jyoti, R.; , "Design of Dual-Band Multimode Profiled Smooth-Walled Horn Antenna for Satellite Communication," *Antennas and Wireless Propagation Letters, IEEE* , vol.9, no., pp.338-341, 2010

[5].- Herd, J.S.; Kao, P.S.;, "Broadband TEM horn array for FOPEN radar," *Antennas and Propagation Society International Symposium,* 2001. IEEE, vol.2, no., pp.452-455 vol.2, 2001.

[6].- Rong-Xing Zhang; Ze-Ming Xie; Qing-Xin Chu; , "A New Horn Array Feed for Parabolic Cylindrical Reflector Antenna," *Microwave Conference*, 2007. APMC 2007. Asia-Pacific, vol., no., pp.1-4, 11-14 Dec. 2007.

[7].- Rathod, J.M.; Kosta, Y.P.;, "Low cost design & development of conical horn feed for parabolic reflector antenna," *Recent Advances in Microwave Theory and*

Applications, 2008. MICROWAVE 2008. International Conference on, vol., no., pp.775-777, 21-24 Nov. 2008

[8].- Bayat, A.; Khaleqi, A.; , "Design and implementation of an X-band corrugated feed horn for offset reflector antenna," *Antennas, Propagation and EM Theory*, 2003. Proceedings. 2003 6th International SYmposium on , vol., no., pp. 157-160, 28 Oct.-1 Nov. 2003

[9].- A. Tayebi, J. Gómez, F. Cátedra, "Optimización de Sondas Para Medidas de Antenas en Campo Cercano", *XXIV Simponsium Nacional de la Unión Científica Internacional de Radio*, Santander, 2009.

[10].- Kuhn, E.; Fasold, D.; Klefenz, F.;, "Design, optimization and test of high-performance circular corrugated feed horns for full V-Band (50 to 75 GHz) coverage," *Antennas and Propagation, 2009. EuCAP 2009. 3rd European Conference on*, vol., no., pp.3462-3466, 23-27 March 2009.

[11].- Karimi, J.; Blostein, S.D.;,"Parabolic reflector array signal processing for improved rural area coverage in personal satellite communications," Universal Personal Communications, 1996. Record., 1996 *5th IEEE International Conference on* , vol.1, no., pp.453-457 vol.1, 29 Sep-2 Oct 1996

[12].- Ramanujam, P.; Lopez, L.F.; Shin, C.; Chwalek, T.J.; , "A shaped reflector design for the DirecTv direct broadcast satellite for the United States," *Antennas and Propagation Society International Symposium,* 1993. AP-S. Digest , vol., no., pp.788-791 vol.2, 28 Jun- 2 Jul 1993

[13].- Terada, M.A.B.; Stutzman, W.L.; , "Computer-aided design of the Green Bank radio telescope reflector antenna," *Antennas and Propagation Society International Symposium*, 1997. IEEE., 1997 Digest , vol.3, no., pp.1630-1633 vol.3, 13-18 Jul 1997

[14].- Jaldehag, R.T.K.; Kildal, P.-S.; Ronnang, B.O.; , "Dual-band reflector feed system for classical Cassegrain radio telescopes," *Antennas and Propagation, IEEE Transactions on* , vol.41, no.3, pp.325-332, Mar 1993

[15].- Terada, M.; Bludworth, N.; Moore, J.; Sullivan, J.; , "Deployable reflector system for satellite applications," *Microwave and Optoelectronics, 2005 SBMO/IEEE MTT-S International Conference on* , vol., no., pp. 647- 649, 25-28 July 2005.

[16].- Tankersley, B.;, "Deployable, parabolic reflectors for ku-band operation," *Antennas and Propagation Society International Symposium*, 1972 , vol.10, no., pp. 311- 314, Dec 1972.

[17].- Martinez-Lorenzo, J.A.; Arias, M.; Rubinos, O.; Gutierrez, J.; Garcia-Pino, A.; , "A shaped and reconfigurable reflector antenna with sectorial beams for LMDS base station," *Antennas and Propagation, IEEE Transactions on ,* vol.54, no.4, pp. 1346- 1349, April 2006.

[18].- Monk, A.D.; Clarricoats, P.J.B.;, "Reconfigurable reflector antenna producing pattern nulls," *Microwaves, Antennas and Propagation, IEE Proceedings* - , vol.142, no.2, pp.121-128, Apr 1995.

[19].- Hoferer, R.A.; Rahmat-Samii, Y.;, "Inflatable parabolic torus reflector antenna for space-borne applications: concept, design and analysis*," Aerospace Conference, 1999. Proceedings. 1999 IEEE* , vol.3, no., pp.249-263 vol.3, 1999.

[20].- Savini, D.; Besso, P.; Tatalias, P.; Klooster, K.vt.; Ritz, W.; , "Electrical performance of a 10 meter inflatable reflector for land mobile communications," *Antennas and Propagation, 1991. ICAP 91., Seventh International Conference on* (IEE) , vol., no., pp.853-856 vol.2, 15-18 Apr 1991.

[21].- http://www.ticra.com/

[22].- http://www.fomento.es/MFOM/LANG_CASTELLANO/DIRECCIONES_GENERALES/INSTITUTO_GEOGRAFICO/Astronomia/instalaciones/cay/

[23].- Haeger, T.A. and Lee, J. J. "Comparison Between a Shaped and Nonshaped Small Cassegrain Antenna". *IEEE Trans. on Antennas and Propagation*, Vol. 38, Nº 12, pp. 1920-1924.

[24].- Antenna Theory: Analysis and Design (John Wiley & Sons, 2005) by Constantine A. Balanis.

[25].- Altshuler, E.; , "The traveling-wave linear antenna," *Antennas and Propagation, IRE Transactions on* , vol.9, no.4, pp.324-329, July 1961.

[26].- Iizuka, K.; , "The traveling-wave v-antenna and related antennas," *Antennas and Propagation, IEEE Transactions on* , vol.15, no.2, pp. 236- 243, Mar 1967.

[27].- Strait, B.J.; Adams, A.T.; , "Analysis and Design of Wire Antennas with Applications to EMC," *Electromagnetic Compatibility, IEEE Transactions on* , vol.EMC-12, no.2, pp.45-54, May 1970.

[28].- Zainud-Deen, S.H.; Salem, N.A.-D.M.; Ibrahem, S.M.M.; Motaafy, H.A.; , "Investigation of an octafilar helix antenna," *Radio Science Conference, 2002. (NRSC 2002). Proceedings of the Nineteenth National* , vol., no., pp. 72- 80, 2002.

[29].- Wen-Yi Qin; Jing-Hui Qiu; Qi Wang; , "A novel multi-frequency quadrifilar helix antenna," *Antennas and Propagation Society International Symposium,* 2005 IEEE , vol.1B, no., pp.467-470 vol. 1B, 2005.

[30].- Sharma, S.K.; Shafai, L.; , "Investigations on miniaturized endfire vertically polarized quasi-fractal log-periodic zigzag antenna," *Antennas and Propagation, IEEE Transactions on* , vol.52, no.8, pp. 1957- 1962, Aug. 2004.

[31].- W. Rotman and R. F. Turner, "Wide-angle microwave lens for line source application," *IEEE Trans. Antennas Propagat*., vol. AP-1 1, pp. 623-632, November 1963.

[32].- H. A. Wheeler, "Transmission-Line Properties of Parallel Strips Separated by a Dielectric Sheet", *IEEE Trans. Microwave Theory and Techniques*, MTT-3, No. 3, March 1965, pp. 172-185.

[33].- Gallegro, A.; Esposito, F.; Pallas, M.;, "Dipole Antenna on a High-Dielectric Substrate," *Microwave Conference,* 1969. 1st European, vol., no., pp.387-390, 8-12 Sept. 1969.

[34].- Mittra, R.; Itoh, T.;, "Theory of Shielded Microstrip Lines," *Microwave Conference,* 1969. 1st European , vol., no., pp.14-17, 8-12 Sept. 1969.

[35].- Sabban, A.;, "Ka band microstrip antenna arrays with high efficiency," *Antennas and Propagation Society International Symposium*, 1999. IEEE, vol.4, no., pp.2740-2743 vol.4, Aug 1999.

[36].- Sabban, A.;, "Applications of MM Wave Microstrip Antenna Arrays," Signals, Systems and Electronics, 2007. *ISSSE '07. International Symposium on* , vol., no., pp.119-122, July 30 2007-Aug. 2 2007.

[37].- T. Taga and D. Tsunekawa, "Performance Analysis of a Built-In Planar Inverted F Antenna for 800 MHz Band Portable Radio Units", *IEEE Journal on Selected areas in communications*, vol. SAC-5, No. 5. June 1987.

[38].- B. Munk, Frequency Selective Surfaces: Theory and Design, John Wiley and Sons, 2000.

5.- Tratamiento de la Geometría.

5.1.- INTRODUCCIÓN.

Los modelos geométricos creados o importados en la herramienta son procesados posteriormente en una etapa de análisis para estudiar su comportamiento electromagnético. En este capítulo se describe el tratamiento que se realiza sobre la geometría cuando comienza el proceso de simulación. En primer lugar se presenta la descripción matemática de las superficies NURBS que describen la geometría bajo análisis. A continuación, se expone el mecanismo de interpolación necesario para adaptar la malla de puntos proporcionada por el fichero geométrico, a los parámetros que definen las superficies NURBS. Por último se describe a grandes rasgos cómo se lleva a cabo el proceso de análisis electromagnético mediante la aplicación del Método de los Momentos sobre la versión discretizada del modelo geométrico.

5.2.- SUPERFICIES NURBS.

Una superficie NURBS se puede definir como una representación matemática de una geometría en 3D capaz de describir cualquier forma arbitraria con mucha precisión. Una característica importante de este tipo de representaciones es que las ecuaciones que definen las superficies NURBS se pueden implementar de manera eficaz y precisa. Además, la cantidad de información que requiere su representación es muy inferior a la que necesitan por separado otras aproximaciones comunes. Respecto a la compatibilidad de formatos, existen varios estándares industriales para intercambiar los modelos geométricos modelados con superficies NURBS, como .DXF e .IGES.

Los parches NURBS se descomponen en parches de Bezier, debido a que son más estables matemáticamente, utilizando el algoritmo Cox-de-Boor [1]. Básicamente, dicho algoritmo describe una curva B-Spline como un conjunto de curvas de Bezier. En

los siguientes apartados se explican las características de estas curvas con más detalle, así como su definición matemática.

5.2.1.- Descripción Matemática.

Una curva de Bezier [2, 3] se puede expresar matemáticamente mediante una combinación de bases de Bernstein, de la siguiente forma:

$$\vec{c}(t) = \sum_{i=0}^{n} \overrightarrow{b_\iota}\, B_i^n(t) \tag{5.1}$$

donde *t* es la variable paramétrica, situada dentro del rango [0, 1], *n* es el grado de la curva, y el vector $\overrightarrow{b_\iota}$, contiene los vértices del polígono de control de la curva. Los polinomios de Bernstein [4, 5] vienen dados por:

$$B_i^n(t) = \frac{n!}{i!\,(n-i)!}\, t^i (1-t)^{n-i} \tag{5.2}$$

Una de las propiedades de estos polinomios es que se pueden obtener de forma recursiva a partir de la siguiente ecuación:

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t) \tag{5.3}$$

Con $B_0^0(t) \equiv 1$ y $B_j^n(t) \equiv 0$ si $j \notin \{0, \dots, n\}$

Algunas características importantes de este tipo de curvas son las siguientes:

- El número de puntos de control que definen la curva es igual al orden de la misma, siendo el grado de la curva igual al orden menos uno.
- La curva se encuentra contenida en la envolvente convexa del polígono de control, por lo que éste ofrece una aproximación intuitiva de la forma de la misma.

- La curva tiende a acercarse a los puntos de control más próximos, pasando necesariamente por el primero y el último de ellos. Además, en ambos puntos el vector tangente a la curva es el resultado de la unión de cada uno de ellos con el punto de control más cercano dentro del segmento paramétrico.
- Este tipo de curvas es invariante frente a transformaciones afines (rotaciones, traslaciones, escalado, etc.).

Modificando la expresión 5.1 se llega a la definición de curva de Bezier racional, dada por:

$$\vec{c}(t) = \frac{\sum_{i=0}^{n} w_i \overrightarrow{b_i} B_i^n(t)}{\sum_{i=0}^{n} w_i B_i^n(t)} \tag{5.4}$$

donde se ha asociado un peso a cada punto de control de la curva, lo que supone la inclusión de un nuevo grado de libertad para el diseño de la misma, haciendo que la curva se aproxime en mayor o menor medida a cada punto de control dependiendo del peso asociado.

Resulta evidente que la expresión no racional (5.1) constituye una particularización de (5.4) haciendo todos los pesos iguales a la unidad. La inclusión de este tipo de curvas permite la representación de curvas cónicas (circunferencias, elipses, hipérbolas y parábolas).

Después de analizar las expresiones por las que se definen las curvas racionales de Bezier, así como sus características generales, se puede generalizar el estudio al caso de superficies paramétricas, incluyendo los elementos necesarios para considerar una nueva dimensión. De esta forma, a partir de (5.4) se obtiene la expresión general de una superficie de Bezier:

$$\vec{r}(u,v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} \overrightarrow{b_{ij}} B_i^m(u) B_j^n(v)}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{ij} B_i^m(u) B_j^n(v)} \tag{5.5}$$

donde se han incluido las dos variables paramétricas *u* y *v*, limitadas al dominio [0, 1], *m* y *n* son los grados de la superficie para cada una de las coordenadas paramétricas, y los puntos de control por extensión al caso bidimensional forman la malla o red de control de la superficie.

Al igual que ocurría con las curvas, la superficie se encuentra encerrada por la envolvente convexa de la malla de control, asegurando que no se produzcan oscilaciones fuera de ella. Puede destacarse también que en los cuatro vértices de la superficie los puntos de control coinciden con los vértices de la propia superficie.

A pesar de que las curvas de Bezier son muy útiles para representar diseños, presentan algunos inconvenientes. Por ejemplo, cuando la curva que se quiere modelar es compleja, su representación obliga a usar una curva de un grado excesivamente alto, lo que puede complicar su tratamiento posterior en aplicaciones prácticas. Además, cuando se modifica la posición de uno de los puntos de control, toda la curva se ve afectada, lo que supone un gran inconveniente. Estos problemas se pueden solucionar utilizando conexiones de diferentes curvas de Bezier de grado bajo. Las curvas compuestas de esta forma se denominan curvas B-Spline.

Una curva B-Spline [6, 7] es una función paramétrica que, al igual que en el caso de las curvas de Bezier, puede expresarse por medio de una expansión con una serie de funciones denominadas bases B-Spline de la siguiente forma:

$$\vec{c}(t) = \sum_{i=1}^{n} \overrightarrow{d_i} N_i^k(t) \qquad 2 \leq k \leq n+1 \tag{5.6}$$

donde el polígono de control está dado por los puntos $\overrightarrow{d_i}$ y posee las mismas características que su homólogo en las superficies de Bezier, *k* es el grado de la curva, y $N_i^k(t)$ son las bases B-Spline, expresadas de forma recursiva según la siguiente fórmula:

$$N_i^k(t) = \frac{(t - t_i)N_i^{k-1}(t)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - t)N_{i+k}^{k-1}(t)}{t_{i+k} - t_{i+1}} \tag{5.7}$$

donde: $N_i^1(t) = \begin{cases} 1 & si\ t_i \leq t \leq t_{i+1} \\ 0 & en\ otro\ caso \end{cases}$

Los valores t_i forman un conjunto de puntos denominado vector de nudos. Dicho vector es una sucesión creciente de números reales dentro del segmento paramétrico [0, 1]. La multiplicidad de un nudo es el número de veces que aparece repetido en el vector de nudos. La distribución del vector de nudos determina las relaciones de continuidad a lo largo de la curva, de forma que la multiplicidad de un determinado nudo tiene influencia sobre el tipo de continuidad de la curva en el punto que se corresponde con el valor paramétrico del nudo.

Cuando los nudos forman una sucesión de valores equiespaciados se dice que el vector de nudos es uniforme. En el caso de que en los extremos del dominio paramétrico la multiplicidad de los nudos sea igual al orden de la curva, se dice que el vector de nudos es abierto. De esta manera, el vector de nudos de una curva abierta y no uniforme (como es el caso de las curvas NURBS), de orden k+1 y con n+1 puntos de control, en un caso general debe tener la siguiente forma:

$$0 = t_1 = t_2 = \cdots = t_k < t_{k+1} \leq t_{k+2} \leq \cdots \leq t_n < t_{n+1} = \cdots = t_{n+k} = 1 \qquad (5.8)$$

Las bases B-Spline poseen además las siguientes propiedades:

$$\sum_{i=0}^{n} N_i^k(t) = 1 \qquad (5.9)$$

$$N_i^k(t) \geq 0 \qquad (5.10)$$

Las curvas B-Spline son invariantes ante transformaciones afines. Además, en una curva B-Spline se debe cumplir la siguiente relación entre el grado k de la curva, el número de puntos de control n+1 y el número de nudos l:

$$l = (n + 1) + (k + 1) \qquad (5.11)$$

Una propiedad muy importante de estas curvas que no es compartida por las curvas de Bezier es que la modificación de uno de los puntos de control afecta únicamente a la zona que se encuentra en la cercanía del mismo, sin extenderse al resto de la curva.

Al igual que ocurría con las curvas de Bezier, se puede asociar un peso a cada punto de control y obtener una curva B-Spline racional (curvas NURBS, Non Uniform Rational B-Spline) cuya expresión matemática viene dada por:

$$\vec{c}(t) = \frac{\sum_{i=0}^{n} w_i \overrightarrow{d_i} N_i^k(t)}{\sum_{i=0}^{n} w_i N_i^k(t)}, \qquad 2 \leq k \leq n+1 \tag{5.12}$$

De igual forma es inmediato introducir el concepto de superficie racional B-Spline, considerando que, al añadir una nueva dimensión espacial, cada punto de la curva B-Spline original se mueve a través de otra curva. La expresión matemática de la superficie puede ser escrita de la siguiente forma:

$$\vec{r}(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij} \overrightarrow{d_{ij}} N_i^k(u) N_j^l(v)}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij} N_i^k(u) N_j^l(v)}, 2 \leq k \leq m+1, 2 \leq l \leq n+1 \tag{5.13}$$

donde k y l son los grados de la superficie para cada una de las coordenadas paramétricas, y $N_i^k(u)$ y $N_j^l(v)$ son las bases vistas en (5.7). La superficie está contenida en la envolvente convexa de la malla de control, y los cuatro vértices de la misma coinciden con los puntos de control de los extremos en ambas coordenadas paramétricas. En el caso de superficies, se define un vector de nudos en cada coordenada paramétrica.

5.2.2.- Proceso de Interpolación.

Los ficheros en formato .DXF almacenan la información geométrica en función de mallas de puntos, y no tratando directamente sobre superficies paramétricas como hacen otros formatos. Teniendo en cuenta que el tratamiento geométrico que lleva a cabo el núcleo electromagnético se hace sobre superficies NURBS, se debe realizar una interpolación sobre los puntos que componen la malla dada por el fichero .DXF para

obtener la descripción matemática de las superficies paramétricas, en este caso, el valor de los puntos de control.

Considerando la expresión que describe una curva de Bezier (5.1), se deduce que los puntos de control $\vec{b}_i$ se pueden obtener fácilmente a partir de n+1 puntos $\vec{p}_i$ equiespaciados en el espacio paramétrico por ejemplo, por los que pasa la superficie:

$$\begin{aligned}
&t = t_0 = 0 \quad , \quad \vec{p}_0 = \vec{b}_0 B_0^n(0) + \vec{b}_1 B_1^n(0) + \ldots\ldots + \vec{b}_n B_n^n(0) \\
&t = t_1 = 1/n \quad , \quad \vec{p}_1 = \vec{b}_0 B_0^n(1/n) + \vec{b}_1 B_1^n(1/n) + \ldots\ldots + \vec{b}_n B_n^n(1/n) \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
&t = t_n = 1 \quad , \quad \vec{p}_n = \vec{b}_0 B_0^n(1) + \vec{b}_1 B_1^n(1) + \ldots\ldots + \vec{b}_n B_n^n(1)
\end{aligned} \tag{5.14}$$

A partir de (5.14) se obtiene un sistema lineal de n ecuaciones con n incógnitas que se puede expresar en notación matricial del siguiente modo:

$$\begin{bmatrix} \vec{p}_0 & \cdots & \vec{p}_n \end{bmatrix} = \begin{bmatrix} B_0^n(t_0) & \cdots & B_n^n(t_0) \\ \vdots & \ddots & \vdots \\ B_0^n(t_n) & \cdots & B_n^n(t_n) \end{bmatrix} \begin{bmatrix} \vec{b}_0 \\ \vdots \\ \vec{b}_n \end{bmatrix} \tag{5.15}$$

$$P = B\,b \tag{5.16}$$

despejando en (5.16) se obtienen los puntos de control *b*:

$$b = B^{-1}\,P \tag{5.17}$$

Si los puntos sobre la curva $\vec{p}_i$ se expresan en coordenadas cartesianas, se obtiene un sistema de ecuaciones para cada una de las coordenadas:

$$\vec{p}_i = x_i\hat{x} + y_i\hat{y} + z_i\hat{z} \tag{5.18}$$

$$\begin{aligned} b_x &= B^{-1}X \\ b_y &= B^{-1}Y \\ b_z &= B^{-1}Z \end{aligned} \tag{5.19}$$

Análogamente, la interpolación realizada para las curvas de Bezier se puede extender al caso de las superficies de Bezier. De la expresión (5.5) se puede obtener:

$$\vec{P}(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} w_{ij}\vec{b}_{ij}\ \mathrm{B_i^m}(u)\mathrm{B_j^n}(v) \tag{5.20}$$

Si se asume que todos los pesos w_{ij} toman valor 1, la ecuación dada por (5.20) se puede expresar como sigue:

$$\vec{P}(u,v) = \sum_{j=0}^{n} B_j^n(v)\sum_{i=0}^{m}\vec{b}_{ij}B_i^m(u) \tag{5.21}$$

Definiendo el siguiente vector $\vec{d}_j(u)$:

$$\vec{d}_j(\mathrm{u}) = \sum_{i=0}^{m}\vec{b}_{ij}B_i^m(u) \tag{5.22}$$

el cual modela una serie de puntos de control de una nueva curva de Bezier de grado n:

$$\vec{P}(\mathrm{v}) = \sum_{j=0}^{n}\vec{d}_j B_j^n(v) \tag{5.23}$$

se puede obtener una expresión matricial del conjunto de puntos $\vec{P}(u,v)$ por los que pasa la superficie:

$$\vec{P}(u,v) = \begin{bmatrix} B_0^m(u) & \cdots & B_m^m(u) \end{bmatrix} \begin{bmatrix} \vec{b}_{00} & \cdots & \vec{b}_{0n} \\ \vdots & \ddots & \vdots \\ \vec{b}_{m0} & \cdots & \vec{b}_{mn} \end{bmatrix} \begin{bmatrix} B_0^n(v) \\ \vdots \\ B_n^n(v) \end{bmatrix} \tag{5.24}$$

Partiendo de una malla de puntos conocida de *(m+1)·(n+1),* se puede definir un sistema de ecuaciones a resolver como el siguiente:

$$P = U\, b\, V \tag{5.25}$$

donde se definen las siguientes matrices:

$$P = \begin{bmatrix} \vec{p}_{00} & \cdots & \vec{p}_{0n} \\ \vdots & \ddots & \vdots \\ \vec{p}_{m0} & \cdots & \vec{p}_{mn} \end{bmatrix}_{(m+1)(n+1)} \tag{5.26}$$

$$b = \begin{bmatrix} \vec{b}_{00} & \cdots & \vec{b}_{0n} \\ \vdots & \ddots & \vdots \\ \vec{b}_{m0} & \cdots & \vec{b}_{mn} \end{bmatrix}_{(m+1)(n+1)} \tag{5.27}$$

$$U = \begin{bmatrix} B_0(u_0) & \cdots & B_n(u_0) \\ \vdots & \ddots & \vdots \\ B_0(u_m) & \cdots & B_m(u_m) \end{bmatrix}_{(m+1)(n+1)} \qquad V = \begin{bmatrix} B_0(v_0) & \cdots & B_n(v_0) \\ \vdots & \ddots & \vdots \\ B_0(v_m) & \cdots & B_m(v_m) \end{bmatrix}_{(m+1)(n+1)} \tag{5.28}$$

para un equiespaciamiento de los puntos de la superficie en el espacio paramétrico:

$$\begin{aligned} u &= \begin{bmatrix} u_0 & u_1 & \cdots & u_{m-1} & u_m \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{m} & \cdots & \frac{m-1}{m} & 1 \end{bmatrix} \\ v &= \begin{bmatrix} v_0 & v_1 & \cdots & v_{n-1} & v_n \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{n} & \cdots & \frac{n-1}{n} & 1 \end{bmatrix} \end{aligned} \tag{5.29}$$

Para obtener los puntos de control se resuelve el sistema de ecuaciones (5.25) despejando *b* de la ecuación:

$$b = U^{-1} P\, V^{-1} \tag{5.30}$$

Si los puntos sobre la superficie $\vec{p}_{ij}$ vienen representados por sus coordenadas cartesianas como en (5.18), hay que resolver 3 sistemas de ecuaciones, un sistema para cada coordenada cartesiana.

$$\begin{aligned} b_x &= U^{-1} X V^{-1} \\ b_y &= U^{-1} Y V^{-1} \\ b_z &= B^{-1} Z V^{-1} \end{aligned} \tag{5.31}$$

En la figura 5.1 se muestra un ejemplo de interpolación de una superficie dada por 5 x 2 puntos en una superficie de Bezier. En dicha figura se puede observar la diferencia entre los dos tipos de modelado: el de color negro hace referencia a la malla de puntos dada por el fichero .DXF, y el de color rojo hace referencia a la superficie paramétrica obtenida definida a partir de los puntos de control obtenidos.

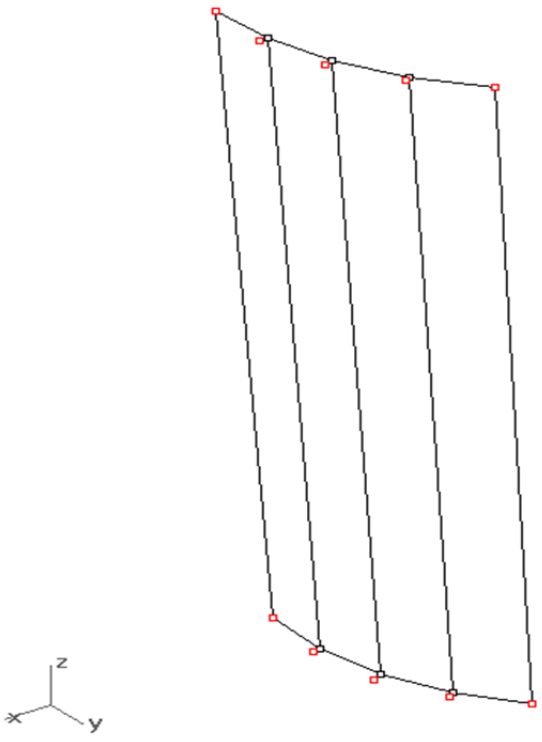


Figura 5.1.- Resultado de la interpolación. Malla de puntos original en color negro. Puntos de control de la superficie de Bezier obtenidos en color rojo.

5.3.- DISCRETIZACIÓN DE LA GEOMETRÍA.

El procedimiento de análisis electromagnético mediante métodos rigurosos como el Método de los Momentos lleva asociada una etapa de discretización de la geometría bajo análisis. Para ello, se ha desarrollado un algoritmo de mallado dentro del grupo de investigación en el que se ha desarrollado la presente tesis, que descompone el modelo geométrico en parches cuadrangulares/triangulares curvos.

Partiendo de superficies paramétricas, se dividen en una cantidad finita de elementos preferentemente cuadrilaterales curvos, si bien algunos pueden ser triangulares, cuyo tamaño de borde se calcula en función de la frecuencia a la que se desee realizar el mallado y del número de divisiones a emplear por longitud de onda. Una característica importante del algoritmo es que el mallado respeta fielmente las formas de la geometría original.

El método de mallado se basa en el algoritmo de enladrillado o paving [8]-[10], combinado con otras técnicas de pre-procesado, como puede ser el cálculo previo de puntos de mallado en los contornos de las superficies que garanticen la continuidad de los cuadrángulos entre superficies que tengan topología. El algoritmo de enladrillado consiste en ir mallando las superficies desde sus fronteras hacia dentro (o fuera, si se empieza en la frontera de un agujero interior a la superficie) completando filas de elementos adosados. Cuando se enfrentan varios frentes de mallado, se tratan especialmente para unirlos de forma que se mantenga la forma ideal de los elementos cuadrangulares. El algoritmo también cuenta con técnicas de postprocesado, como la etapa de limpiado o *clean up* [11] que mejora la calidad del mallado, consiguiendo que los elementos se aproximen todavía más a cuadrángulos. Las figuras 5.2 y 5.3 muestran el resultado del algoritmo aplicado sobre la geometría de un reflector alimentado mediante una bocina piramidal. En la figura 5.3 se pueden apreciar los detalles del mallado.

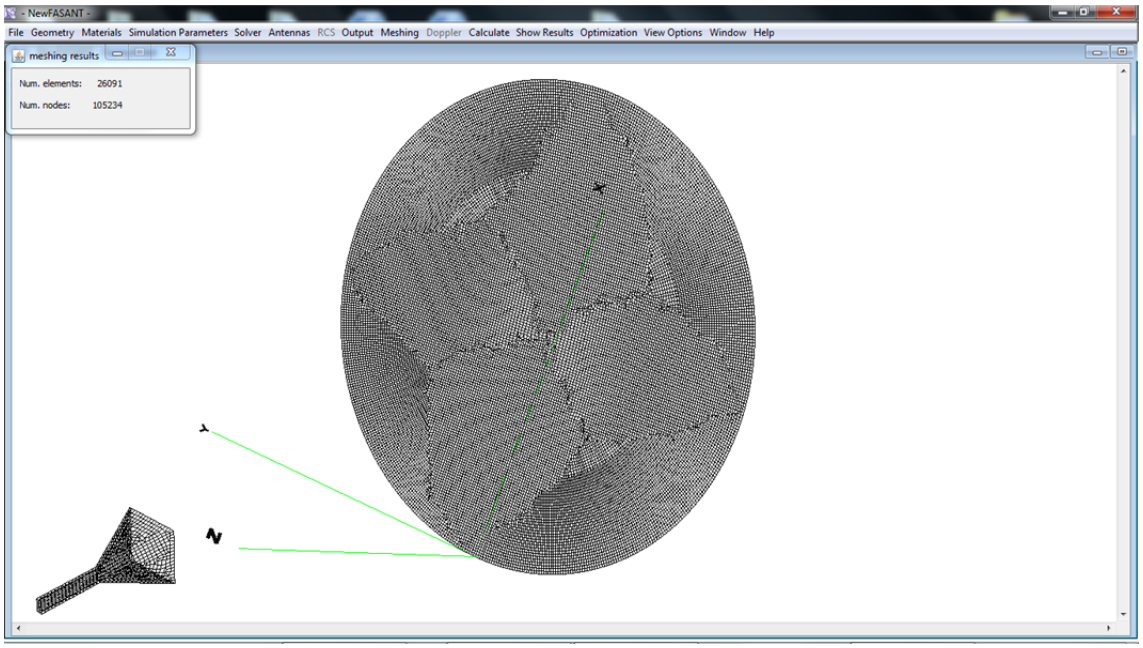


Figura 5.2.- Algoritmo de mallado aplicado sobre el modelo geométrico de un reflector alimentado por una bocina piramidal.

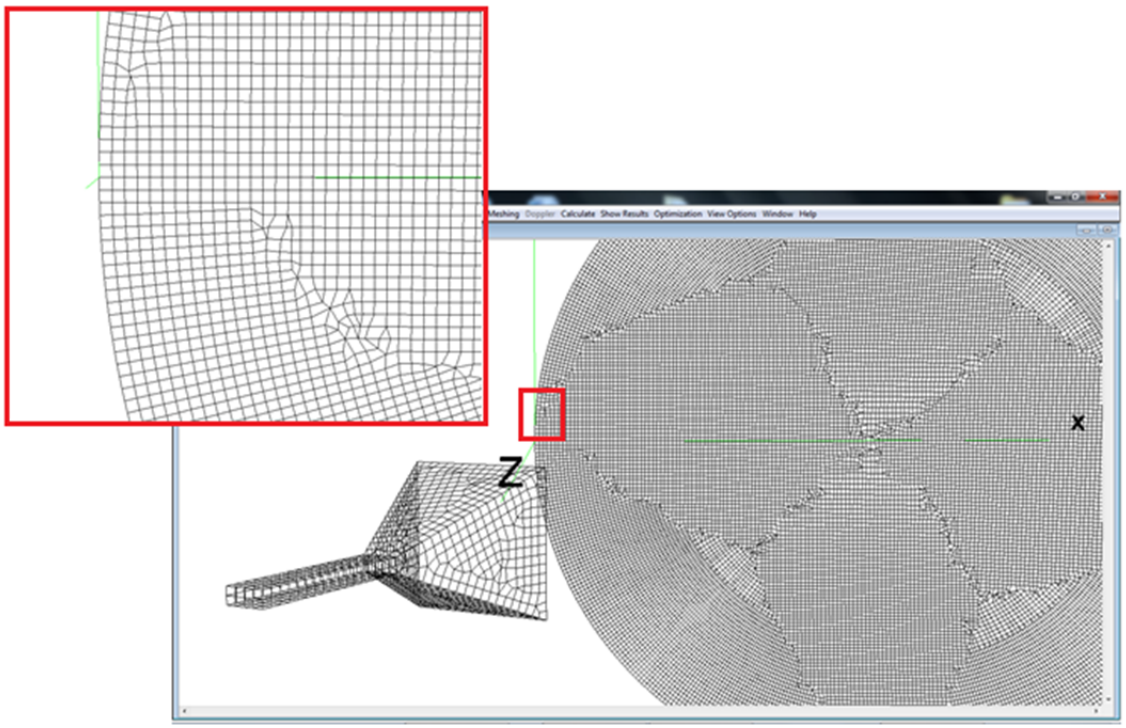


Figura 5.3.- Detalle del mallado.

5.4.- ANÁLISIS ELECTROMAGNÉTICO.

Para describir el tratamiento que realiza el núcleo electromagnético sobre la geometría, se parte del teorema de equivalencia aplicado a un caso sencillo en el que

una superficie conductora es iluminada mediante un campo incidente creado por una fuente de corriente en un punto del espacio. Este campo induce unas corrientes en la superficie de la geometría, las cuales generan un campo dispersado. Para poder calcular la distribución de corrientes sobre la estructura conductora en presencia de unas determinadas fuentes, es necesario plantear una ecuación que relacione la excitación establecida con los campos producidos, a través de unas condiciones de contorno determinadas, impuestas por la presencia de estructuras que afectan al comportamiento de los campos.

Dependiendo de las condiciones de contorno aplicadas, se obtienen las siguientes ecuaciones integrales: de campo eléctrico, la cual se basa en la condición de contorno sobre la componente tangencial de campo eléctrico; de campo magnético, relacionada con la condición de contorno que vincula la componente tangencial de campo magnético con la densidad de corriente inducida; y de campo combinado, que surge como combinación lineal de las dos anteriores.

La ecuación integral obtenida se resuelve numéricamente aplicando el Método de los Momentos. En los siguientes sub-apartados se presentan las características más importantes de dicho método, base a partir de la cual se ha implementado el núcleo electromagnético de NewFasant.

5.4.1.- Método de los Momentos.

El Método de los Momentos [12, 13] es una de las técnicas rigurosas de análisis electromagnético más utilizadas mundialmente para resolver problemas de radiación y dispersión. Básicamente se trata de un procedimiento que permite analizar un determinado problema reduciéndolo a la resolución de un sistema de ecuaciones lineales, como el indicado en (5.32).

$$\mathfrak{L}(I) = Y \tag{5.32}$$

donde $\mathfrak{L}$ es un operador lineal, en este caso un operador integro-diferencial, I es la incógnita a resolver y representa la distribución de corrientes e Y es una función conocida, en este caso un campo impreso o un voltaje.

Dado que no existe una solución analítica exacta a la ecuación (5.32), la función incógnita *I* se aproxima por una serie de funciones base, de la siguiente forma:

$$I = \sum_{n=1}^{N} \alpha_n B_n \tag{5.33}$$

donde $\{\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_N\}$ son los coeficientes que ponderan a cada una de las funciones base y que deberán ser determinados.

Gracias a la propiedad de linealidad del operador integro-diferencial, se puede sustituir (5.33) en (5.32), obteniéndose la siguiente expresión:

$$\sum_{n=1}^{N} \alpha_n \mathfrak{L}(B_n) \cong Y \tag{5.34}$$

La ecuación (5.34) establece un sistema de ecuaciones con N incógnitas $\{\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_N\}$. Este tipo de problemas se puede resolver estableciendo un conjunto de funciones prueba en el dominio del operador $\{W_1, W_2, W_3, \dots, W_M\}$ y realizando el producto interno de (5.34) con cada una de ellas, obteniendo:

$$\sum_{n=1}^{N} \alpha_n \langle W_m, \mathfrak{L}(B_n) \rangle = \langle W_m, Y \rangle \tag{5.35}$$

donde aparece el producto interno que define en el dominio del operador $\mathfrak{L}$ como:

$$\langle f, g \rangle = \int f(x) g^*(x) dx \tag{5.36}$$

siendo $g^*(x)$ el complejo conjugado de $g(x)$.

Utilizando notación matricial se puede observar que la expresión dada en (5.35) representa un sistema de *M* ecuaciones con *N* incógnitas.

$$[Z_{mn}][\alpha_n] = [W_m] \tag{5.37}$$

donde

$$[Z_{mn}] = \begin{bmatrix} \langle W_1, \mathfrak{L}(B_1)\rangle & \langle W_1, \mathfrak{L}(B_2)\rangle & \cdots & \langle W_1, \mathfrak{L}(B_N)\rangle \\ \langle W_2, \mathfrak{L}(B_1)\rangle & \langle W_2, \mathfrak{L}(B_2)\rangle & \cdots & \langle W_2, \mathfrak{L}(B_N)\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle W_M, \mathfrak{L}(B_1)\rangle & \langle W_M, \mathfrak{L}(B_2)\rangle & \cdots & \langle W_M, \mathfrak{L}(B_N)\rangle \end{bmatrix} \tag{5.38}$$

$$[\alpha_n] = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix} \tag{5.39}$$

$$[W_m] = \begin{bmatrix} \langle W_1, Y\rangle \\ \langle W_2, Y\rangle \\ \vdots \\ \langle W_M, Y\rangle \end{bmatrix} \tag{5.40}$$

Si la matriz $[Z_{mn}]$ no es singular, la solución se obtiene calculando la matriz inversa:

$$[\alpha_n] = [Z_{mn}]^{-1}[W_m] \tag{5.41}$$

Aparte del modo de resolución expuesto, existe otra forma de resolver el sistema de ecuaciones mediante métodos iterativos, como por ejemplo el Método del Gradiente Conjugado o Biconjugado Estabilizado L [14]. Desde el punto de vista computacional, es más eficiente resolver el sistema hallando una solución aproximada con un grado de error admisible establecido a priori.

5.4.2.- Funciones Base y Funciones Prueba.

La discretización en subdominios que realiza el Método de los Momentos se basa en un conjunto de funciones base y un conjunto de funciones prueba. Las primeras tienen como objetivo realizar un muestreo de las incógnitas del sistema de ecuaciones, mientras que las segundas verifican posteriormente el cumplimiento de las condiciones de contorno sobre puntos o zonas discretas de la superficie de la estructura.

Según se ha comentado en el apartado anterior, para resolver la ecuación integro-diferencial hay que discretizar el problema, expresando la función incógnita en

una serie de funciones base, que pueden ser de dos tipos: de dominio completo y de subdominio. Las más usadas son las de subdominio debido a que no es necesario un conocimiento a priori de la forma de la función a la que representan.

Las funciones base se definen sobre una serie de subdominios eléctricamente pequeños, los cuales se obtienen según el algoritmo de mallado descrito en el apartado 5.3. Su función es definir cómo se modela la corriente conformada a la superficie en todos los puntos de la estructura. El uso de un tipo de función base determinado está relacionado con el tipo de modelado utilizado para representar el cuerpo. Los tres tipos de funciones más utilizados son:

- Funciones rooftop de Glisson-Wilton [15]. Las superficies se modelan mediante parches rectangulares planos. La función incógnita a determinar son las componentes de la corriente en las dos direcciones del plano que contiene a la superficie.

- Funciones triángulo de Rao-Wilton-Glisson [16]. Modelan las superficies con mayor precisión y flexibilidad que las anteriores, ya que incluyen funciones base entre parches triangulares planos.

- Funciones rooftop sobre superficies paramétricas [17]. Son las que se utilizan en el núcleo de NewFasant, aprovechando que los subdominios obtenidos en el proceso de discretización de la geometría son superficies paramétricas curvas. Estas funciones se establecen sobre dos subdominios adyacentes, de modo que la corriente fluye de un parche a otro a través del lado común a ambos subdominios, siguiendo líneas isoparamétricas. Al contrario de los dos esquemas de modelado anteriores, estas funciones no representan una aproximación faceteada de la geometría, sino que se adaptan perfectamente a la forma de la geometría original. En la figura 5.4 se puede ver un ejemplo de este tipo de funciones base.

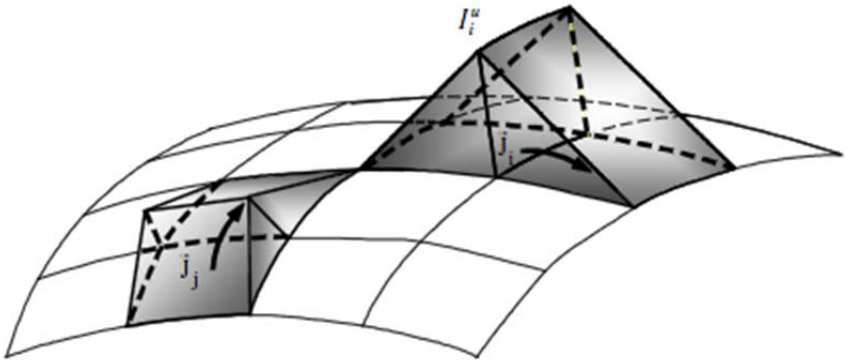


Figura 5.4.- Funciones base conformadas a la superficie. Imagen obtenida de [18].

Por su parte, las funciones prueba también se definen en función de la naturaleza de las superficies que modelan la geometría que se esté analizando. Las utilizadas dentro del núcleo de NewFasant son las funciones conocidas *razor-blade* [15], las cuales indican cómo se promedia la ecuación integral en todos los lados de la estructura.

Para su definición se consideran dos subdominios adyacentes *A* y *B*, que comparten el lado *i*. La función prueba asociada al lado une el centro de ambos subdominios a través del punto medio del lado y tiene altura unidad. Su descripción matemática es la siguiente:

$$W_i^A(u,v) = \begin{cases} 1 & si\ v = 0.5\ y\ 0.5 \leq u \leq 1 \\ 0 & en\ el\ resto \end{cases} \tag{5.42}$$

$$W_i^B(u,v) = \begin{cases} 1 & si\ v = 0.5\ y\ 0 \leq u \leq 0.5 \\ 0 & en\ el\ resto \end{cases} \tag{5.43}$$

La figura 5.5 muestra un ejemplo de función prueba conformada a la superficie bajo análisis.

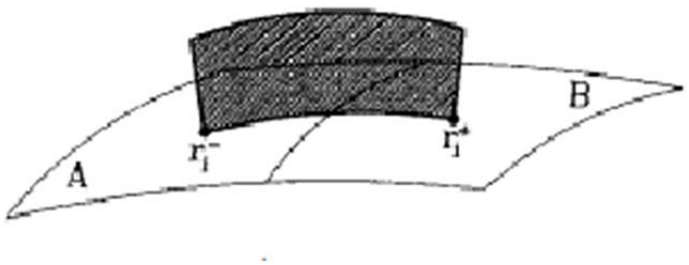


Figura 5.5.- Función prueba conformada a la superficie. Imagen obtenida de [17].

Una vez que las funciones base y las funciones prueba han sido definidas, se procede al cálculo de la matriz de impedancias del sistema de ecuaciones propio del Método de los Momentos.

5.4.3.- Matriz de Impedancias.

Como se ha comentado anteriormente, el Método de los Momentos sirve para determinar el valor de las corrientes inducidas en la superficie conductora mediante la aplicación de un operador integro-diferencial con el que se llega a un sistema de M ecuaciones y N incógnitas. Cada una de esas incógnitas representa un coeficiente que determina la contribución de cada una de las funciones base en la solución final.

Una vez obtenido el sistema de ecuaciones, el siguiente paso consiste en resolverlo de forma eficiente para obtener la matriz de impedancias. Para ello, se debe tener en cuenta que la complejidad del sistema a resolver está en función del tamaño eléctrico de la estructura bajo análisis, ya que cuanto mayor sea la estructura (definida por un conjunto de subdominios) mayor será el número de incógnitas que se tendrán que obtener. Además, hay que tener en cuenta que esta etapa de cálculo es la más crítica del proceso, ya que es donde se requiere una mayor demanda de tiempo y memoria.

Cada elemento de la matriz de impedancias representa la interacción entre dos subdominios y está formado por dos términos, los cuales se calculan de forma independiente: el término inductivo, que depende de la corriente asociada a cada subdominio y el término capacitivo, que depende de la distribución de cargas asociada a ese mismo subdominio [17, 18].

5.4.4.- Método Rápido de los Multipolos Multinivel.

A pesar de proporcionar resultados muy precisos, la desventaja del Método de los Momentos es que consume muchos recursos cuando la estructura bajo análisis es eléctricamente muy grande. Dichos recursos son, principalmente, el tiempo de CPU y la memoria de almacenamiento. A lo largo de los últimos años, la principal línea de

investigación en este campo ha sido el desarrollo de nuevos algoritmos que permitan reducir los requerimientos de tiempo y memoria.

En este apartado se describe cómo se acelera el cálculo de las corrientes mediante la aplicación del Método Rápido de los Multipolos Multinivel (MLFMA, *Multilevel Fast Multipole Method*) [19] al análisis electromagnético de cuerpos complejos modelados por superficies NURBS.

El MLFMA constituye una técnica muy rápida y eficaz para reducir los requisitos de tiempo y memoria necesarios de la aplicación del Método de los Momentos. Se trata de un método muy popular debido a la considerable reducción de recursos computacionales derivados de su aplicación en el análisis de cuerpos dispersores grandes, obteniendo buenos resultados cuando se tratan cuerpos complejos de forma arbitraria.

Los términos de agregación y desagregación se obtienen utilizando la formulación del Método Rápido de los Multipolos (FMM, *Fast Multiple Method*) [20] en el nivel más bajo siguiendo el procedimiento de interpolación para realizar el producto matriz-vector en los niveles más altos.

La aplicación del MLFMA implica la división de la geometría en cubos volumétricos o regiones de primer orden que, a su vez, se agrupan dando lugar a regiones de orden superior. Se recomienda que el tamaño de las regiones de nivel más bajo se establezca en $\lambda/4$. La figura 5.6 muestra la distribución de los cubos volumétricos dada una geometría genérica.

La ventaja de aplicar el MLFMA estriba en que no es necesario almacenar los términos de campo cercano de la matriz de acoplos. Respecto al cálculo de las interacciones entre elementos lejanos, se lleva a cabo mediante un proceso iterativo de resolución del sistema de una manera eficiente usando los términos de agregación, desagregación y traslación. De este modo, la complejidad del problema se reduce de $O(N^2)$ a $O(NlogN)$.

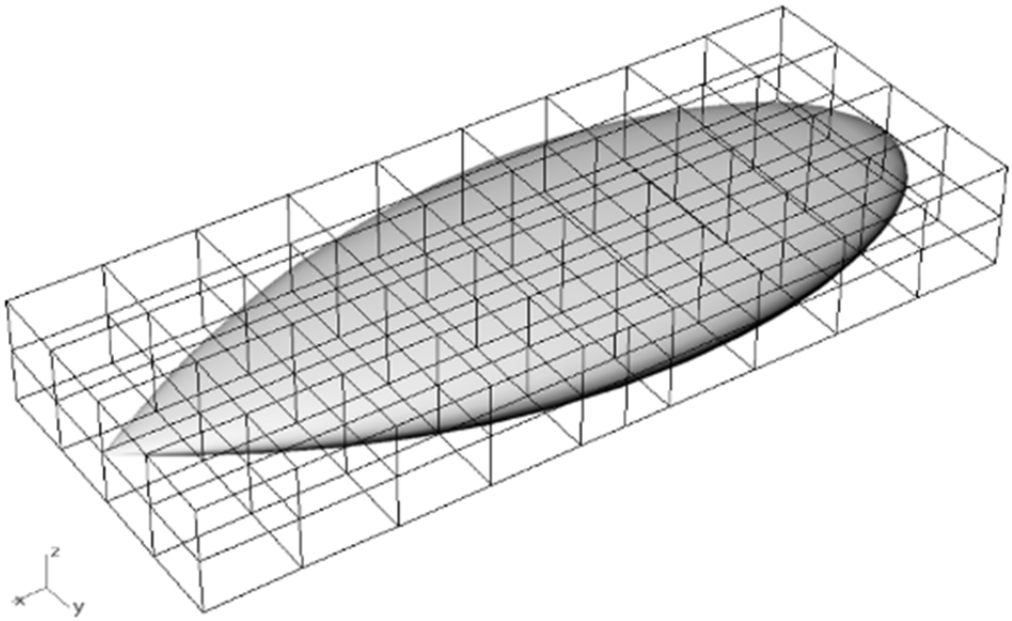


Figura 5.6.- División en regiones del MLFMA. Imagen obtenida de [21].

Por otro lado, aprovechando las ventajas que proporcionan los clusters de ordenadores, se ha llevado a cabo la paralelización del núcleo electromagnético, de modo que las simulaciones se puedan realizar sobre varios procesadores simultáneamente. Esta paralelización se ha implementado utilizando el paradigma MPI [22], que permite el intercambio de información entre procesadores por medio de mensajes, ya sea para sincronización o para intercambiar datos, con lo cual se dispone de una memoria distribuida e independiente para cada procesador, que optimiza la velocidad y el rendimiento de trabajo.

5.4.5.- Aplicación del Método de los Momentos al Cálculo de la RCS de una Esfera.

En este apartado se presenta el cálculo de la RCS biestática de una esfera de 1m de radio a 300MHz. La figura 5.7 muestra el cuadro de diálogo utilizado para establecer la configuración de la onda plana incidente. La polarización se ha fijado según $\hat{\theta}$ y la dirección de incidencia se ha fijado en la dirección normal al plano XY, es decir, viene dada por $\theta=0^{\circ}$ y $\varphi=0^{\circ}$.

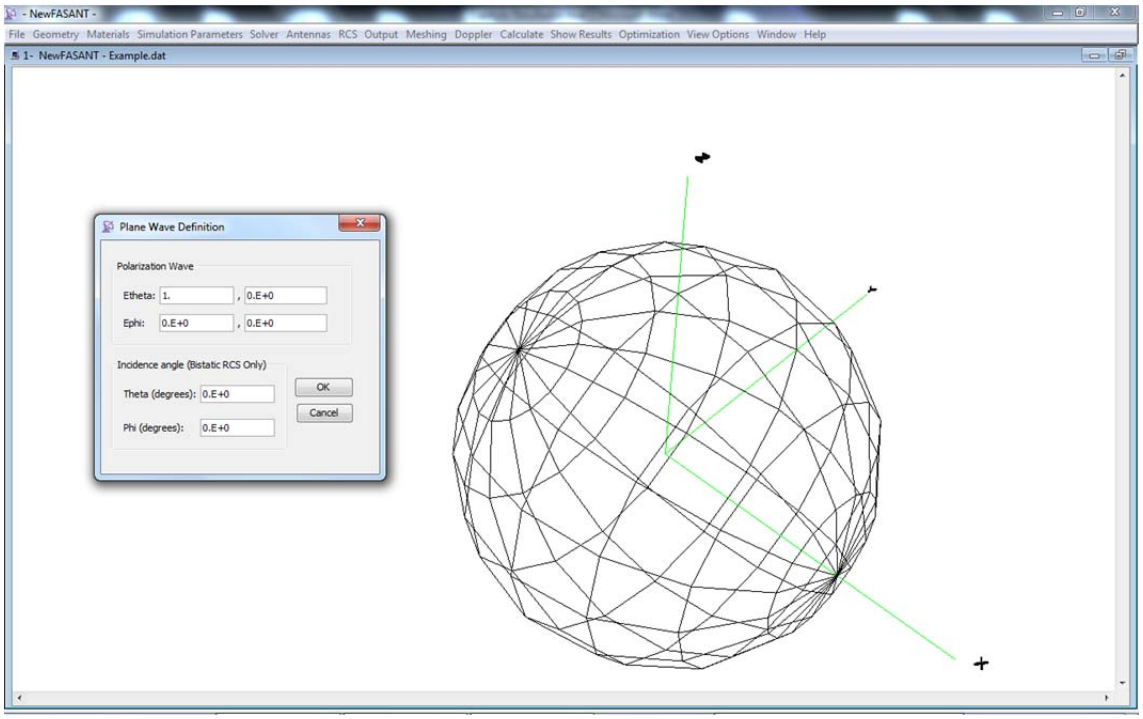


Figura 5.7.- Configuración de la polarización y de la dirección de la onda incidente.

El análisis se lleva a cabo considerando un corte en φ=0º y estableciendo un barrido angular en θ desde 0º hasta 360º. En la figura 5.8 se puede ver el modelo discretizado de la esfera requerido para iniciar la simulación.

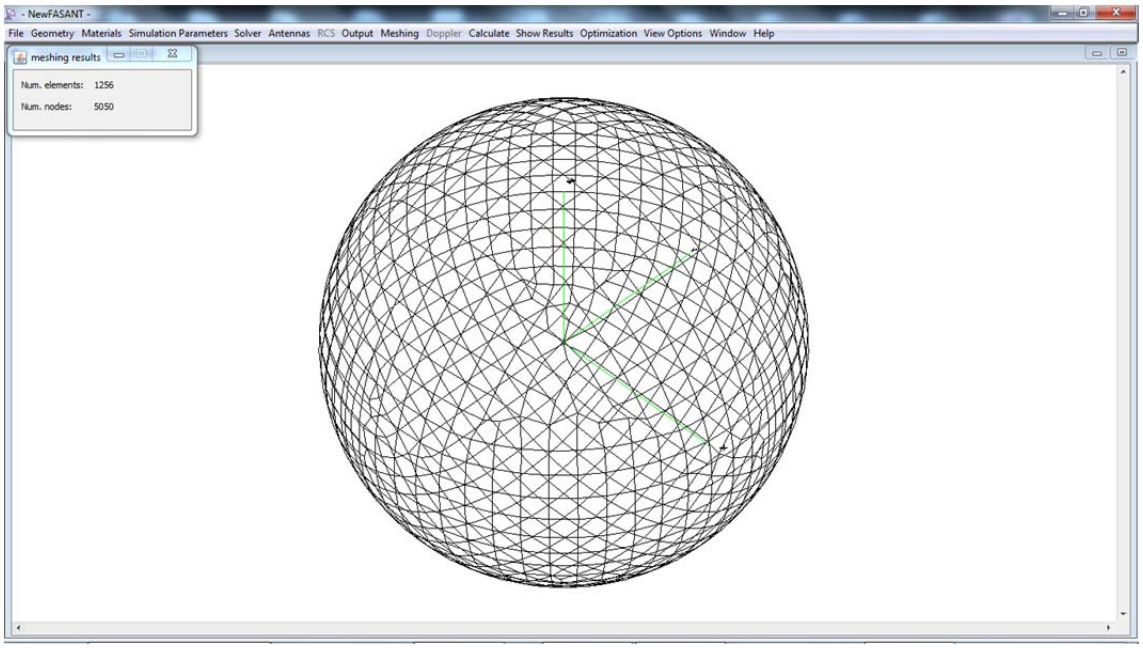


Figura 5.8.- Modelo geométrico de la esfera tras aplicar el algoritmo de mallado.

La figura 5.9 muestra el resultado de la RCS biestática para el corte φ=0º. Por otro lado, en la figura 5.10 se puede ver la distribución de corrientes sobre la superficie de la esfera.

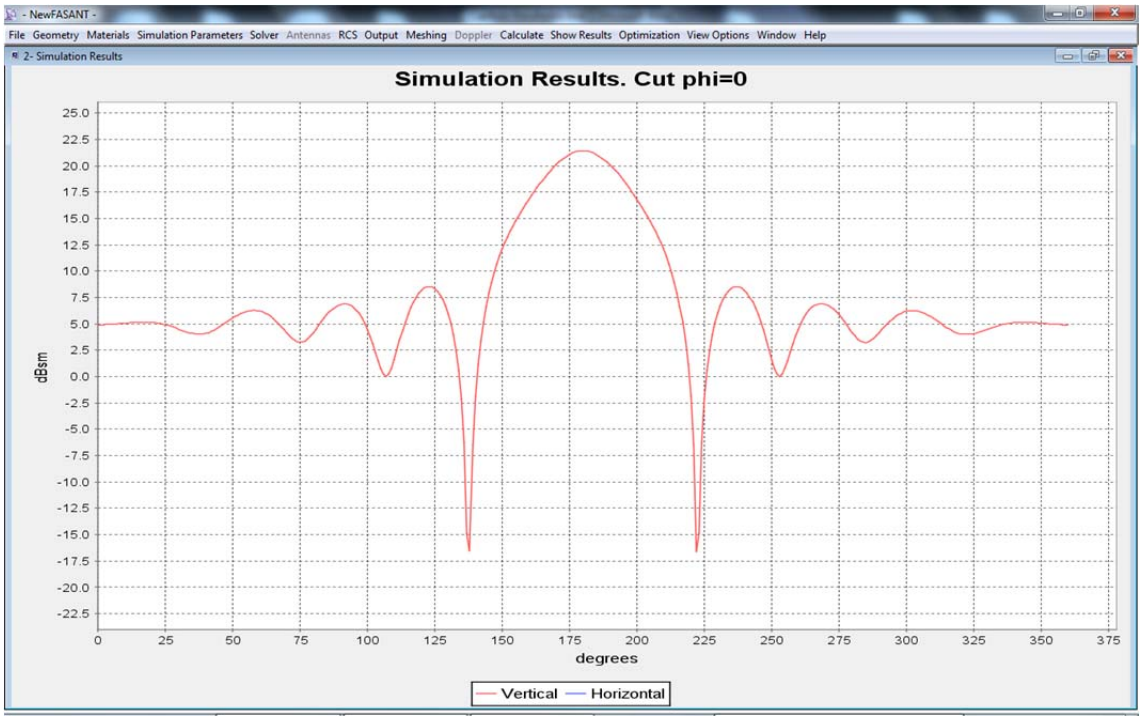


Figura 5.9.- RCS biestática en el plano E.

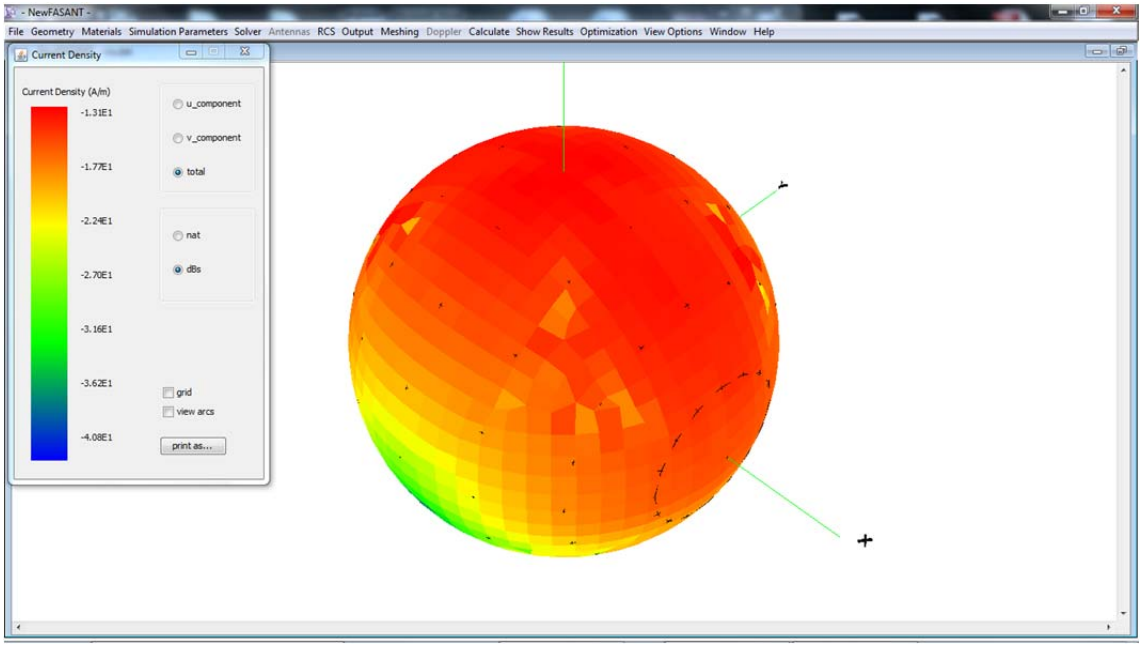


Figura 5.10.- Distribución de corriente sobre la superficie de la esfera.

5.5.- REFERENCIAS.

[1].- W. Boehm. "Generating the Bezier points of B-Spline Curves and Surfaces". *Computer Aided Design*, vol. 13, no. 32, 1981.

[2].- P. Bezier, "The Mathematical Basis of UNISURF CAD System", *Butterworths*, London, 1986.

[3].- P. Bezier, "Numerical Control: Mathematics and Applications", *Wiley*, 1972.

[4].- G. Lorentz, "Bernstein Polynomials", *Toronto Press*, Toronto, 1953.

[5].- R.T. Farouki, V.T.Rajan, "On the numerical condition of polynomials in Bernstein form", *Computer Aided Geometric Design*, vol.4, no. 3, pp. 191-216, 1987.

[6].- C. De Boor, "A Practical Guide to Splines", *Springer*, 1978.

[7].- S. Coons, "Computer Aided Geometric Design, Chaper Surface Patches and B-Spline Curves", pp. 1-16, *Academic Press*, 1974.

[8].- Ted D. Blacker and Michael B. Stephenson. "Paving: A New Approach to Automated Quadrilateral Mesh Generation". *International Journal for Numerical Methods in Engineering,* Vol. 32, 811-847 (1991).

[9].- David R. White and Paul Kinney. "Redesign of the Paving Algorithm: Robustness Enhancements through Element by Element Meshing". *In Proceedings 6th International Meshing Roundtable,* pages 323–335, Oct. 1997.

[10].- Cass, R.J., Benzley. S.E., Meyers, R.J., and Blacker, T.D., "Generalized 3D Paving: An Automated Quadrilateral Surface Mesh Generation Algorithm," *IJNME*, Vol. 39, No 9, pp. 1475-1490, May 1996.

[11].- Paul Kinney. "Clean Up: Improving Quadrilateral Finite Elements Meshes*"*. *Proceedings, 6th International Meshing Roundtable*, pp.437- 447.

[12].- R.F. Harrington. "Field Computation by Moment Methods". *MacMillan*, New York, 1968.

[13].- R.F. Harrington. "Matrix Methods for Field Problems". *Proceedings of the IEEE,* Vol. 55, pp.136-149, February 1967.

[14].- L. Valle López. "Aplicación de técnicas híbridas basadas en el método de los momentos para el análisis de Antenas en presencia de geometrías arbitrarias eléctricamente grandes modeladas con superficies NURBS". Tesis Doctoral. Universidad de Cantabria, Octubre de 1995.

[15].- A.W. Glisson, D.R. Wilton. "Simple and Efficient Numerical Methods for Problems of Electromagnetic Radiation and Scattering from Surfaces*"*. *IEEE Transactions on Antennas and Propagation*, Vol. AP-28, No.5, pp. 593-603, September 1980.

[16].- S.M. Rao, D.R. Wilton, A.W. Glisson. "Electromagnetic Scattering by Surfaces of Arbitrary Shape". *IEEE Transactions on Antennas and Propagation*, Vol. AP-30, No. 3, pp. 409-418, May 1982.

[17].- L. Valle, F. Rivas, M.F. Cátedra. "Combining the Moment Method with Geometrical Modeling by NURBS Surfaces and Bezier Patches". *IEEE Transactions on Antennas and Propagation*, Vol. 42, No. 3, pp. 373-381, March 1994.

[18].- E. García García, "Contribución al análisis de problemas electromagnéticos mediante el Método de los Momentos con bajo coste computacional", Tesis Doctoral. Universidad de Alcalá, Marzo 2005.

[19].- W. C. Chew, J. Jin, E. Michielssen, J. Song, Editors: "Fast and Efficient Algorithms in Computational Electromagnetics". *Artech House* Inc. 2001.

[20].- N. Engheta, W. D. Murphy, V. Rokhlin, and M. S. Vassiliou, "The fast multipole method (FMM) for electromagnetic scattering problems", *IEEE Trans. Antennas Propagat*., vol. 40, no. 6, pp. 634-641, Jun. 1992.

[21].- Garcia, E.; Delgado, C.; Diego, I.G.; Catedra, M.F.; , "An Iterative Solution for Electrically Large Problems Combining the Characteristic Basis Function Method and the Multilevel Fast Multipole Algorithm," *Antennas and Propagation, IEEE Transactions on* , vol.56, no.8, pp.2363-2371, Aug. 2008.

[22].- http://www.mcs.anl.gov/research/projects/mpi/

6.- Ejecución y Optimización.

6.1.- INTRODUCCIÓN.

Este capítulo presenta el proceso de ejecución, que incluye la configuración de los parámetros de simulación, el análisis electromagnético de la geometría y la representación de los resultados obtenidos. También se describe el proceso de optimización, que obtiene las dimensiones óptimas del modelo geométrico, en función de unas ciertas especificaciones representadas mediante una función de coste. Como se verá a lo largo del capítulo, ambos procesos pueden llevarse a cabo en modo local o en modo remoto.

6.2.- PARÁMETROS DE SIMULACIÓN.

Este apartado describe la primera fase del proceso de ejecución en la que se especifican los parámetros de simulación necesarios para ejecutar el núcleo electromagnético.

6.2.1.- Parámetros Generales.

Antes de iniciar el análisis, el sistema configura los parámetros que contienen información general sobre la simulación. En primer lugar, se especifica la localización del archivo de geometría que almacena la descripción del modelo. A continuación se indica el tipo de simulación, que puede ser una de las siguientes:

- Análisis de RCS.
  Existen dos opciones de cálculo de RCS: monoestática, cuando el emisor y el receptor se encuentran en el mismo punto, y biestática, cuando se encuentran en diferentes posiciones angulares.

- Análisis de antenas.

  Dependiendo del tipo de alimentación seleccionado, se pueden realizar tres tipos de análisis: a través de un conjunto de dipolos, importando un patrón de radiación externo o mediante puntos de alimentación dados por un voltaje y una impedancia determinados.

- Análisis de circuitos.

  Este tipo de simulación permite analizar circuitos microstrip definidos sobre una o más capas de estructuras dieléctricas o conductoras.

- Análisis de coeficientes de transmisión y reflexión.

El siguiente parámetro a especificar es el método de resolución aplicado, que puede ser el Método de los Momentos o la técnica de Física Óptica. Ésta se ha introducido en el núcleo para analizar estructuras eléctricamente grandes a altas frecuencias. Se trata de una técnica asintótica que no presenta los problemas de los métodos rigurosos como el Método de los Momentos ante ese tipo de análisis.

A continuación, si se ha indicado la opción de resolución mediante el Método de los Momentos, se elige el tipo de ecuación integral que resuelva el problema electromagnético, EFIE, MFIE o CFIE. Cada una de ellas plantea diferentes condiciones de contorno sobre la estructura: en la EFIE, el campo tangencial eléctrico total debe ser nulo sobre la superficie de la estructura, mientras que la MFIE fuerza la condición de contorno que se establece con las componentes tangenciales del campo magnético. La CFIE es una combinación lineal de las dos anteriores.

Por defecto, la ecuación integral establecida es la de campo eléctrico, pues incluye la formulación general y se puede aplicar tanto a cuerpos abiertos como a cuerpos cerrados. En cambio, la CFIE, sólo se puede aplicar sobre cuerpos cerrados que dispongan sus vectores normales hacia el exterior. En los casos en que este tipo de ecuación es aplicable, se convierte en una solución particularmente útil, sobre todo cuando se usa conjuntamente con el método rápido de los multipolos, dado que requiere menos memoria y converge mucho antes que la solución basada en la EFIE. El sistema permite aplicar la EFIE sobre ciertos objetos de la geometría, analizando el resto con la

CFIE. De este modo, se pueden distinguir dos tipos de análisis dependiendo de las características geométricas del modelo a analizar. En una misma simulación, para analizar objetos cerrados se puede aplicar la CFIE, mientras que para analizar objetos abiertos se puede aplicar EFIE.

La figura 6.1 muestra el cuadro de diálogo que aparece cuando se asigna la EFIE a una estructura periódica de cruces metálicas, habiendo establecido un análisis general mediante CFIE para todos los objetos del modelo. De este modo, el análisis de la bocina corrugada, al tratarse de un cuerpo volumétrico y cerrado, se realiza aplicando la CFIE, mientras que el análisis de la estructura periódica se realiza mediante la EFIE.

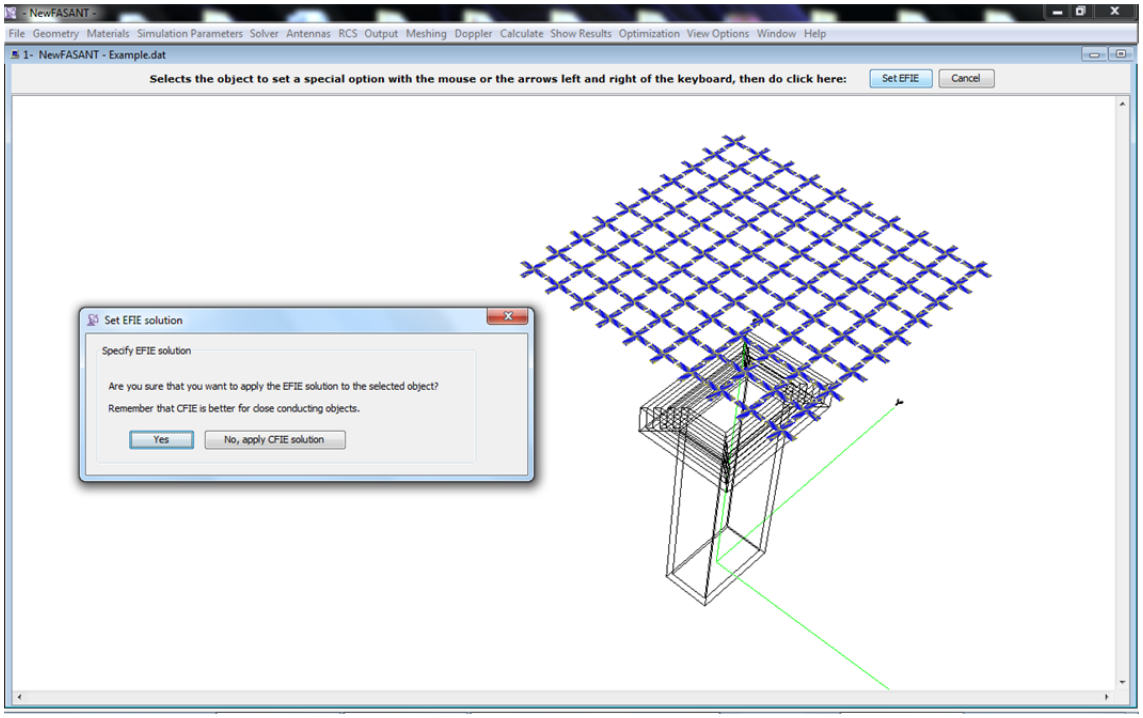


Figura 6.1.- Asignación de la EFIE a una parte de un modelo geométrico.

Aparte de indicar el tipo de ecuación integral, también se deben especificar otros parámetros relacionados con el método de resolución, como son: el número máximo de iteraciones que se pueden ejecutar durante el proceso de resolución de la matriz de impedancias, el error máximo admisible establecido en el método del gradiente conjugado y el número de divisiones por longitud de onda. Este último valor hace referencia a la densidad de mallado, estableciéndose normalmente a un valor típico de 10 divisiones. Dicho valor es aplicable sobre todos los objetos que componen la geometría bajo análisis. Sin embargo, hay situaciones en las que, debido a las

características del modelo geométrico, es preferible establecer un menor número de divisiones a una determinada parte de la geometría. Este es el caso, por ejemplo, de una antena reflectora iluminada por una bocina. En situaciones como esta, en las que la diferencia de tamaño es tan notable, el sistema ofrece la posibilidad de especificar un menor número de divisiones sobre el objeto de mayores dimensiones, siempre que éste no presente detalles geométricos pequeños, lo que implicaría realizar una discretización más densa en dichas zonas.

En la figura 6.2 se puede ver un ejemplo en el que se establecen 5 divisiones por longitud de onda para realizar el análisis de la superficie reflectora, mientras que para analizar la bocina se mantiene el número de divisiones especificado en la configuración general, cuyo valor por defecto es de 10.

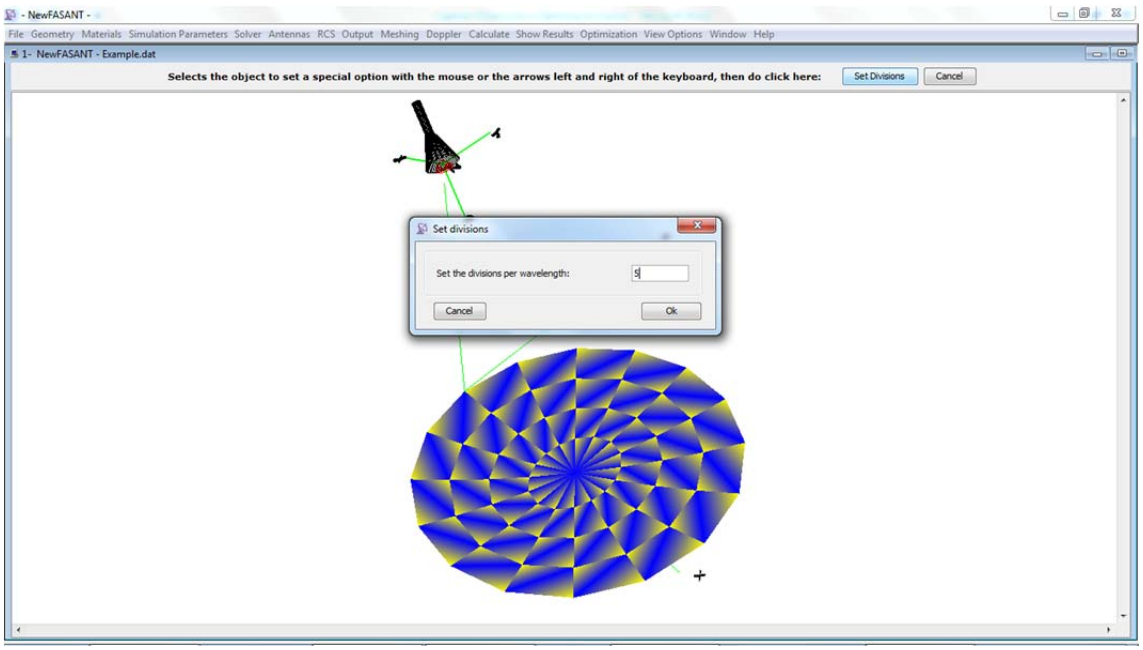


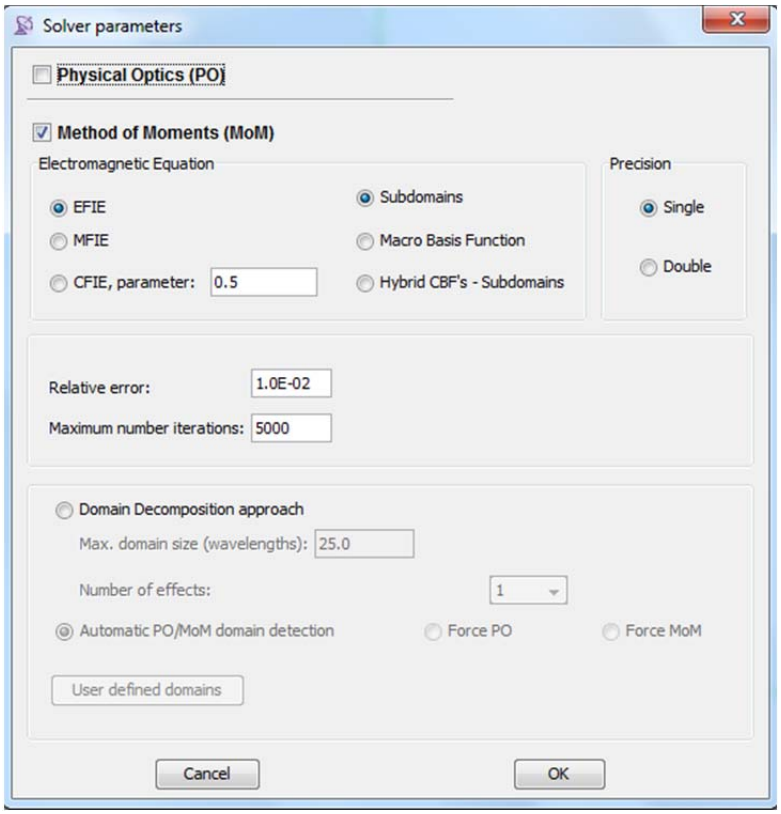
Figura 6.2.- Asignación de 5 divisiones por longitud de onda a una parte del modelo geométrico.

La figura 6.3 muestra el cuadro de diálogo en el que se establece la configuración del método de resolución. En la imagen se puede apreciar que después de especificar el tipo de ecuación integral, aparecen tres posibilidades de análisis:

- *Subdomains*: hace referencia a la implementación tradicional del Método de los Momentos discretizando la geometría en función de subdominios, sobre los que aplicar las funciones base, tal y como se comentó en el capítulo anterior.

- *Macro Basis Function*: se trata de una nueva técnica implementada [1], cuyo objetivo es minimizar el número de incógnitas de la matriz de impedancias. Para ello, se define un determinado conjunto de funciones base de alto nivel, que modelan las corrientes inducidas en la superficie de la estructura bajo análisis, obteniendo una matriz de acoplos de menor tamaño.

- *Hybrid CBFs – subdomains*: esta opción permite aplicar una combinación de los dos métodos anteriores.



Fig**ura 6.3.-** Cuadro de diálogo para establecer el método de resolución del problema electromagnético.

En la parte inferior de la figura 6.3 se puede seleccionar una técnica de análisis distinta a las anteriores, denominada DDM (*Domain Decomposition Method*), que consiste en la descomposición de la geometría en bloques volumétricos llamados ventanas [2]. Su propósito coincide con el de las técnicas mencionadas anteriormente: minimizar los recursos que consume el Método de los Momentos cuando la estructura bajo análisis es eléctricamente muy grande. La novedad que presenta este método respecto a los anteriores es que soluciona los posibles problemas de convergencia que

pueden surgir en las zonas de la geometría débilmente acopladas. Para ello, este método incluye técnicas de alta frecuencia, como el trazado de rayos.

A continuación se indica la frecuencia de análisis, que puede ser un único valor o varios, puesto que se ofrece la posibilidad de realizar un barrido de frecuencias indicando el margen de variación y el incremento.

En cuanto al método MLFMA [3], el sistema asigna regiones volumétricas de tamaño λ/4, estableciendo un número de niveles determinado, hasta englobar a la geometría en un solo cubo.

Por último, se especifican los cortes del diagrama de radiación en campo lejano que se desean visualizar cuando finalice la simulación. A pesar de que por defecto se pueden visualizar tanto la distribución de corrientes como la distribución de cargas sobre la superficie de la estructura, los diagramas de campo en 2D deben indicarse como un parámetro más de simulación. Se pueden definir tantos cortes como se deseen en ambas coordenadas θ y φ, especificando su valor, y la variación angular de la otra coordenada en función de un valor inicial, un incremento angular y un número de muestras. Como se verá en el apartado 6.5, también es posible visualizar el diagrama de radiación y la ganancia en 3D.

6.2.2.- Parámetros de Alimentación.

Dependiendo del tipo de simulación especificado, se debe establecer un tipo de alimentación u otro. Por ejemplo, para el caso de RCS, ya sea monoestática o biestática, es necesario especificar los parámetros relacionados con la iluminación mediante un frente de onda plana. Según este tipo de alimentación, el campo eléctrico incidente viene dado por:

$$\overrightarrow{E_i}(\vec{r}) = \overrightarrow{E_0} \cdot e^{-j\overrightarrow{\beta_0}\vec{r}} \tag{6.1}$$

donde $\overrightarrow{\beta_0}$ es la dirección de incidencia y $\overrightarrow{E_0}$ es el vector de polarización.

Cuando el tipo de simulación establecido es análisis de antenas, hay varias posibilidades para determinar la alimentación del modelo. Por ejemplo, el caso más común es definir uno o más dipolos, configurando su amplitud, fase, posición y orientación, en base al sistema absoluto representado por el sistema antena. Para indicar la orientación, de los dipolos o del sistema antena, hay dos posibilidades. La primera consiste en especificar los valores de los ángulos θ_Z y φ_Z del eje Z' de la antena y un ángulo φ_X para orientar los ejes X' e Y', y la segunda consiste en aplicar una matriz de cosenos directores. Ambos modos de definición se relacionan mediante las siguientes expresiones:

$$M = \begin{bmatrix} Dx(x) & Dx(y) & Dx(z) \\ Dy(x) & Dy(y) & Dy(z) \\ Dz(x) & Dz(y) & Dz(z) \end{bmatrix} \tag{6.2}$$

donde *M* es la matriz de cosenos directores y *Dx*, *Dy* y *Dz* son los vectores de dirección que indican la orientación del dipolo. Su valor se obtiene según las expresiones (6.3)-(6.5).

$$Dx = \begin{bmatrix} S_1(x)\cdot T_1(x) + S_1(y)\cdot T_2(x) + S_1(z)\cdot T_3(x) \\ S_1(x)\cdot T_1(y) + S_1(y)\cdot T_2(y) + S_1(z)\cdot T_3(y) \\ S_1(x)\cdot T_1(z) + S_1(y)\cdot T_2(z) + S_1(z)\cdot T_3(z) \end{bmatrix} \tag{6.3}$$

$$Dy = \begin{bmatrix} S_2(x)\cdot T_1(x) + S_2(y)\cdot T_2(x) + S_2(z)\cdot T_3(x) \\ S_2(x)\cdot T_1(y) + S_2(y)\cdot T_2(y) + S_2(z)\cdot T_3(y) \\ S_2(x)\cdot T_1(z) + S_2(y)\cdot T_2(z) + S_2(z)\cdot T_3(z) \end{bmatrix} \tag{6.4}$$

$$Dz = \begin{bmatrix} S_3(x)\cdot T_1(x) + S_3(y)\cdot T_2(x) + S_3(z)\cdot T_3(x) \\ S_3(x)\cdot T_1(y) + S_3(y)\cdot T_2(y) + S_3(z)\cdot T_3(y) \\ S_3(x)\cdot T_1(z) + S_3(y)\cdot T_2(z) + S_3(z)\cdot T_3(z) \end{bmatrix} \tag{6.5}$$

siendo S_1, S_2, S_3, T_1, T_2 y T_2:

$$S_1 = [\cos\varphi_x \quad -\sin\varphi_x \quad 0] \tag{6.6}$$

$$S_2 = \left[\cos\left(\varphi_x + \frac{\pi}{2}\right) \quad -\sin\left(\varphi_x + \frac{\pi}{2}\right) \quad 0\right] \tag{6.7}$$

$$S_3 = [0 \quad 0 \quad 1] \tag{6.8}$$

$$T_1 = \begin{bmatrix} \sin\left(\theta_z + \frac{\pi}{2}\right) \cdot \cos\varphi_z \\ \sin\left(\theta_z + \frac{\pi}{2}\right) \cdot \sin\varphi_z \\ \cos\left(\theta_z + \frac{\pi}{2}\right) \end{bmatrix} \tag{6.9}$$

$$T_3 = \begin{bmatrix} \sin\theta_z \cdot \cos\varphi_z \\ \sin\theta_z \cdot \sin\varphi_z \\ \cos\theta_z \end{bmatrix} \tag{6.10}$$

$$T_2 = T_1 \times T_3 \tag{6.11}$$

Los dipolos pueden ser eléctricos o magnéticos y se pueden definir tantos como sean necesarios. En la figura 6.4 se muestran los cuadros de diálogo implementados para establecer la configuración de este tipo de alimentación. En el ejemplo de la imagen se ha establecido el sistema antena en el punto central de la apertura de la bocina, indicando el origen de fase, con la orientación por defecto. También se han especificado dos dipolos, cada uno caracterizado por sus propios parámetros de orientación, posición, amplitud y fase.

Para el caso de las antenas creadas mediante el módulo descrito en el capítulo 4, no es necesario especificar ningún tipo de alimentación, ya que se define automáticamente cuando se genera el modelo geométrico de las antenas.

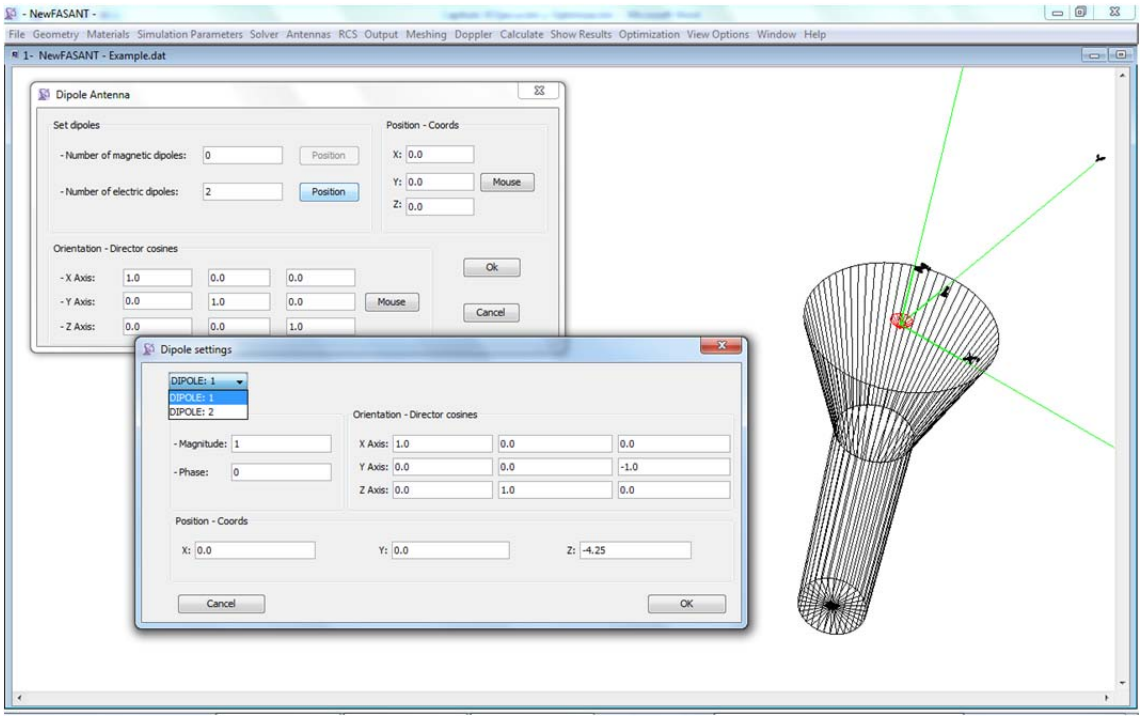


Figura 6.4.- Cuadros de diálogo para establecer la configuración de los dipolos que alimentan una antena de bocina.

Otra posibilidad para establecer la iluminación de una antena es mediante un diagrama de radiación externo. En ese caso, se debe especificar su posición y la orientación que debe tener el diagrama de radiación. El diagrama puede tener distintos formatos (simetría de revolución REV, simetría de semi-revolución RV2 o diagrama tridimensional 3DE) y distintas polarizaciones, lineal o circular. La opción de análisis mediante un diagrama de radiación de antena es muy útil, por ejemplo, cuando se desea analizar una antena reflectora. En vez de incluir una bocina física en el modelo geométrico, se puede establecer esta opción de alimentación, evitando tener que analizar la bocina. A continuación se muestra un extracto del archivo en el que se especifican los valores del diagrama de radiación de la antena de tipo RV2 donde se define el plano E y el plano H de la bocina.

```
RV2
LIN
Eth

 .0  181   1.0    .0    1    .0

  .0          .00000   .0         .00000   180.0
 1.0          .00000   .0         .00000   180.0
```

```
 2.0      .00000   .0      .00000   180.0
 3.0      .00000   .0      .00000   180.0
 4.0      .00000   .0      .00000   180.0
 5.0     -.10000   .0      .00000   180.0
 6.0     -.10000   .0     -.10000   180.0
 7.0     -.10000   .0     -.10000   180.0
 8.0     -.20000   .0     -.10000   180.0
 9.0     -.20000   .0     -.20000   180.0
10.0     -.30000   .0     -.20000   180.0
 ...        ...     ...        ...     ...

175.0    -22.40000  .0    -20.60000  180.0
176.0    -22.40000  .0    -20.30000  180.0
177.0    -22.40000  .0    -20.10000  180.0
178.0    -22.40000  .0    -19.90000  180.0
179.0    -22.40000  .0    -19.60000  180.0
180.0    -22.40000  .0    -19.50000  180.0
```

En primer lugar se especifica el tipo de diagrama y la polarización. Después se indican las direcciones en θ (barrido angular de 181 muestras desde 0º con incrementos de 1º) y en φ (una única muestra en 0º). Por último se muestran los valores de campo eléctrico en el plano E (columnas 2 y 3) y en el plano H (columnas 4 y 5) en términos de amplitud y fase.

La tercera y última alternativa para establecer la excitación de una antena es la alimentación por campo impreso, a través de puntos de alimentación, determinados mediante un voltaje y una impedancia. Esta opción es ampliamente utilizada, sobre todo para alimentar antenas de hilo, tal y como se explicó en el capítulo 4. Al igual que ocurre con el número de dipolos, el sistema permite establecer un número arbitrario de puntos de alimentación. Para indicar el borde donde se desea aplicar el voltaje, se seleccionan las dos superficies que comparten dicho borde.

En la figura 6.5 se muestra un ejemplo de este tipo de alimentación, en el que se analiza una antena espiral circular a una frecuencia de 100MHz. En la parte izquierda de la figura se puede ver que el campo impreso se ha definido mediante un 1V y 0Ω. Dicha información se almacena en la tabla mostrada en la figura 6.6, junto con los identificadores de las dos superficies involucradas. En las figuras 6.7 y 6.8 se pueden ver los resultados obtenidos tras finalizar la simulación.

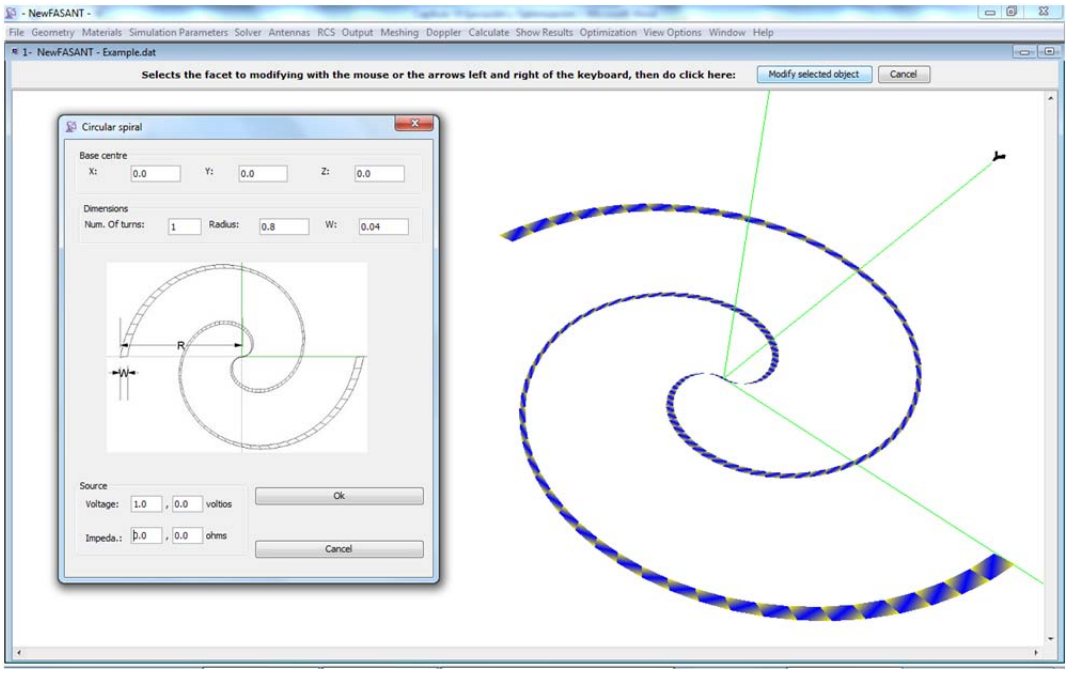


Figura 6.5.- Modelo geométrico de la antena alimentada por campo impreso.

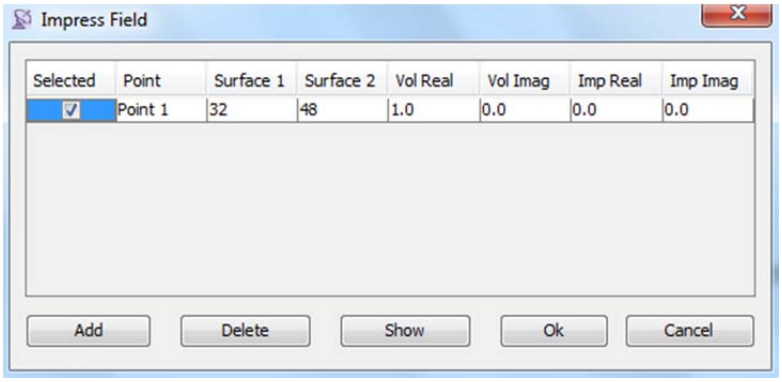


Figura 6.6.- Cuadro para establecer la alimentación por campo impreso.

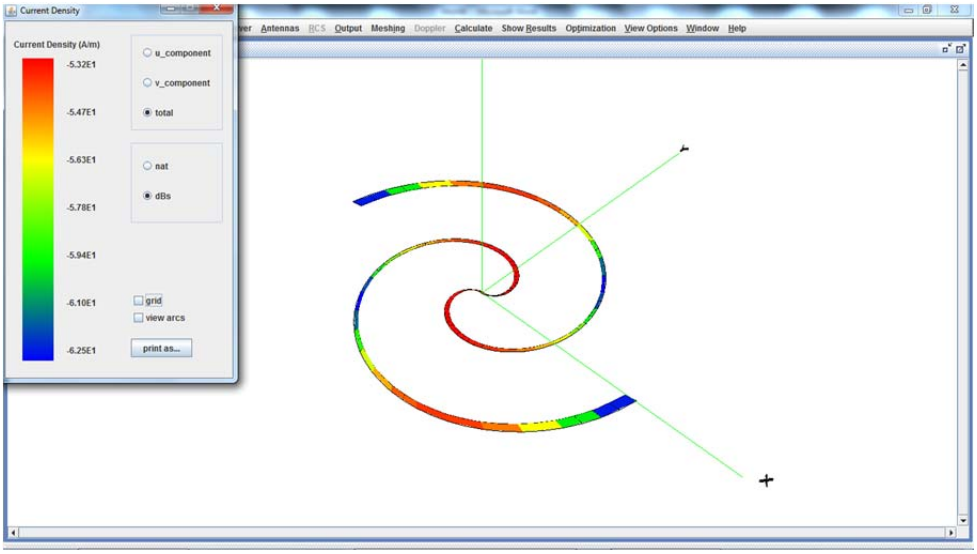


Figura 6.7.- Densidad de corriente sobre la superficie de la espiral.

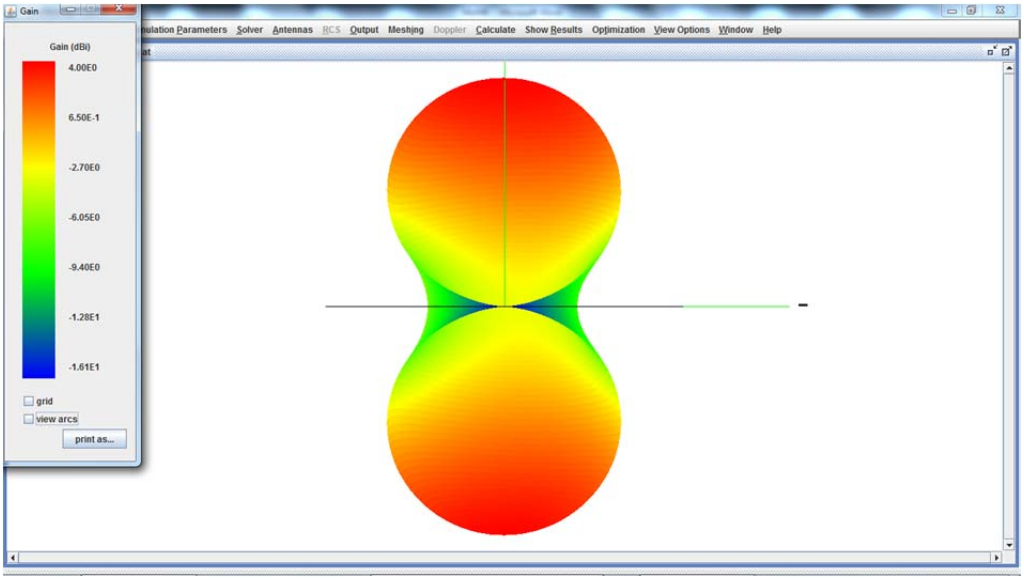


Figura 6.8.- Vista del perfil del diagrama de radiación de la antena.

6.2.3.- Parámetros de Materiales.

Por defecto, se asume que todos los objetos creados en la herramienta están formados por superficies metálicas conductoras PEC. Sin embargo, se pueden definir otros materiales, a partir de parámetros como la permitividad eléctrica relativa, la permeabilidad magnética relativa, el número de capas y el espesor. También es posible indicar el modo de análisis de las superficies que tengan asignado un material distinto del PEC. Las dos opciones disponibles son modo volumétrico o aproximación de capa fina [4]. El fichero de materiales generado almacena la información referente a cada material, junto con los índices de las superficies que tengan asignado dicho material.

Se puede definir un número indeterminado de materiales, de modo que se disponga de una base de datos, donde cada uno de ellos se identifique mediante un nombre y se incluya su definición particular: número de capas, espesor, permitividad y permeabilidad. La figura 6.9 muestra la ventana utilizada para definir las propiedades de los materiales. Se puede observar que es posible distinguir las capas definidas por medio de su color.

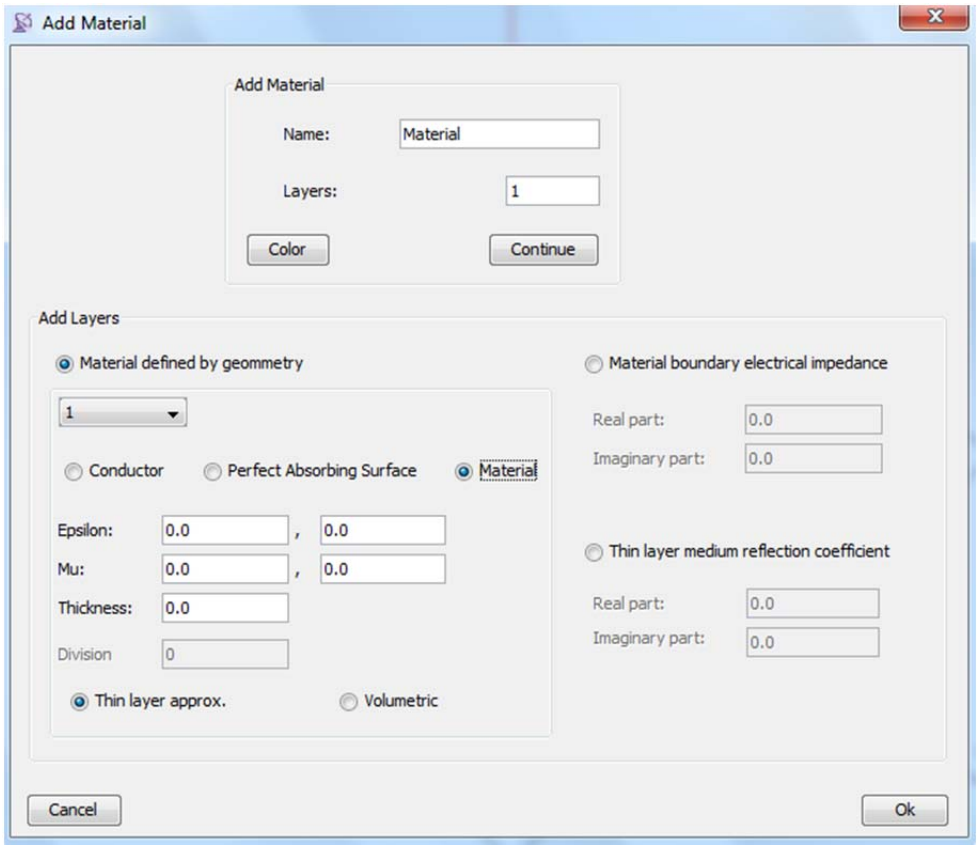


Figura 6.9.- Definición de puntos de observación sobre una superficie esférica para determinar cuál es el campo dispersado por un aerogenerador.

6.2.4.- Parámetros de Observación.

Los parámetros de observación para simulaciones en campo cercano se pueden fijar a lo largo de rectas, planos, superficies esféricas o superficies cilíndricas. También existe la posibilidad de cargar un fichero externo que incluya las coordenadas de los puntos de observación. Estableciendo la configuración de un modo u otro, se debe crear un archivo en el que se indique el número de puntos de observación generados y a continuación se incluyan las coordenadas *x, y, z* de cada uno de ellos.

En la figura 6.10 se puede ver un ejemplo en el que se analiza el campo dispersado por un aerogenerador y se establecen puntos de observación sobre dos esferas de radio 60m y 70m respectivamente.

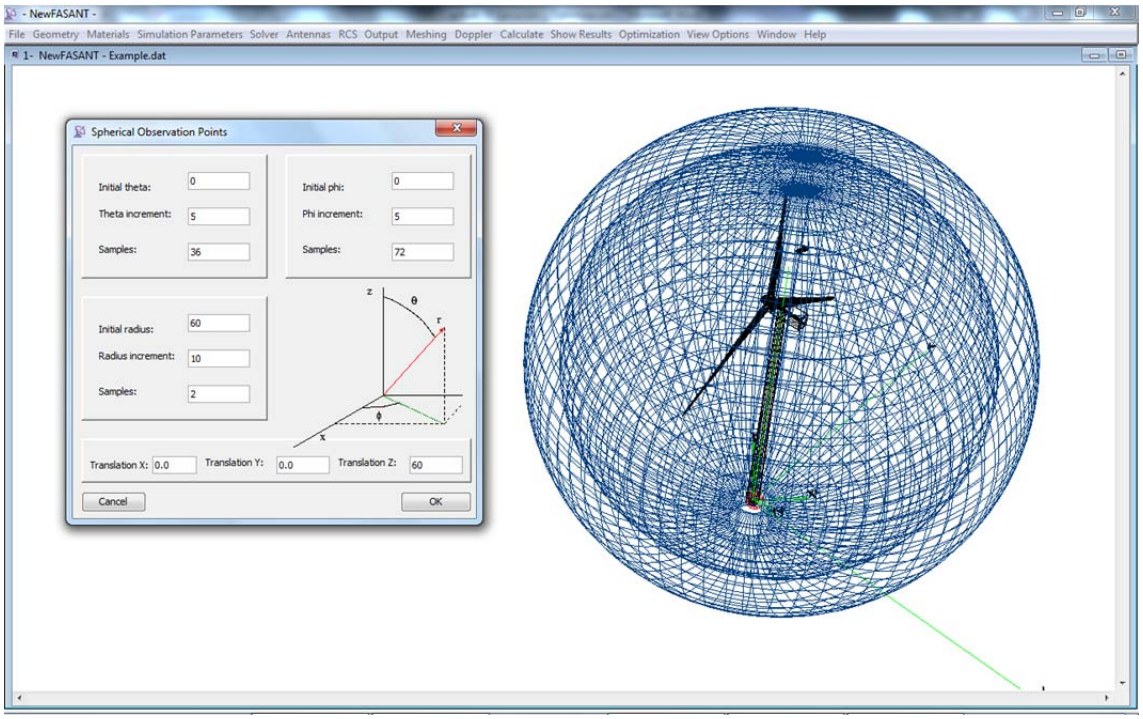


Figura 6.10.- Definición de puntos de observación sobre una superficie esférica para determinar cuál es el campo dispersado por un aerogenerador.

La figura 6.11 muestra otro ejemplo, en el que los puntos de observación se han establecido sobre un plano en Z constante, tal y como se indica en la imagen.

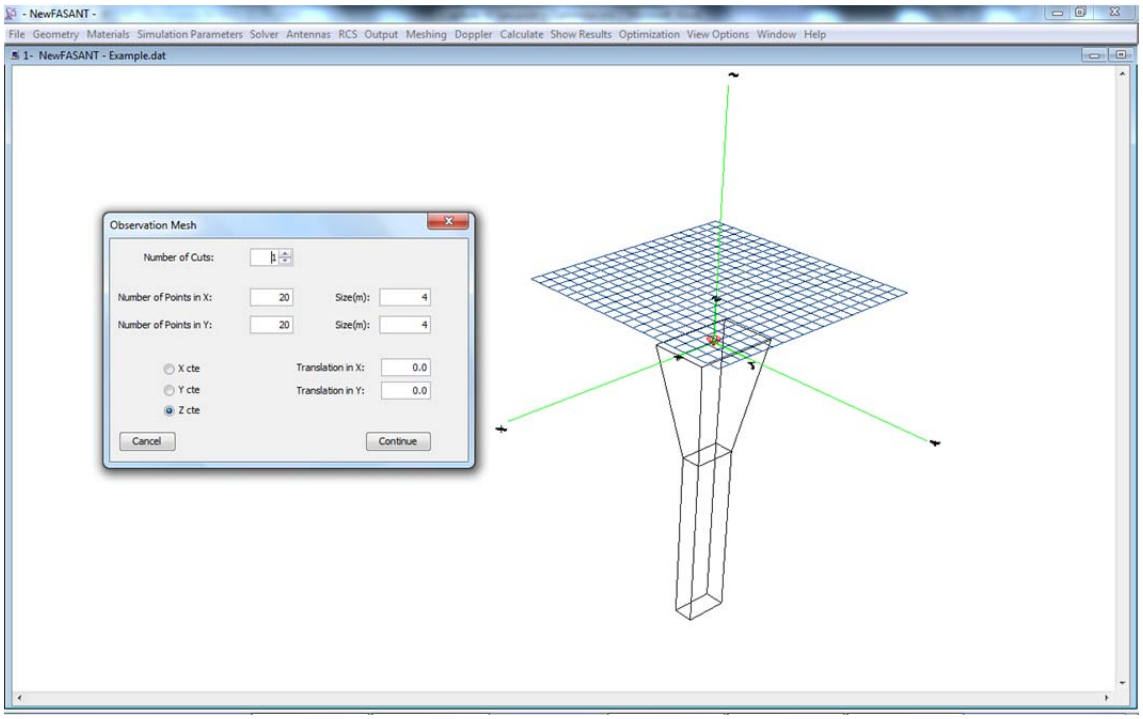


Figura 6.11.- Definición de puntos de observación sobre un plano para determinar cuál es el valor del campo radiado por una bocina piramidal en campo cercano.

6.3.- EJECUCIÓN LOCAL DEL NÚCLEO ELECTROMAGNÉTICO.

Después de establecer todos los parámetros de configuración necesarios para poder iniciar la simulación, se ejecuta el algoritmo de mallado, de modo que se obtenga un modelo geométrico discretizado sobre el que poder aplicar el Método de los Momentos.

Ejecutar el núcleo electromagnético de forma local implica que la simulación se lleve a cabo en el mismo equipo en el que está instalado el sistema. Debido a que hoy en día es muy común que los equipos incluyan más de un procesador, también se da la opción de ejecutar una versión paralelizada del núcleo. De este modo, se reduce el tiempo de cálculo considerablemente, sobre todo cuando se trata de analizar estructuras eléctricamente muy grandes.

En el siguiente apartado se presentan los aspectos más relevantes considerados en la implementación de esta parte del proceso de ejecución local.

6.3.1.- Características de la Implementación.

Hoy en día, cualquier aplicación permite que el flujo del programa se divida en varios sub-flujos, de tal forma que cada uno realice una tarea determinada de forma independiente y simultánea. En el caso que nos ocupa, la GUI se ejecuta sobre un hilo principal que se encarga de la comunicación con los usuarios. Simultáneamente, se puede ejecutar otro proceso mediante la creación de un nuevo hilo, de modo que hasta que éste no finalice, ambos estarán ejecutándose de forma paralela.

La creación de nuevos hilos se realiza para llevar a cabo varias funciones dentro del sistema, siendo la más importante la ejecución del núcleo electromagnético. Durante una simulación, el control del programa pasa al hilo creado, quedando el principal en segundo plano, de forma que el usuario tenga la posibilidad de abortar la simulación. Cuando la ejecución finaliza, la Máquina Virtual de Java elimina al hilo secundario y devuelve el control del programa al hilo principal.

En la figura 6.12 se puede observar el cuadro mostrado por la GUI mientras se está ejecutando el núcleo, en el que se informa del estado de la ejecución. Entre otros datos, se muestran la hora y fecha de inicio, el número total de incógnitas, la fase del método de resolución que se está ejecutando en cada instante, etc.

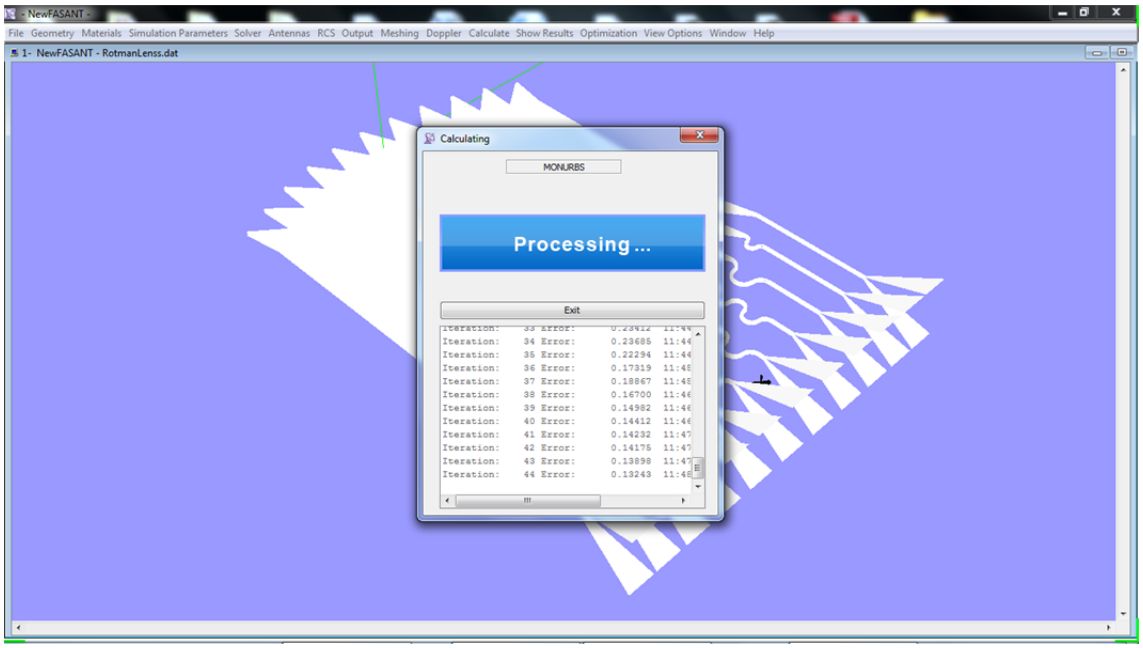


Figura 6.12.- Cuadro informativo del estado de la simulación en proceso.

Para implementar esta parte de la herramienta ha sido necesario el uso de los hilos (*Thread*) de Java, los cuales son creados y gestionados por la Máquina Virtual de Java. En el paquete *java.lang* se pueden encontrar las clases e interfaces que dan soporte al manejo de hilos. La forma directa de crear un hilo es extender de la clase *Thread* y redefinir el método *run()*. Sin embargo, cuando la clase que se desea ejecutar en un hilo independiente hereda de alguna otra clase, no puede extender a la vez la clase *Thread* y otra más, debido a que Java no soporta la herencia múltiple. En estos casos, los hilos se pueden crear implementando la interfaz *Runnable*.

El ciclo de vida de un hilo comienza cuando se invoca el método *start()*. En ese momento, la Máquina Virtual de Java establece los recursos necesarios y ejecuta el método *run()*. A continuación, el hilo puede invocar al método *wait()*, de forma que quede a la espera de que se produzca una condición de finalización. Por ejemplo, para el caso del algoritmo de mallado, dicha condición consiste en la creación del fichero que

contenga la geometría discretizada. Por último, el hilo muere cuando finaliza el método *run()*.

6.4.- EJECUCIÓN REMOTA DEL NÚCLEO ELECTROMAGNÉTICO.

Cuando el caso a analizar es muy grande en términos de longitud de onda, la mejor solución es ejecutar la simulación en máquinas más potentes (servidores o *clusters*) con mucha más memoria que un ordenador de mesa convencional, y que incluyen varias decenas o incluso centenas de procesadores, para distribuir la carga computacional y ejecutar el núcleo en paralelo.

Los *clusters* están formados por un conjunto de procesadores, conectados entre sí mediante una red de alta velocidad, que trabajan en paralelo para proporcionar un mayor rendimiento, disponibilidad y eficiencia, de tal forma que el conjunto sea visto como un único ordenador.

En los siguientes apartados se describe cómo se lleva a cabo el proceso de ejecución remota del núcleo electromagnético haciendo uso de los servidores disponibles en el grupo de investigación, los protocolos involucrados para realizar la conexión, la transferencia de ficheros, etc.

6.4.1.- Características de la Implementación.

Desde un punto de vista práctico, el mecanismo de ejecución remota presenta algunas diferencias con respecto al proceso de ejecución local. La primera y fundamental es que el núcleo no se ejecuta en el ordenador local, sino que la simulación se lleva a cabo en el servidor. Una vez que ésta finaliza, los resultados son devueltos al ordenador local, donde son procesados y visualizados. Para que el acceso a la máquina remota sea posible, hay que realizar una solicitud de conexión con el servidor especificando los datos de acceso mostrados en la figura 6.13.

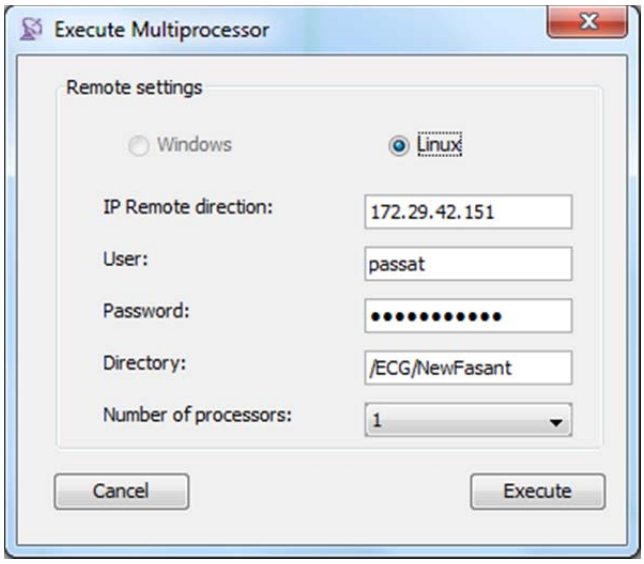


Figura 6.13.- Datos necesarios para acceder al servidor de forma remota.

Si la información especificada es válida, el sistema establece la conexión y se realiza la transferencia de los ficheros de configuración, necesarios para ejecutar el núcleo. A continuación, se inicia la simulación teniendo en cuenta tantos procesadores como se hayan especificado en el cuadro de la figura 6.13. Tanto para el modo de ejecución local como para el modo de ejecución remoto, el sistema detecta el número de procesadores disponibles en el equipo, ofreciendo la posibilidad de elegir cuántos se quieren usan en cada simulación.

Durante la simulación remota, el aspecto de la GUI es el mismo que en el caso de una ejecución local. A pesar de que internamente el proceso se lleva a cabo de forma distinta, la única diferencia que se percibe externamente es el paso previo a la ejecución en el que hay que identificarse para tener acceso al servidor.

6.4.2.- Conexión y Transferencia de Datos.

Aparte de utilizar hilos como ocurría en el modo de ejecución local, la implementación del módulo de ejecución remota se basa en el uso de los protocolos SSH (*Secure SHell*) y SCP (*Secure CoPy*). El primero sirve para establecer conexiones remotas de forma segura, mientras que el segundo se utiliza para realizar transferencias de archivos en modo cifrado.

El mecanismo aplicado para establecer la conexión con el servidor consiste en utilizar el comando PLINK, un cliente SSH que permite ejecutar comandos en un servidor remoto desde un terminal local. Por otro lado, para transferir los ficheros desde el terminal hasta el servidor y para devolver los resultados, se usa PSCP, un cliente SCP que permite transferir archivos desde un host local a un host remoto o viceversa.

El proceso de establecimiento de la conexión se inicia con la solicitud que realiza el host local. En ese momento, el servidor comprueba si éste dispone de permisos de conexión. En caso afirmativo, se intercambian sus identificadores de versión y verifican que los protocolos utilizados en ambas máquinas coincidan para poder llevar a cabo el intercambio de información. A continuación, el servidor transfiere al host local su clave pública para que éste encripte los mensajes, de modo que sólo los pueda descifrar el propio servidor.

Si no es la primera vez que el ordenador local se conecta a ese servidor, puede verificar la autenticidad de la clave pública recibida comparándola con la que tiene almacenada en su lista de hosts conocidos. Si es la primera vez, el host local solicita una confirmación para aceptar la clave, ya que no puede saber si la clave pública recibida pertenece o no al servidor. Tras aceptar la confirmación, la clave pública del servidor se añade al archivo `/ssh/known_hosts`, donde se almacenan los hosts conocidos. De esta forma, cuando el ordenador local vuelva a conectarse a dicho servidor, comprobará si su clave coincide con alguna de la lista de hosts conocidos y no será necesario solicitar la confirmación para aceptar la clave.

Por otra parte, para evitar tener que introducir la contraseña cada vez que se realiza la conexión con la máquina remota, se pueden generar un par de claves de autorización (pública y privada), de modo que el host local pueda identificarse mediante su clave pública. Para ello, se debe teclear el comando:

```
ssh –keygen –t rsa
```

Como consecuencia de dicha orden, se generan dos archivos: `id_rsa`, que contiene la clave privada, y `id_rsa.pub`, que contiene la clave pública. A continuación, se debe copiar el contenido de `id_rsa.pub` en el fichero `authorized_keys` de la

máquina remota, de modo que el servidor almacene la clave pública del host local y no sea necesaria la autenticación mediante contraseña.

6.4.3.- Gestión de las Simulaciones.

Hay procesos que implican un gran consumo de recursos temporales, como pueden ser las optimizaciones o las simulaciones a varias frecuencias. En estos casos, lo común es esperar a que se generen los ficheros de resultados, manteniendo la sesión abierta para no perder los datos.

Esta situación no se produce con NewFasant, ya que éste dispone de un gestor de simulaciones, válido tanto para modo local como para modo remoto, que permite ejecutar dichos procesos en *background*, de forma que se pueda seguir trabajando en otros diseños mientras se espera la finalización de la simulación en curso. En el momento en que el proceso ejecutado en segundo plano finaliza, se realiza la correspondiente notificación, ofreciéndose la posibilidad de visualizar los resultados.

Otra funcionalidad avanzada del gestor de simulaciones es que incluso permite cerrar la sesión mientras se está ejecutando un proceso (local o remoto) sin perder la conexión con dicho proceso. De esta forma, cuando se vuelve a inicial el sistema, éste informa de que hay una simulación pendiente, como se puede observar en la figura 6.14. En el cuadro que aparece al iniciar la sesión se da la opción de visualizar el estado de la simulación ejecutada en *background*.

Durante el tiempo de inactividad pueden haber ocurrido dos cosas:

1. La simulación ha finalizado. En ese caso se notifica y se visualizan los resultados.
2. La simulación continúa ejecutándose. En ese caso se ofrece la posibilidad de visualizar su estado, mostrando el cuadro informativo de la figura 6.11.

Por último, tanto si el modo de ejecución es local o remoto, el sistema también da la opción de abortar la simulación. Esta acción conlleva la eliminación de los procesos activos relacionados con la ejecución del núcleo electromagnético.

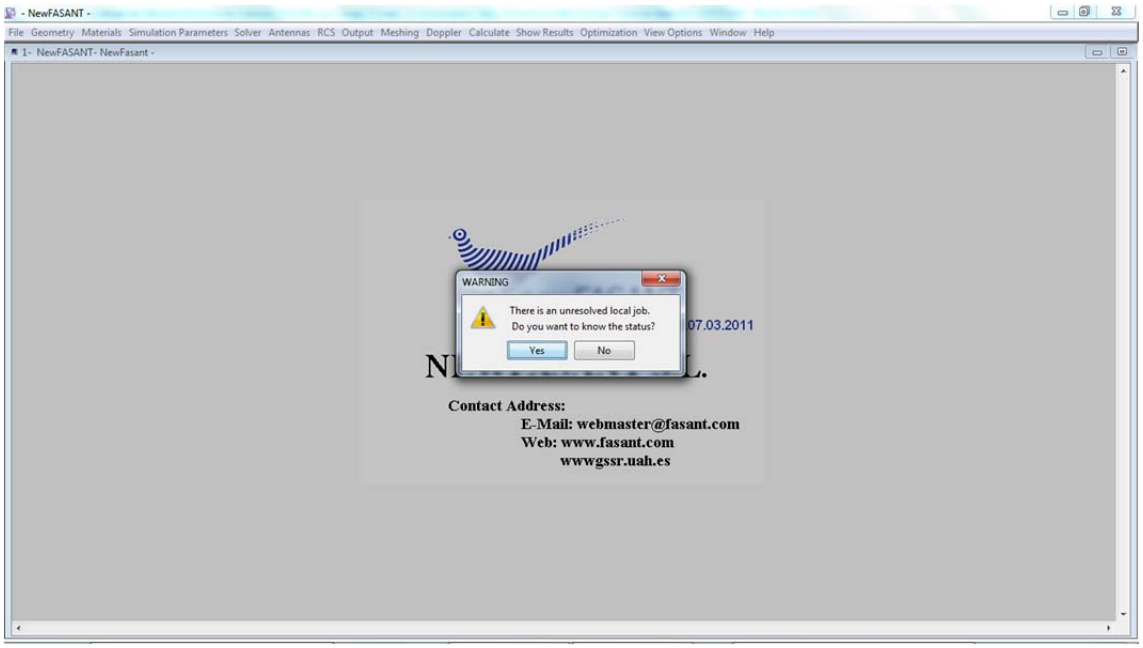


Figura 6.14.- Cuadro informativo que notifica de la existencia de una simulación local ejecutada en segundo plano.

6.5.- VISUALIZACIÓN DE RESULTADOS.

Como resultado de un análisis, se genera un conjunto de ficheros de resultados, que son procesados por el sistema para ser mostrados en función de las opciones que se indican a continuación:

1. Visualización de gráficas en dos dimensiones de intensidad de campo radiado, de campo dispersado, de ganancia o de coeficientes de transmisión y reflexión.
   Esta opción permite representar tanto valores de campo como de ganancia en función del corte en θ o en φ especificado. En el caso de los coeficientes de transmisión y reflexión, se representa su variación en función de la frecuencia. Por ejemplo, la figura 6.15 muestra el cuadro de diálogo donde se especifica el modo de visualización de un corte del diagrama de radiación. La figura 6.16 muestra la curva obtenida, junto con la ventana donde se muestran las opciones disponibles para el tratamiento del gráfico: hacer zooms, imprimir o guardar la gráfica obtenida, editar las propiedades gráficas como el título, los colores de las curvas, etc.

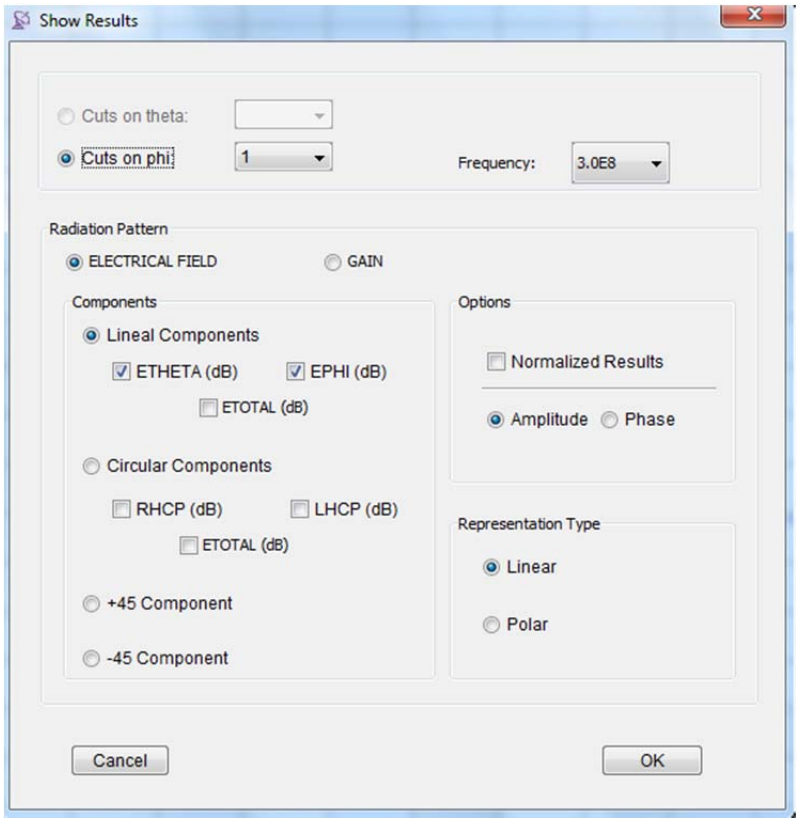


Figura 6.15.- Cuadro de diálogo donde se especifica el modo de visualización de los resultados de campo en 2D.

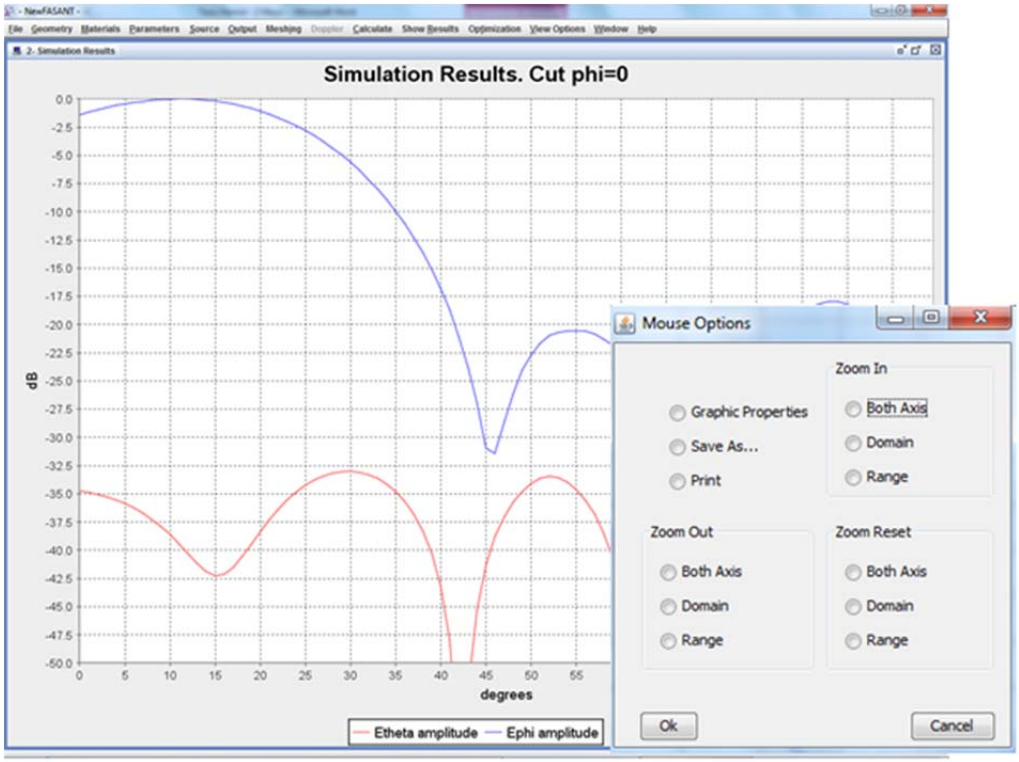


Figura 6.16.- Gráfica en dos dimensiones que muestra el corte φ=0º del diagrama de radiación de una bocina seccionada.

2. Mostrar la distribución de corrientes sobre las superficies de la estructura analizada.

   La densidad de corriente se representa mediante un código de colores donde los valores más elevados de corriente se corresponden con colores cálidos y los valores más bajos se identifican con colores fríos. Las unidades se pueden fijar en dBs o en unidades naturales. En la figura 6.17 se muestra un ejemplo de este tipo de representación.

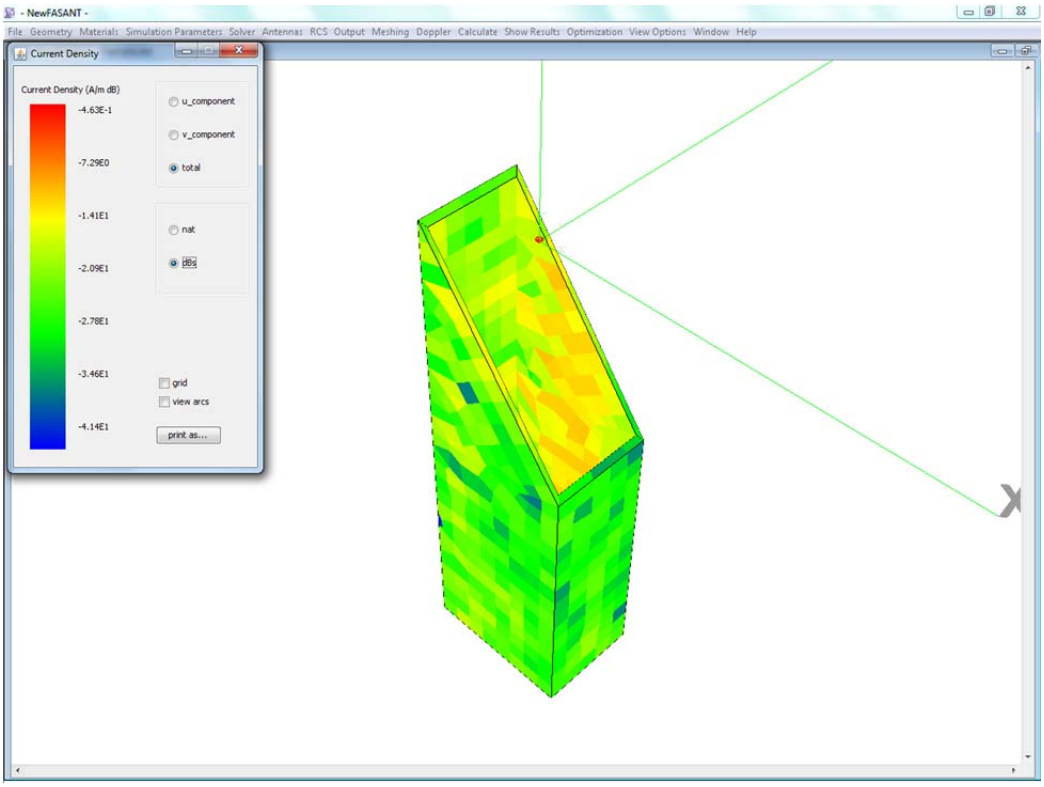


Figura 6.17.- Representación de la distribución de corrientes sobre la estructura.

3. Representar la densidad de carga sobre la estructura.

   La densidad de carga se representa de forma similar a la distribución de corrientes. En la figura 6.18 se puede ver un ejemplo.

4. Visualizar el diagrama de radiación en tres dimensiones.

   El diagrama de radiación tridimensional se muestra sobre el modelo geométrico, en tres dimensiones, representado mediante zonas de diferentes colores que indican mayor o menor nivel de radiación. La figura 6.19 muestra un ejemplo de este tipo de representación.

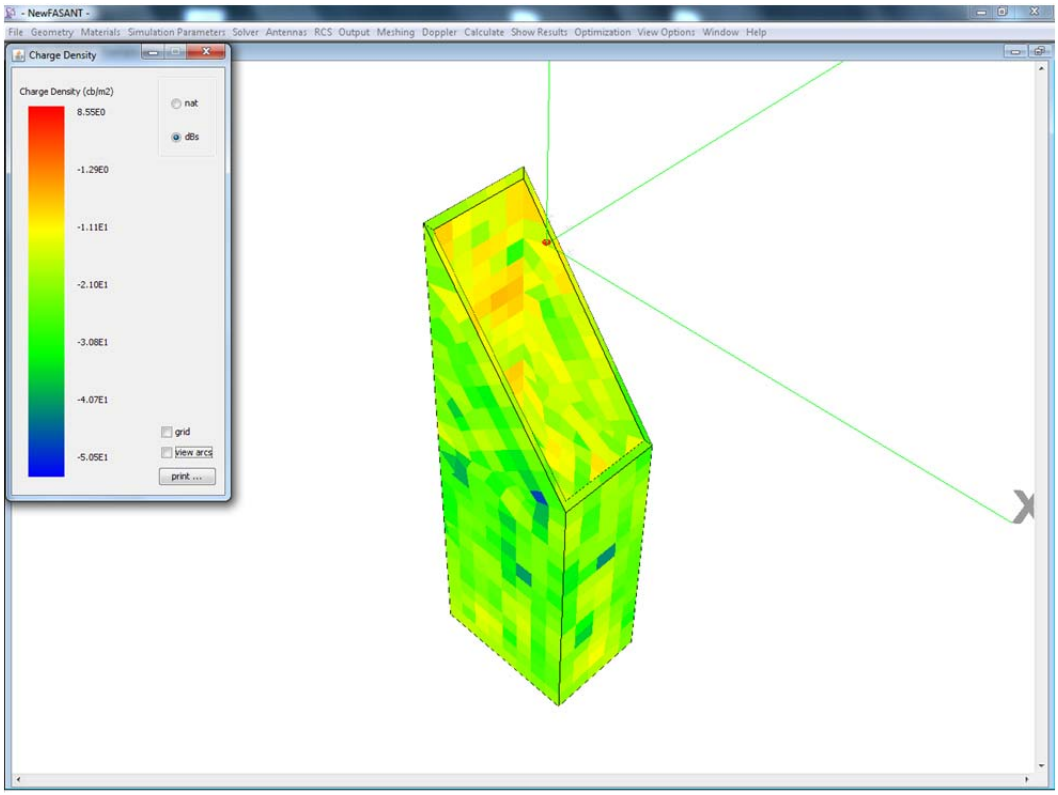


Figura 6.18.- Representación de la densidad de carga sobre una estructura.

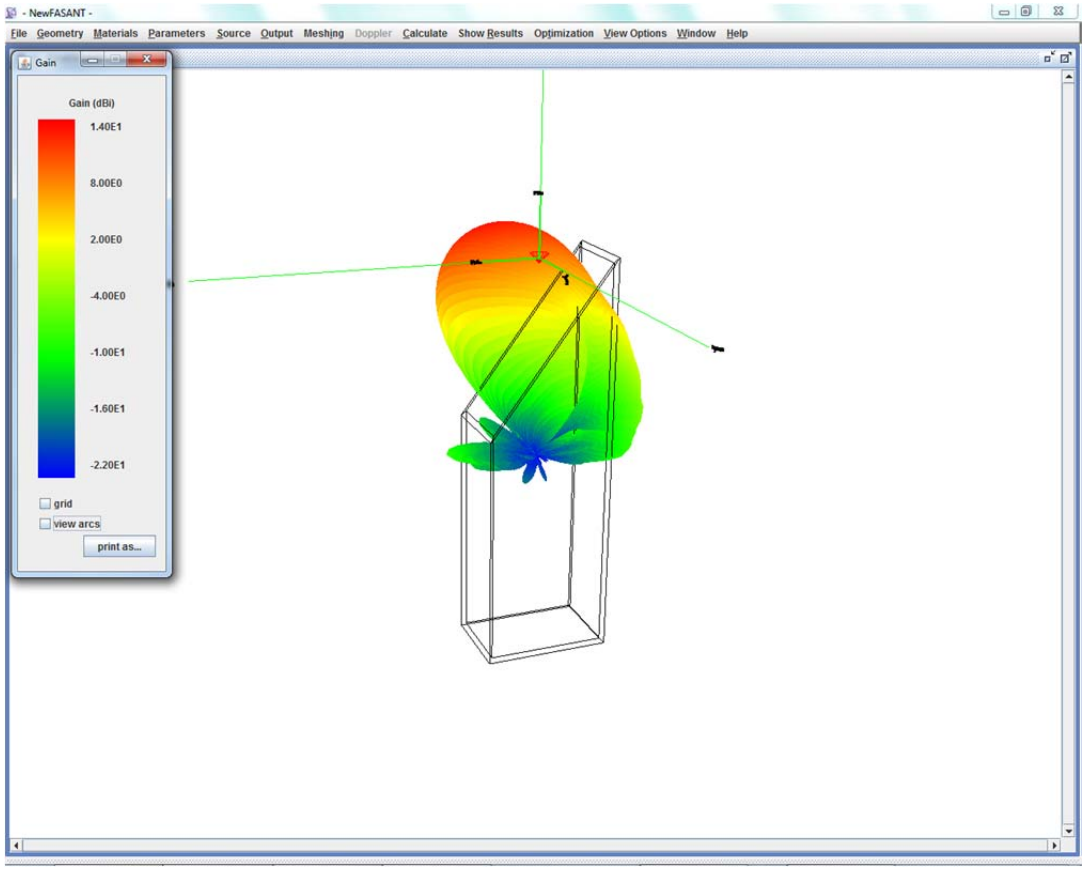


Figura 6.19.- Diagrama de radiación en tres dimensiones.

6.6.- PROCESO DE OPTIMIZACIÓN.

El proceso de optimización consiste en la variación de los parámetros de un modelo para conseguir una determinada respuesta. Por ejemplo, la ganancia máxima de una antena se puede optimizar variando las dimensiones de su apertura.

El módulo de optimización implementado permite calcular el valor óptimo de cualquier parámetro que forme parte de la caracterización de un objeto, como puede ser el radio de una esfera, el número de vueltas de una hélice, la anchura de las corrugaciones de una bocina, etc. La optimización se realiza en base a unas especificaciones fijadas inicialmente a través de una función de coste, de forma que los parámetros seleccionados varíen dentro de un margen conocido, hasta encontrar la solución óptima que minimice dicha función de coste. En la figura 6.20 se muestra su funcionamiento de forma esquemática.

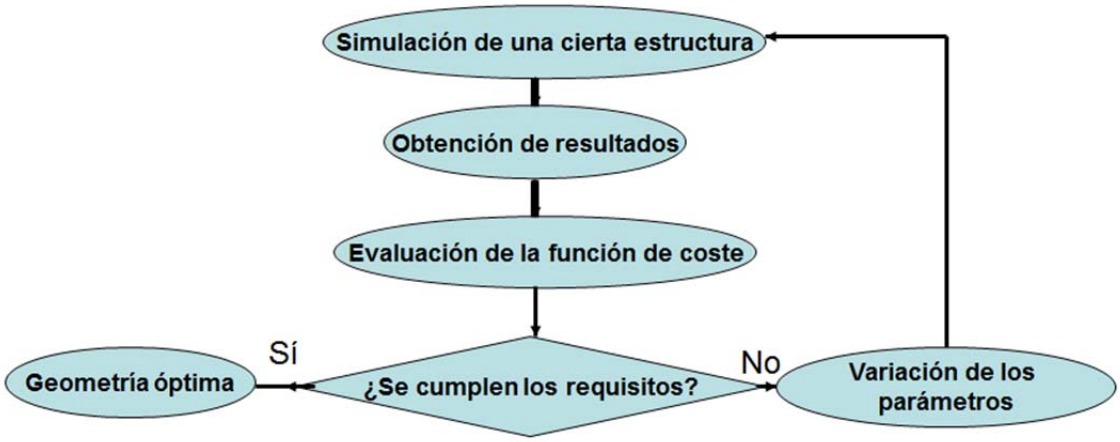


Figura 6.20.- Diagrama del proceso de optimización.

El proceso comienza analizando una determinada estructura, definida por los parámetros iniciales. Al finalizar la simulación, se procesan los resultados y se evalúa la función de coste establecida previamente. Si todos los requerimientos impuestos son alcanzados, se devuelven los valores de los parámetros que han proporcionado dichos resultados. En caso de que no se cumplan las especificaciones establecidas, se modifican los parámetros, se actualiza la geometría y se vuelve a analizar, repitiendo todo el proceso.

Si en alguna iteración los resultados obtenidos dan lugar a una función de coste igual a cero, el proceso finaliza mostrando la geometría optimizada junto con los resultados obtenidos. Si, por el contrario, la función de coste no se anula en ninguna de las iteraciones, los parámetros devueltos serán aquellos que impliquen un coste más bajo.

6.6.1.- Selección de parámetros.

El primer paso del proceso consiste en seleccionar los parámetros que se desean optimizar. Tal y como se comentó en el capítulo 3, todos los valores editables que definen la forma de un objeto se han parametrizado, de modo que su edición y optimización sea más eficiente. El sistema permite establecer un margen de variación para cada parámetro, de tal forma que al seleccionar uno de ellos se establezcan los valores máximos y mínimos de dicho margen. La figura 6.21 muestra los parámetros optimizables de una estructura periódica definida mediante superficies rectangulares. Entre ellos se encuentran las coordenadas del centro de la estructura (*x, y, z*), las dimensiones de la celda unidad (anchura y altura), el número de elementos en *x* y en *y*, y el periodo de repetición en ambas direcciones.

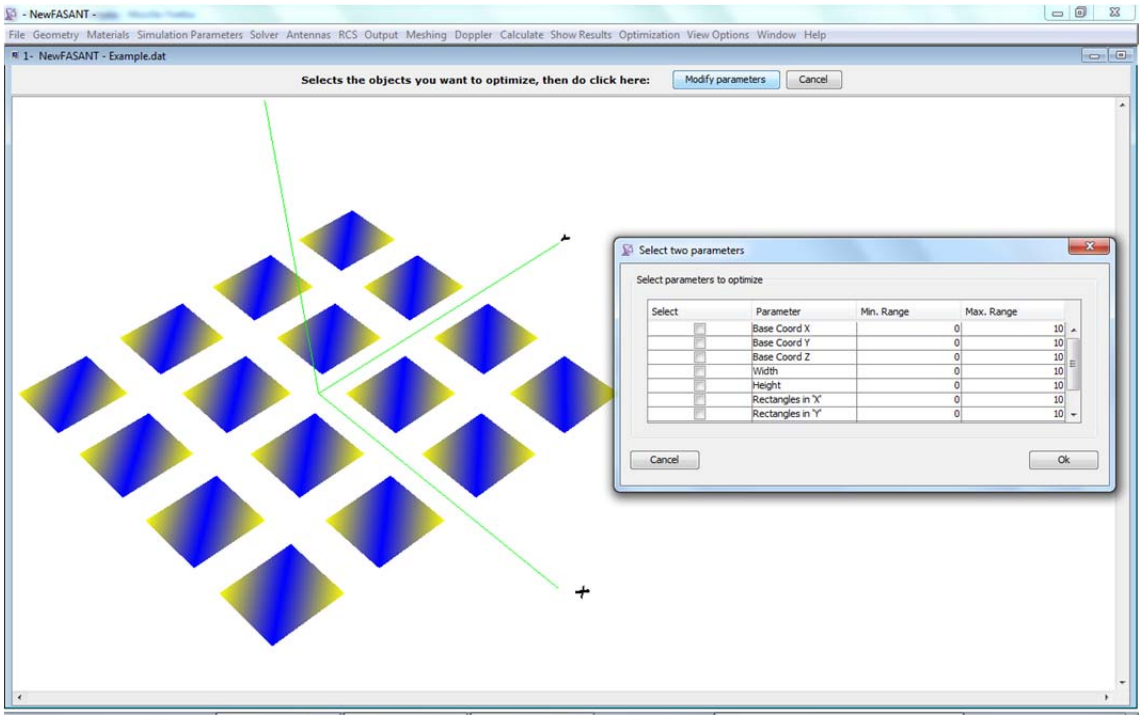


Figura 6.21.- Selección de los parámetros a optimizar de una estructura periódica.

6.6.2.- Algoritmos de Optimización.

Según el diagrama de la figura 6.20, a lo largo del proceso de minimización de la función de coste se establecen nuevos valores para los parámetros de acuerdo un algoritmo de optimización. Cada método posee unas características diferentes, apropiadas para diferentes tipos de problemas. Saber cuál es el más adecuado para un caso determinado no es una tarea fácil. Depende del número de parámetros, del rango de cada uno, de la salida deseada, del tamaño del modelo y de los recursos disponibles [5].

Los algoritmos de optimización implementados hasta ahora son el método de variación lineal, el método de gradiente descendente y el método de PSO (*Particle Swarm Optimization*).

El algoritmo de variación lineal no se considera estrictamente un método de optimización, puesto que los parámetros a optimizar varían linealmente entre los valores máximos y mínimos. Sin embargo, un análisis de estas características puede ser útil para estudiar el efecto general que produce la variación de un parámetro concreto antes de iniciar el proceso de optimización propiamente dicho. La aplicación de esta técnica implica un gran consumo computacional de recursos, ya que para cada posible combinación de los valores asignados a cada parámetro, se ejecuta una simulación y se evalúa la función de coste. Por esta razón no se recomienda analizar más de dos variables en cada optimización. Por ejemplo, de acuerdo a la expresión dada en (6.12), si se estudia la variación de dos parámetros estableciendo a cada uno de ellos 8 valores distintos, en total se realizarán 64 simulaciones.

$$N_{simulaciones} = \prod_{i=1}^{N_{parámetros}} N_{valores}(i) \tag{6.12}$$

El algoritmo de gradiente descendente [6] es un método de optimización que recalcula el valor de los parámetros en cada iteración considerando la dirección opuesta al gradiente de la función de coste en la iteración anterior. El algoritmo de minimización es el siguiente:

$$X_{k+1} = X_k - \alpha_k \nabla q(X_k)^T \tag{6.13}$$

donde *X* representa el valor de los parámetros, *q(x)* es la función a minimizar, y $\alpha_k \geq 0$ es el escalar que minimiza la siguiente expresión:

$$q(X_k - \alpha_k \nabla q(X_k)^T) \tag{6.14}$$

La inclusión del signo negativo en (6.14) implica un desplazamiento en el sentido opuesto al gradiente de la función de coste, es decir, en la dirección de máxima variación del error. Por tanto, el valor de los parámetros se actualizará siguiendo la dirección de máximo decrecimiento, hasta que se alcance el mínimo de la función.

El valor que toma α_k en cada iteración es un factor importante a tener en cuenta. Si se trata de un valor muy pequeño, la optimización se ralentiza. En cambio, si es demasiado grande, se pueden producir oscilaciones en torno al mínimo, apareciendo problemas de convergencia. En dicha situación, se devuelven los valores de los parámetros que impliquen el menor coste de entre todas las opciones evaluadas.

Respecto al algoritmo de PSO [7], se trata de un método de optimización que se basa en el comportamiento de las colonias de aves, peces o insectos cuando se desplazan de un sitio a otro. Si un miembro descubre un mejor camino, lo comunica al resto, de tal forma que en la siguiente iteración todos actualizan su posición y su velocidad en función de la información transmitida. De este modo, los miembros de la colonia tienden a dirigirse hacia un mejor espacio de búsqueda en el proceso de migración, es decir, los parámetros geométricos tienen a dirigirse hacia un mejor espacio de búsqueda en el proceso de minimización de la función de coste.

6.6.3.- Función de Coste.

La función de coste sirve para cuantificar el objetivo que se desea alcanzar determinando cuánto se aproxima la solución actual a la solución óptima. Desafortunadamente, ningún algoritmo de optimización puede garantizar una solución óptima especificando unos ciertos parámetros y una determinada función de coste. Aun así, el sistema siempre devuelve la mejor solución encontrada que se ajuste lo máximo posible a las condiciones establecidas.

La función de coste representa propiedades relacionadas con el diagrama de radiación como la ganancia, el nivel de lóbulo principal a secundario o la pureza de polarización, las cuales normalmente necesitan ser optimizadas para cumplir unas ciertas especificaciones.

Cuando se lleva a cabo una optimización, a menudo interesa optimizar más de un objetivo. Por ejemplo, se puede querer optimizar la ganancia de una antena, pero al mismo tiempo mejorar su pureza de polarización. O se puede querer optimizar la forma del diagrama de radiación aplicando distintas especificaciones para el plano E y para el plano H. En estas situaciones es posible definir varias funciones locales, de modo que la función de coste global se defina como una combinación lineal de todas ellas.

La función de coste global permite optimizar el diagrama de radiación de acuerdo a una forma arbitraria especificada mediante varios bloques. Por ejemplo, se pueden definir condiciones locales sobre la forma del haz principal y de los lóbulos secundarios en un plano determinado. Dichas condiciones se pueden expresar, por ejemplo, de la siguiente manera: desde θ=0º hasta θ=40º, el nivel de campo debe superar los 25dB para caracterizar el haz principal y desde θ=40º hasta θ=70º el nivel debe ser menor de 10dB para minimizar los lóbulos secundarios. La figura 6.22 representa estas funciones de coste locales de forma gráfica.

En la figura 6.22 se observan dos tipos de niveles: los indicados en línea discontinua son los niveles deseados y los representados en línea continua son los reales. Se puede ver que en el primer bloque ($0 \leq \theta \leq 40$) se intenta maximizar el lóbulo principal, mientras que en el segundo bloque ($40 \leq \theta \leq 70$) el objetivo es minimizar el nivel de lóbulo secundario.

La penalización aplicada en ambos casos se establece en una unidad por cada dB fuera del margen especificado. En el caso del ejemplo, la penalización en el punto θ=0º es de 4 unidades. Sin embargo, en el punto θ=40º, la diferencia entre el valor de la curva en ese punto (10dB) y el valor establecido como el ideal (25dB), es de 15dB, por lo que la penalización en θ=40º es de 15 unidades. Análogamente, se puede aplicar el mismo razonamiento sobre el segundo bloque obteniendo, en este caso, penalizaciones máximas de 3 unidades.

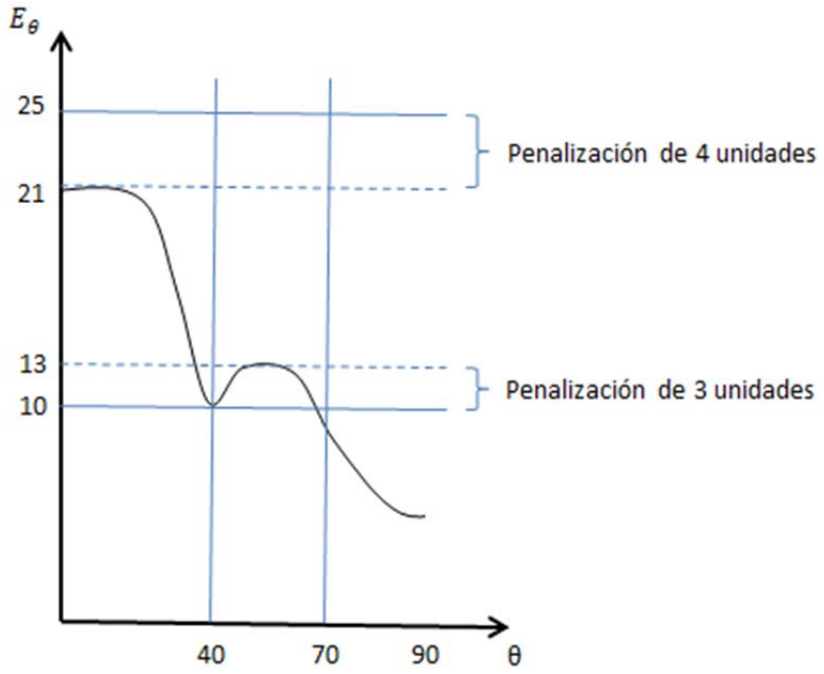


Figura 6.22.- Especificación de una función de coste mediante dos bloques.

Para establecer la función de coste, la herramienta dispone del cuadro de diálogo que se muestra en la figura 6.23, mediante el cual se puede establecer el número de tramos que definen la función de coste global y el tipo de componente que se desea optimizar, polar o contrapolar. Para cada uno de los bloques hay que especificar el tipo de corte (en θ o en φ) que se va a procesar, los valores que determinan el margen angular y los niveles de campo máximos y mínimos que se desean obtener.

Matemáticamente, la expresión que determina la función de coste global del ejemplo anterior y, en general, de cualquier optimización, es la siguiente:

$$Z = \sum_{i=1}^{N_B} \sum_{j=1}^{N_S} \left[\Gamma\left(\left| E_{\theta,\varphi}\left(\theta_j^i, \varphi_j^i\right) - P_{max} \right| \right) + \Gamma\left(P_{min} - \left| E_{\theta,\varphi}\left(\theta_j^i, \varphi_j^i\right) \right| \right) \right] \qquad (6.15)$$

donde

$$\Gamma(x) = \begin{cases} 0 & si\ x < 0 \\ x & si\ x \geq 0 \end{cases} \qquad (6.16)$$

N_B es el número de bloques, N_S es el número de valores angulares de cada bloque, y P_{max} y P_{min} son los niveles máximos y mínimos que definen la forma del diagrama de radiación ideal dentro de cada bloque. Puede ocurrir, como en el ejemplo, que en cada

bloque sólo se defina uno de los dos niveles. De acuerdo a la figura 6.22, en el primer bloque se ha establecido que el valor mínimo del campo P_{min} dentro del rango $0 \leq \theta \leq 40$ sea 25dB. El valor máximo P_{max} se obvia ya que no aporta ninguna información útil. En el segundo bloque ocurre lo contrario, se especifica el valor máximo P_{max} que puede alcanzar el lóbulo secundario sin especificar el mínimo P_{min}.

$E_{\theta,\varphi}(\theta_j^i, \varphi_j^i)$ representa la potencia de campo radiado para el valor angular *j*-ésimo del bloque *i*-ésimo. La componente considerada E_θ o E_φ dependerá de la configuración que se haya indicado en el cuadro mostrado en la figura 6.24.

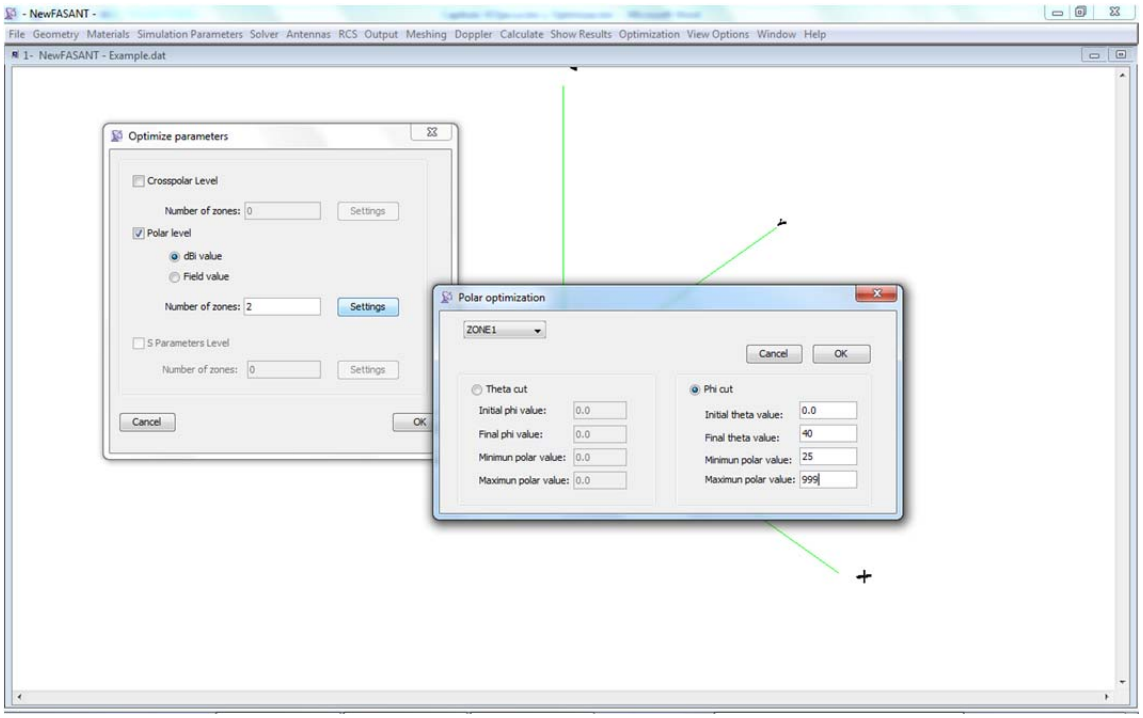


Figura 6.23.- Especificación de una función de coste mediante dos bloques.

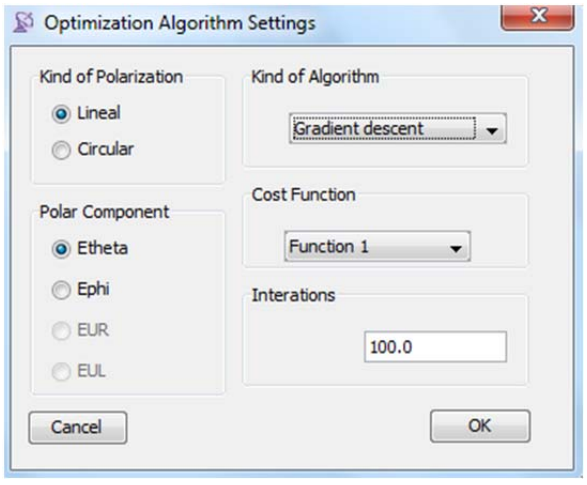


Figura 6.24.- Parámetros del proceso de optimización.

Por último, indicar que la optimización se puede llevar a cabo a distintas frecuencias, de forma que se aplique la misma función de coste sobre todas ellas.

6.7.- REFERENCIAS.

[1] C. Delgado, M. F. Cátedra and R. Mittra, "Application of the Characteristics Basis Functions Method Utilizing a Class of Basis and Testing Functions Defined on NURBS Patches", *IEEE Transactions on Antennas and Propagation*, Vol. 56, No. 3, March 2008.

[2] Gonzalez, I.; Gomez, J.; Tayebi, A.; Delgado, C.; Catedra, F.; , "A domain decomposition moment method approach for analysis of complex reflectorarrays," *Antenna Technology and Applied Electromagnetics & the American Electromagnetics Conference (ANTEM-AMEREM),* 2010 14th International Symposium on , vol., no., pp.1-4, 5-8 July 2010.

[3].- E. García García, "Contribución al análisis de problemas electromagnéticos mediante el Método de los Momentos con bajo coste computacional", Tesis Doctoral. Universidad de Alcalá, Marzo 2005.

[4].- I. Gónzalez, F. Saez de Adana, F. Cátedra: "Application of the Multilevel Fast Multipole Method to the Analysis of Conformed Multilayered Periodic Structures". *IEEE AP-S International Symposium on Antennas and Propagation*, Honolulu, Hawaii, USA, 10-15 June 2007.

[5].- Pardalos, P. M., Resende, Mauricio, G. C. "Handbook of applied optimization", Oxford University Press, 2002.

[6].- Edwin K. P., Chong, Stanislaw H. Zak, "An introduction to optimization", John Wiley & Sons, New Jersey, 2008.

[7].- J. Kennedy, R. Eberhart. "Particle Swarm Optimization," *Proceedings of IEEE International Conference on Neural Networks*. Perth (Australia) Vol. 4. 1995. pp. 1942-1948.

7.- Resultados.

7.1.- INTRODUCCIÓN.

A pesar de que durante toda la memoria se han ido presentando algunos resultados cuando se explicaban las funcionalidades de la herramienta, éstos no se pueden considerar pruebas definitivas de la viabilidad del sistema desarrollado. En este capítulo se presentan algunos de los resultados que se han ido obteniendo a lo largo de la realización de la tesis. Se organiza en varios apartados en los que se muestran diseños y simulaciones de antenas, algunas de cuales se han optimizado, construido y medido para poder comparar los resultados de las simulaciones con las medidas reales.

7.2.- ANÁLISIS DE UNA BOCINA PIRAMIDAL.

En este primer apartado se presenta el análisis de una bocina piramidal. El diseño de la antena es inmediato, pues, como se indicó en el capítulo 4, el sistema dispone de un módulo específico de creación de antenas.

La bocina se alimenta mediante un dipolo eléctrico situado a $\lambda/4$ de la base, orientado en el eje Y, de forma que se propague el modo fundamental a lo largo de la guía y la antena radie con polarización lineal. El momento dipolar se define con una amplitud igual a 1 y una fase nula. En la figura 7.1 se pueden ver los cuadros de diálogo implementados para facilitar la definición de los dipolos y del sistema antena. A la derecha de la imagen se muestra el modelo geométrico generado por la herramienta. Se puede observar que el sistema antena se sitúa en la apertura de la bocina para establecer el origen de fases, manteniendo su orientación por defecto.

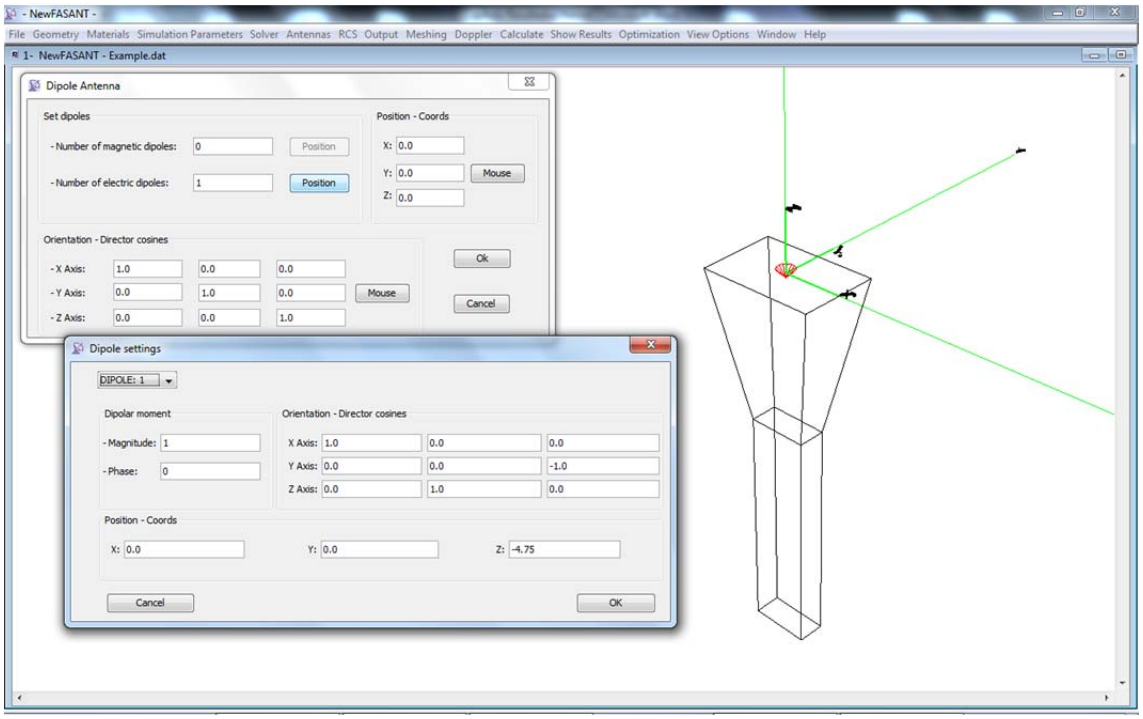


Figura 7.1.- Modelo geométrico de la bocina junto con los cuadros de diálogo que definen la alimentación mediante dipolos.

Una vez establecida la alimentación, se especifica que la simulación se lleva a cabo para analizar el diagrama de radiación en campo lejano, definiendo los planos del diagrama que se desean visualizar, en este caso, el plano E y el plano H y un plano diagonal fijando φ=45º. A continuación, se ejecuta el proceso de discretización de la geometría, obteniendo como resultado la estructura mostrada en la figura 7.2. En la parte izquierda de la imagen se muestra un cuadro que informa sobre la frecuencia, el número de superficies (planas y curvas) obtenidas, el número de elementos totales y el número de nodos.

Una vez que todos los parámetros relacionados con el método de resolución han sido definidos, se inicia el proceso de análisis, en el que el núcleo del sistema resolverá el problema electromagnético aplicando la ecuación integral de campo eléctrico. Tras finalizar el análisis, se ofrece la posibilidad de mostrar los resultados mediante distintas opciones. Por ejemplo, en la figura 7.3 se pueden visualizar los tres cortes principales del diagrama de radiación (φ=0º, φ=45º y φ=90º) previamente definidos. Se puede ver que el margen angular establecido para la variable θ en los tres gráficos varía desde 0º hasta 180º. Otra característica destacable es que los resultados se han normalizado, estableciendo el nivel de máxima radiación con un valor de 0 dB.

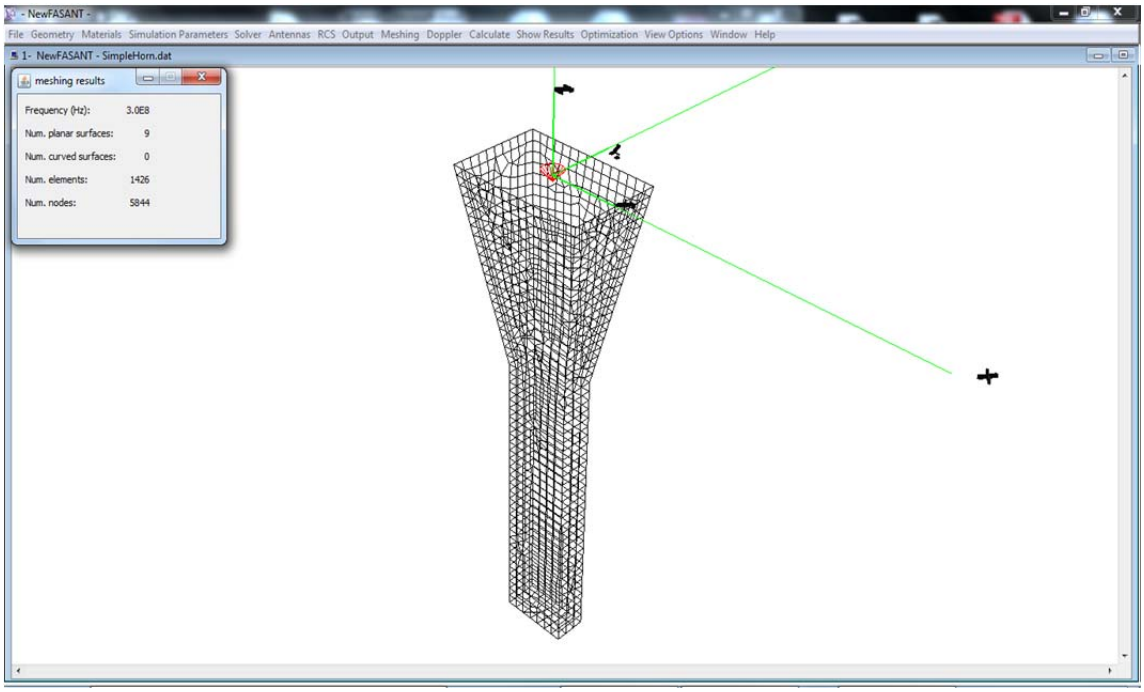


Figura 7.2.- Resultado del proceso de mallado de la geometría.

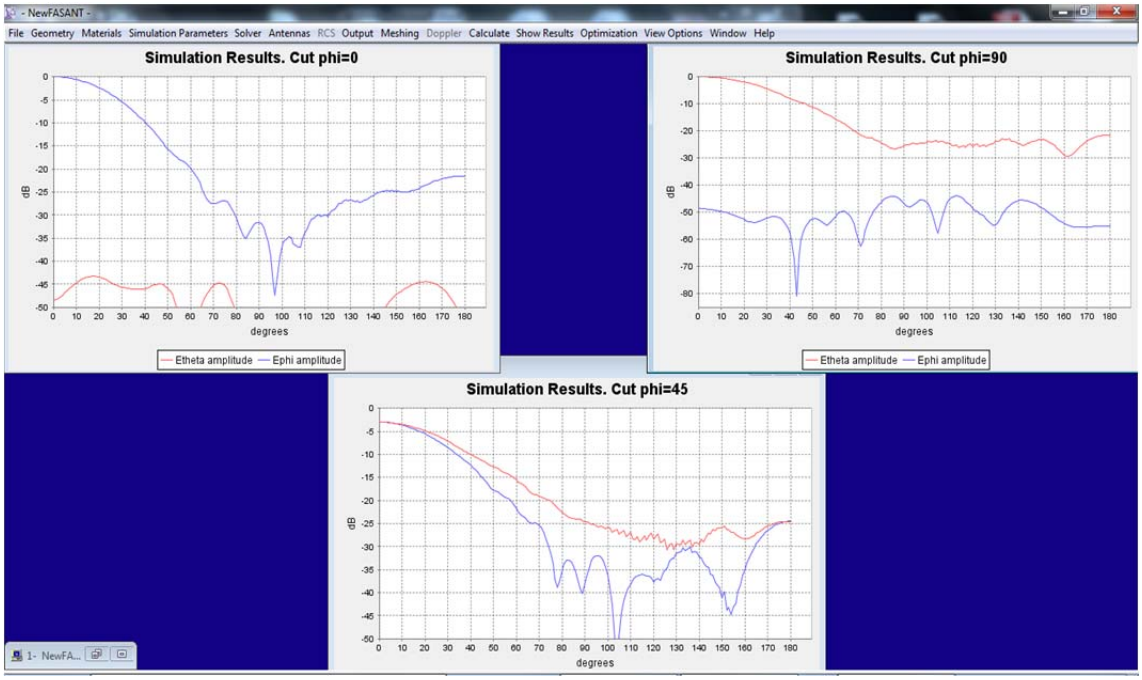


Figura 7.3.- Resultados obtenidos tras finalizar la simulación.

Por otro lado, puede resultar interesante analizar la distribución de corrientes y de cargas sobre la superficie de la bocina. Ambas magnitudes se pueden ver de forma gráfica en la figura 7.4.

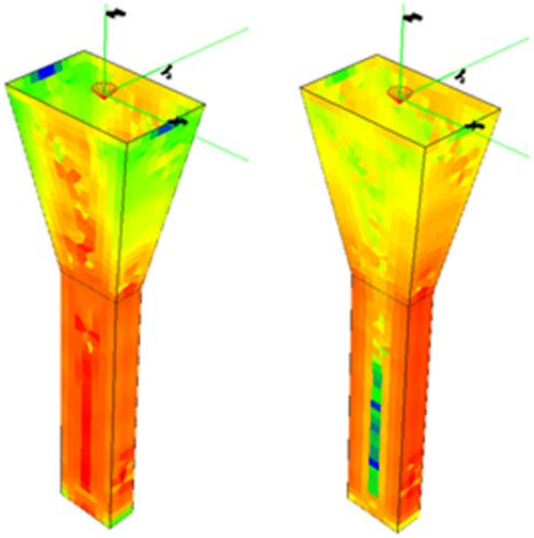


Figura 7.4.- Distribución de corriente (izquierda) y distribución de carga (derecha).

Por último, la figura 7.5 muestra el diagrama de radiación tridimensional superpuesto sobre el modelo geométrico de la bocina. Mediante este tipo de representación se puede analizar más fácilmente la respuesta de la antena.

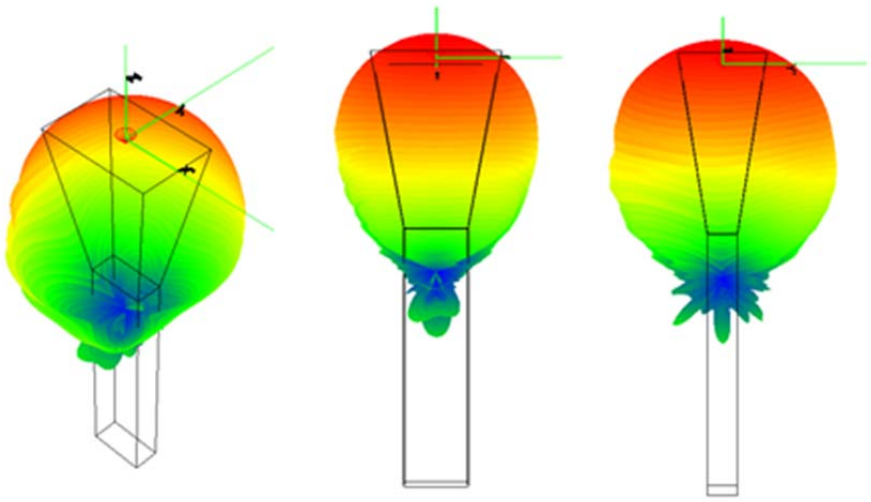


Figura 7.5.- Diagrama de radiación 3D. Vista en perspectiva (izquierda), vista de frente (centro) y vista de perfil (derecha).

7.3.- ANÁLISIS DE UN REFLECTOR PARABÓLICO.

El módulo de antenas incluido en la herramienta también facilita la creación de antenas reflectoras parabólicas. En este apartado se presenta el análisis de un reflector simple con simetría de revolución a una frecuencia de 10 GHz. Se trata de un reflector centrado de 0.3 m de diámetro y 0.2 m de distancia focal.

Para definir la iluminación del reflector se especifica la opción de alimentación mediante el diagrama de radiación de una bocina cónica que radie con polarización circular. Una vez establecida la configuración de todos los parámetros, se procede a la ejecución de la simulación. En la figura 7.6 se puede ver el modelo de la antena discretizado, mientras que en la figura 7.7 se muestran los resultados obtenidos para el corte del diagrama de radiación en φ=0º.

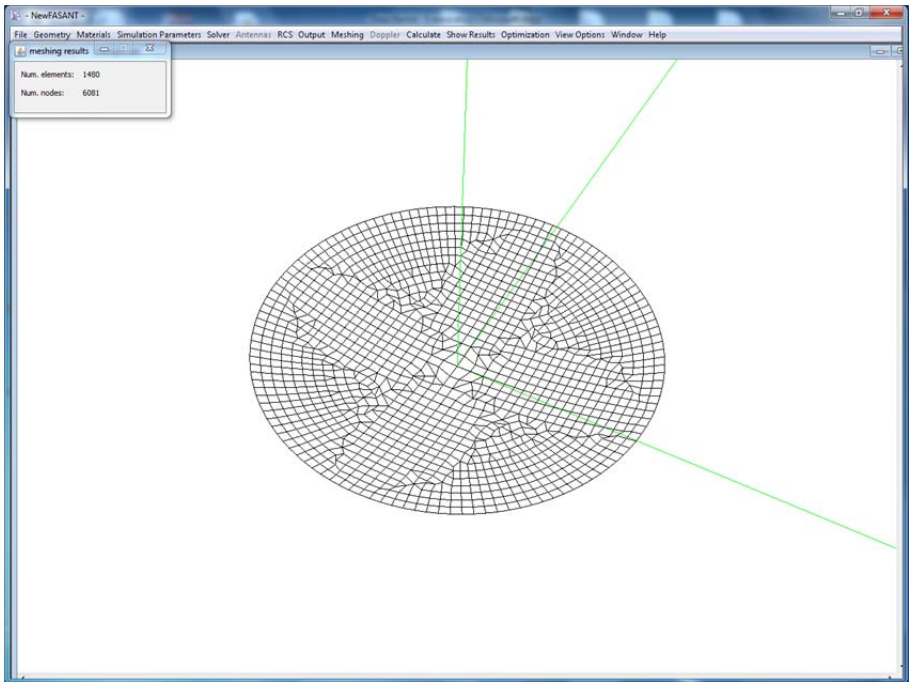


Figura 7.6.- Resultado obtenido tras aplicar el algoritmo de mallado.

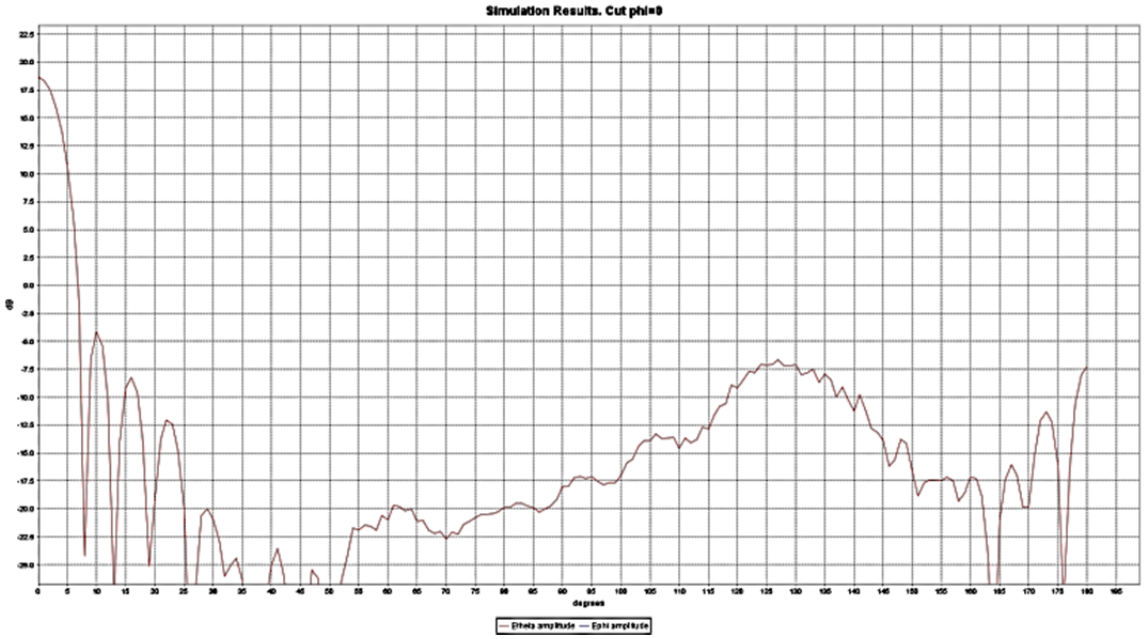


Figura 7.7.- Diagrama de radiación en 2D. Corte en φ=0º.

Por último, en la figura 7.8 se visualiza la densidad de corriente y en la figura 7.9 se muestra el diagrama de radiación tridimensional.

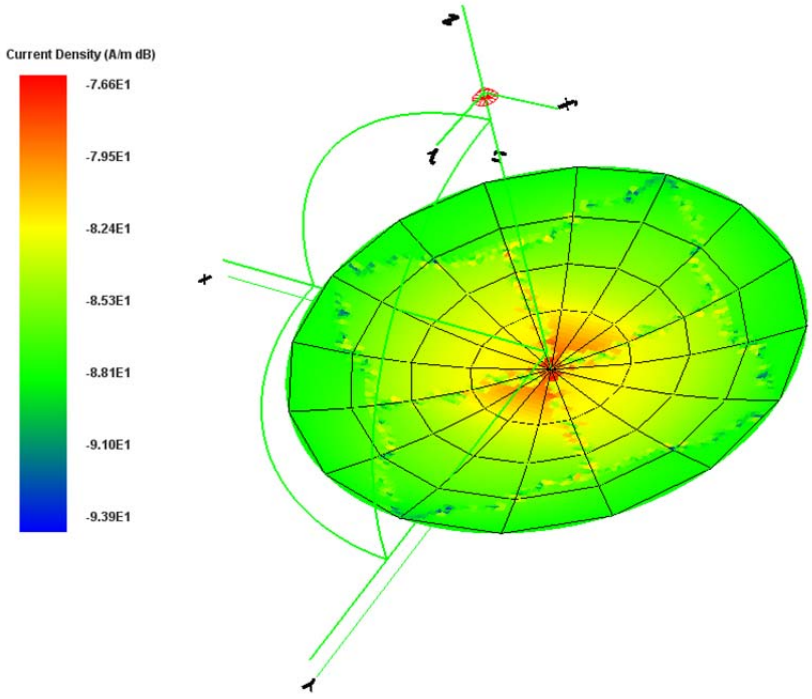


Figura 7.8.- Densidad de corriente sobre la superficie de la antena reflectora.

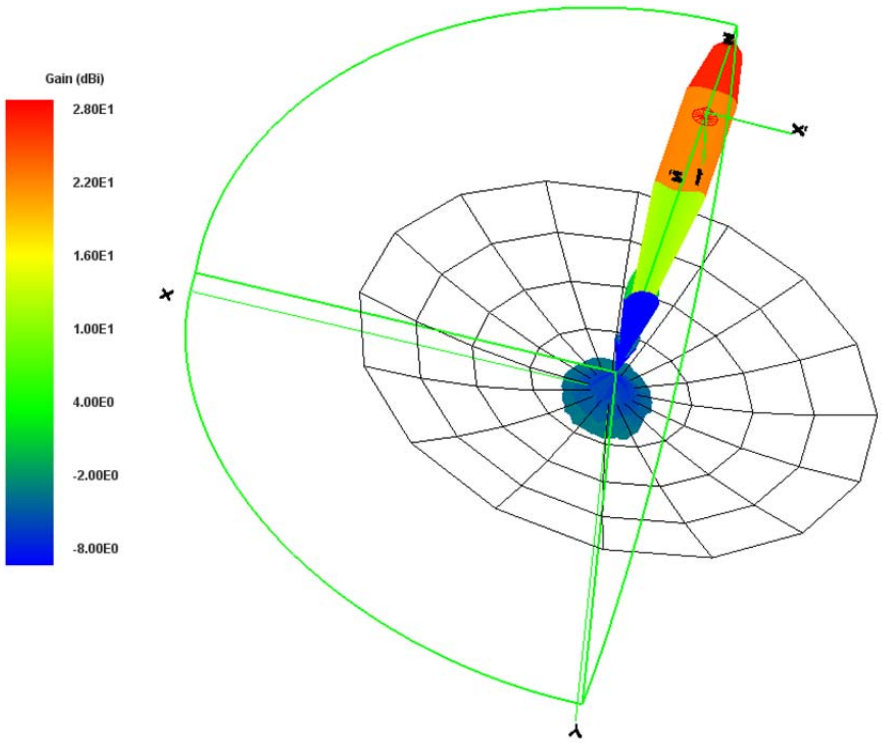


Figura 7.9.- Diagrama de radiación 3D.

7.4.- ANÁLISIS DE UNA ANTENA REFLECTARRAY DUAL.

En este apartado se expone el análisis de una antena en el que su modelo geométrico no se crea utilizando las facilidades que incluye la interfaz, sino que se importa directamente de un fichero externo. La geometría importada corresponde a una antena reflectarray compuesta por una superficie plana reflectante situada encima un plano de masa. Sobre dicha superficie se sitúan los elementos radiantes modelados mediante parches microstrip.

Las antenas reflectarray se utilizan en aplicaciones radar y en comunicaciones de larga distancia. A pesar de que suelen tener un ancho de banda estrecho, son muy apropiadas para funcionar en frecuencia dual. Hoy en día las antenas reflectoras microstrip se usan ampliamente como alternativa a las antenas reflectoras convencionales por sus ventajas, como por ejemplo peso y volumen reducidos, robustez mecánica, compatibilidad con dispositivos activos, bajo coste y rapidez en la fabricación.

La figura 7.10 muestra el modelo geométrico importado en el escenario de la herramienta.

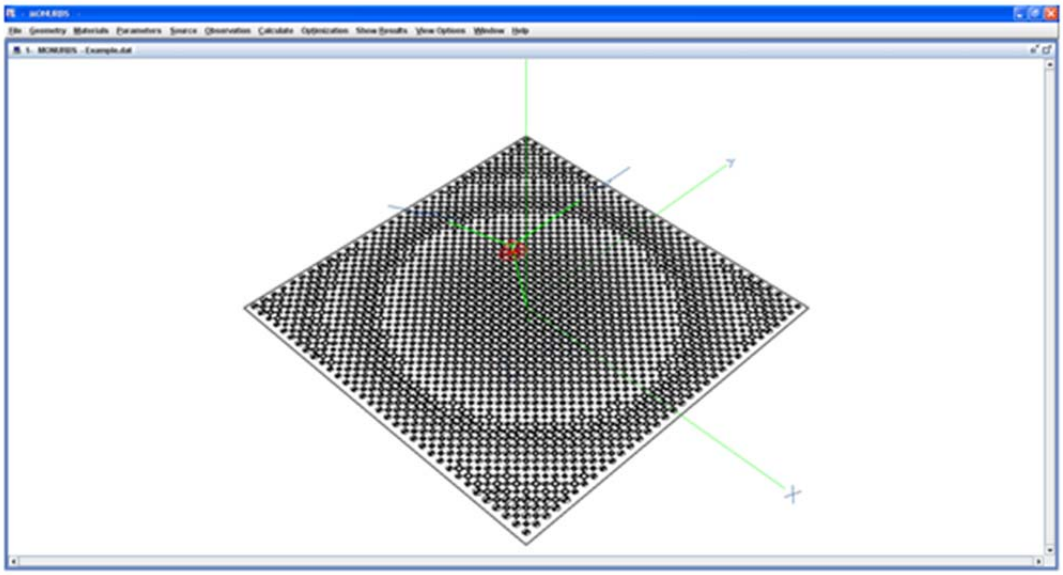


Figura 7.10.- Modelo geométrico del reflectarray importado a partir de un fichero existente creado con una herramienta de CAD externa.

El reflectarray bajo estudio se compone de un plano de masa metálico sobre el que se sitúa un panel de material dieléctrico, cuyo espesor es de 3.175mm y su permitividad es de 2.17. Las celdas metálicas reflectantes se encuentran incrustadas en el dieléctrico, tal y como se puede ver en la figura 7.11, siguiendo un patrón de repetición de 12mm tanto en el eje X como en el eje Y. Las dimensiones totales de la estructura son 40cm x 40cm.

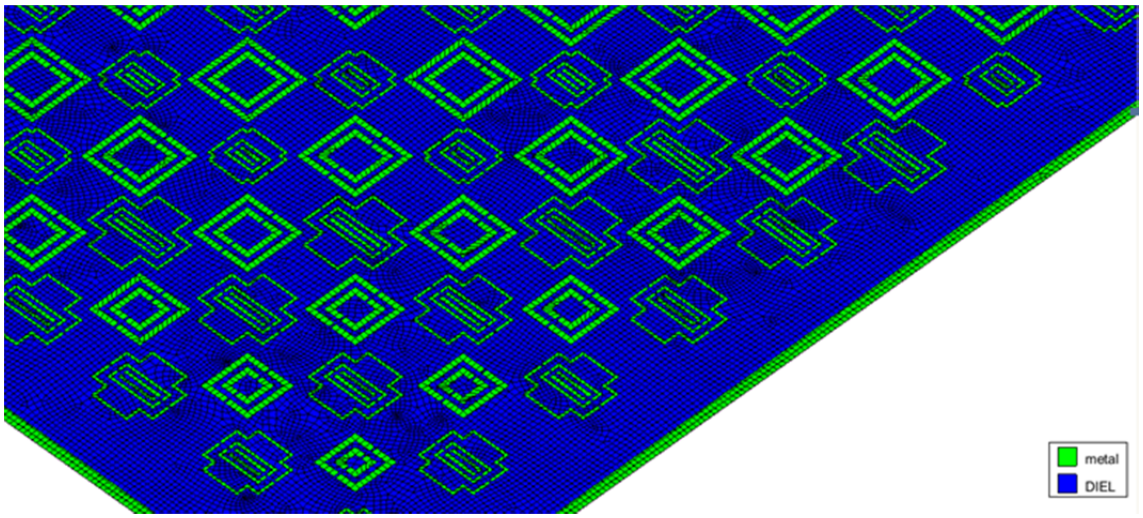


Figura 7.11.- Diferenciación entre los componentes metálicos y dieléctricos.

Debido a que la herramienta considera, por defecto, que todas las superficies que componen el modelo geométrico son conductoras, es necesario crear un nuevo tipo de material que modele el comportamiento del dieléctrico. Por tanto, para analizar correctamente el reflectarray, primero se define el material dieléctrico y se almacena en la base de datos, como se puede ver en la figura 7.12.

El reflectarray es capaz de operar en dos frecuencias distintas, 12GHz y 14GHz, una para cada polarización. Su objetivo es conformar un haz estrecho que radie en una dirección del espacio específica, en este caso dicha dirección viene dada por θ=19º y φ=0º para ambas frecuencias. La alimentación de la antena se realiza mediante una bocina piramidal con 10dB de caída en los bordes, situada en el foco, cuya posición viene determinada por (x_f=-0.124m, y_f=0m, z_f=0.432m), y cuyo diagrama de radiación es de tipo gaussiano.

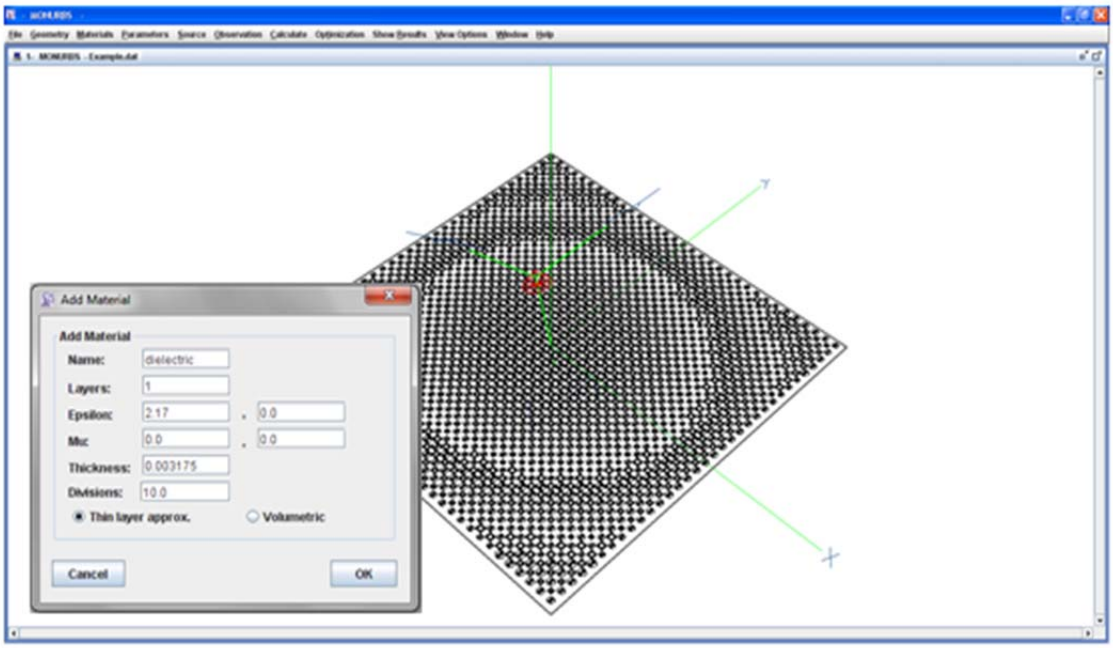


Figura 7.12.- Configuración del material dieléctrico.

Para llevar a cabo el análisis del reflectarray, se aplica el método de las ventanas, que consiste, básicamente, en dividir la geometría en varias bloques y resolver cada uno de ellos de forma independiente. De este modo, se consigue resolver varios problemas pequeños en vez de tener que resolver uno grande. Según se comentó en el capítulo 4, el objetivo de esta técnica es reducir los requerimientos de tiempo y memoria, que son el principal problema del Método de los Momentos al analizar estructuras eléctricamente grandes. La figura 7.13 muestra un cuadro de diálogo en el que se configuran los parámetros del método de resolución.

El reflectarray dual se ha construido y medido en el *Communications Research Centre* de Canadá. El profesor Reza Chaharmir [1] ha proporcionado las medidas para poder compararlas con los resultados obtenidos en las simulaciones. Las figuras 7.14 y 7.15 muestran los resultados de las comparaciones para ambas bandas de frecuencia. En ellas se puede observar que la componente polar del diagrama simulado en ambas gráficas se ajusta muy bien a los resultados reales. También se comprueba que el haz principal apunta en la dirección prevista dada por θ=19º y φ=0º.

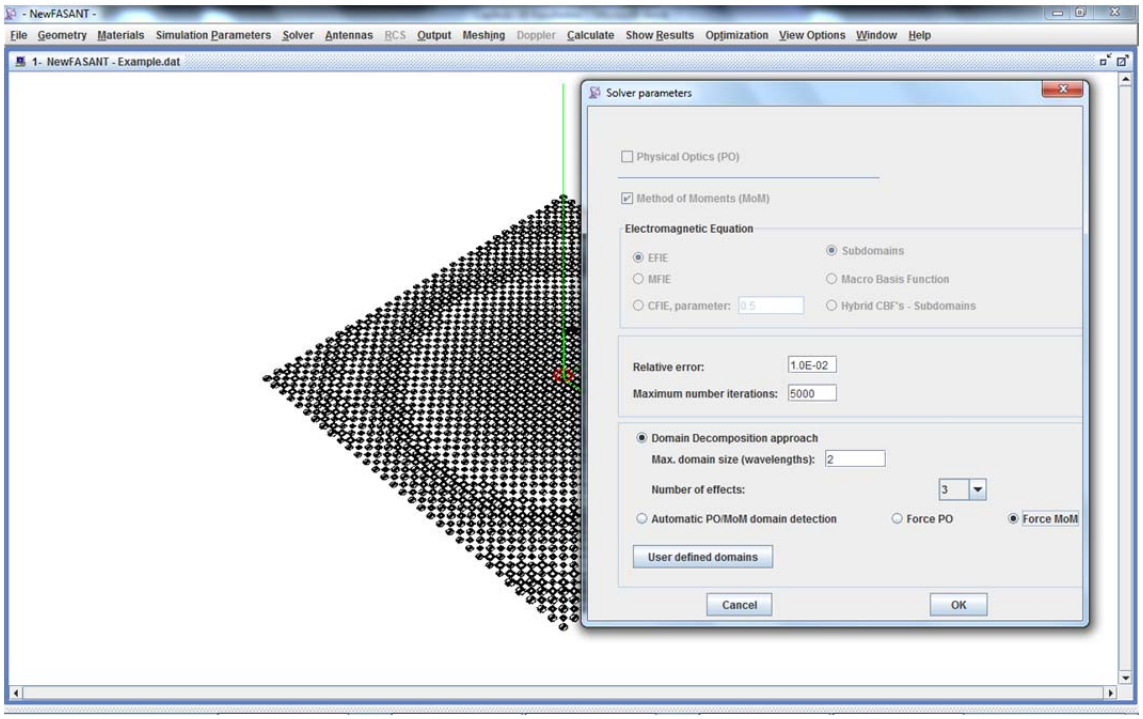


Figura 7.13.- Configuración de los parámetros del método de las ventanas.

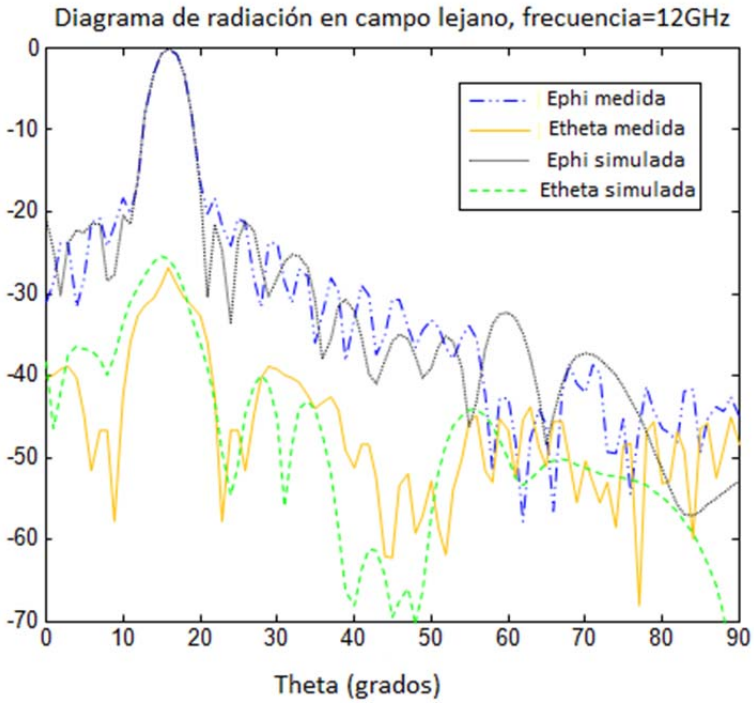


Figura 7.14.- Comparación entre medidas y simulaciones para la frecuencia de 12GHz.

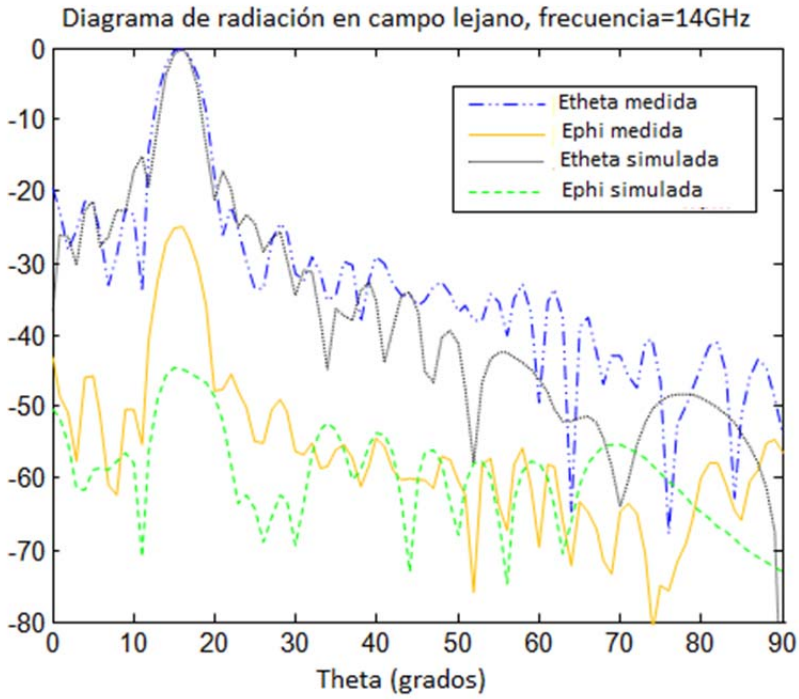


Figura 7.15.- Comparación entre medidas y simulaciones para la frecuencia de 14GHz.

7.5.- DISEÑO Y ANÁLISIS DE UNA ANTENA EBG.

El diseño de la antena que se describe en este apartado se basa en el uso de una estructura EBG (*Electromagnetic BandGap*) compuesta por dos capas metálicas de estructuras periódicas que contienen cavidades rectangulares. Al colocar estos elementos sobre la apertura de una guía de onda circular, se consigue una mejora considerable en sus prestaciones, en términos de ganancia, ancho de banda y pureza de polarización. Durante los últimos años se han presentado varios diseños de antenas, filtros, amplificadores, etc. que incorporan este tipo de estructuras para conseguir mejoras en el rendimiento de varios dispositivos [2-5].

El proceso de diseño de la antena es bastante rápido debido a la variedad de entidades geométricas y operaciones disponibles en la herramienta. La figura 7.16 muestra el modelo geométrico de la antena.

Se trata de una antena compacta, geométricamente compuesta por dos cilindros que modelan la guía de onda, una placa con agujero circular que simula un plano de masa y dos capas de estructuras periódicas que modelan la estructura EBG.

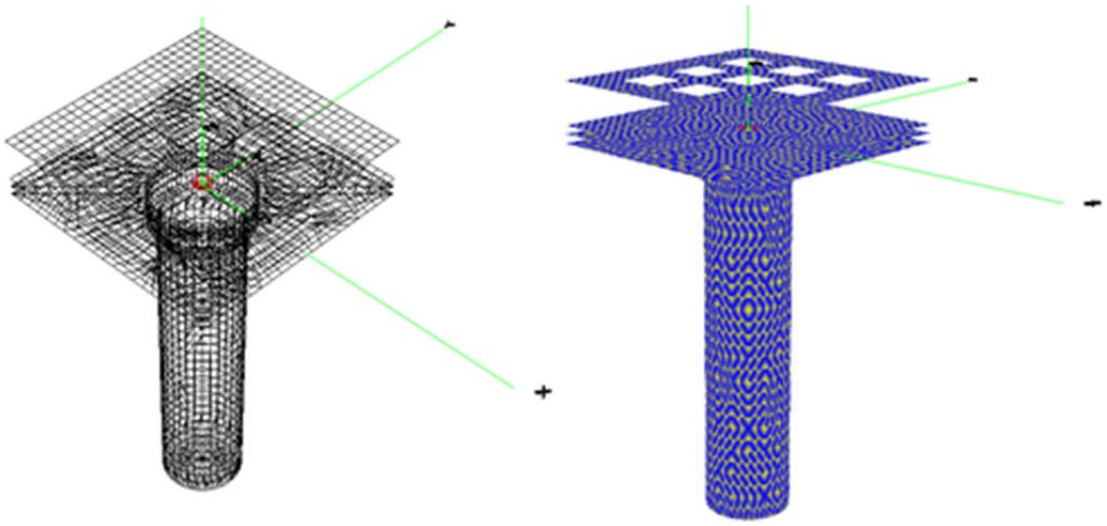


Figura 7.16.- Modelo geométrico de la antena EBG.

La configuración de los parámetros de simulación se ha realizado estableciendo la alimentación mediante un dipolo eléctrico situado a λ/4 del fondo de la guía orientado en el eje Y para conseguir que la antena radie con polarización lineal, la frecuencia de análisis se ha fijado en 9 GHz y se han especificado los dos cortes principales del diagrama de radiación para visualizar los resultados. Respecto al método de resolución, esta vez se ha aplicado el Método de los Momentos tradicional, resolviendo el problema electromagnético mediante la ecuación integral de campo eléctrico.

Después de realizar el mallado de la geometría y llevar a cabo la simulación numérica, se han obtenido los resultados mostrados en las figuras 7.17, 7.18 y 7.19.

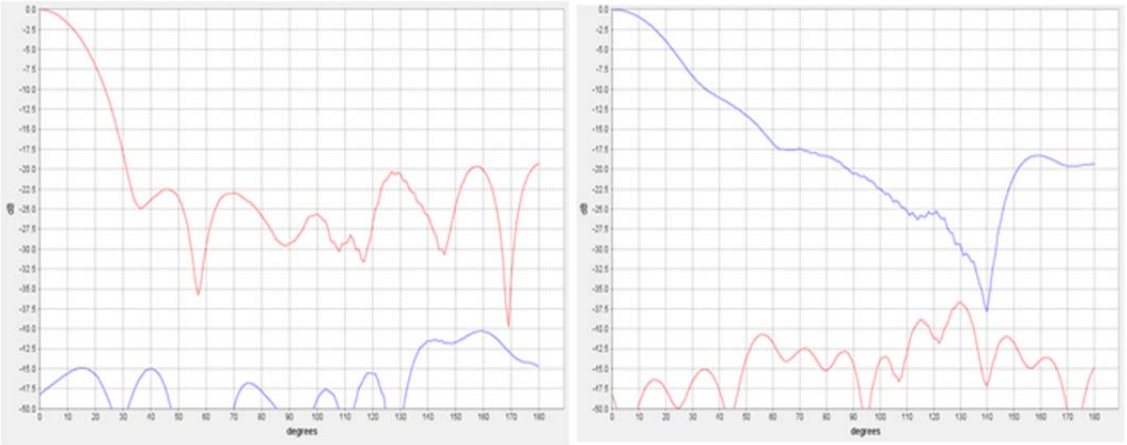


Figura 7.17.- Resultados obtenidos considerando los cortes en el plano E y en el plano H del diagrama de radiación.

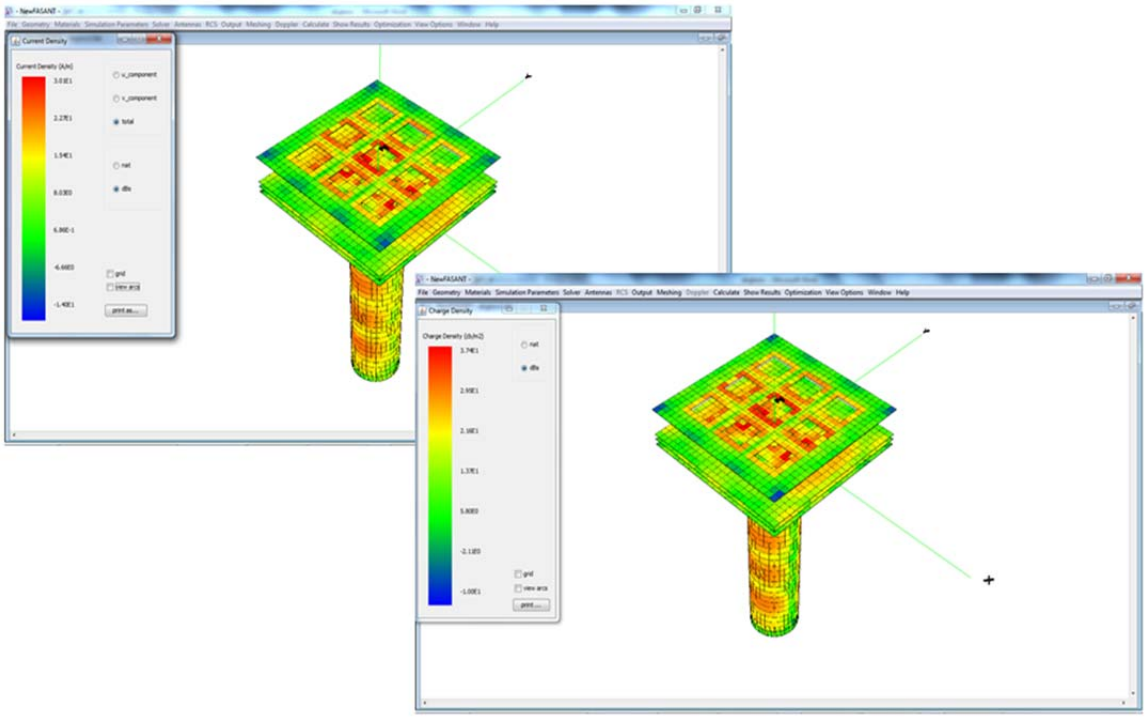


Figura 7.18.- Visualización de la densidad de corriente (izquierda) y de la densidad de carga (derecha) en la superficie de la antena.

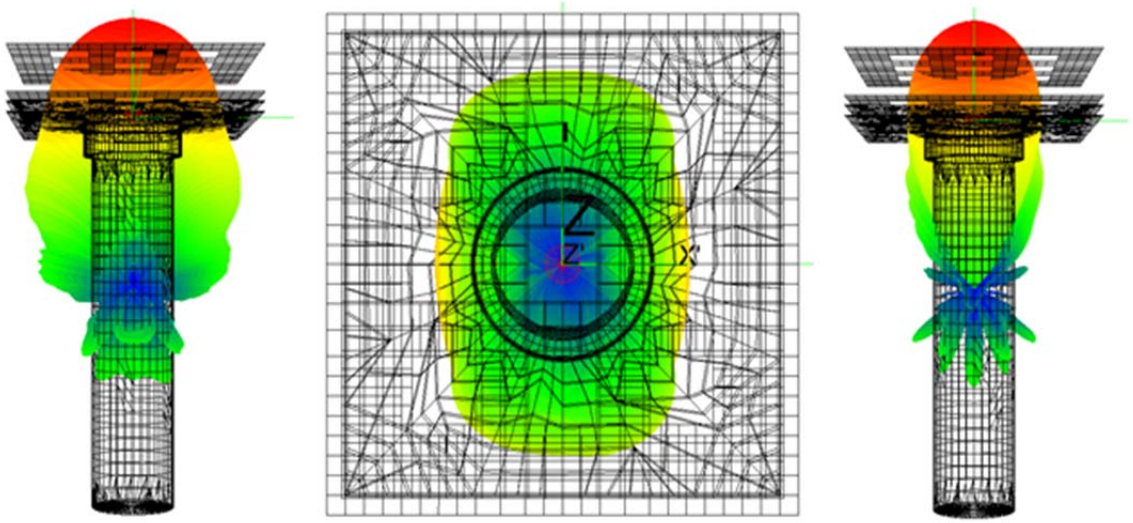


Figura 7.19.- Diagrama de radiación en 3 dimensiones. Vista de frente (izquierda), vista en planta (centro) y vista de frente (derecha).

Una vez finalizada la etapa de simulación, se procede a la construcción de un prototipo real que valide los resultados proporcionados por la herramienta. La figura 7.22 muestra una fotografía de la antena construida.

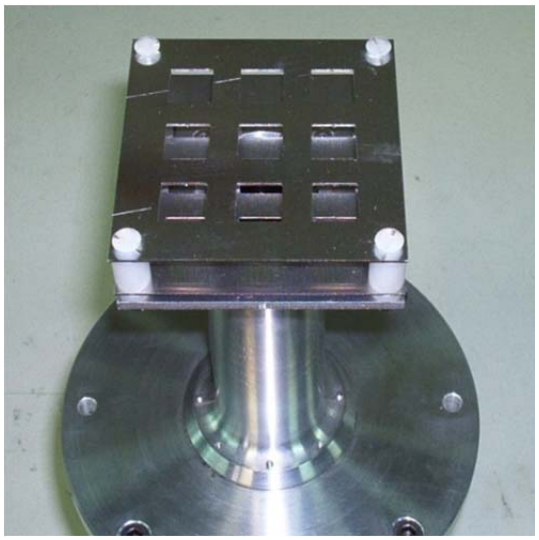


Figura 7.20.- Prototipo de la antena EBG.

La antena está hecha de aluminio y pesa 400gr. aproximadamente. En la imagen se puede ver que las rejillas están sujetas mediante cuatro tornillos de plástico, los cuáles se espera que no disturben demasiado el diagrama de radiación. El prototipo se ha medido en la cámara anecoica del grupo, obteniéndose muy buenos resultados, tal y como se aprecia en las figuras 7.21 y 7.22. La figura 7.23 muestra una imagen de la cámara anecoica.

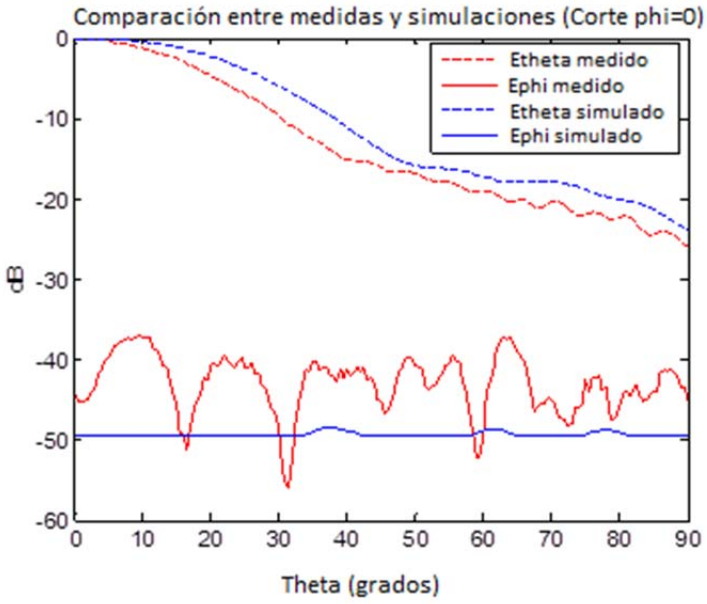


Figura 7.21.- Comparación de los diagramas de radiación en campo lejano. Corte en el plano E a 9GHz.

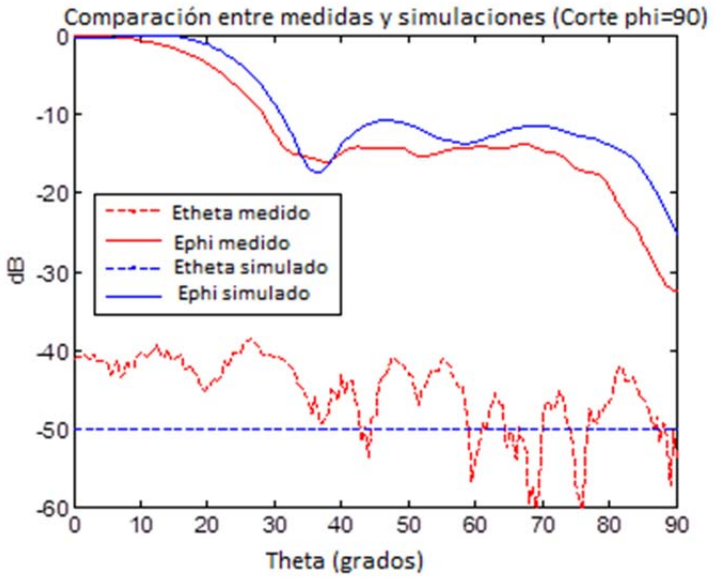


Figura 7.22.- Comparación de los diagramas de radiación en campo lejano. Corte en el plano H a 9GHz.

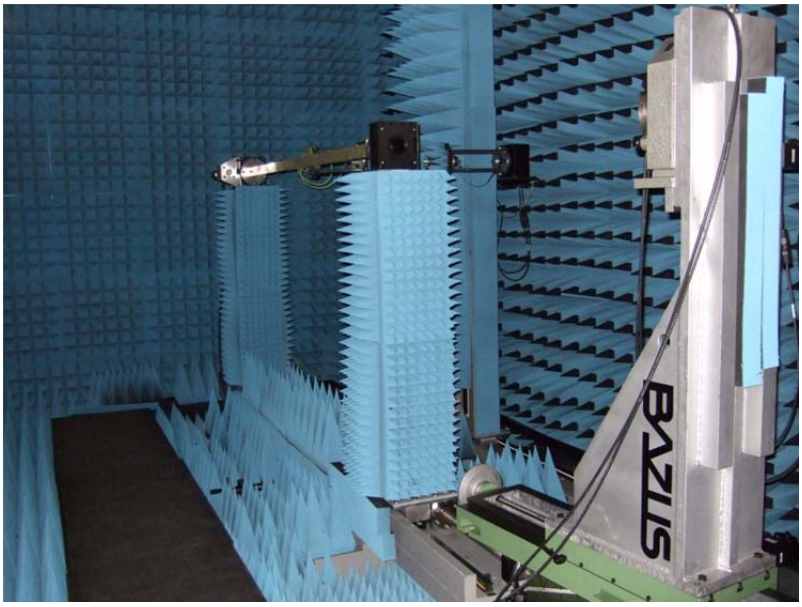


Figura 7.23.- Cámara anecoica donde se ha medido el prototipo de la antena.

Se observa que las medidas reales se ajustan bastante bien a los resultados de las simulaciones. Las pequeñas discrepancias que pueda haber se deben a errores de fabricación que introducen efectos indeseados en el diagrama de radiación. La antena proporciona el máximo de radiación en la dirección axial y presenta un nivel de componente contrapolar por debajo de -40dB tanto para el corte en el plano E como para el corte en el plano H.

A partir de los resultados obtenidos se deduce que esta antena es una buena candidata para un gran número de aplicaciones debido a su excelente pureza de polarización y su pequeño tamaño. Por ejemplo, se puede emplear para el diseño de arrays de bocinas usando la antena EBG como celda unidad. El array obtenido podría proporcionar un diagrama de radiación apropiado para alimentar a un reflector multi-haz. Para esta aplicación en particular, el uso de un array que utilice bocinas convencionales no es posible, ya que para proporcionar una iluminación correcta, el tamaño de la apertura de dichas bocinas debería ser excesivamente grande, lo que provocaría colisiones entre las ellas al intentar ocupar el mínimo espacio posible para bloquear al menor número de rayos.

7.6.- ANÁLISIS Y OPTIMIZACIÓN DE UNA ANTENA DE HÉLICE CUADRIFILAR.

Este apartado presenta el proceso de diseño y optimización de una antena de hélice cuadrifilar para aplicaciones TTC (*Telemetry, Tracking and Control*) [6, 7]. Las antenas TTC son un componente esencial en las comunicaciones por satélite, puesto que llevan a cabo la monitorización y el control de las aeronaves desde estaciones terrestres. Además, estos sistemas permiten medir, recoger y transmitir comandos, así como calcular posiciones, distancias y otras medidas. Típicamente, las características electromagnéticas de este tipo de antenas son ganancia baja, cobertura hemisférica y polarización circular. Las estructuras tipo hélice han sido usadas ampliamente a lo largo de la historia para operar en este tipo de sistemas [8]. Entre sus ventajas destacan su tamaño compacto, comportamiento resonante y capacidad de funcionamiento en varias frecuencias.

El modelo geométrico de la antena diseñada se compone de un cilindro, que representa la base de la antena, y un array circular de cuatro hélices. Como se puede ver en la figura 7.24, las hélices opuestas están cortocircuitadas en la zona superior. Tal y como se explicó en el capítulo 4, la herramienta desarrollada facilita la creación de varios tipos de antenas, entre ellas, la antena de hélice.

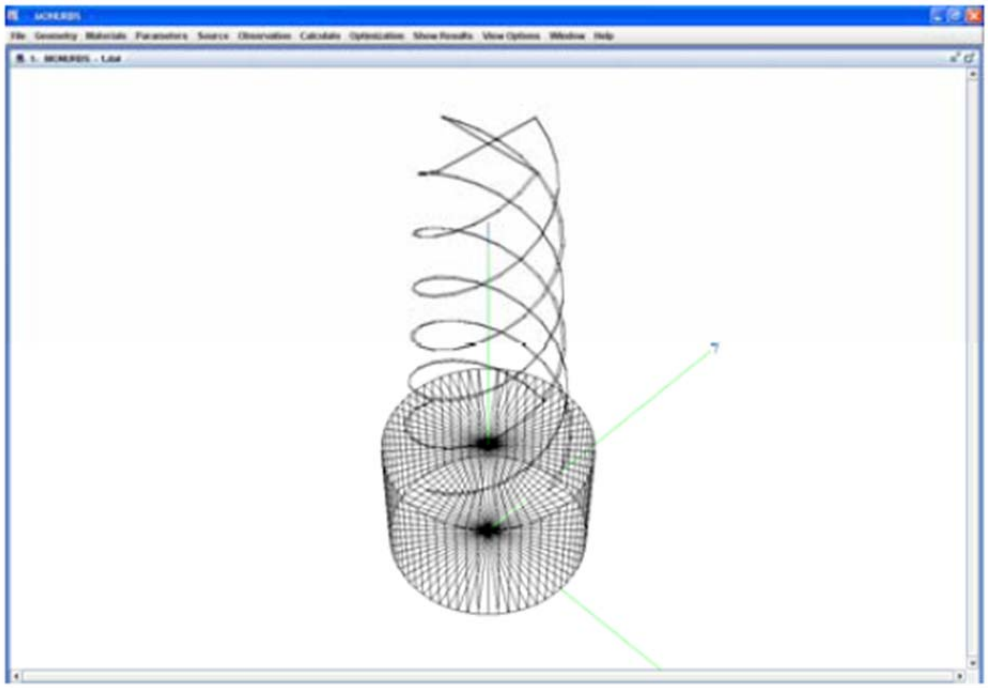


Figura 7.24.- Modelo geométrico de la antena de hélice cuadrifilar.

Debido a que la antena de hélice es una estructura resonante, sus dimensiones físicas deben ajustarse para proporcionar una buena respuesta considerando una frecuencia de operación determinada. En este caso particular, el proceso de optimización de parámetros se realiza teniendo en cuenta una serie de requisitos que la antena debe satisfacer. Las especificaciones que debe cumplir son las siguientes:

1. La antena debe funcionar correctamente en la banda S con operación dual tanto a 1.81GHz como a 2.25GHz.
2. La polarización debe ser circular a derechas.
3. Respecto a la ganancia, el máximo debe estar por encima de 2dBi dentro del rango determinado por los valores de θ=70º hasta θ=110º.
4. La diferencia entre las componentes polar y contrapolar debe ser mayor de 12dB en dicho margen angular.
5. El peso del prototipo debe ser lo más bajo posible. El límite permisible está en 550gr, incluyendo el peso del sistema de alimentación.
6. Otro aspecto a considerar es la robustez, puesto que la antena debe ajustarse muy bien a la superficie del satélite y soportar fuerzas de fricción, altas temperaturas, fuerzas gravitacionales, etc.

La alimentación de la antena se lleva a cabo especificando cuatro puntos de alimentación en la zona de contacto entre las hélices y la base. El voltaje aplicado en cada punto es de 1V de amplitud y una fase que va variando estableciendo una diferencia de fase de 90º entre puntos adyacentes. De esta forma, se consigue que la antena radie con polarización circular. Para que el sentido sea circular a derechas, las fases de los puntos alimentados se fijan en 0º, 90º, 180º y 270º girando el sentido contrario al de las agujas del reloj.

La antena presenta un diagrama de radiación con simetría de revolución referida al eje Z, por lo que sólo es necesario establecer un corte en φ=0º con θ variando desde 0º hasta 180º.

El diagrama de radiación de este tipo de antenas depende principalmente del número de vueltas de la hélice, de la altura y de los radios superior e inferior. Para determinar cuáles son los valores óptimos de cada parámetro, considerando las especificaciones mencionadas anteriormente, se ha llevado a cabo un exhaustivo estudio paramétrico.

La figura 7.25 muestra el cuadro de diálogo en el que se indican los parámetros que se desean optimizar, junto con su rango de variación. Como se puede ver, el radio inferior variará de 1.5cm hasta 2.5cm, el radio superior de 0.5cm hasta 1.5cm, la altura de 10cm hasta 15cm y el número de vueltas de 0.5 hasta 1.5.

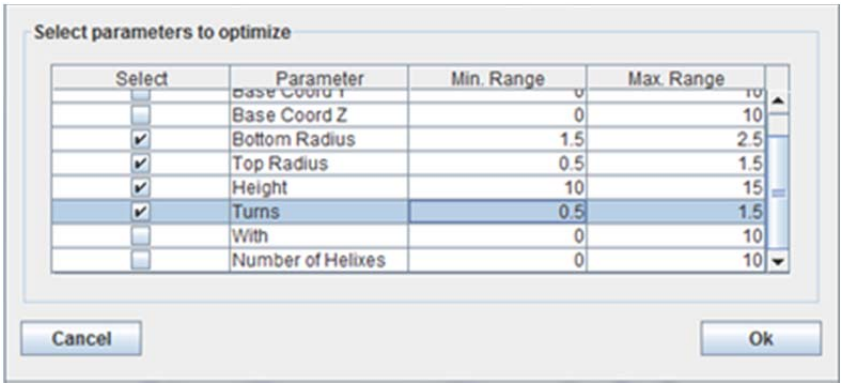


Figura 7.25.- Definición de los parámetros a optimizar y selección de rango de variación.

A continuación, se establece el algoritmo de optimización 'Gradiente Descendente' y se define la función de coste. Para ello, se especifica una zona de

optimización del nivel contrapolar, en la que se debe seleccionar el corte en φ, con los valores que se muestran en la figura 7.26. Según las especificaciones, la diferencia entre las componentes polar y contrapolar debe ser mayor de 12dB en el rango comprendido entre θ=70º y θ=110º.

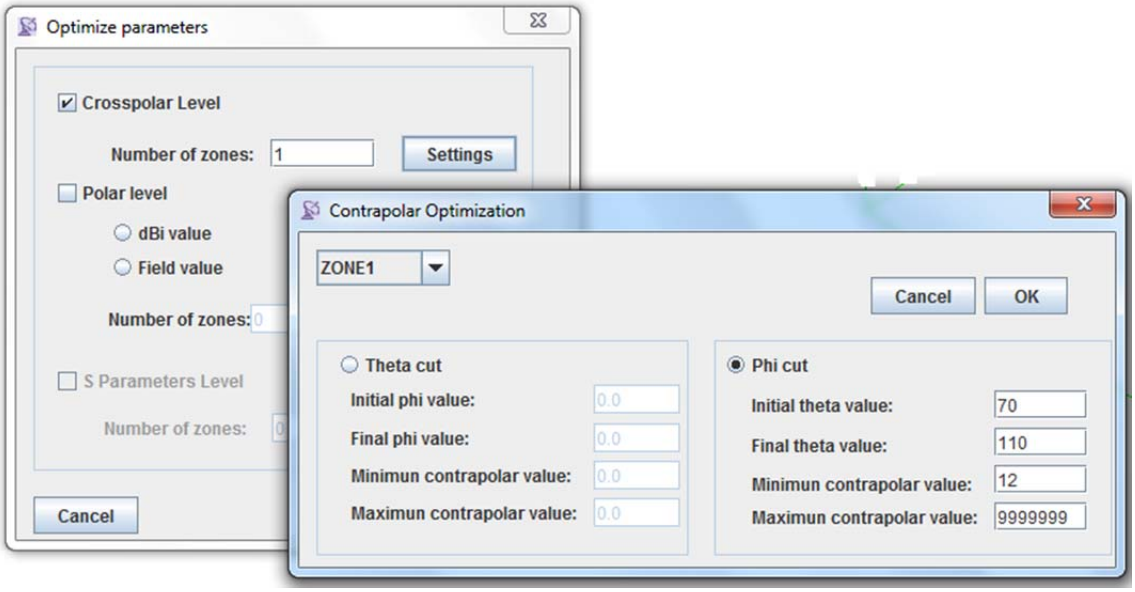


Figura 7.26.- Definición de la función de coste.

Los parámetros óptimos obtenidos al finalizar el proceso de optimización son:

-Número de vueltas. 0.831
-Radio inferior: 1.945cm
-Altura: 13.8cm
-Radio superior: 1.022cm

La figura 7.27 muestra la componente polar en dBi para las dos bandas de frecuencia, obtenida para un barrido angular desde 0º a 180º en la componente θ, fijando las dimensiones óptimas. Se observa que ambas curvas cumplen los requisitos establecidos en el punto 3 de la lista, en el que se indica que el valor de ganancia en el rango angular comprendido entre θ=70º y θ=110º debe ser mayor de 2dBi.

Por otra parte, la figura 7.28 muestra la diferencia en dB de los niveles polar y contrapolar, donde la restricción de 12dB se cumple también para ambas frecuencias.

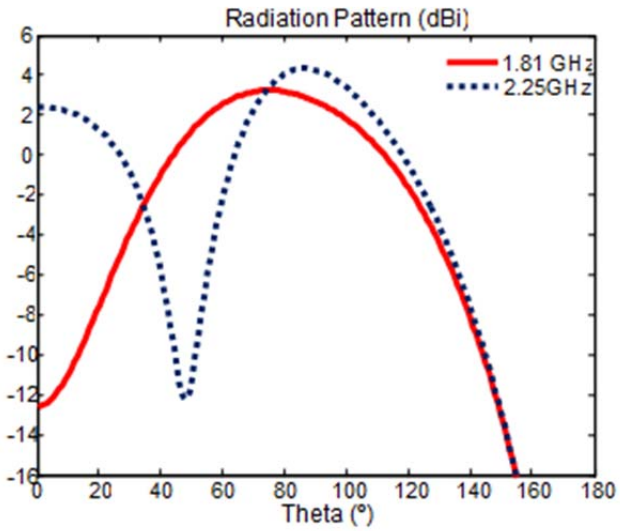


Figura 7.27.- Resultados de la simulación para el corte φ=0º. Componente polar en dBi.

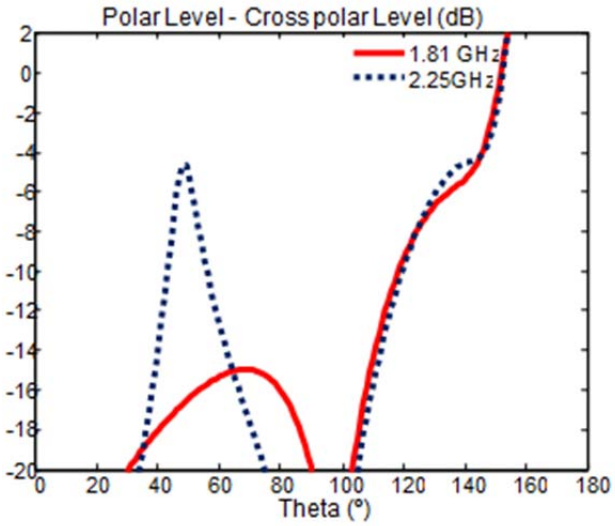


Figura 7.28.- Resultados de la simulación para el corte φ=0º. Diferencia entre los niveles polar y contrapolar en dB.

Una vez que los resultados de las simulaciones cumplen las especificaciones impuestas, se procede a la fabricación de un prototipo real. La figura 7.29 muestra una fotografía de la antena construida. Se observa que se ha añadido un poste metálico en su parte interior, necesario para fijar fuertemente las tiras de la hélice y proporcionar mayor robustez mecánica de acuerdo con el requisito número 6. Respecto a los requerimientos de tamaño y peso, éstos también se cumplen satisfactoriamente con las dimensiones óptimas proporcionadas por el módulo de optimización de la herramienta.

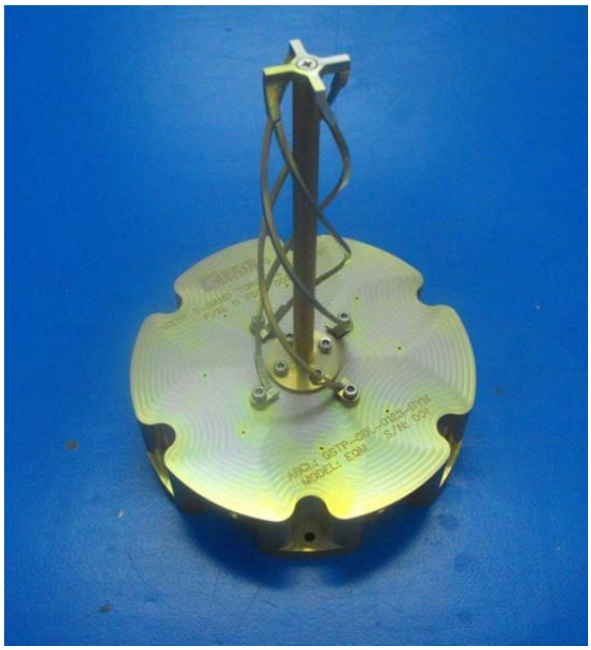


Figura 7.29.- Prototipo construido.

La última etapa del proceso consiste en medir el prototipo en una cámara anecoica y verificar que las medidas reales se corresponden con las predicciones obtenidas. Las comparaciones entre medidas y simulaciones a 1.81GHz y a 2.25GHz se muestran en las gráficas de las figuras 7.30 y 7.31 respectivamente. En ellas se puede ver que los resultados de las medidas reales se ajustan bastante bien a los que se han obtenido en las simulaciones. Las posibles discrepancias entre ambos resultados se pueden deber a modificaciones en la geometría debidas a la inserción de tornillos y otros elementos mecánicos incluidos en el proceso de construcción del prototipo.

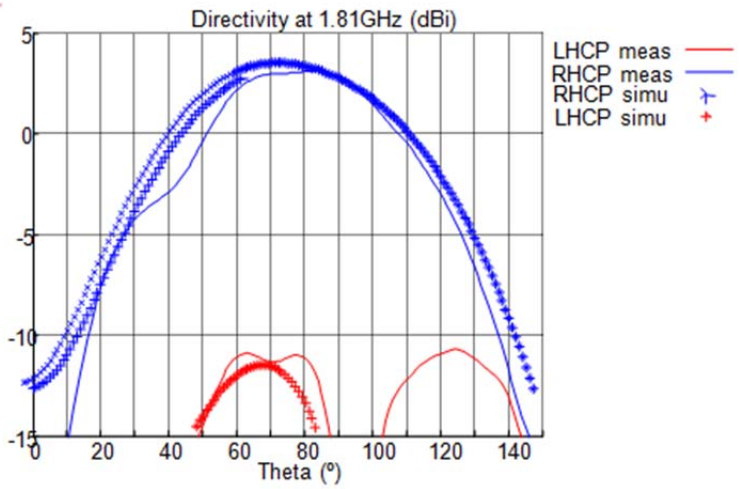


Figura 7.30.- Comparación entre medidas y simulaciones a la frecuencia de 1.81GHz.

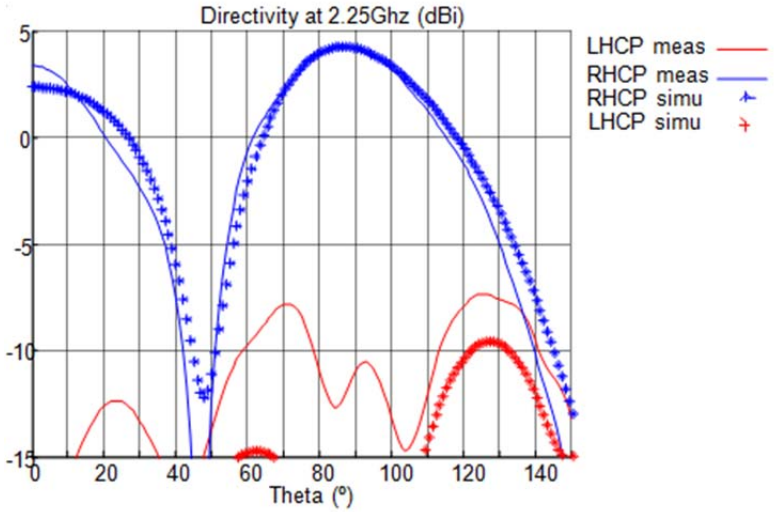


Figura 7.31.- Comparación entre medidas y simulaciones a la frecuencia de 2.25GHz.

Por último, mencionar que este proyecto ha sido posible gracias a la colaboración con la empresa RYMSA, para llevar a cabo el proyecto denominado "S band Toroidal Antenna".

7.7.- ANÁLISIS Y OPTIMIZACIÓN DE UNA SONDA CORRUGADA.

En este apartado se describe el proceso de diseño, análisis y optimización de una sonda rectangular corrugada para es realizar medidas de antenas en campo cercano.

La dificultad principal para medir directamente antenas grandes eléctricamente en campo lejano estriba en las dimensiones tan enormes que debe tener el campo de medida. Una solución a este problema es medir el campo cercano y, mediante un proceso matemático, hallar el campo radiado lejano. En las medidas en campo lejano es habitual emplear como sondas antenas directivas, con el fin de mejorar el margen dinámico y reducir el efecto de las reflexiones. Por el contrario, en las medidas de campo cercano es importante que la sonda presente un diagrama de radiación plano y omnidireccional con una gran pureza de polarización para proporcionar un margen angular amplio. Estos aspectos son fundamentales para evitar problemas en la posterior corrección de sonda al convertir los resultados a campo lejano [9].

A la hora de diseñar la sonda, se deben considerar las siguientes especificaciones:

1. El diagrama de radiación debe ser omnidireccional plano con nivel de componente polar por encima de -10dB al menos hasta 60º.
2. La sonda debe proporcionar buena pureza de polarización con nivel de componente contrapolar por debajo de -35dB.
3. El prototipo debe presentar unas dimensiones y peso mínimos.
4. La frecuencia de trabajo se establece en 1GHz.

Al igual que ocurría con la antena de hélice, el modelo de la sonda corrugada también se genera directamente mediante el módulo de antenas disponible en la interfaz.

En este caso, la alimentación de la sonda se modela mediante un dipolo orientado en el eje X situado a $\lambda/4$ de la base, tal y como se puede ver en la figura 7.32.

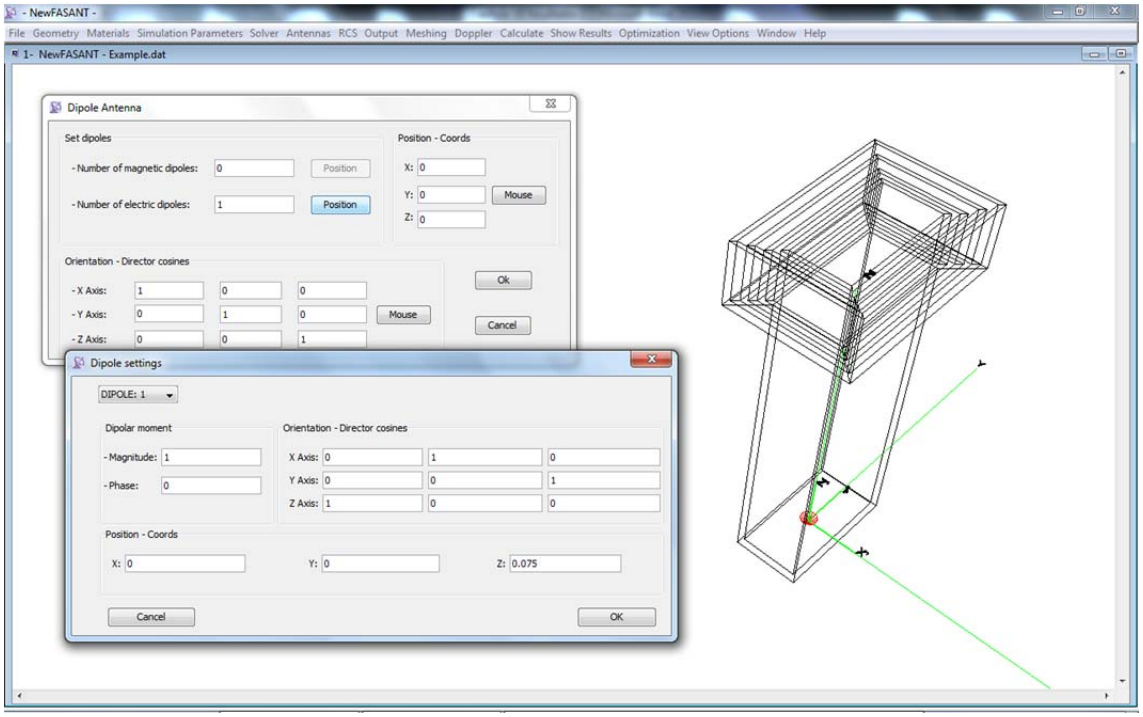
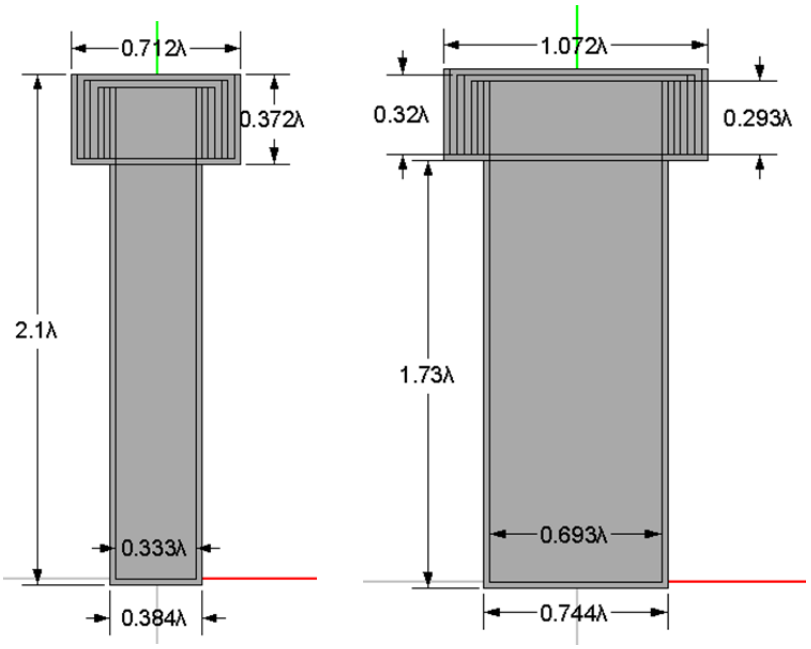


Figura 7.32.- Alimentación de la sonda mediante un dipolo eléctrico.

Para poder afirmar que la sonda tiene buena pureza de polarización es necesario que el nivel de componente contrapolar en los diagramas de radiación de los tres cortes

principales (plano E, plano H y plano φ=45º) esté por debajo de, al menos, -35dB [10]. Como este requerimiento es el más difícil de alcanzar, se especificará en la función de coste, de forma que se obtengan las dimensiones óptimas considerando el nivel de pureza de polarización en los tres cortes.

Entre los parámetros a optimizar, destacan el número de corrugaciones y las diferentes alturas y anchuras de cada una de ellas. El modelo geométrico de la sonda obtenido tras ejecutar el proceso de optimización se muestra en la figura 7.33, donde las dimensiones se dan en función de la longitud de onda.



Figu**ra 7.33.-** Dimensiones óptimas de la sonda.

Las 4 corrugaciones presentan una anchura de 0.0253λ. Las alturas optimizadas de cada una son 0.293λ, 0.293λ, 0.32λ y 0.347λ y las distancias entre ellas se establecen en 0.0267λ, 0.0293λ y 0.032λ.

En la figura 7.34a se muestra el modelo renderizado, mientras que en la figura 7.34b se puede observar el resultado del algoritmo de mallado.

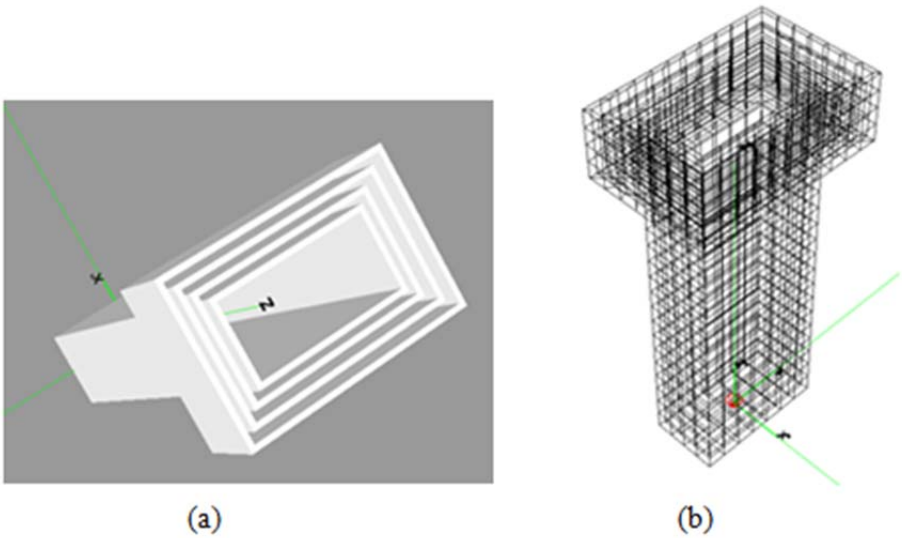


Figura 7.34.- Modelo geométrico de la sonda con las dimensiones óptimas. (a) Modelo renderizado. (b) Modelo discretizado.

La figura 7.35 muestra los resultados proporcionados por la herramienta para los cortes en los planos E y H respectivamente una vez finalizado el proceso de optimización.

Figura 7.35.- Resultados obtenidos para los cortes φ=0º y φ=90º.

Para poder verificar si se cumplen los requisitos establecidos inicialmente, se han calculado las componentes de campo en X y en Y para los tres cortes principales del diagrama. Las curvas obtenidas se muestran en la figura 7.36, de donde se desprenden las siguientes conclusiones:

- El nivel de la componente polar se mantiene por encima de -10dB hasta los 60º, lo que asegura un diagrama muy plano, casi omnidireccional.
- El nivel de la componente contrapolar se mantiene claramente por debajo de -41dB a lo largo de los tres cortes principales, por lo que la sonda proporciona una excelente pureza de polarización.
- Se satisfacen todos los requisitos impuestos inicialmente, pues aparte de los dos puntos anteriores, se ha obtenido una sonda compacta con unas dimensiones óptimas mínimas, lo que la convierte en un diseño muy apropiado para realizar mediciones en campo cercano.

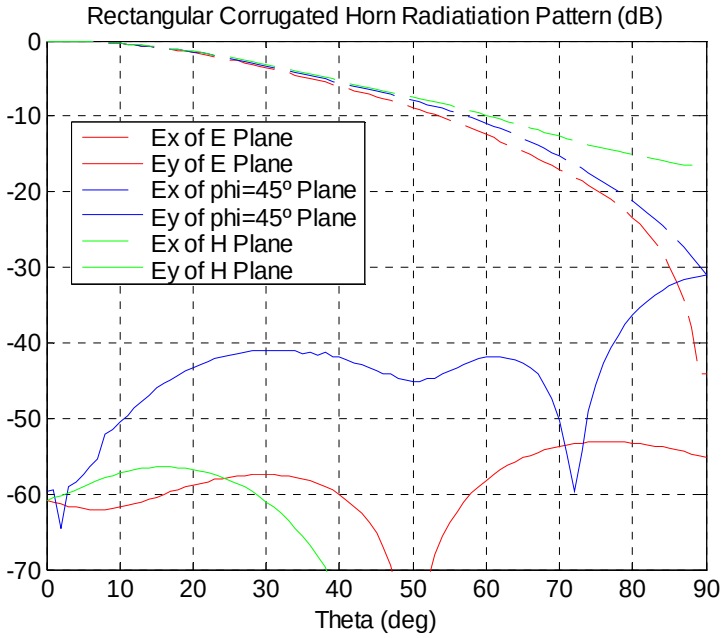


Figura 7.36.- Resultados obtenidos tras optimizar la sonda.

Respecto al resto de resultados obtenidos, en la parte izquierda de la figura 7.37 se puede ver la representación de la densidad de corriente sobre la superficie de la sonda. Por otro lado, en la parte derecha de la imagen se muestra la densidad de carga, estableciendo la opción de visualización de vista en planta para poder ver la distribución de la carga sobre las corrugaciones.

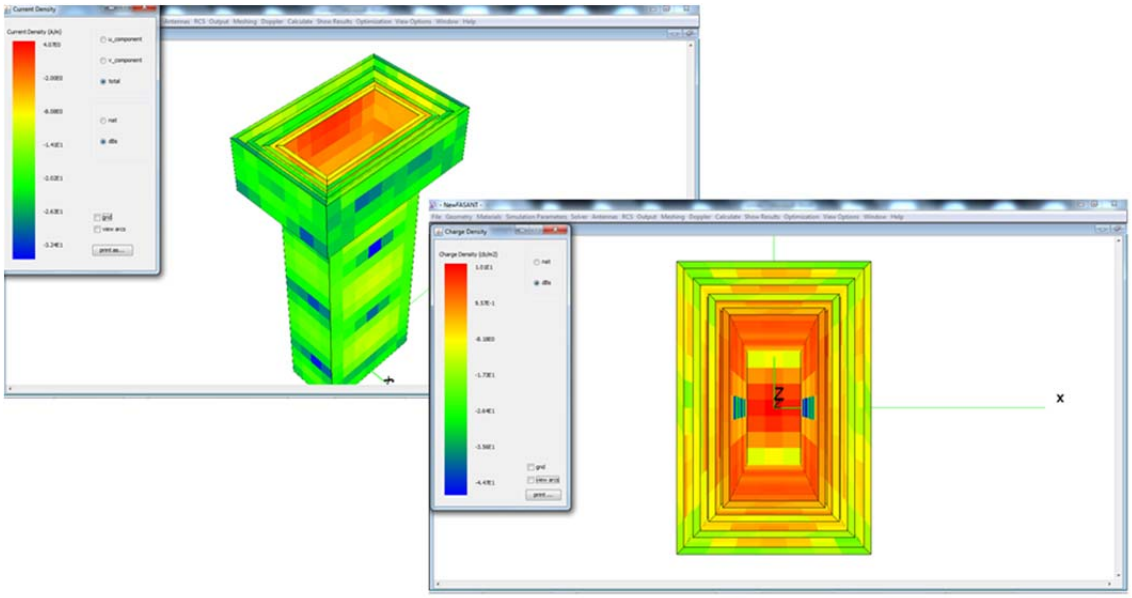


Figura 7.37.- Densidad de corrientes (izquierda) y de carga (derecha) sobre la superficie de la sonda.

Por último, la figura 7.38 muestra el diagrama de radiación en tres dimensiones sobre el modelo geométrico de la sonda. En dicha imagen se puede comprobar que es muy poco directivo, prácticamente omnidireccional, tal y como se requería en las especificaciones.

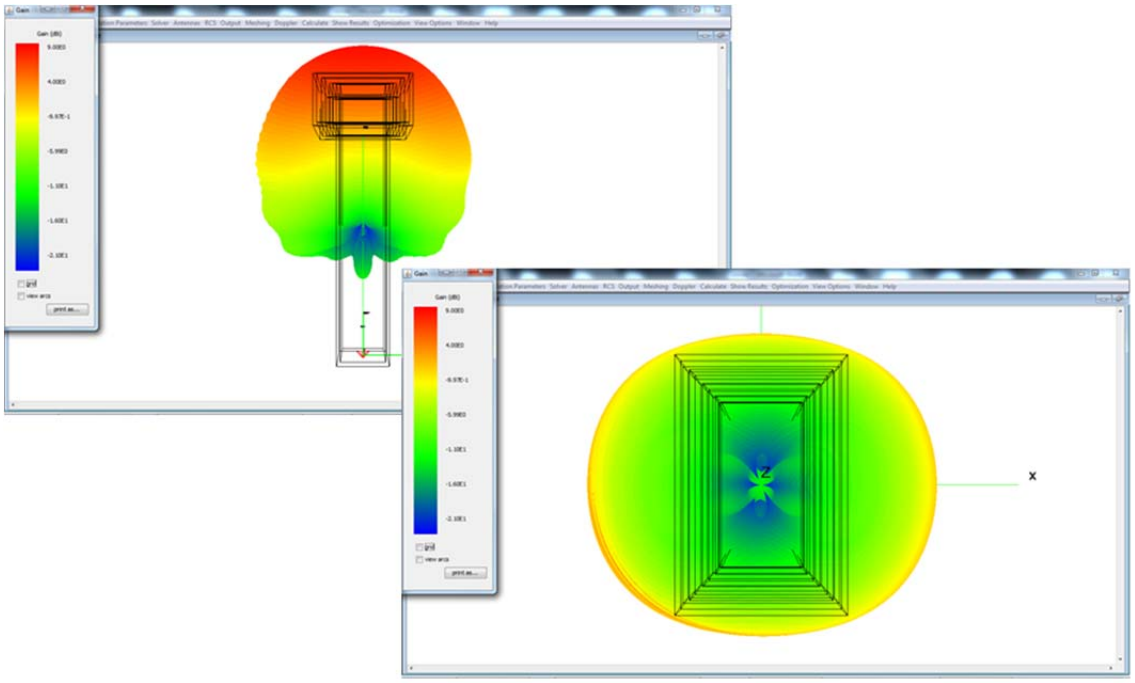


Figura 7.38.- Diagrama de radiación tridimensional de la sonda. Vista de perfil (izquierda) y vista en planta (derecha).

7.8.- REFERENCIAS.

[1].- M.R. Chaharmir, J. Shaker, N. Gagnon, "Novel broadband dual-band linear orthogonal polarization reflectarray", *Electronics Letters*, Jan. 2009, Vol. 45, No. 1, pp. 13-14.

[2].- G. K. Palikaras, A. P. Feresidis, J. C. Vardaxoglou, "Cylindrical Electromagnetic Bandgap Structures for Directive Base Station Antennas", *IEEE Antennas and Wireless Propagation Letters*, vol. 3, 2004.

[3].- A.R. Weily, K.P. Esselle, T.S. Bird, and B.C. Sanders, "Dual resonator 1-D EBG antenna with slot array feed for improved radiation bandwidth", *IET Microwaves, Antennas & Propagation*, vol. 1, Issue 1, February 2007.

[4].- J.-W. Baik, S.-M. Han, C. Jeong, J. Jeong and Y.-S. Kim, "Compact Ultra-Wideband Bandpass Filter With EBG Structure", *Microwave and Wireless Components Letters*, vol. 18, Issue 10, October 2008.

[5].- Yahya Rahmat-Samii, "The Marvels of Electromagnetic Band Gap (EBG) Structures," *ACES Journal*, vol. 18, no. 3, pp. 1-10, 2003.

[6].- Albertsen, N. C., Balling, P., Gregersen, F. R., and Laursen, F., "Low gain S-band TT and C antennas. Development of cardioid coverage antenna". TICRA final report S-44-02, Denmark, 1976.

[7].- M. F. Cátedra, J. C. Cruellas, J. F. Díaz, "Dual Band Antenna for Telemetry, Tracking and Control (TTC) of Satellites at Ku-Band", *Electronic Letters*, vol.22, December 1986.

[8].- S. Wang, S.J. Chung, "A Miniature Quadrifilar Helix Antenna for Global Positioning Satellite Reception", *IEEE Transactions on Antennas and Propagation,* vol.57, No.12, December 2009.

[9].- Yaghjian, A. "An overview of near-field antenna measurements", *IEEE Transactions on Antenna and Propagation*, vol. 34, Jan.1986, pp. 30-45.

[10].- G. Fedi, S. Manetti, G. Pelosi and S. Selleri, "Profiled Corrugated Circular Horns Analysis and Synthesis Via an Artificial Neural Network", *IEEE Tansactions on Antennas and Propagation*, Vol. 49, No.11, Nov 2001.

8.- Conclusiones y Futuras Líneas de Trabajo.

8.1.- CONCLUSIONES.

En esta tesis se ha presentado el desarrollo de una interfaz gráfica de usuario implementada en Java para el diseño, optimización y análisis de estructuras complejas mediante el Método de los Momentos. Todo la parte relacionada con visualización del modelo geométrico, así como su manipulación mediante la aplicación de transformaciones geométricas, ha sido posible gracias al uso de las librerías gráficas de Java3D.

Se ha llevado a cabo la implementación de un sistema fácil de usar, robusto y extensible que integra todas las etapas del proceso de simulación: creación del modelo geométrico, configuración de los parámetros de simulación, ejecución del núcleo electromagnético, visualización de los resultados obtenidos y optimización de parámetros geométricos. De este modo, se consigue unificar todas las etapas mencionadas en un único sistema, sin tener que depender de herramientas externas.

Además, la herramienta desarrollada proporciona opciones gráficas avanzadas como por ejemplo, un módulo especializado en diseño de antenas, mediante el cual se agiliza el proceso de creación del modelo geométrico, estableciendo el tipo de alimentación de forma sencilla e intuitiva. Las antenas disponibles son de varios tipos de bocinas, reflectores simples y dobles, lentes, antenas de hilo, de parche, etc.

Otra novedad interesante desde el punto de vista computacional ha sido la incorporación de un gestor de simulaciones que permite realizar análisis en modo local, ejecutando el núcleo en el propio ordenador donde está instalada la herramienta, o en modo remoto, ejecutando el núcleo en máquinas ajenas computacionalmente muy

potentes que permiten llevar a cabo complejas simulaciones numéricas. Utilizando cualquiera de estos dos modos es posible ejecutar también la versión paralelizada del núcleo para obtener un mejor rendimiento del sistema.

Por último, destacar la implementación del módulo de optimización, desarrollado para calcular las dimensiones óptimas de las estructuras bajo análisis, en base a unas especificaciones establecidas mediante una función de coste.

8.2.- FUTURAS LÍNEAS DE TRABAJO.

El trabajo desarrollado hasta ahora se ha realizado de forma que se permita incluir ampliaciones y mejoras en el sistema sin demasiado esfuerzo. La programación modular permite extender la GUI fácilmente, manteniéndola siempre actualizada en caso de incluir nuevas funcionalidades en el núcleo electromagnético que involucren a la parte gráfica de la herramienta.

Respecto a la inclusión de nuevas funcionalidades, se destacan las siguientes:

- Implementar un sistema de colas que permita gestionar varias solicitudes para ejecutar simulaciones remotas en la parte del servidor. Se puede establecer un orden de prioridad en función de la complejidad de cada caso, en función del usuario, etc.

- Actualizar el módulo de antenas de la herramienta de acuerdo con las investigaciones e innovaciones que se vayan realizando. También se deberán añadir algunos modelos de antenas utilizados ampliamente en la actualidad, como antenas Yagi, bicónicas, Vivaldi, antenas fractales, etc.

- Implementar nuevos métodos de optimización, como algoritmos genéticos, método de Newton, métodos Quasi-Newton, etc.

- Añadir más opciones para optimizar otros parámetros que definen el comportamiento electromagnético, como impedancias o valores de campo cercano.

- Modificar la forma de configurar la función de coste para permitir establecer una penalización mayor o menor en función de unos determinados criterios. La penalización actual se ha fijado en una unidad por cada dB que caiga fuera del margen determinado por la función de coste. La inclusión de esta opción permitiría establecer un mayor o menor coste (número de unidades por dB) dependiendo de la zona del diagrama, o del corte a optimizar.

- Mejorar el módulo de visualización de resultados utilizando paquetes especializados como por ejemplo DISLIN, cuya función es representar datos de resultados científicos. Se trata de unas librerías de alto nivel capaces de dibujar diversos tipos de diagramas y gráficos 2D/3D.

- Aprovechar las herramientas libres que hay disponibles para realizar interfaces gráficas de alto nivel. Por ejemplo, el paquete VTK (*Visualization Toolkit*) incluye técnicas avanzadas de visualización y modelado e incluso algoritmos de mallado propios. Además, dispone de una serie de capas de enlace con lenguajes como Python y Java, por lo que permite desarrollar interfaces gráficas programando en alguno de estos lenguajes. Otras bibliotecas que proporcionan funcionalidades gráficas similares son DrawP3D, MAM-VRS (*Modeling and Animation Machine, Virtual Rendering System*), OpenRM, VolPack y Math3D.