



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Desarrollo de herramienta informática de apoyo a la docencia de
Sistemas de Radiocomunicaciones.*

Grado en Ingeniería Mecánica

ALUMNO: Gonzalo José Peñuelas Pérez-Cuadrado

DIRECTORES: Sandra Castro Cao

CURSO ACADÉMICO: 2015-2016

UniversidadeVigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Desarrollo de herramienta informática de apoyo a la docencia de
Sistemas de Radiocomunicaciones.*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

En el ámbito de la docencia, las clases prácticas suponen una herramienta de apoyo fundamental para alcanzar el objetivo de fijar los contenidos teóricos de las materias mediante la aplicación práctica de los mismos. El objetivo de este trabajo de fin de grado es crear una herramienta que permita al alumno, mediante la simulación de escenarios de enlaces y mecanismos de radiopropagación, así como con la resolución de cuestionarios, reforzar los conceptos ligados a la asignatura de Sistemas de Radiocomunicaciones. Para ello se implementará una aplicación basada en un interfaz gráfico de usuario sencillo e intuitivo. Dicha aplicación se implementará como un módulo del programa Matlab® utilizado en el laboratorio de la asignatura.

PALABRAS CLAVE

Sistemas de Radiocomunicaciones, MATLAB, aplicación, GUI, ayuda a la docencia.

AGRADECIMIENTOS

Quiero agradecer a todos aquellos que a lo largo de estos cinco años han estado a mi lado.

Mi familia, sin la cual nada de esto habría sido posible, los cuales han estado siempre apoyándome y de quienes me siento muy orgulloso.

A mi tutora por guiarme durante estos últimos meses, ayudándome en todo momento y siempre con una sonrisa.

Por último, pero no menos importante a mis compañeros de camareta, Carlos y Miguel, sin los cuales, las horas de TFG habrían sido horas de TFG.

CONTENIDO

Contenido	1
Índice de Figuras	3
1 Introducción.....	5
1.1 Motivación	5
1.2 Objetivos	9
1.3 Metodología	9
2 Estado del arte	11
2.1 Tecnología en la docencia.....	11
2.1.1 Píldoras Educativas.....	12
2.1.2 Aulas Virtuales	13
2.2 Herramientas informáticas	13
2.2.1 R.....	14
2.2.2 Octave	14
2.2.3 MATLAB	15
3 Aplicación Desarrollada	17
3.1 Metodología de desarrollo de la aplicación	17
3.2 Desarrollo de la aplicación.....	19
3.2.1 Ventana principal.....	19
3.2.2 Ventana dos	20
3.2.3 Problemas	21
3.2.4 Cuestiones.....	22
3.2.5 Tema 1	23
3.2.6 Tema 2. Modulaciones.....	25
3.2.7 Tema 3	31
3.2.8 Tema 4	33
4 Presentación de Resultados	41
4.1 Ejemplo de uso.....	41
4.1.1 Tema 1	42
4.1.2 Tema 2	43
4.1.3 Tema 3	44
4.1.4 Tema 4	45
5 Conclusiones y líneas futuras	49

5.1 Conclusiones	49
5.2 Líneas Futuras	49
6 Bibliografía.....	51
Anexo I: Código	53
Anexo II: Curvas de propagación de Ondas de Superficie.....	135

ÍNDICE DE FIGURAS

Figura 1-1-1 Simulación del experimento de Oersted en 1819. Tomada de [2]	5
Figura 1-1-2 Simulación del Experimento de Faraday en 1831. Tomada de [4]	6
Figura 1-1-3 Ecuaciones de Maxwell en 1873. Tomada de [6]	6
Figura 1-1-4 Imagen del TELSTAR I antes de su lanzamiento. Tomada de [9].....	7
Figura 1-1-5 Coche con teléfono incorporado, 1924. Tomada de [10]	7
Figura 1-1-6 Martin Cooper, utilizando un Motorola DynaTAC . Tomada de [11]	8
Figura 2-1 Maquina de Sidney Pressey. Tomada de [12]	11
Figura 2-2 Presentación de Power Point, programa utilizado en la docencia. Tomada de [13].....	12
Figura 2-3 Ejemplo de píldora educativa. Tomada de [14].....	12
Figura 2-4 Faitic (Aula virtual de la Universidad de vigo). Tomada de [16].....	13
Figura 2-5 Interfaz grafico RKWard. Tomada de [18].....	14
Figura 2-6 Octave. Tomada de [20]	15
Figura 3-1 Diagrama de flujo de la aplicación	18
Figura 3-2 VPrincipal, vista de desarrollo.....	19
Figura 3-3 VPrincipal, vista de ejecución	20
Figura 3-4 V2, vista de desarrollo	20
Figura 3-5 V2, vista de ejecución.....	21
Figura 3-6 T1P, vista de desarrollo	21
Figura 3-7 T2P, vista en ejecución	22
Figura 3-8 T1C, viste de desarrollo	22
Figura 3-9 T1C, vista en ejecución.....	23
Figura 3-10 T1P1, en desarrollo.....	24
Figura 3-11 T1P1, vista de ejecución	24
Figura 3-12 mod_ana, vista en desarrollo	26
Figura 3-13 mod_ana, vista de ejecución	27
Figura 3-14 ASK, vista de desarrollo	28
Figura 3-15 ASK, vista de ejecución.....	28
Figura 3-16 FSK, vista de desarrollo.....	29
Figura 3-17 FSK, vista de ejecución	30
Figura 3-18 PSK, vista de desarrollo.....	30
Figura 3-19 PSK, vista de ejecución	31
Figura 3-20 Representación de diagrama de radiación en 3D.....	31

Figura 3-21 Representación de diagrama de radiación en coordenadas polares	32
Figura 3-22 antenarad, vista de desarrollo	32
Figura 3-23 antenarad, vista de ejecución	33
Figura 3-24 T4P1, vista de desarrollo	33
Figura 3-25 T4P1, vista de ejecución	34
Figura 3-26 Distintos tipos de terreno que atraviesa una onda de superficie.....	35
Figura 3-27 ond_sup, vista de desarrollo	35
Figura 3-28 ond_sup, vista de ejecución	36
Figura 3-29 Elipsoide de Fresnel.....	37
Figura 3-30 Alturas a considerar para la condición de espacio libre.	38
Figura 3-31 Fresnel, vista de desarrollo	38
Figura 3-32 Fresnel, vista de ejecución.....	39
Figura 4-1 Ejemplo de uso 1	41
Figura 4-2 Ejemplo de uso 2	42
Figura 4-3 Ejemplo de uso 2, tema 1.....	42
Figura 4-4 Ejemplo de uso 3, Tema 1	43
Figura 4-5 Ejemplo de uso 1, Tema 2	43
Figura 4-6 Ejemplo de uso 2, Tema 2	44
Figura 4-7 Ejemplo de uso 1, Tema 3	44
Figura 4-8 Ejemplo de uso 1, Tema 4	45
Figura 4-9 Ejemplo de uso 2, Tema 4	46
Figura 4-10 Ejemplo de uso 3, Tema 4	47

1 INTRODUCCIÓN

1.1 Motivación

La telecomunicación es un sistema de comunicación a distancia que emplea medios eléctricos o electromagnéticos.

La radiocomunicación es un tipo de telecomunicación que se lleva a cabo mediante ondas de radio, las cuales se caracterizan por el movimiento del campo eléctrico y magnético.

En el año 1819, el profesor de física de la Universidad de Copenhague, Hans Christian Ørsted, descubre el electromagnetismo de manera accidental, cuando en uno de sus experimentos para sus estudiantes la aguja de una brújula magnética que se encontraba cerca de un cable se movió. Este descubrimiento fue crucial, ya que puso en evidencia la relación que existe entre el magnetismo y la electricidad. [1]

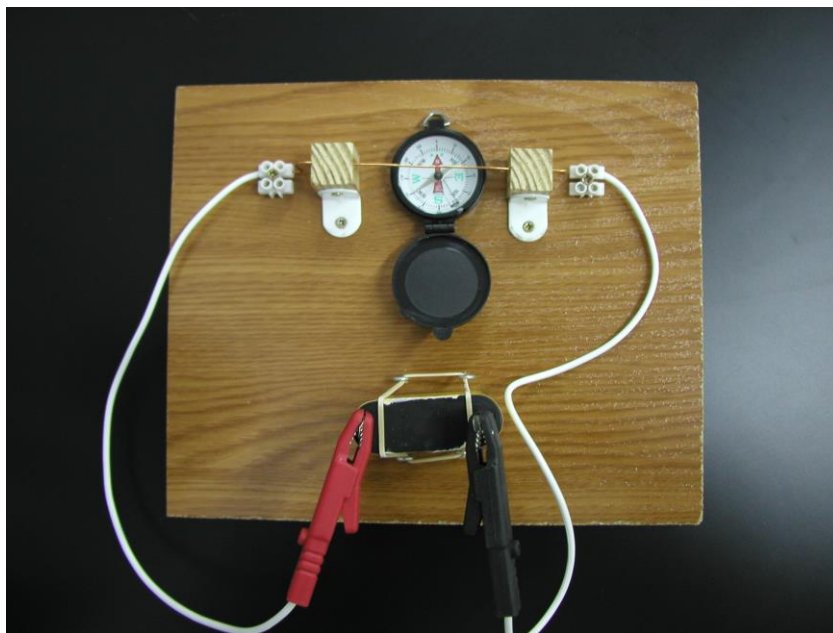


Figura 1-1-1 Simulación del experimento de Oersted en 1819. Tomada de [2]

Unos años más tarde, en 1831, Michael Faraday demuestra el efecto inverso al colocar un imán de barra dentro de un solenoide de cable en el que no había conectada ninguna batería. Cuando metió el imán, se produjo una descarga de corriente eléctrica en una dirección y cuando lo retiró, se produjo una descarga de electricidad en la dirección opuesta. [3]

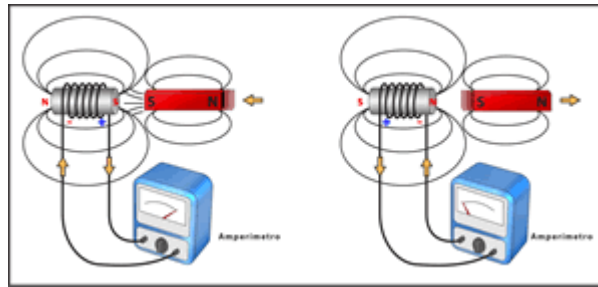


Figura 1-1-2 Simulación del Experimento de Faraday en 1831. Tomada de [4]

En 1873, el matemático escocés James Clerk Maxwell, justificó matemáticamente los conceptos descritos hasta el momento, como las leyes de la inducción electromagnética y de los campos de fuerza en su obra "*Treatise on Electricity and Magnetism*". Introdujo el concepto de onda electromagnética, que permite una descripción matemática adecuada de la interacción entre electricidad y magnetismo mediante sus ecuaciones. [5]

	Forma Integral	Forma Diferencial
Ley de Gauss	$\oiint_s \vec{D} \cdot d\vec{S} = q$	$\nabla \cdot \vec{D} = \rho$
Ley de Faraday	$\oint_c \vec{E} \cdot d\vec{l} = -\frac{\partial}{\partial t} \iint_s \vec{B} \cdot d\vec{S}$	$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t}$
Flujo del campo Magnético	$\oiint_s \vec{B} \cdot d\vec{S} = 0$	$\nabla \cdot \vec{B} = 0$
Ley de Ampère	$\oint_c \vec{H} \cdot d\vec{l} = I + \frac{\partial}{\partial t} \iint_s \vec{D} \cdot d\vec{S}$	$\nabla \times \vec{H} = \vec{J} + \frac{\partial \vec{D}}{\partial t}$

Figura 1-1-3 Ecuaciones de Maxwell en 1873. Tomada de [6]

En 1887, Heinrich Rudolf Hertz, descubrió la propagación de las ondas electromagnéticas en el espacio y estudió la naturaleza y propiedades de las mismas, sentando las bases que llevarían a Marconi a una invención destinada a revolucionar las comunicaciones: la radio. [7]

En 1899, Marconi consiguió establecer comunicación a través del Canal de la Mancha entre Dover (Reino Unido) y Wimereux (Francia), a través de una comunicación sin hilos.

En 1901 se produce la primera comunicación transatlántica sin hilos entre Newfoundland (Canadá) y Polou (Reino Unido).

En 1922 nace en el Reino Unido la British Broadcasting Company (comúnmente conocida como BBC), una compañía encargada de transmitir un servicio de radio de forma experimental.

En 1945, Arthur C. Clark, publicó en la revista británica *Wireless World* un artículo técnico titulado "*Extra-terrestrial Relays — Can Rocket Stations Give World-wide Radio Coverage?*", en el cual estudiaba la posibilidad de utilizar satélites artificiales como repetidores para comunicaciones terrenales. [8]

En 1957 se consiguió lanzar el satélite Sputnik.

En 1962, se lanzó el TELSTAR I, el primer satélite de comunicaciones comerciales de la historia, que permitió crear un canal de televisión entre EEUU y Europa.

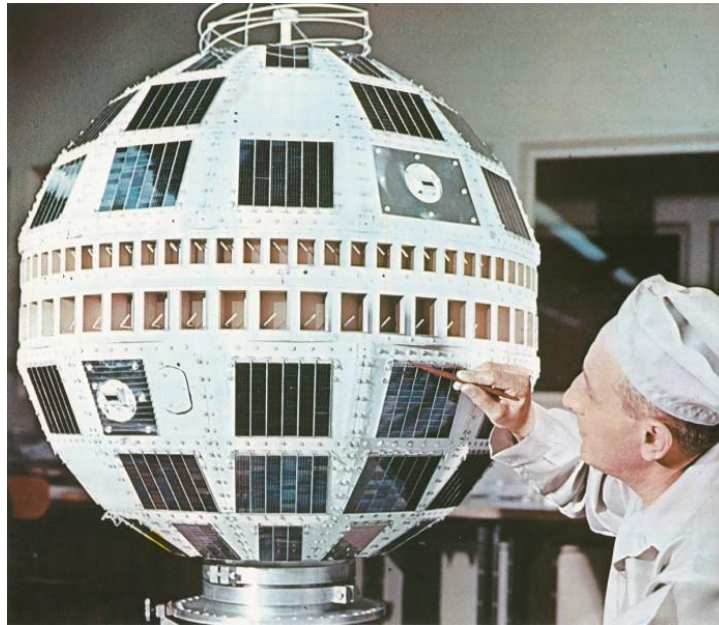


Figura 1-1-4 Imagen del TELSTAR I antes de su lanzamiento. Tomada de [9]

Entre 1993 y 1995 se pone en marcha el sistema de geolocalización por GPS.

A partir de 1946 comienzan los servicios de telefonía móvil. Si bien los primeros equipos eran grandes y pesados, por lo que estaban limitados casi exclusivamente a ser transportados en vehículos.



Figura 1-1-5 Coche con teléfono incorporado, 1924. Tomada de [10]

No fue hasta 1984 cuando se creó el primer teléfono móvil realmente portátil. El DynaTAC 8000X es presentado por Motorola y pesaba cerca de 1 kg.



Figura 1-1-6 Martin Cooper, utilizando un Motorola DynaTAC . Tomada de [11]

A partir de la década de los 90, la tecnología se ha ido desarrollando y han aparecido sistemas desde GSM, IS-136 o iDEN, pasando por EMS, MMS, GPRS y EDGE, hasta UMTS y LTE.

Debido a la utilidad de los Sistemas de Radiocomunicación en el día a día de los ciudadanos, y a que esta tecnología se encuentra en constante desarrollo y toma cada vez más importancia en nuestras vidas, se ve de gran utilidad estudiar la asignatura de “*Sistemas de Radiocomunicaciones*”.

En esta asignatura no se entra al detalle de las radiocomunicaciones, pero si se estudian algunos aspectos básicos muy útiles que todo ingeniero debe conocer.

Aunque la asignatura de Sistemas de Radiocomunicaciones no se estudia en el Grado de Ingeniería Mecánica en el Campus de Vigo, en la Escuela Naval Militar se optó por incluirla en el expediente académico dentro de la intensificación en Tecnología Naval. Y es que las radiocomunicaciones son el día a día de los miembros de las Fuerzas Armadas. Desde el infante de marina que desembarca en una playa hasta el marino que navega por los océanos, se hace necesario el uso de estos sistemas para poder mantener comunicaciones, puesto que comunicaciones por cable es algo que resulta inviable en la mayor parte de los casos.

Para una mejor comprensión y estudio de la asignatura, se ve necesario el uso de herramientas informáticas que sirvan de ayuda. Por ello, en este trabajo de fin de grado (más adelante lo llamaremos TFG), se pretende desarrollar una interfaz gráfica o GUI (Guided User Interface) para el estudio de la asignatura.

1.2 Objetivos

Este TFG consiste en una interfaz gráfica con la que el usuario/alumno pueda asimilar los conceptos explicados en clase mediante la realización de ejercicios prácticos y/o preguntas teóricas y teórico-prácticas. Se pretende:

- Desarrollar una interfaz gráfica simple e intuitiva para el usuario.
- Motivar al alumno en el estudio de la asignatura.
- Mejorar, mediante la búsqueda de información, la madurez, independencia y crítica del autor para la obtención de información de diversas fuentes.

1.3 Metodología

Para conseguir estos objetivos se han planificado los siguientes hitos:

1. Un estudio de la evolución las tecnologías en la ayuda a la docencia a lo largo de la historia (recogido en el [apartado 2.1](#) de la presente memoria).
2. Revisión de algunos de los métodos más utilizados hoy en día como son las píldoras educativas y las aulas virtuales (recogidos en los [apartado 2.1.1](#) y [apartado 2.1.2](#) de la presente memoria respectivamente).
3. Estudio de posibles programas para realizar la herramienta (recogido en el [apartado 2.2](#) de la presente memoria).
4. Selección de programa y desarrollo de la herramienta con el programa seleccionado (recogido en el [capítulo 3](#) de la presente memoria).
5. Validación del programa desarrollado (recogido en el [capítulo 4](#) de la presente memoria).
6. Conclusiones y líneas futuras (recogidas en el [capítulo 5](#) de la presente memoria).

2 ESTADO DEL ARTE

2.1 Tecnología en la docencia

La tecnología toma cada vez un papel más importante en las aulas, ya que supone una ayuda a la enseñanza y una reducción de los costes. Gracias a la utilización de las tecnologías se consigue una mayor implicación por parte del alumno.

El uso de la tecnología como ayuda a la docencia se pudo ver por primera vez en los Estados Unidos de América, a principios del siglo XX, con las primeras películas educativas en las aulas y pocos años después con la máquina de enseñanzas de Sidney Pressey, fue la primera máquina de respuestas múltiples que se utilizó en un primer momento para medir el coeficiente intelectual de los militares estadounidenses.



Figura 2-1 Maquina de Sidney Pressey. Tomada de [12]

Desde ese momento, los medios tecnológicos utilizados en las aulas han ido mejorando a la par que el desarrollo tecnológico de la sociedad en general.

Se han utilizado desde punteros laser, pasando por proyectores y otros medios audiovisuales, hasta el uso de los ordenadores y tabletas. Con ellos podemos obtener gran variedad de programas que nos son de gran ayuda como puede ser el Microsoft Office, Adobe, Photoshop e infinidad de programas que nos facilitan el aprendizaje y estudio de diversos campos de estudio.

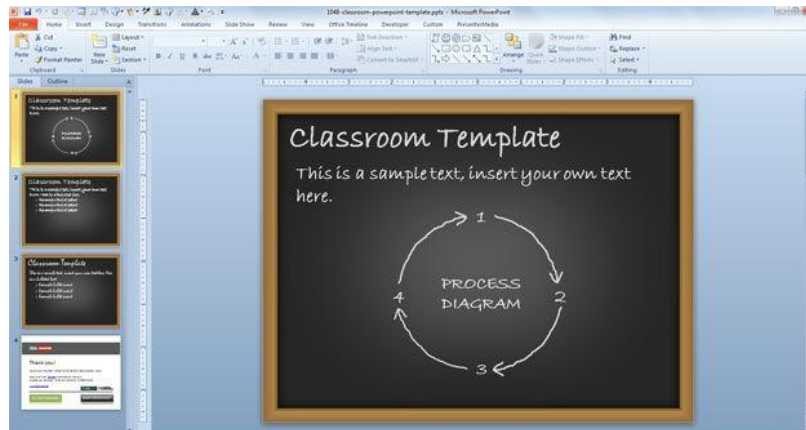


Figura 2-2 Presentación de Power Point, programa utilizado en la docencia. Tomada de [13]

Hoy en día difícilmente se concibe la enseñanza sin estas aplicaciones o programas, puesto que cada vez se desarrollan más y más sofisticadas. En los siguientes apartados se comentarán algunas de las más importantes.

2.1.1 Píldoras Educativas

Las píldoras educativas son videos, generalmente de corta duración, en las que el autor explica una serie de conceptos acerca de un determinado tema. En la mayoría de los casos, los autores (profesores) se apoyan en transparencias para hacer estas píldoras.

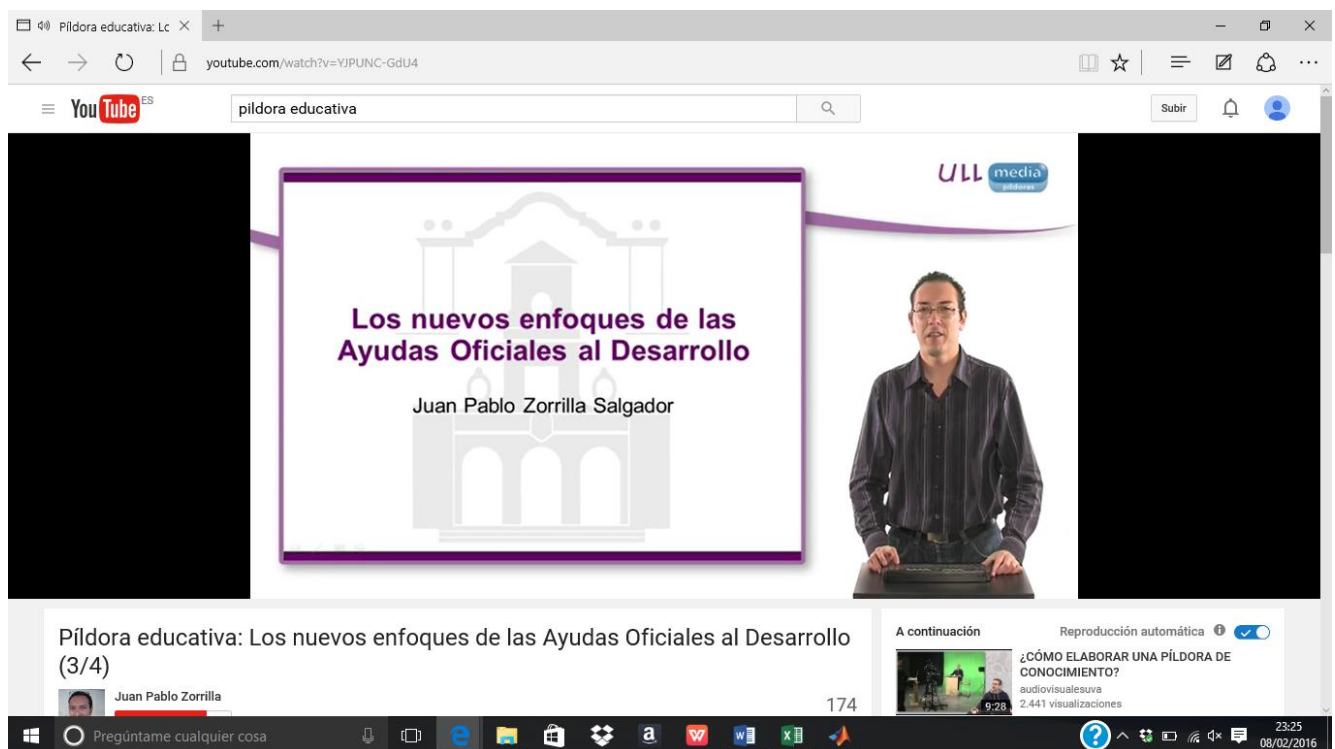


Figura 2-3 Ejemplo de píldora educativa. Tomada de [14]

2.1.2 Aulas Virtuales

A medida que Internet se ha ido desarrollando, han aparecido las llamadas Aulas Virtuales, que son mecanismos mediante los cuales profesores y alumnos pueden interactuar sin necesidad de una clase presencial.

A través de las aulas virtuales los alumnos pueden acceder y desarrollar una serie de acciones que son propias de un proceso de enseñanza presencial como conversar, leer documentos, realizar ejercicios, formular preguntas al docente, trabajar en equipo, etc. Todo ello de forma simulada sin que medie una

interacción física entre docentes y alumnos. [15]



Figura 2-4 Fatic (Aula virtual de la Universidad de vigo). Tomada de [16]

Aunque las aulas virtuales y las píldoras educativas son de gran ayuda, para el propósito de este TFG se necesita una herramienta informática que permita desarrollar una interfaz gráfica de usuario a medida.

2.2 Herramientas informáticas

Existen diversos programas con lo que desarrollar interfaces gráficas, entre ellos se encuentra R, Octave y MATLAB.

2.2.1 R

R es un lenguaje y entorno de programación para realizar análisis estadísticos y gráficos. Es un software de libre distribución el cuál está disponible para los sistemas operativos Windows, Macintosh, GNU/Linux y Unix. [7]

En R nos encontramos múltiples herramientas gráficas y estadísticas (Figura 2-5). Además, puede integrarse con distintas bases de datos y se han creado bibliotecas para facilitar su utilización desde otros lenguajes de programación. [7]

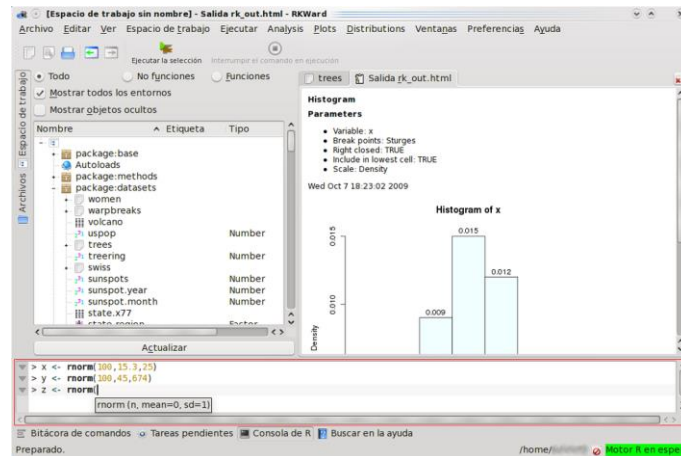


Figura 2-5 Interfaz gráfico RKWard. Tomada de [18]

R forma parte de un proyecto abierto en el que todos los usuarios colaboran subiendo paquetes de software los cuales en 2009 superaron la cifra de los dos mil. Estos paquetes son extensiones que el R necesita para poder realizar ciertas funciones. Para almacenarlos, existe un repositorio oficial de paquetes, que se divide por temas o vistas.

La interfaz gráfica para el lenguaje R se llama RKWard y también utiliza bibliotecas libres.

2.2.2 Octave

Octave o GNU Octave es un programa de libre distribución para realizar cálculos numéricos. Está escrito en C++ y su lenguaje puede ser extendido con funciones y procedimientos, por medio de módulos dinámicos.

Utiliza otros programas GNU para ofrecer al usuario la posibilidad de crear gráficos para luego imprimirlos o guardarlos. Además de estar disponible en plataformas Unix, también lo está para Windows. [8]

¹ El sistema GNU nació del ‘Proyecto GNU’ iniciado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre.

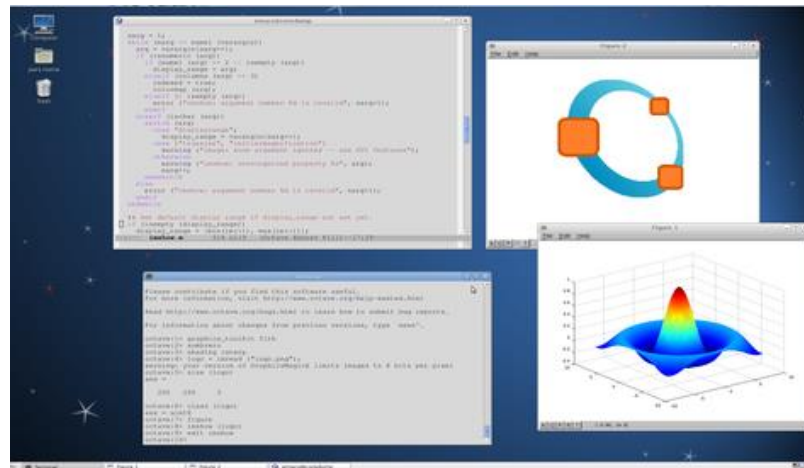


Figura 2-6 Octave. Tomada de [20]

2.2.3 MATLAB

MATLAB (MATrix LABoratory) es una herramienta de software matemático que utiliza un lenguaje de alto nivel. Es un software muy utilizado en universidades y centros de investigación y desarrollo.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las *cajas de herramientas* (toolboxes); y las de Simulink con los *paquetes de bloques* (blocksets). [21]

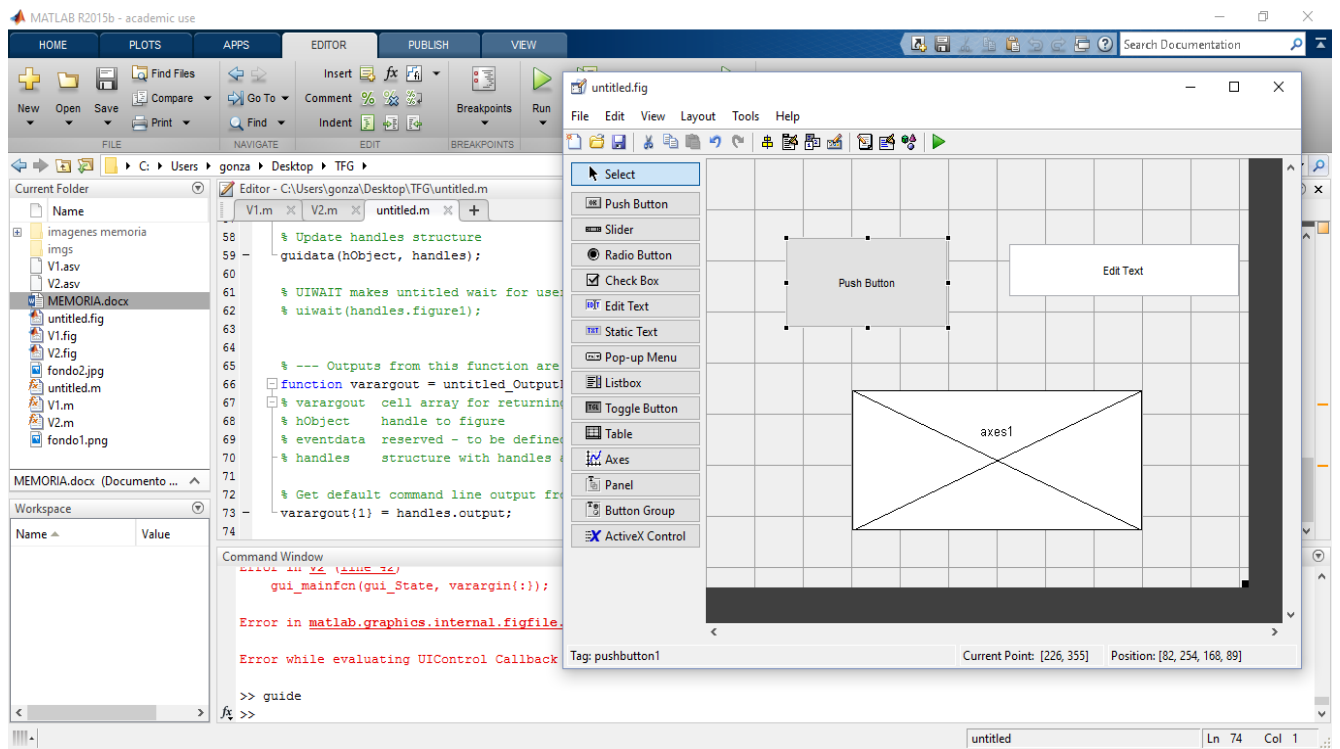


Figura 2 -4 Interfaz gráfico de MATLAB. GUI.

3 APLICACIÓN DESARROLLADA

3.1 Metodología de desarrollo de la aplicación

A la hora de desarrollar una aplicación, se debe tener en cuenta cuatro pasos fundamentales: identificar la idea, dibujar la aplicación, desarrollar la aplicación y mejorar la aplicación.

1. **Identificar la idea:** Es necesario saber con claridad el fin último que se quiere conseguir. En el caso de esta aplicación, el fin último es el de ayudar al alumno a estudiar la asignatura de Sistemas de Radiocomunicaciones. Pero para que el alumno quiera utilizar esta aplicación, debe de tener algún atractivo, algo que le llame la atención. Este atractivo es la simplicidad de la herramienta, la cual hará que el alumno piense que es tan fácil de usar que merece la pena utilizarla para el estudio. Una vez resuelto el problema principal, y encontrado el atractivo para la solución, se puede continuar con la creación de la aplicación.

La idea de este TFG surge de la necesidad de ayudar en la docencia de Sistemas de Radiocomunicaciones.

Título	Desarrollo de herramienta informática de apoyo a la docencia de Sistemas de Radiocomunicaciones	Director/es	Sandra Castro Cao	Ámbito	TEL
Prerrequisitos o conocimientos previos deseables:	Sistemas de radiocomunicación; Nociones de programación en Matlab	Asignatura/s de la titulación con la/s que el TFG tiene relación directa:	Sistemas de radiocomunicaciones	Código	TFG47
Resumen	En el ámbito de la docencia, las clases prácticas son una herramienta de apoyo fundamental para lograr fijar los contenidos teóricos de las materias mediante la aplicación práctica de los mismos. El objetivo de este trabajo de fin de grado es crear una herramienta que permita al alumno, mediante la simulación de escenarios de enlaces y mecanismos de radiopropagación, así como con la resolución de cuestionarios, reforzar los conceptos ligados a la asignatura de Sistemas de Radiocomunicaciones. La herramienta se implementará como un módulo del programa Matlab utilizado en el laboratorio de la asignatura				

Figura 3-1 Oferta de TFG, en el que se propone la idea a desarrollar.

2. **Diseñar la aplicación:** Es en este momento cuando la aplicación comienza a tomar forma. Es conveniente hacer varios bocetos en papel, sobre las distintas funciones de la aplicación. Lo importante es plasmar las ideas que se tienen en la cabeza sobre papel, o formato informático, como Photoshop.

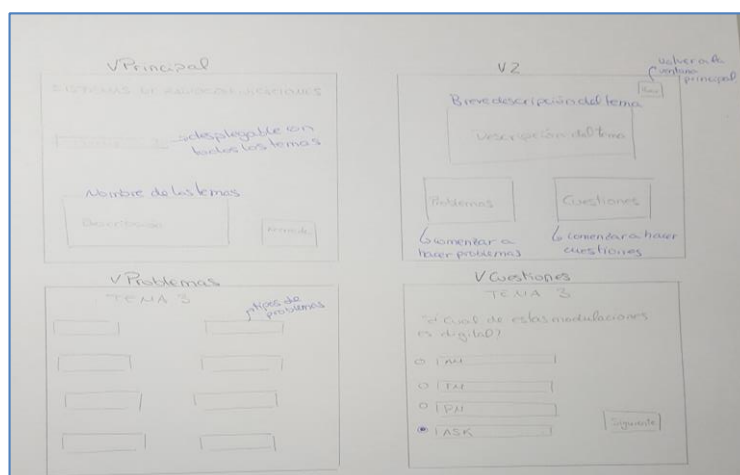


Figura 3 – 2 Boceto de la aplicación

3. **Desarrollar la aplicación:** Si se tienen los dos puntos anteriores bien claros y definidos, este apartado no debe de ser excesivamente complicado. El desarrollo de la app debe de ser de manera bien estructurada, lo cual no quiere decir que a medida que se avance en ciertos aspectos no se puedan modificar anteriores, pero es preferible hacerlo por partes. Es necesario para además elegir el software con el que crear la aplicación. En el caso de este TFG, aunque a priori ya se había optado por MATLAB también se ha considerado otras posibles opciones.

Tanto R como Octave son programas muy completos, ya que se puede encontrar todo tipo de ayuda. No se empleará en este TFG, debido a que el tiempo para la realización es escaso y utilizarlo supondría aprender a programar sin tener ninguna base de conocimientos. Por ello se utilizará MATLAB, un programa con el que se ha trabajado en varias ocasiones, al cual se tiene acceso en el ordenador personal por ser alumno de la Universidad de Vigo.

4. **Validar la aplicación:** Una vez la aplicación está terminada y lista para empezar a usarse, será una aplicación bastante básica con infinidad de fallos. Estos fallos son descubiertos por los usuarios y deben de ser corregidos para que se pueda seguir disfrutando de ella. Por ello es una buena idea crear un entorno donde los usuarios de la aplicación nos puedan dejar sus comentarios acerca de la misma, donde incluyan fallos y posibles aspectos a mejorar.

Como ya se ha comentado en el punto 3 de “Cuatro pasos fundamentales para desarrollar una aplicación”, el desarrollo debe de ser de forma estructurada, y para ello se empezará con la ventana inicial a la que se la ha llamado *Ventana principal*.

Para facilitar la comprensión de la aplicación, se presenta a continuación un diagrama de flujo.

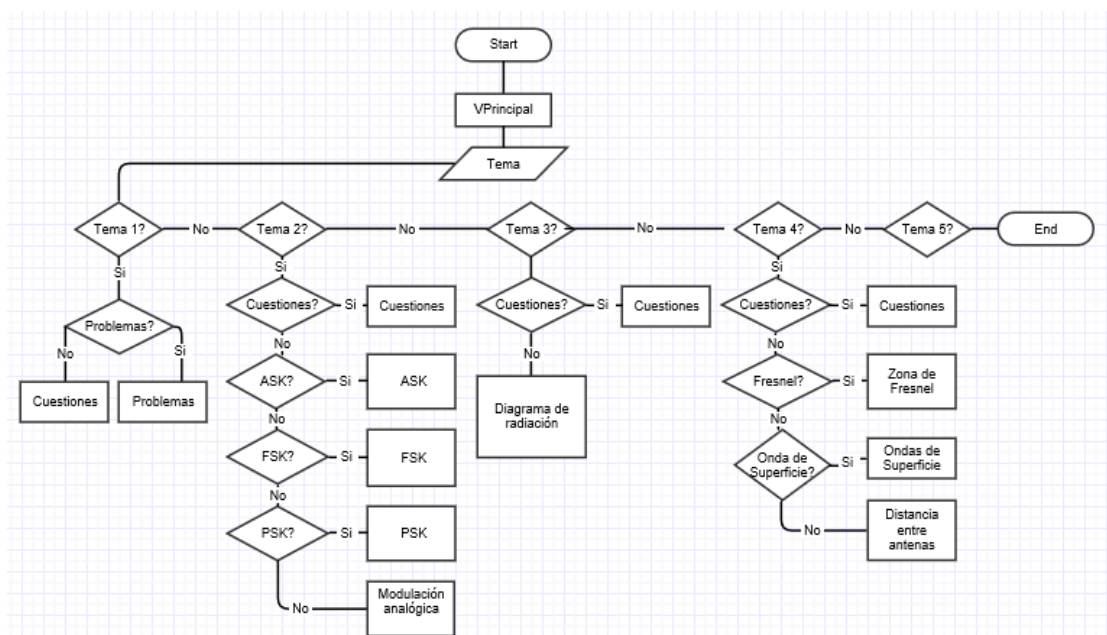


Figura 3-1 Diagrama de flujo de la aplicación

3.2 Desarrollo de la aplicación

El desarrollo de la aplicación se ha realizado mediante varios interfaces gráficos de usuario o ventanas. A continuación, se explicará de que se compone cada ventana, introduciendo en los casos que sean necesarios algunos conceptos teóricos tomados de [22] y de [23] y presentando dos imágenes, una en vista de programación en el guide de MATLAB y la segunda en vista de ejecución. El código de cada ventana queda reflejado en el [Anexo I](#).

3.2.1 Ventana principal

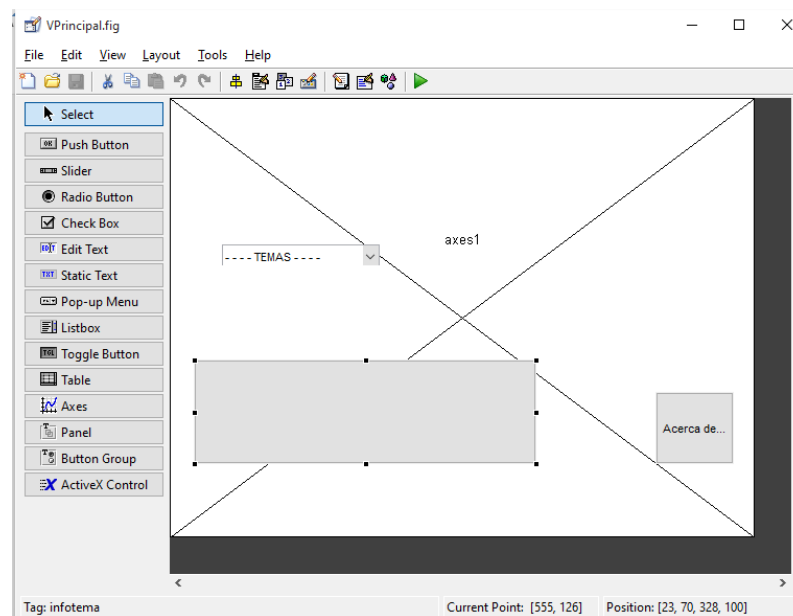


Figura 3-2 VPrincipal, vista de desarrollo

Es la primera ventana que aparece cuando se ejecuta la aplicación. En MATLAB se la ha llamado *VPrincipal* para que se pueda hacer referencia a ella de forma más sencilla.

Esta ventana se compone del título de la asignatura en la parte superior, un pop-up menu, en el cual se despliegan los distintos temas para que el usuario escoja uno de ellos. Los temas siguen la programación desarrollada para la asignatura durante el curso 2015/2016.

Cuando se ha elegido el tema, en un Push-Button, aparecerá el nombre del tema que se ha seleccionado y al pulsarlo abrirá la siguiente ventana.

Por último, la ‘Ventana Principal’ cuenta con un pequeño Push-Button, llamado “Acerca de...”, en el cual se hace referencia al autor del programa.

Una vez creada la Ventana Principal, se le pone una imagen de fondo, para que al ejecutarla aparezca de la siguiente manera.



Figura 3-3 VPrincipal, vista de ejecución

3.2.2 Ventana dos

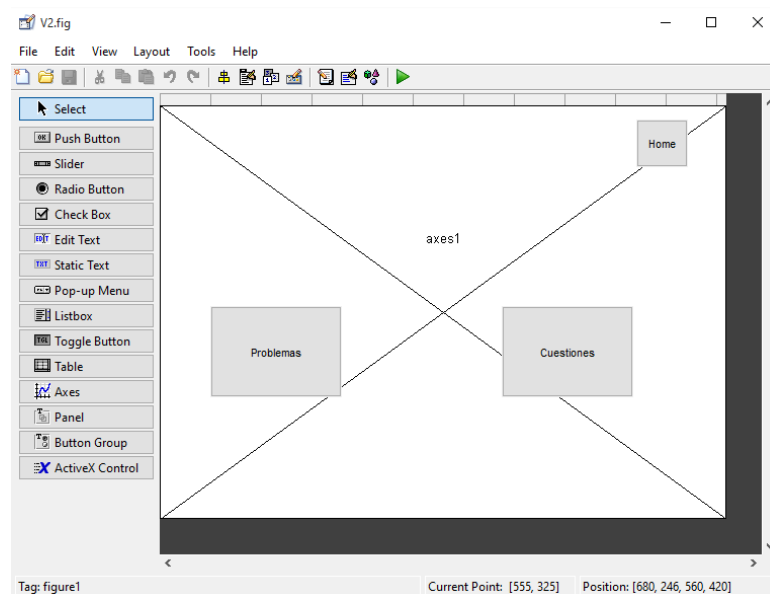


Figura 3-4 V2, vista de desarrollo

La Ventana dos, o como se llamará en MATLAB, V2, está formada por tres Push-Buttons, cada uno de los cuales tiene una función distinta.

El que se encuentra en la esquina superior derecha, redirige a la Ventana Principal, cerrando V2.

El Push-button en el que pone Problemas, cierra V2 y abre una nueva ventana en la cual se puede elegir entre los distintos tipos de problemas dentro del mismo tema.

El Push-button en el que pone Cuestiones, cierra V2, al igual que los anteriores, y abre una ventana en la cual se pueden realizar distintas cuestiones teórico prácticas de ese tema.

Además, en la parte superior de la ventana nos aparece el título del tema en el que estamos.



Figura 3-5 V2, vista de ejecución

3.2.3 Problemas

Una vez se ha seleccionado Problemas en V2, aparece una nueva ventana para realizar problemas del tema escogido previamente. A esta nueva ventana se le llama T (*número del tema*) P, por ejemplo, si se ha escogido el tema 1, T1P (Tema 1 Problemas).

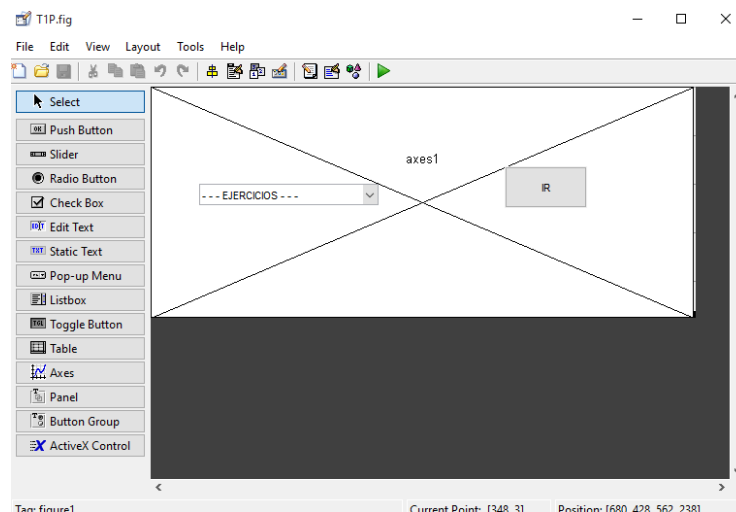


Figura 3-6 T1P, vista de desarrollo

Esta nueva ventana se compone de un pop-up menu, donde aparecen los distintos tipos de ejercicios del tema. Y un Push-Button el cual nos lleva hacia el tipo de ejercicio elegido. En todos los temas, el contenido de esta ventana será muy similar.

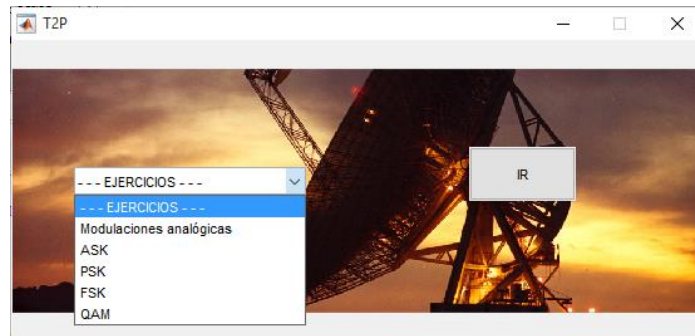


Figura 3-7 T2P, vista en ejecución

3.2.4 Cuestiones

En caso de seleccionar la opción de cuestiones en V2, aparece una nueva ventana en la que se pueden realizar distintas preguntas tipo test.

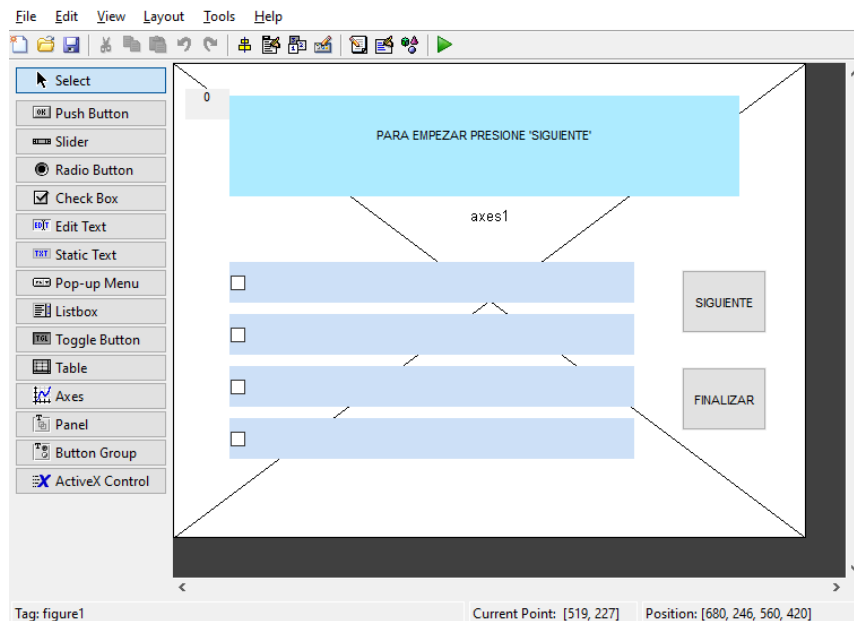


Figura 3-8 T1C, viste de desarrollo

Las ventanas de cuestiones se componen de dos Static Text, uno en la parte superior izquierda que muestra en número de la pregunta y otro en el centro superior que redacta la pregunta.

En el centro hay cuatro Check Boxes, en cada una de ellas irá una posible respuesta.

A la derecha se encuentran dos Push Buttons, uno de ellos para avanzar a la siguiente pregunta y el otro para finalizar el test.

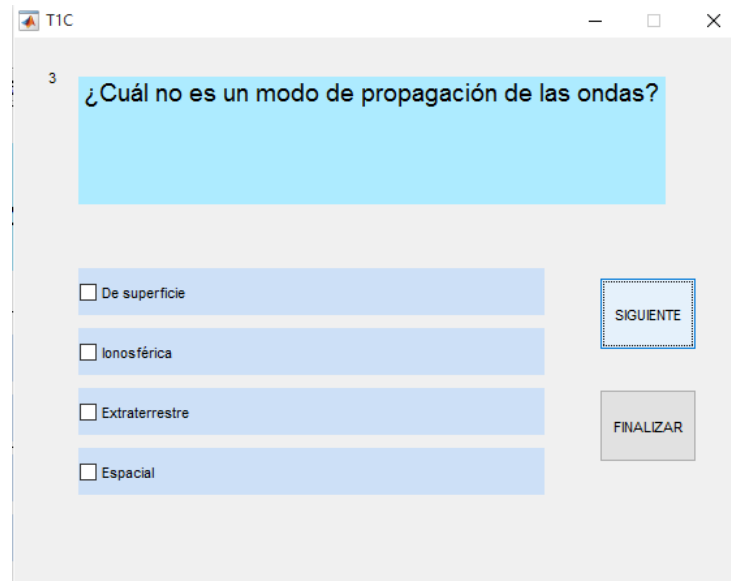


Figura 3-9 T1C, vista en ejecución

Hasta ahora se han visto las primeras ventanas introductorias a los distintos temas. En los siguientes apartados se expondrán los distintos tipos de ejercicios y/o cuestiones de para tema por separado.

3.2.5 Tema 1

El primer tema de la asignatura sirve como introducción a los conceptos básicos. Se propone como problemas la conversión de distintas unidades a unidades logarítmicas. Para un valor g en unidades naturales, y un valor G en unidades logarítmicas:

$$G = 10 * \log_{10}(g)$$

$$g = 10^{\frac{G}{10}}$$

Esto ocurre siempre y cuando sea una relación lineal. Pero cuando se comparan tensiones o intensidades de campo eléctrico debe tenerse en cuenta que la potencia es proporcional a V^2 o equivalentemente a $|E|^2$. Por ello, si comparamos dos tensiones o dos intensidades de campo eléctrico E_1 y E_2 , la relación evaluada en dB es:

$$20 * \log_{10}\left(\frac{|e_1|}{|e_2|}\right)$$

3.2.6 Tema 2. Modulaciones

En este tema se trabaja el concepto de modulación, y se analizan una serie de modulaciones de forma básica.

Dependiendo de la naturaleza de la señal moduladora (analógica o digital) existen:

- Modulaciones analógicas: AM, FM, PM.
- Modulaciones digitales: ASK, PSK, FSK, QAM, OFDM...

A continuación, se mostrarán los distintitos tipos de modulaciones empleados en este TFG.

3.2.6.1 Modulaciones analógicas

En función del parámetro modificado en la portadora existen:

- Modulación en amplitud (AM), donde la amplitud de la portadora varía en función de la moduladora. (AM convencional, DBL, BLU, BLR).
- Modulación angular donde el ángulo de la portadora varía en función de la moduladora. FM varía en frecuencia y PM varía en fase. En una modulación FM, la frecuencia instantánea es proporcional a la moduladora, mientras que en una modulación PM es la fase instantánea la que es proporcional a la moduladora.

➤ AM Convencional

Siendo la señal moduladora $m(t)$ y la señal portadora:

$$c(t) = A_c * \cos(2 * \pi * f_c * t)$$

La señal modulada $s(t)$ será:

$$s(t) = A_c[1 + a * m(t)] * \cos(2 * \pi * f_c * t)$$

Donde 'a' es el índice de modulación.

➤ DBL

Con la AM convencional se desperdicia potencia, ya que sólo una parte de la potencia transmitida lleva información. La solución es suprimir la componente de portadora.

La señal modulada $s(t)$ será:

$$s(t) = c(t) * m(t) = A_c * m(t) * \cos(2 * \pi * f_c * t)$$

➤ FM

Siendo una señal portadora $c(t)$:

$$c(t) = A_c * \cos(2 * \pi * f_c * t)$$

Y una señal moduladora $m(t)$, la señal modulada $s(t)$ será:

$$s(t) = A_c * \cos(\theta_i(t)) = A_c * \cos(2 * \pi * f_c * t + \phi(t))$$

Donde $\theta_i(t)$ es la fase instantánea y varía en función de la señal moduladora $m(t)$:

$$f_i(t) = \frac{1}{2 * \pi} \frac{d\theta_i(t)}{dt} = f_c + \frac{1}{2 * \pi} \frac{d\phi_i(t)}{dt}$$

$f_i(t)$: frecuencia instantánea.

$$f_i(t) = f_c + k_f * m(t) \rightarrow \theta_i(t) = 2\pi \int f_i(t) dt = 2 * \pi * f_c * t + 2 * k_f \int m(t) dt$$

K_f : sensibilidad en frecuencia.

$$s(t) = A_c * \cos[2 * \pi * f_c * t + 2 * \pi * k_f * \int m(t) dt]$$

➤ PM

Siendo una señal portadora $c(t)$:

$$c(t) = A_c * \cos(2 * \pi * f_c * t)$$

Y una señal moduladora $m(t)$, la señal modulada $s(t)$ será:

$$s(t) = A_c * \cos(\theta_i(t)) = A_c * \cos(2 * \pi * f_c * t + \phi(t))$$

Donde $\theta_i(t)$ es la fase instantánea y varía en función de la señal moduladora $m(t)$:

$$\theta_i(t) = 2 * \pi * f_c * t + k_p * m(t)$$

$f_i(t)$: frecuencia instantánea.

$$f_i(t) = \frac{1}{2 * \pi} \frac{\partial \theta_i(t)}{\partial t} = f_c + \frac{1}{2 * \pi} \frac{\partial \phi(t)}{\partial x}$$

K_p : sensibilidad en fase.

$$s(t) = A_c * \cos[2 * \pi * f_c * t + k_p * m(t)]$$

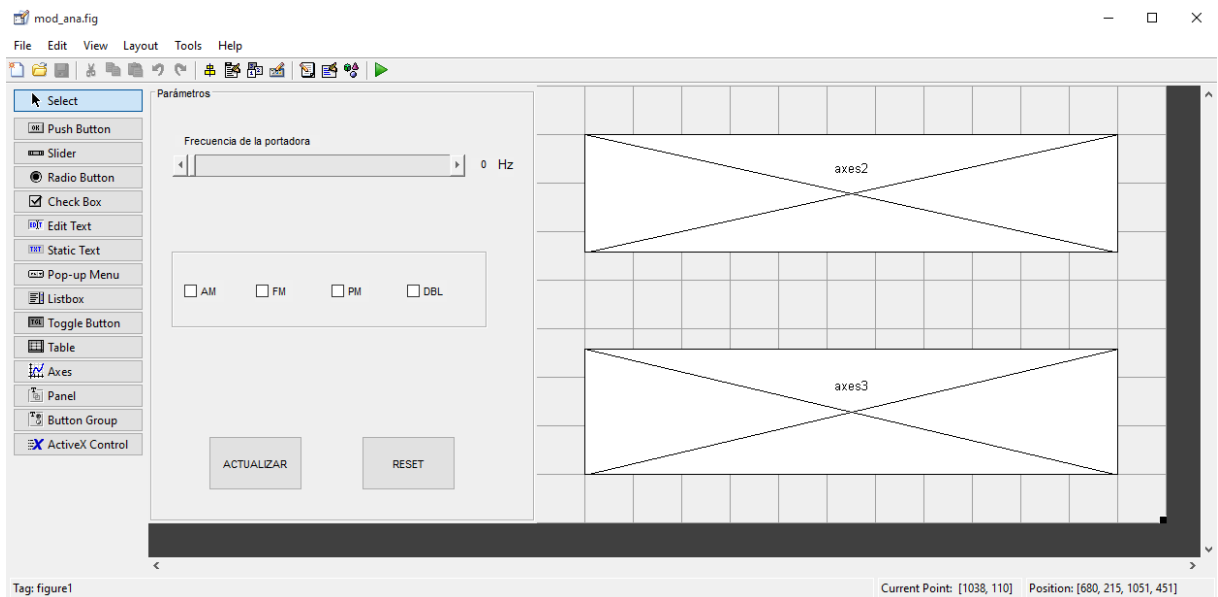


Figura 3-12 mod_ana, vista en desarrollo

Para estudiar las modulaciones analógicas hay un panel y dos axes a la derecha. En la parte superior hay un Slider el cual sirve para seleccionar la frecuencia de la portadora.

En el panel se puede elegir entre modulaciones AM convencional, FM, o PM. Además, existe la posibilidad de escoger una modulación Doble Banda Lateral (DBL).

A la derecha se encuentran tres Axes para la representación de los distintos tipos de modulaciones escogidos.

Abajo a la izquierda hay dos Push Buttons, uno para actualizar los Axes (actualizar) y otro para borrar todo (reset).

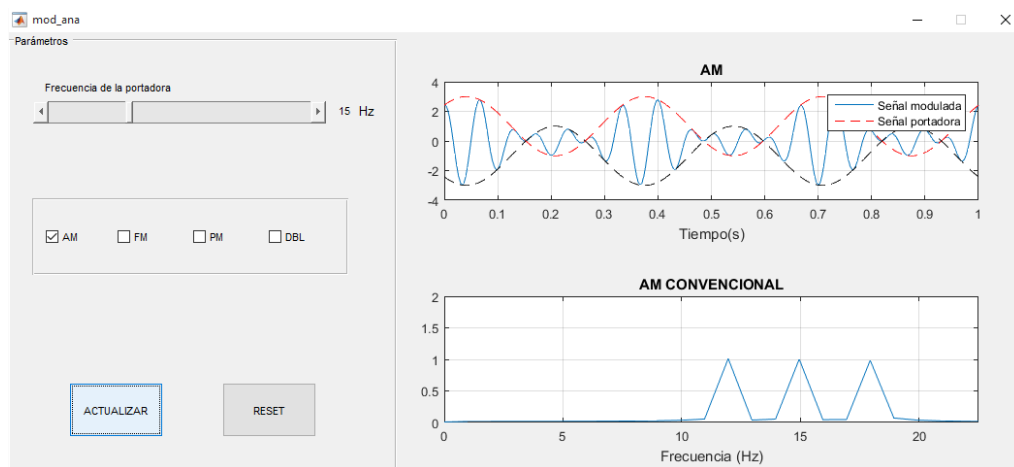


Figura 3-13 mod_ana, vista de ejecución

3.2.6.2 Modulaciones Digitales

Las modulaciones digitales tienen como objetivos convertir los bits en señales adecuadas para su transmisión. Los bits se agrupan formando símbolos y para cada símbolo se escoge una señal $s_m(t)$ diferente de duración T_s (periodo de símbolo).

$$1 \text{ símbolo} = k \text{ bits} \rightarrow M = 2^k \text{ símbolos distintos}$$

3.2.6.2.1 ASK

ASK (Amplitud Shift Keying), o modulación por desplazamiento en amplitud.

$$s_m(t) = A_m * \cos(2 * \pi * f_c * t) \rightarrow m = 0, 1, 2, \dots, M - 1$$

Ejemplo: 4-ASK.

$$M = 4 \rightarrow k = 2$$

Bits	00	01	10	11
A_m	$A_0 = 0$	$A_1 = 2$	$A_2 = 4$	$A_3 = 8$

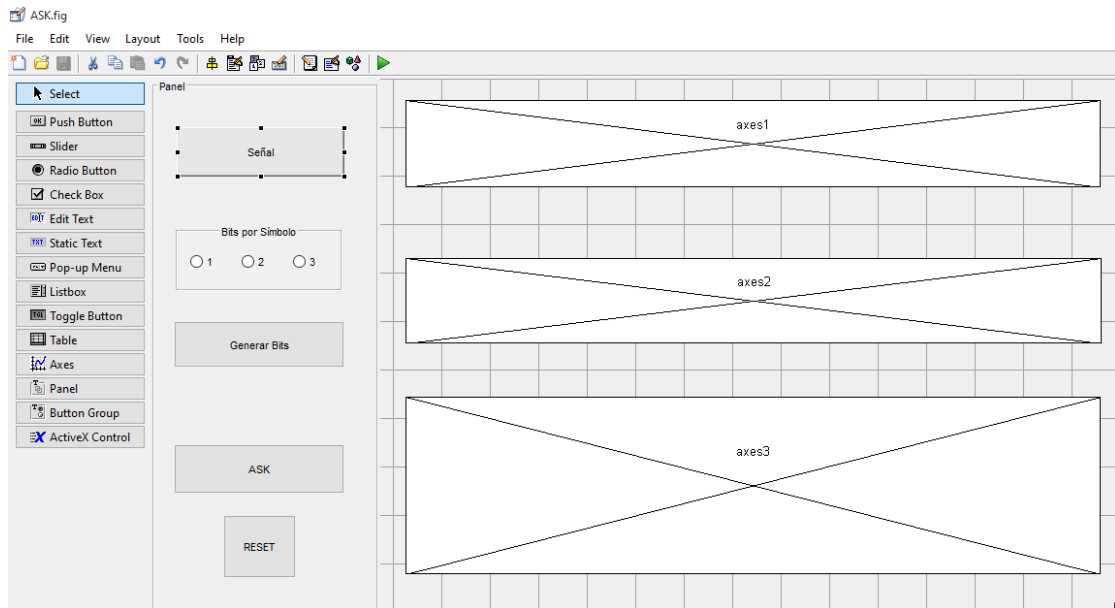


Figura 3-14 ASK, vista de desarrollo

Esta venta consta de cuatro Push Buttons, tres Axes y tres Radio Buttons.

El primer Push Button, el que está más arriba, imprime en el primer Axes la portadora. El segundo Push Button, sirve para generar bits aleatorios y agruparlos en símbolos, dependiendo de cuantos bits por símbolo se escojan, lo cual se hace en los Radio Buttons justo encima. El tercer Push Button, sirve para realizar la modulación. El último Push Button es para borrar todo.

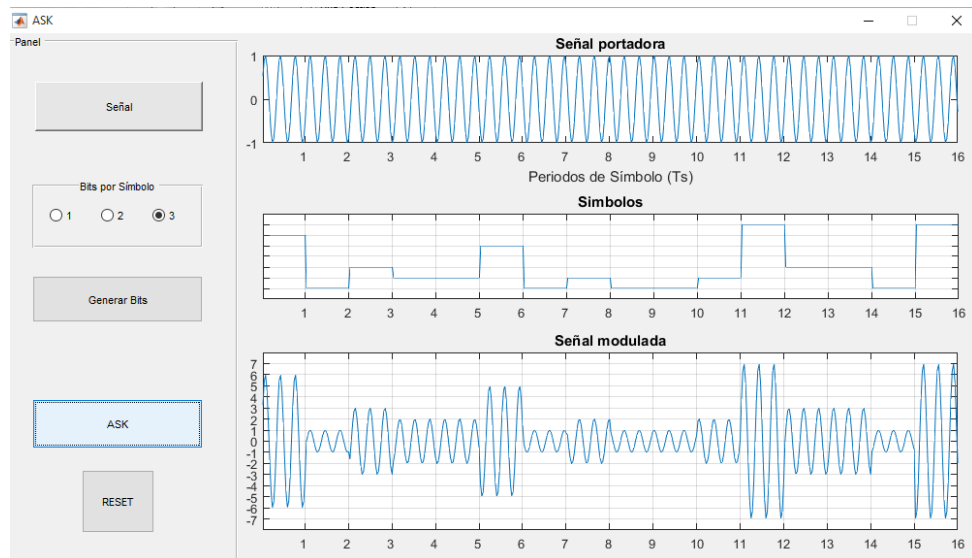


Figura 3-15 ASK, vista de ejecución

3.2.6.2.2 FSK

FSK (Frequency Shift Keying) o modulación por desplazamiento de frecuencia.

$$s_m(t) = A * \cos(2 * \pi * f_m * t)$$

$$f_m = f_c + \frac{(2m - (M - 1))}{2} * \Delta f$$

Ejemplo: 4-FSK

M = 4 → k = 2

Bits	00	01	10	11
f_m	$f_1 = f_c - \frac{3}{2}\Delta f$	$f_2 = f_c - \frac{1}{2}\Delta f$	$f_3 = f_c + \frac{1}{2}\Delta f$	$f_4 = f_c + \frac{3}{2}\Delta f$

Tabla 1 Ejemplo de 4-FSK

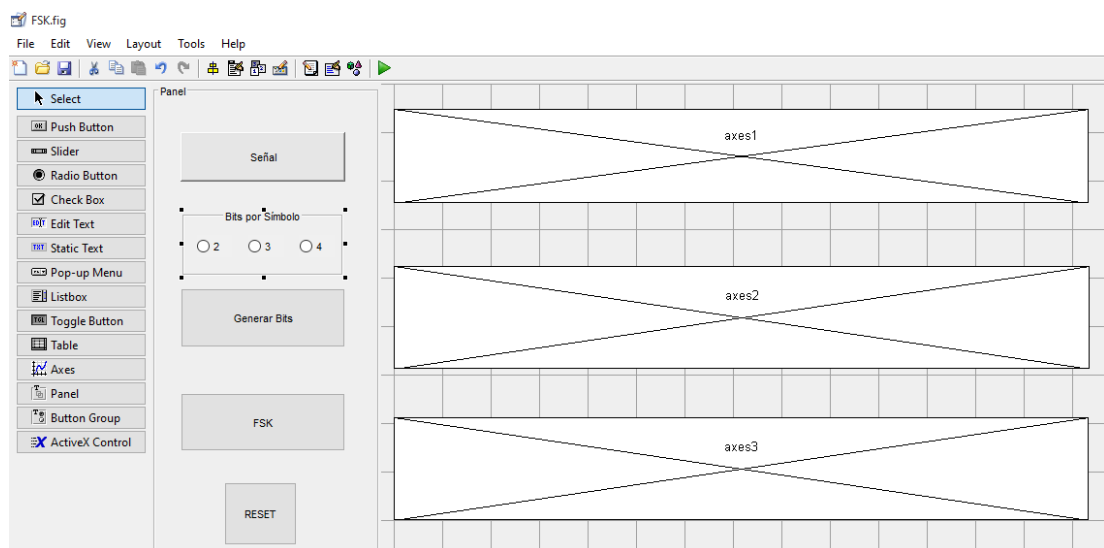


Figura 3-16 FSK, vista de desarrollo

Esta ventana es muy parecida a la ASK. La única diferencia notable a primera vista es que el tercer Push Button se calcula una modulación FSK.

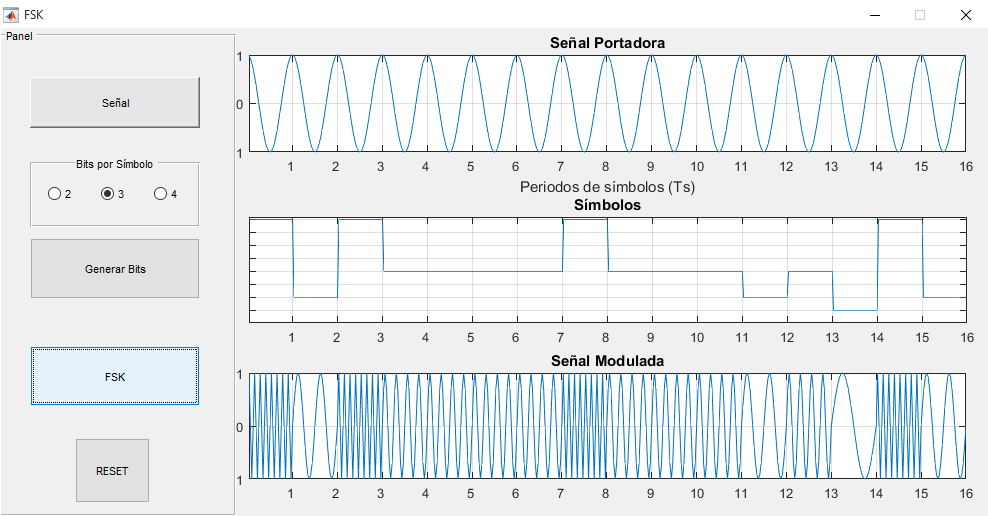


Figura 3-17 FSK, vista de ejecución

3.2.6.2.3 PSK

PSK (Phase Shift Keying) o modulación por desplazamiento de fase.

$$s_m(t) = A_c \cos(2\pi f_c t + \phi_m)$$

Ejemplo: 4-PSK

M = 4 → k = 2

Bits	00	01	10	11
ϕ_m	$\phi_1 = \frac{\pi}{4}$	$\phi_2 = \frac{3\pi}{4}$	$\phi_3 = \frac{5\pi}{4}$	$\phi_4 = \frac{7\pi}{4}$

Tabla 2 Ejemplo de 4-PSK

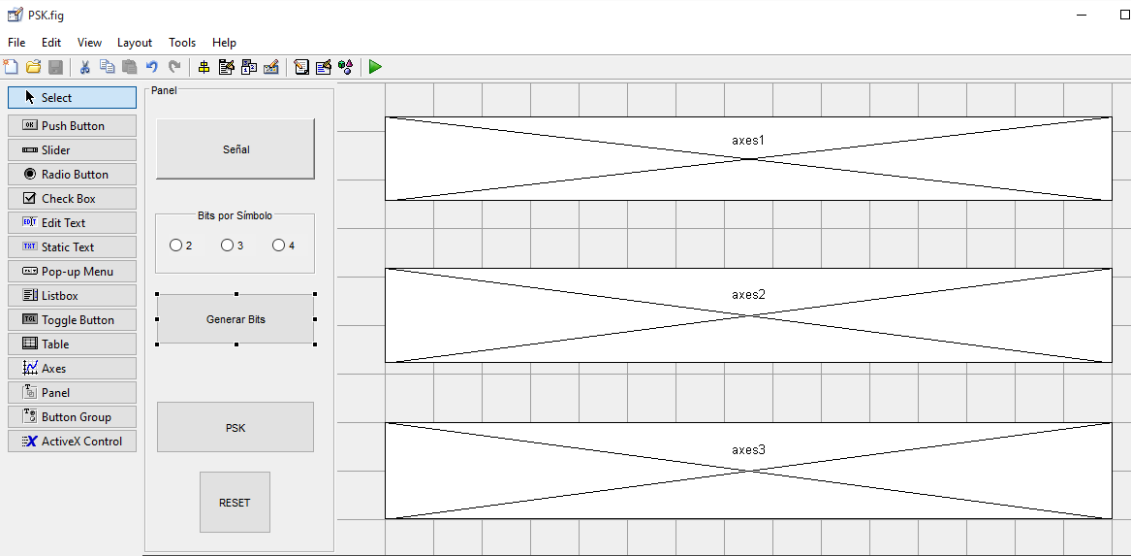


Figura 3-18 PSK, vista de desarrollo

Esta ventana es muy parecida a la *ASK* y *FSK*. La única diferencia notable a primera vista es que el tercer Push Button se calcula una modulación *PSK*.

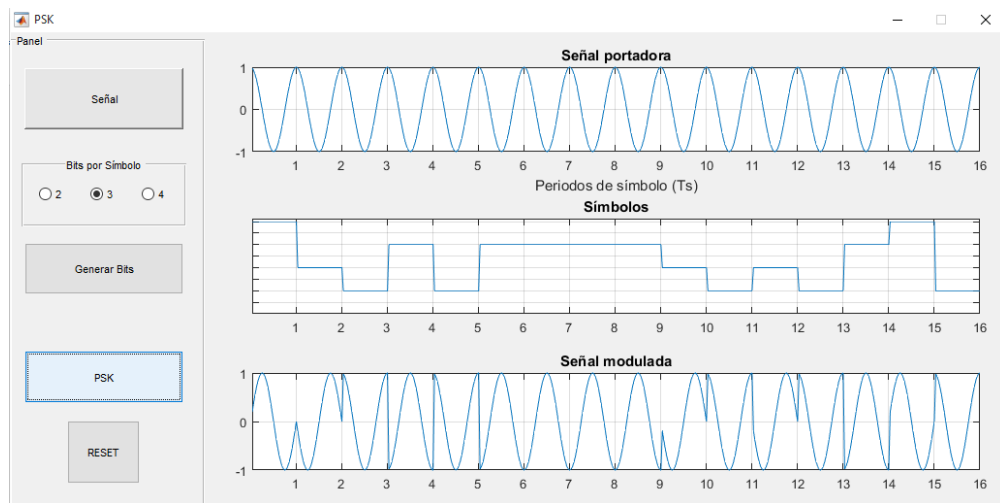


Figura 3-19 PSK, vista de ejecución

3.2.7 Tema 3

Una antena es una caja negra caracterizada por una serie de parámetros, que sirven para calcular los balances en un sistema radio. Entre estos parámetros se encuentran los diagramas de radiación.

Un diagrama de radiación de una antena es la representación gráfica de las características de radiación en función de la dirección angular.

Para representarlo se utiliza normalmente un sistema de coordenadas esférico. Pero debido a la complejidad de representar diagramas 3D (en tres dimensiones), su puede optar por hacer una representación de cortes del diagrama en coordenadas polares o cartesianas.

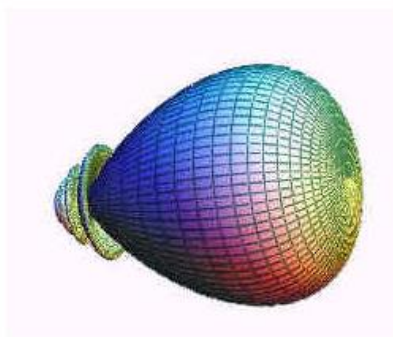


Figura 3-20 Representación de diagrama de radiación en 3D

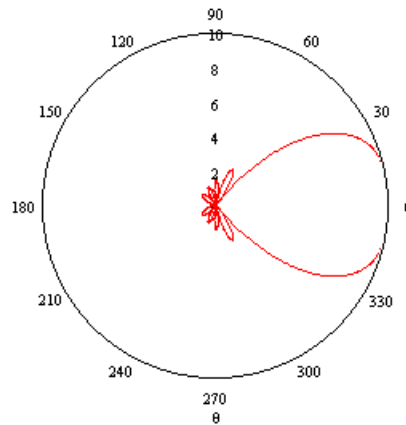


Figura 3-21 Representación de diagrama de radiación en coordenadas polares

Los diagramas de radiación se componen de lóbulos, los cuales se definen como porciones del diagrama delimitado por zonas de radiación más débil. Existen varios tipos de lóbulos:

- Principal: el que contiene la dirección de máxima radiación.
- Secundario: el resto de lóbulos.
- Laterales: los secundarios adyacentes al lóbulo principal.
- Posterior: el que ocupa la dirección opuesta al principal.

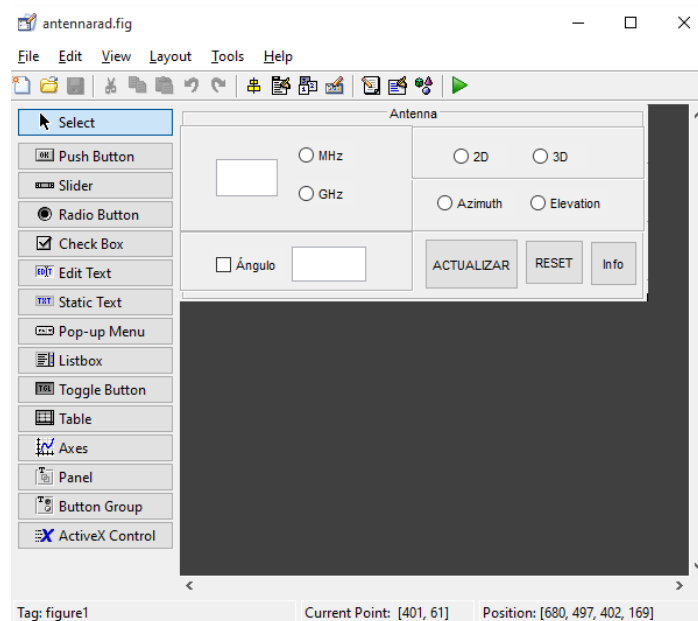


Figura 3-22 antennarad, vista de desarrollo

La ventana ‘*antennarad*’ se compone de cinco paneles. El de la esquina superior izquierda está formado por un Edit Text y dos Radio Buttons y sirve para introducir la frecuencia, ya sea en Megahercios (MHz) o en Gigahercios (GHz).

Arriba a la derecha tenemos dos Radio Buttons para elegir la vista del diagrama de radiación en dos o tres dimensiones.

Inmediatamente debajo, se encuentra un panel con dos Radio Buttons en los que se puede elegir si la vista en dos dimensiones se quiere en azimuth o en elevación.

El panel de la esquina inferior izquierda contiene un Check Box y un Static Text para introducir el ángulo en grados por el cual se quiere hacer el corte en dos dimensiones.

Por último, en la esquina inferior derecha se encuentran tres Push Buttons, para actualizar, resetear la ventana y para dar información acerca del tipo de antena.

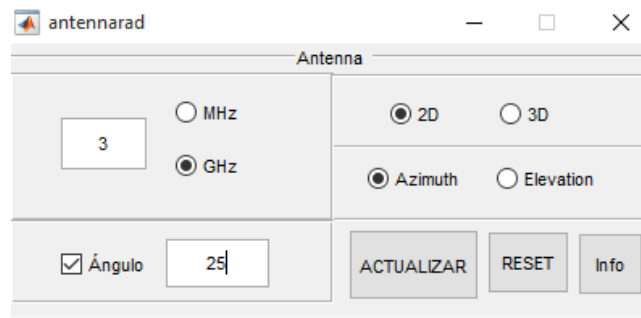


Figura 3-23 antennarad, vista de ejecución

3.2.8 Tema 4

En este tema se exponen los distintos modos de propagación de las ondas electromagnéticas dependiendo de la frecuencia.

A frecuencias bajas (VLF, ELF, ULF) las ondas se propagan siguiendo la superficie terrestre hasta varios miles de kilómetros. A este se le conoce como onda de superficie. Para frecuencias superiores, las ondas se propagan por la ionosfera (MF, HF) y troposfera. En frecuencias superiores a VHF las ondas se propagan en línea recta en cortas distancias.

Si se desprecian los efectos de reflexión y refracción y suponiendo que las ondas son un rayo directo que va desde una antena transmisora a una antena receptora, el factor a tener en cuenta es la curvatura de la Tierra. Para que haya conexión, debe de haber visibilidad directa entre las antenas.

Esto es lo que se propone en el ejercicio T4P1, calcular la altura mínima de una antena dada la altura de otra antena y la distancia entre ellas o calcular la distancia máxima entre dos antenas de alturas conocidas para que haya visibilidad directa.

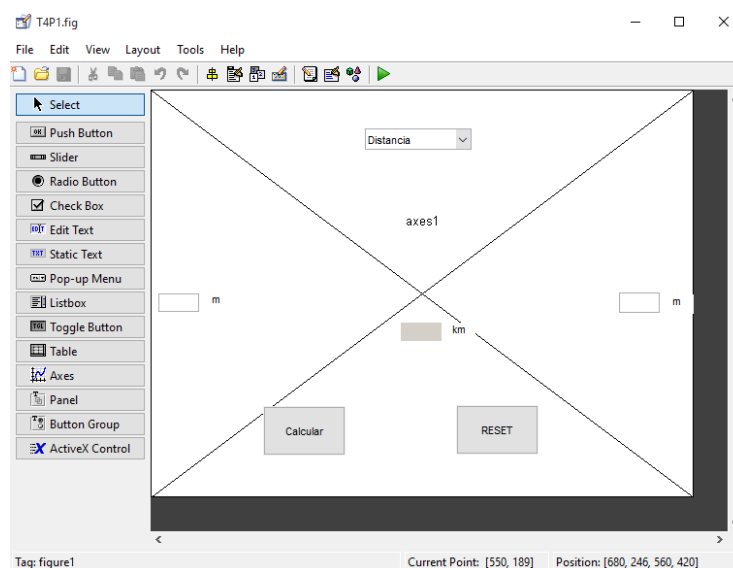


Figura 3-24 T4P1, vista de desarrollo

Esta ventana se compone de un Pop-up Menu donde se puede elegir el dato que se desea calcular.

Dos Edit Text y un Static Text en el centro de la pantalla. Dependiendo de la incógnita del problema (altura de una antena o distancia entre antenas), el Static Text será uno u otro.

Se tienen dos Push Buttons, uno para calcular el problema y otro para resetear los datos.

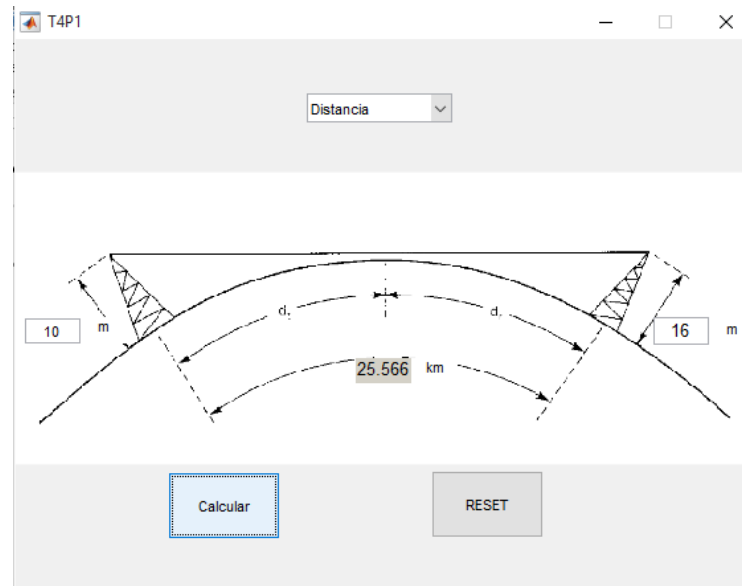


Figura 3-25 T4P1, vista de ejecución

Ondas viajando largas distancias a través de la troposfera e ionosfera son curvadas debido a los efectos de reflexión y refracción.

Ondas de Superficie

En las ondas de superficie una parte de la energía es absorbida por el suelo, induciendo corrientes. Esta absorción aumenta cuanto mayor es la frecuencia y es mayor para polarización horizontal que para polarización vertical.

Además, la conductividad del suelo también influye en la absorción.

Tipo de suelo	S/m
Agua salada	5
Terrenos boscosos	$8 \cdot 10^{-3}$
Terrenos áridos	$2 \cdot 10^{-3}$
Suelo urbano	10^{-3}

Tabla 3 Absorción en distintos tipos de terreno

Debido a la complejidad del cálculo, es necesario disponer de un juego de curvas de propagación por onda de superficie que correspondan a un cierto número de valores-tipo de frecuencias y de características del suelo. [8]

Estas curvas de propagación se pueden ver en el [Anexo II](#). Un problema añadido es el caso en el que la propagación discorra por distintos terrenos. Para ello ITU-R (Radiocommunication Sector of the International Telecommunication Union) recomienda el método de Millington para el cálculo del campo recibido.

A continuación, se puede ver un ejemplo para el cálculo de la intensidad del campo a través del método de Millington con tres terrenos diferentes.

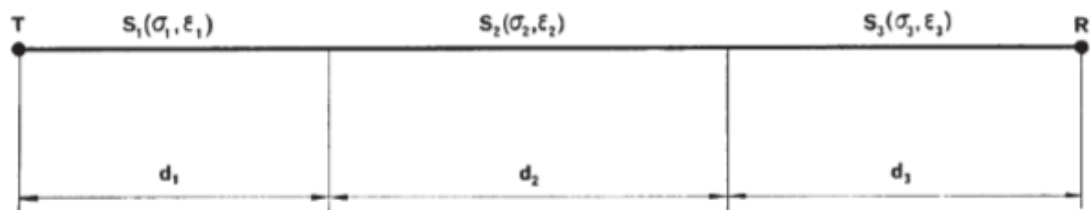


Figura 3-26 Distintos tipos de terreno que atraviesa una onda de superficie.

$$E_R = E_1(d_1) - E_2(d_1) + E_2(d_1 + d_2) - E_3(d_1 + d_2) + E_3(d_1 + d_2 + d_3)$$

$$E_T = E_3(d_3) - E_2(d_3) + E_2(d_3 + d_2) - E_1(d_3 + d_2) + E_1(d_3 + d_2 + d_1)$$

$$E(d) = \frac{E_T + E_R}{2}$$

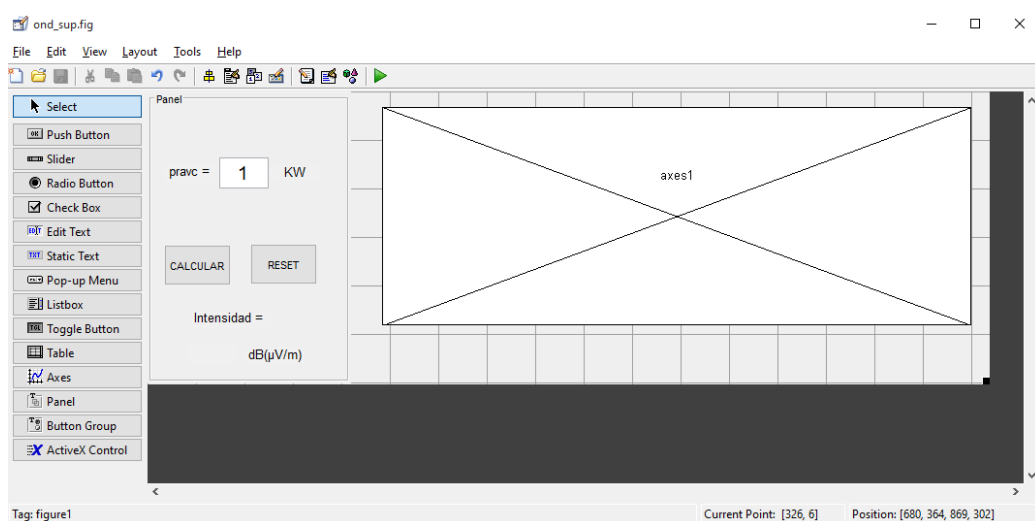


Figura 3-27 ond_sup, vista de desarrollo

Esta ventana se compone de un panel, el cual contiene tres Static Text (en uno de ellos será donde se muestre el resultado) y dos Push Buttons, para comenzar a hacer los cálculos y para resetear. En la parte derecha contiene un Axes en donde se dibujan los distintos terrenos

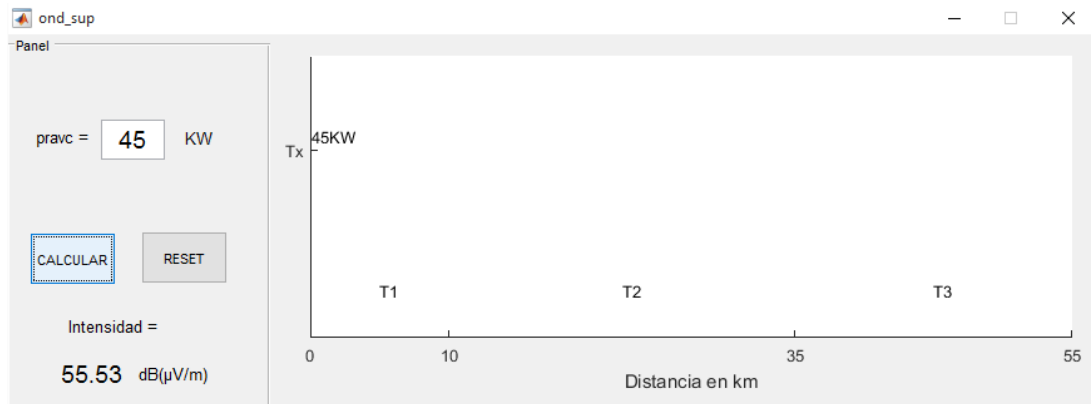


Figura 3-28 ond_sup, vista de ejecución

Onda Espacial

La modalidad de propagación de ondas de espacio es utilizada por radioenlaces en visibilidad o fijos, también denominados LOS (Line of Sight).

La energía asociada a la onda se transmite entre la antena transmisora y la receptora en línea recta (una vez corregida la refracción atmosférica) y en forma de onda reflejada por la superficie terrestre o por la troposfera.

La onda espacial es la única que interesa en UHF, VHF y SHF.

Las ondas electromagnéticas son refractadas cuando pasan de un medio a otro con diferentes índices de refracción.

Índice de refracción:

$$n = n_0 + \frac{dn}{dh}h = n_0 + g * h$$

El valor de g queda definido mediante la fórmula de Booker:

$$g = \frac{\partial n}{\partial h} = \frac{1}{R_0} \left(-0.2 + 30 \frac{\partial P}{\partial h} - 6 \frac{\partial T}{\partial h} \right)$$

Para características de la atmósfera estándar:

Radio ecuatorial terrestre $\rightarrow R_0 = 6378$ km.

Variación de la presión del vapor de agua con la altura $\rightarrow \frac{\partial P}{\partial h} = -0.0033 \text{ mbar/m}$.

Variación de la temperatura con h $\rightarrow \frac{\partial T}{\partial h} = -0.009^\circ\text{C} / \text{m}$

Introduciendo estos valores en la ecuación obtenemos:

$$g = \frac{0.245}{R_0}$$

Al introducir la g en la ecuación del índice de refracción:

$$n \cong n_0 - \frac{n}{4R_0}$$

El gradiente de refractividad G:

$$G = g * 10^6$$

Para efectuar modelos simplificados se suele modificar la curvatura del rayo para considerarlo rectilíneo. Esto implica modificar la curvatura de la tierra en un factor k.

$$k = \frac{157}{157 + G}$$

$$a' = k * a$$

Según el valor de k, tenemos distintos tipos de troposfera:

- Súper-refractiva, $k > 4/3$.
- Normal, $1 \leq k \leq 4/3$.
- Sub-refractiva, $0 \leq k \leq 1$.
- Conductiva, $k < 0$.

Cuando una onda electromagnética incide sobre un obstáculo se difracta. Es necesario tener en cuenta el volumen que ocupa la onda. Se definen los elipsoides de Fresnel como elipsoides de orden n, con los focos en el transmisor y el receptor, siendo el radio del elipsoide en un punto a una distancia (d1) del transmisor y una distancia (d2) del receptor y una longitud de onda λ :

$$r_n = \sqrt{\frac{n * \lambda * d1 * d2}{d1 + d2}}$$

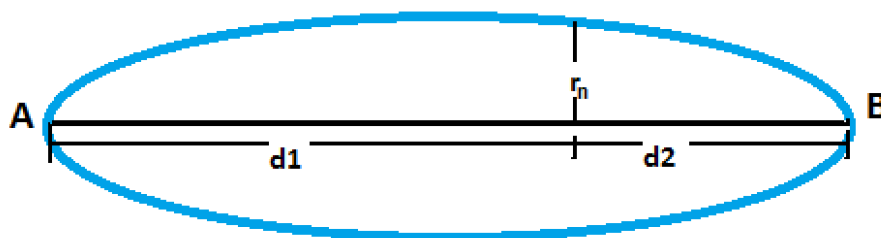


Figura 3-29 Elipsoide de Fresnel.

Para considerar condición de espacio libre el primero lóbulo de Fresnel (aquel de orden $n=1$) no se debe ver obstaculizado en el trayecto emisor-receptor.

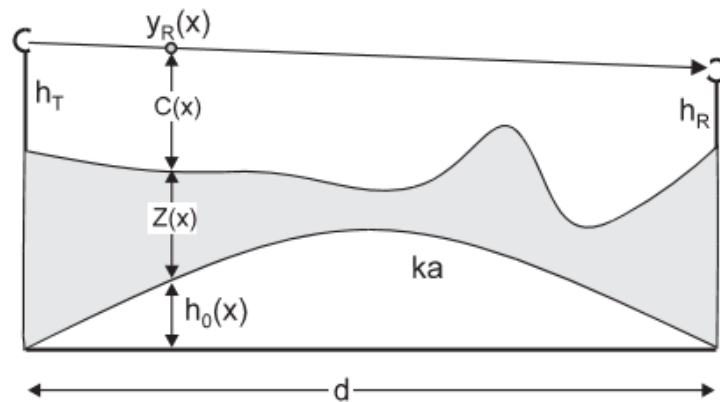


Figura 3-30 Alturas a considerar para la condición de espacio libre.

$h_0(d1) \rightarrow$ un punto sobre la parábola base para un radio ficticio $k*a$.

$$h_0 = \frac{d1 * d2}{2 * k * a}$$

$Z(d1) \rightarrow$ la cota del terreno a una distancia $d1$ del transmisor.

$C(d1) \rightarrow$ distancia que hay a una distancia $d1$ del transmisor entre el terreno y la recta que une transmisor y receptor.

$Y_R(d1) \rightarrow$ la ordenada a una distancia $d1$.

Para el cálculo de la zona de Fresnel está la ventana *Fresnel*.

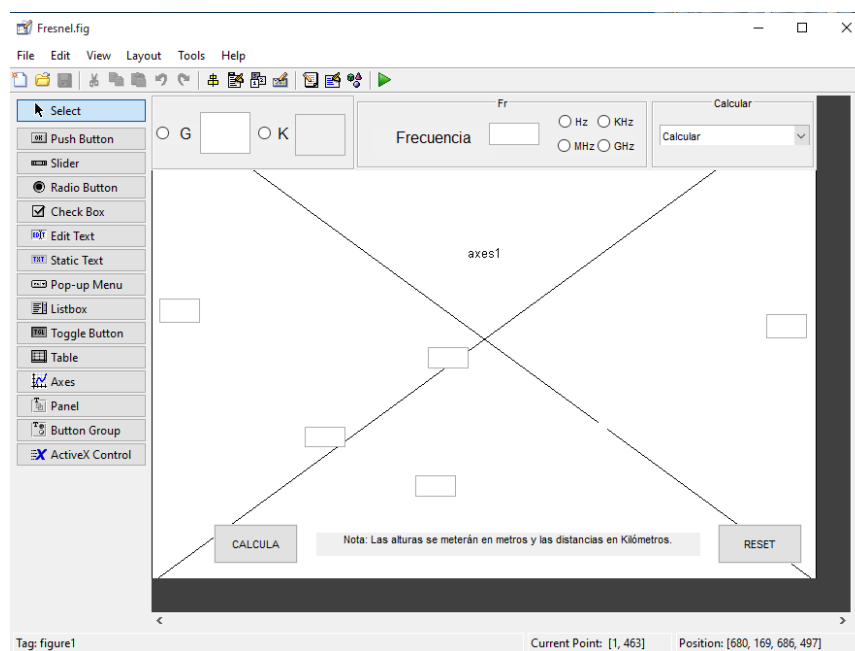


Figura 3-31 Fresnel, vista de desarrollo

Esta ventana se compone de tres paneles en la parte superior. El de la izquierda está formado por dos Radio Buttons y dos Edit Text y sirve para introducir el valor de G o el valor de k .

En el panel central hay un Static Text, un Edit Text y cuatro Radio Buttons. En este panel se introduce la frecuencia de la onda y la magnitud.

El panel de la derecha está formado por un Pop-up Menu, en el que se puede elegir lo que se quiere calcular.

En el resto de la ventana se encuentran una serie de Edit Texts en los que se introducen los valores y dos Push Buttons para calcular el resultado o para borrar todos los datos.

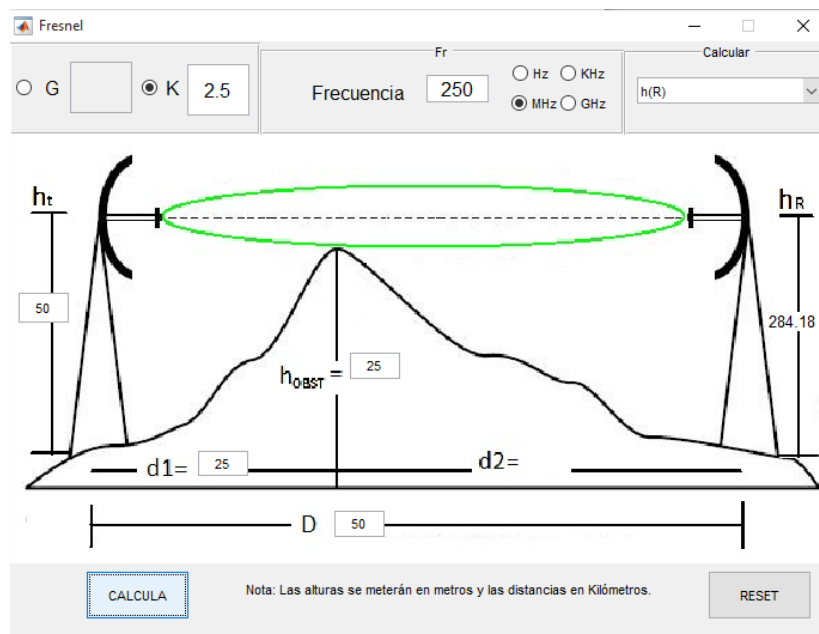


Figura 3-32 Fresnel, vista de ejecución

4 PRESENTACIÓN DE RESULTADOS

Una vez se ha terminado de desarrollar la herramienta hay que probarla. Para abrirla en un ordenador que tenga MATLAB sólo hay que abrir el ejecutable 'VPrincipal'.

En caso de que el ordenador no tenga MATLAB, se debe:

1. Copiar el instalador del RunTime de un ordenador con MATLAB:
<matlabroot>\toolbox\compiler\deploy\win32\MCRInstaller.exe
2. Ya en el ordenador de destino, desempaquetar los instaladores, haciendo doble clic en MCRInstaller.
3. Instalar MCRInstaller.
4. Asegurarse, una vez instalado, de que C:\MCR\v73\runtime\win32 está en el PATH.

Una vez se haya terminado de instalar el RunTime, o se tenga el MATLAB, se procede a abrir la aplicación.

4.1 Ejemplo de uso

Una vez abierta la aplicación, aparece la Ventana Principal. Donde se elige el tema el tema sobre el que se quiere trabajar.



Figura 4-1 Ejemplo de uso 1

Al elegir el tema, se abre la Ventana 2 donde se puede elegir entre hacer problemas o realizar algunas cuestiones teóricas tipo test.

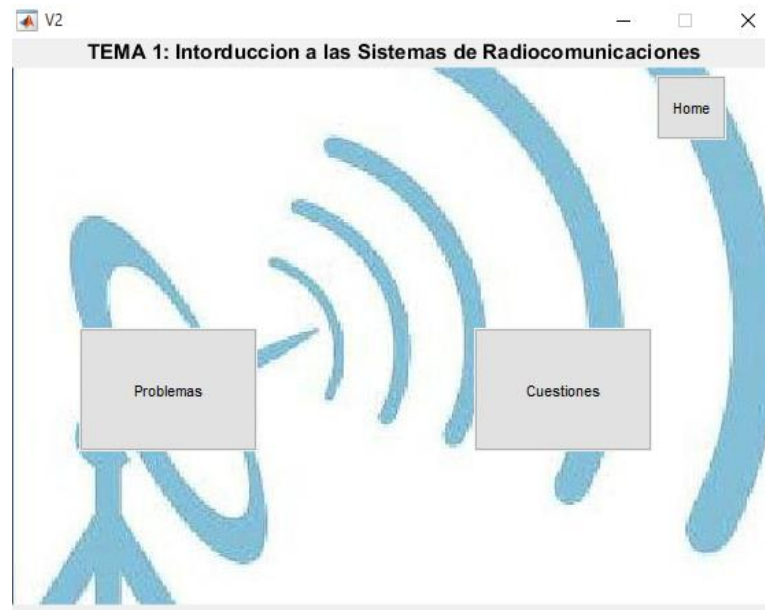


Figura 4-2 Ejemplo de uso 2

4.1.1 Tema 1

Al elegir la opción de problemas, se abre una ventana llamada T1P en la cual sólo se puede seleccionar un tipo de ejercicio. Al seleccionar este ejercicio y apretar el botón de IR, aparece una ventana, la cual son ejercicios para convertir magnitudes a unidades logarítmicas.

Una vez realizadas todas las conversiones, se pulsa el botón corregir. Una vez apretado el botón, las soluciones introducidas se pondrán en color verde o rojo, dependiendo de si están bien o mal calculadas respectivamente. En caso de no ser capaces de realizar alguna conversión se puede ver la solución, aunque habrá que pulsar tres veces el botón ‘Soluciones’, antes de que se muestren.

Convierte las siguientes unidades.					
$g=170$	22.3	G(dB)	$P(\text{dBm})=89$	794	E(dBmV/m)
$p(W)=13$	11.14	P(dBW)	$E(\text{dBmV/m})=43$	127	e(mV/m)
$p(\text{mW})=150$	21.76	P(dBm)	$E(\text{dBu})=76$	6.31	e(mV/m)
$e(\text{mV/m})=35$	30.88	E(dBmV/m)	$P(\text{dBW})=37$	9	P(dBKW)
$e(\mu\text{V/m})=47$	33.44	E(dBu)	$P(\text{dBW})=37$	67	P(dBm)
$G(\text{dB})=13$	19.95	g	$E(\text{dBmV/m})=43$	103	E(dBu)
$P(\text{dBW})=37$	5.01	p(W)	$E(\text{dBu})=76$	44	E(dBV/m)

Figura 4-3 Ejemplo de uso 2, tema 1

Si en vez de elegir la opción de problemas se elige la de Cuestiones se abre otra ventana en la que se permite responder a una serie de preguntas tipo test. Hay cuatro posibles respuestas para cada una de las preguntas y sólo deja pasar a la siguiente pregunta si se ha seleccionado la respuesta correcta.

The screenshot shows a window titled 'T1C' with a question: '¿Que tipos de modo de explotación existen?'. Below the question are four radio button options: 'Simplex', 'Duplex', 'Semiduplex', and 'Todas son correctas'. The 'Todas son correctas' option is selected. To the right of the options are two buttons: 'SIGUIENTE' and 'FINALIZAR'.

Figura 4-4 Ejemplo de uso 3, Tema 1

4.1.2 Tema 2

Al elegir el tema 2 se presentan dos posibilidades, Modulaciones y Cuestiones. La de Cuestiones es similar a la del tema 1 por lo que no se volverá a explicar.

La de Modulaciones da la posibilidad de escoger entre modulaciones analógicas (AM convencional, FM, PM, DBL) y modulaciones digitales.

➤ Modulaciones analógicas

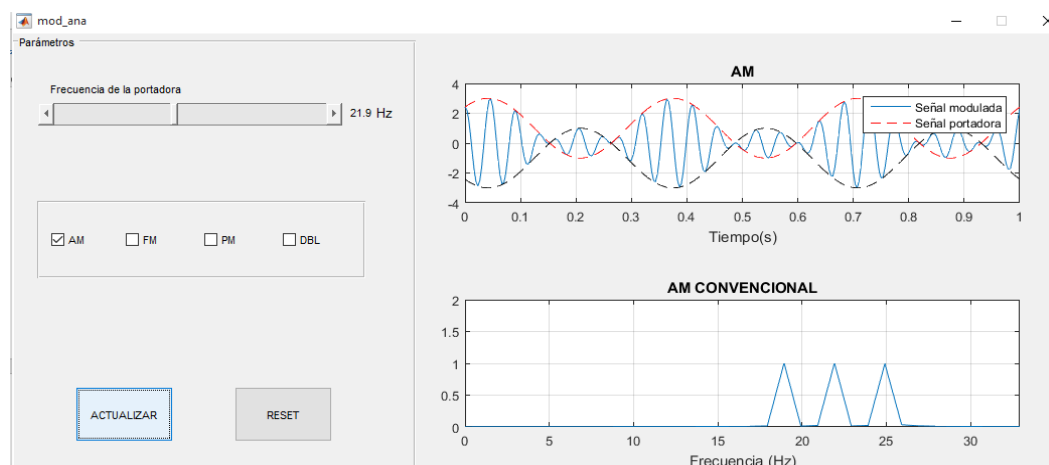


Figura 4-5 Ejemplo de uso 1, Tema 2

Arriba a la izquierda se encuentra un Slider el cual permite escoger la frecuencia de la señal portadora. Esta frecuencia está comprendida entre 25 y 75 Herzios.

A continuación, se muestran los tipos de modulaciones analógicas que se pueden realizar. En este ejemplo se ha escogido AM convencional.

Al apretar el botón de actualizar, dibuja en las gráficas de la derecha, la señal modulada y la portadora (en la gráfica de arriba) con su correspondiente leyenda, en el eje temporal, y en la gráfica de abajo se muestra el espectro de la señal modulada en función de la frecuencia.

➤ ASK, PSK y FSK

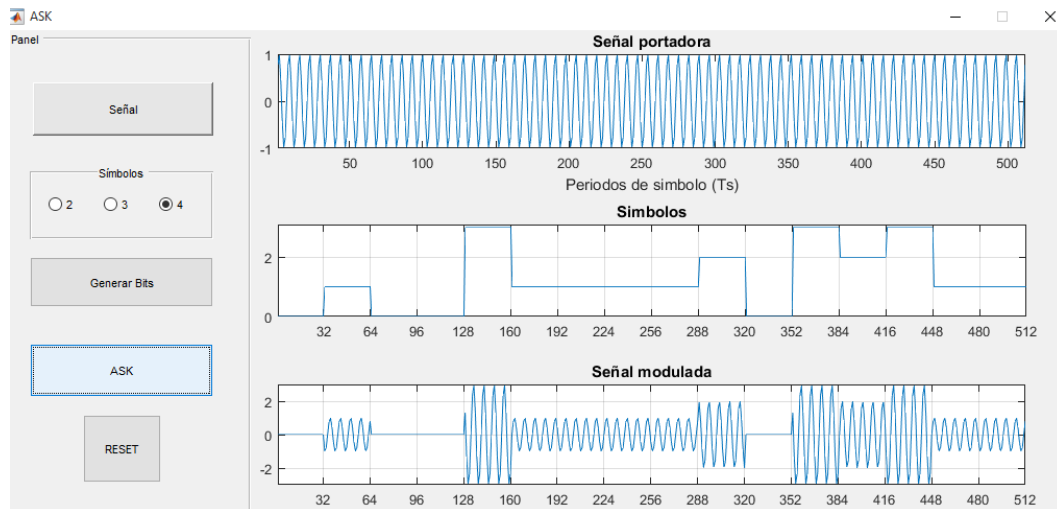


Figura 4-6 Ejemplo de uso 2, Tema 2

Para representar la ASK, PSK o FSK se hace de la misma manera. Primero hay que generar una señal portadora apretando el botón superior izquierdo ‘Señal’.

A continuación, hay que elegir el número de bits por símbolo que tiene la señal modulada y apretar el botón ‘Generar Bits’. Este botón crea una serie de bits aleatorios y automáticamente los presenta agrupados en símbolos.

Por último, pulsar el botón ‘ASK’ (‘PSK’ o ‘FSK’ en caso de estar con cualquiera de las otras modulaciones). Y se genera la señal modulada. Con el botón reset se borran todas las señales generadas.

4.1.3 Tema 3

Al elegir el tema 3 se presentan dos posibilidades, Diagrama de Radiación y Cuestiones. La de Cuestiones es similar a la del resto de temas, por lo que no se va a explicar.

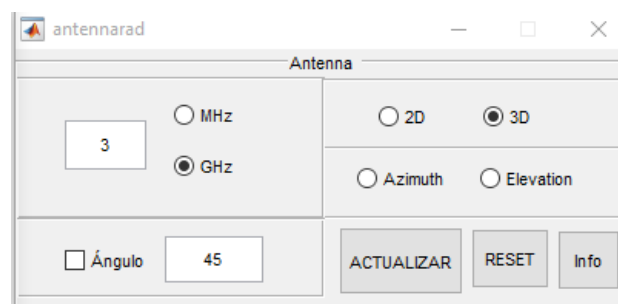


Figura 4-7 Ejemplo de uso 1, Tema 3

Para el diagrama de radiación se introduce la frecuencia en el recuadro superior izquierdo y se elige si la frecuencia está en MHz o en GHz. Acto seguido, se debe elegir si se quiere una representación en dos o en tres dimensiones.

En caso de ser en dos dimensiones, se ha de elegir si se quiere el corte en azimut o en elevación.

El corte se hace por defecto en el ángulo 0° . Si se quiere realizar otro corte y que se represente también, hay que pulsar el botón de ‘Ángulo’, en el recuadro inferior izquierdo e introducir el ángulo en grados.

El botón de ‘Actualizar’ representa el diagrama de radiación que se haya querido generar. El botón de ‘Reset’ borra los datos introducidos. Y por último existe un botón ‘Info’ el cual informa de que tipo de antena es.

4.1.4 Tema 4

Al elegir el tema 3 se presentan dos posibilidades, Problemas y Cuestiones. La de Cuestiones es similar a la del resto de temas, por lo que no se va a explicar.

Dentro de los problemas nos encontramos con tres tipos distintos.

➤ Fresnel

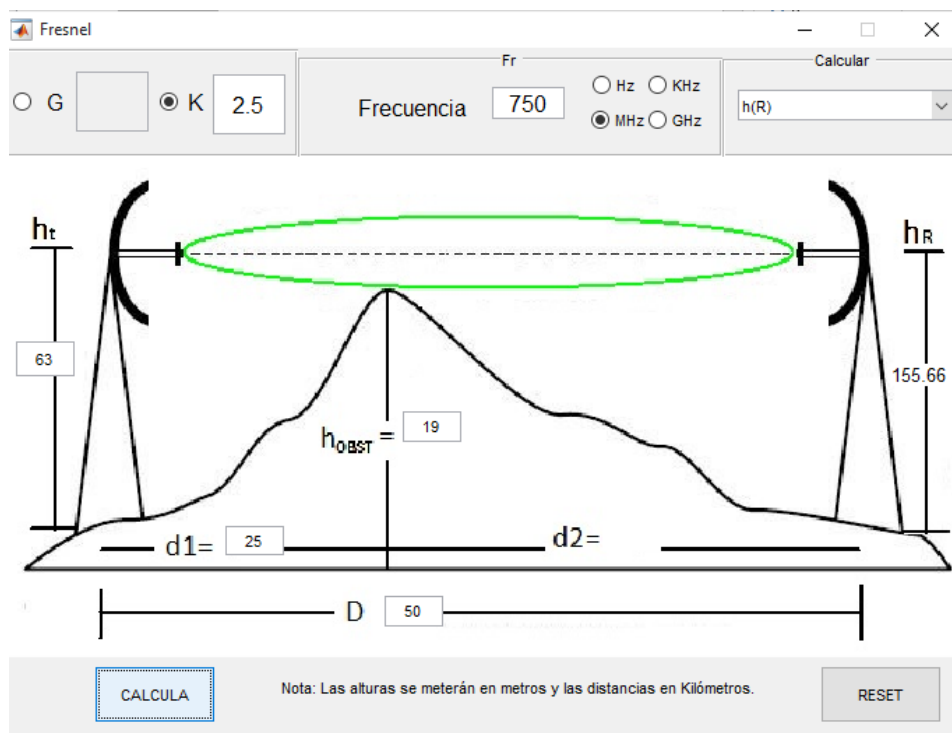


Figura 4-8 Ejemplo de uso 1, Tema 4

Aquí se puede elegir entre varios tipos de ejercicios. Calcular altura mínima de una antena, calcular la altura máxima de un obstáculo, calcular la frecuencia mínima o dadas unas condiciones y comprobar si la primera zona de Fresnel libra el obstáculo. Para realizar todos estos ejercicios, se cuenta con una serie de parámetros que deben ser introducidos.

En el panel de arriba a la derecha, es donde se elige el tipo de ejercicio que se va a realizar.

En la parte superior izquierda se encuentra la opción de introducir la G (Gradiente de refractividad) o la k (factor que determina el tipo de troposfera), según cuál de los dos sea conocida.

Lo siguiente que se debe introducir es la frecuencia. Y a continuación todos los datos que están

encima del dibujo, como son la altura de la antena transmisora (h_t), de la antena receptora (h_r), la altura del obstáculo (h_{obst}), la distancia entre la antena transmisora y el obstáculo (d_1) y la distancia entre las dos antenas (D). Se deberán introducir todos estos datos, excepto el que se desee calcular, el cuál la propia aplicación no dejará modificar.

➤ Onda de Superficie

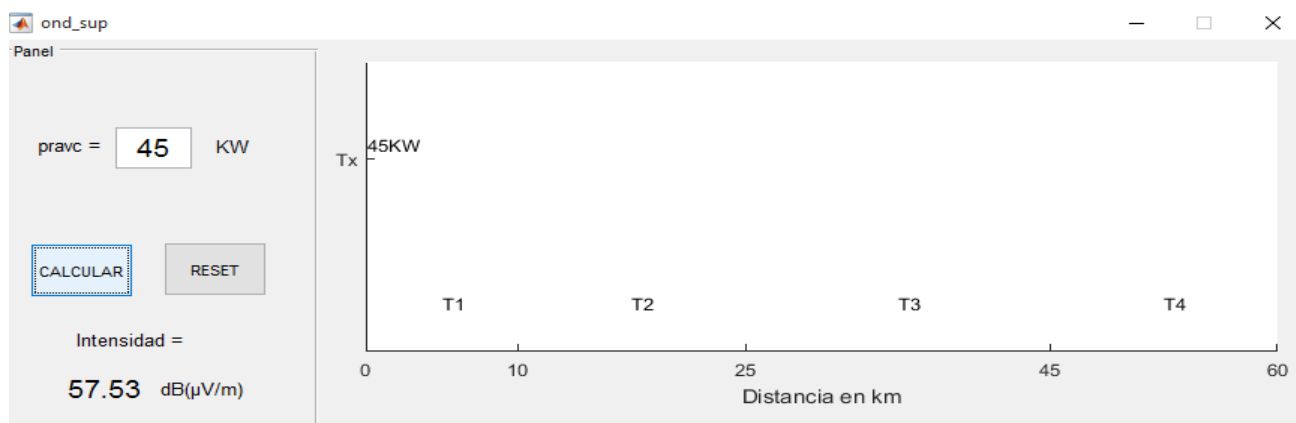


Figura 4-9 Ejemplo de uso 2, Tema 4

Este ejercicio calcula la intensidad de campo recibida por una antena, de una señal enviada por una antena transmisora a través de distintos tipos de terrenos.

Para empezar, se debe introducir la potencia de transmisión de la antena en KW, y a continuación pulsar el botón de calcular y comenzarán a aparecer pantallas en las que introducir datos.

El primer valor que hay que introducir es el número de terrenos. Los siguientes valores son las distancias de cada terreno por separado.

Por último, preguntará el valor de la intensidad de campo para un terreno determinado a una distancia determinada. Este valor se saca de las tablas, las cuales se encuentran en el [Anexo II](#).

Una vez que ya tiene todos los valores que necesita el programa realiza el cálculo y lo pone en pantalla.

➤ *Distancia entre antenas*

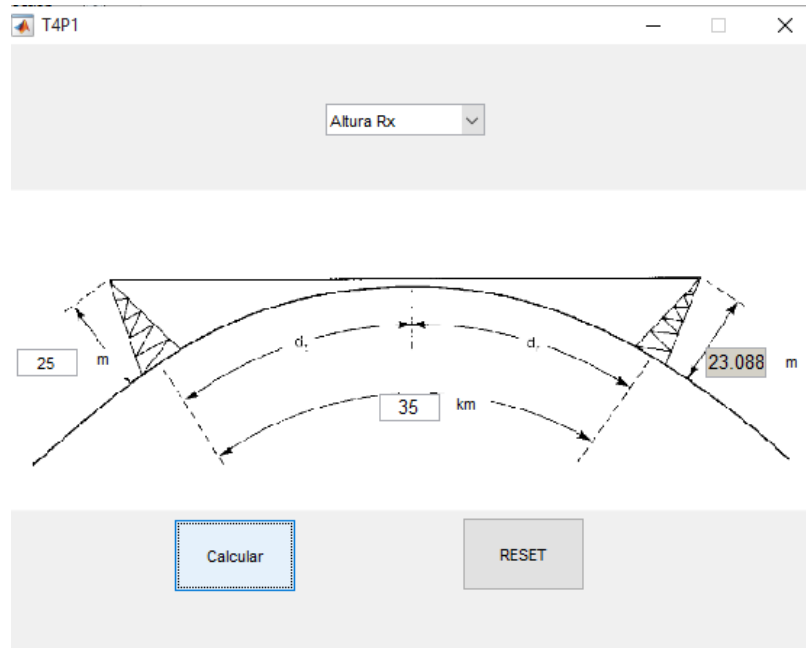


Figura 4-10 Ejemplo de uso 3, Tema 4

En este ejercicio se puede calcular tanto la altura mínima de una antena transmisora y una receptora para que haya visión directa entre ellas, como la distancia máxima.

Para ello se debe en primer lugar seleccionar qué es lo que se desea calcular, y una vez seleccionado se introducen los otros dos valores y se pulsa 'Calcular'.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

Se han conseguido muchos de los objetivos propuestos para este TFG:

- Se ha conseguido desarrollar una herramienta simple e intuitiva de manejar para el usuario que cubre casi todos los temas de la asignatura. Aunque debido al poco tiempo del que se ha dispuesto, no ha sido posible implementar los distintos tipos de ejercicios que cubrirían todos los contenidos de la asignatura.
- Aún no es posible determinar si se va a conseguir motivar a los alumnos en el estudio de la asignatura. Aunque una encuesta realiza a una serie de alumnos que ya la han cursado muestra una alta probabilidad de conseguir este objetivo, ya que el 75% de los encuestados afirman que de haber dispuesto de esta herramienta mientras cursaban la asignatura, la hubieran usado.
- Durante el desarrollo de este TFG, el autor ha mejorado su madurez, independencia y crítica en la búsqueda de información de diversas fuentes.

5.2 Líneas Futuras

Aunque se han conseguido grandes avances, aún queda mucho por desarrollar en esta aplicación. Se proponen como líneas futuras de mejora las siguientes:

1. Realizar una modulación QAM.
2. Representar constelaciones y espectros de las modulaciones digitales.
3. Crear más tipos de antenas para que el alumno pueda compararlas.
4. Implementar problemas de 'Balance de un enlace radioeléctrico'.
5. Opción de elegir problemas para hacer, teniendo una casilla en la que introducir el resultado.
6. Tener la opción de introducir preguntas tipo test, para que sean resueltas posteriormente.

6 BIBLIOGRAFÍA

- [1] [En línea]. Available: <http://www.buscabiografias.com/biografia/verDetalle/4252/Hans%20Christian%20Oersted>. [Último acceso: 10 Enero 2016].
- [2] [En línea]. Available: <http://www10.uniovi.es/semanacyt2007/experimentando/magnetismo/experimentos/e4.htm>. [Último acceso: 03 Marzo 2016].
- [3] [En línea]. Available: <http://bibliadelarazon.blogspot.com/2001/05/1831-electricidad-y-magnetismo-unidos.html>. [Último acceso: 10 Enero 2016].
- [4] [En línea]. Available: <http://fisica3roifd.blogspot.com.es/p/ley-de-faraday.html>. [Último acceso: 03 Marzo 2016].
- [5] [En línea]. Available: <http://bibliadelarazon.blogspot.com/2001/04/1873-electricidad-y-magnetismo-se-unen.html>. [Último acceso: 13 Enero 2016].
- [6] [En línea]. Available: <http://www.escuelapedia.com/las-ecuaciones-de-maxwell/>. [Último acceso: 03 Marzo 2016].
- [7] [En línea]. Available: <http://www.biografiasyvidas.com/biografia/h/hertz.htm>. [Último acceso: 13 Enero 2016].
- [8] [En línea]. Available: <https://www.itu.int/itu-news/manager/display.asp?lang=es&year=2008&issue=03&ipage=Arthur-Clarke&ext=html>. [Último acceso: 15 Enero 2016].
- [9] [En línea]. Available: <http://www.thisdayinaviation.com/tag/telstar/>. [Último acceso: 03 Marzo 2016].
- [10] [En línea]. Available: https://es.wikipedia.org/wiki/Historia_del_tel%C3%A9fono_m%C3%B3vil. [Último acceso: 03 Marzo 2016].
- [11] [En línea]. Available: <http://recursostic.educacion.es/secundaria/tecnologia/tesla/web/>. [Último acceso: 03 Marzo 2016].
- [12] [En línea]. Available: <http://timerime.com/es/evento/2080022/Pressey+teaching+machine/>. [Último acceso: 03 Marzo 2016].
- [13] [En línea]. Available: <http://www.free-power-point-templates.com/articles/free-educational-powerpoint-theme-for-presentations-in-the-classroom/>. [Último acceso: 03 Marzo 2016].
- [14] [En línea]. Available: <https://www.youtube.com/watch?v=YJPUNC-GdU4>. [Último acceso: 03 Marzo 2016].

- [15] [En línea]. Available: http://sisbib.unmsm.edu.pe/bibvirtual/tesis/ingenie/caba%C3%B1as_v_j/cap1.htm. [Último acceso: 21 Enero 2016].
- [16] [En línea]. Available: <http://fatic.uvigo.es/index.php/es/>. [Último acceso: 03 Marzo 2016].
- [17] [En línea]. Available: [https://es.wikipedia.org/wiki/R_\(lenguaje_de_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/R_(lenguaje_de_programaci%C3%B3n)). [Último acceso: 20 Enero 2016].
- [18] [En línea]. Available: https://es.wikipedia.org/wiki/RKward#/media/File:RKward_screenshot.png. [Último acceso: 03 Marzo 2016].
- [19] [En línea]. Available: https://es.wikipedia.org/wiki/GNU_Octave. [Último acceso: 29 Febrero 2016].
- [20] [En línea]. Available: https://es.wikipedia.org/wiki/RKward#/media/File:RKward_screenshot.png. [Último acceso: 03 Marzo 2016].
- [21] [En línea]. Available: <https://es.wikipedia.org/wiki/MATLAB#MATLAB>. [Último acceso: 21 Enero 2016].
- [22] J. M. Hernando Rábanos, L. Mendo Tomás y J. M. Riera Salis, Transmisión por radio, EDITORIAL UNIVERSITARIA RAMON ARECES.
- [23] *Apuntes de la asignatura de Sistemas de Radiocomunicaciones*, 2015.
- [24] La Asamblea de Radiocomunicaciones de la UIT, [En línea]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.368-7-199203-S!!PDF-S.pdf. [Último acceso: 23 Febrero 2016].

ANEXO I: CÓDIGO

VPrincipal

```
function varargout = VPrincipal(varargin)
% VPRINCIPAL MATLAB code for VPrincipal.fig
%     VPRINCIPAL, by itself, creates a new VPRINCIPAL or raises the existing
%     singleton*.
%
%     H = VPRINCIPAL returns the handle to a new VPRINCIPAL or the handle to
%     the existing singleton*.
%
%     VPRINCIPAL('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in VPRINCIPAL.M with the given input arguments.
%
%     VPRINCIPAL('Property','Value',...) creates a new VPRINCIPAL or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before VPrincipal_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to VPrincipal_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help VPrincipal

% Last Modified by GUIDE v2.5 28-Jan-2016 13:47:04

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @VPrincipal_OpeningFcn, ...
                  'gui_OutputFcn',  @VPrincipal_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before VPrincipal is made visible.
function VPrincipal_OpeningFcn(hObject, eventdata, handles, varargin)
background=imread('v0.jpg');
axes(handles.axes1);
axis off
imshow(background)
title('SISTEMAS DE RADIOCOMUNICACIONES','FontSize',16);
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes VPrincipal wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = VPrincipal_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in selecttema.
function selecttema_Callback(hObject, eventdata, handles)
% Aquí se selecciona el tema
global tema %Al definir una variable como global, nos da la oportunidad de
% poder usarla en múltiples ventanas.
tema = get(hObject, 'Value');
if tema > 1 && tema < 7
    get(handles.infotema, 'Enable');
    set(handles.infotema, 'Enable', 'on');
% Si hemos elegido uno de los temas, el botón se activará
else
    get(handles.infotema, 'Enable');
    set(handles.infotema, 'Enable', 'off');
    set(handles.infotema, 'String', ' ');
end
% En caso de no tener seleccionado uno de los temas, el botón no se
% podrá usar, y se borrará todo lo que hubiese escrito en el botón.

if tema == 2
    get(handles.infotema, 'String');
    set(handles.infotema, 'String', 'INTRODUCCIÓN');
elseif tema == 3
    get(handles.infotema, 'String');
    set(handles.infotema, 'String', 'MODULACIONES');
elseif tema == 4
    get(handles.infotema, 'String');
    set(handles.infotema, 'String', 'RADIACIÓN Y ONDAS GUIADAS');
elseif tema == 5
    get(handles.infotema, 'String');
    set(handles.infotema, 'String', 'RADIOPROPAGACIÓN');
elseif tema == 6
    get(handles.infotema, 'String');
    set(handles.infotema, 'String', 'BALANCE DE UN ENLACE RADIOELECTRICO');
end
% hObject handle to selecttema (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
function selecttema_CreateFcn(hObject, eventdata, handles)
```

```
% hObject      handle to selectema (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in infotema.
function infotema_Callback(hObject, eventdata, handles)
global tema
if tema > 1 & tema < 6
    V2;
elseif tema == 6
    msgbox('En proceso de desarrollo');
end
% Nos lleva a la siguiente ventana a la que hemos llamado V2.

% --- Executes on button press in acerca.
function acerca_Callback(hObject, eventdata, handles)
msgbox('Este programa ha sido creado por: Gonzalo Peñuelas','Acerca de...');
% Un mensaje en el cual se pone que el creador del mensaje es Gonzalo Peñuelas
```

V2

```
function varargout = V2(varargin)
% V2 MATLAB code for V2.fig
%      V2, by itself, creates a new V2 or raises the existing
%      singleton*.
%
%      H = V2 returns the handle to a new V2 or the handle to
%      the existing singleton*.
%
%      V2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in V2.M with the given input arguments.
%
%      V2('Property','Value',...) creates a new V2 or raises the
%      existing singleton*. Starting from the left, property value pairs are
%      applied to the GUI before V2_OpeningFcn gets called. An
%      unrecognized property name or invalid value makes property application
%      stop. All inputs are passed to V2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help V2

% Last Modified by GUIDE v2.5 31-Jan-2016 16:57:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
```

```
        'gui_Singleton', gui_Singleton, ...
        'gui_OpeningFcn', @V2_OpeningFcn, ...
        'gui_OutputFcn', @V2_OutputFcn, ...
        'gui_LayoutFcn', [] , ...
        'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before V2 is made visible.
function V2_OpeningFcn(hObject, eventdata, handles, varargin)
background=imread('v2.jpg');
axes(handles.axes1);
axis off
imshow(background)
% Choose default command line output for V2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes V2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = V2_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
global tema %recordamos el valor de la variable tema, de la ventana anterior
get(handles.problemas, 'Enable');
if tema==2
    title('TEMA 1: Introducción a las Sistemas de Radiocomunicaciones');
elseif tema == 3
    set(handles.problemas, 'String', 'Modulaciones');
    set(handles.problemas, 'Enable', 'on');
    title('TEMA 2: Modulaciones');
elseif tema == 4
    set(handles.problemas, 'String', 'Diagramas de radiación');
    set(handles.problemas, 'Enable', 'on');
    title('TEMA 3: Radiación y Ondas Guiadas');
elseif tema == 5
    set(handles.problemas, 'String', 'Problemas');
    set(handles.problemas, 'Enable', 'on');
    title('TEMA 4: Radiopropagación');
elseif tema == 6
    set(handles.problemas, 'String', 'Problemas');
    set(handles.problemas, 'Enable', 'on');
```

```
    title('TEMA 5: Balance de un Enlace Radioeléctrico');
end
%ponemos en la parte superior de la ventana el nombre del tema

% --- Executes on button press in problemas.
function problemas_Callback(hObject, eventdata, handles)
global tema
switch tema
    case 2
        T1P;
    case 3
        T2P;
    case 4
        antennarad;
    case 5
        T4P;
    case 6
        T5P1;
end
close V2
% Este boton nos lleva a los problemas del tema que hayamos elegido.

% --- Executes on button press in cuestiones.
function cuestiones_Callback(hObject, eventdata, handles)
global tema
switch tema
    case 2
        T1C
    case 3
        T2C
    case 4
        T3C
    case 5
        T4C
    case 6
        T5C
    case 7
        T6C
end
close(V2)

% --- Executes on button press in volver.
function volver_Callback(hObject, eventdata, handles)
close all;
VPrincipal;
```

T1P

```
function varargout = T1P(varargin)
% T1P MATLAB code for T1P.fig
%     T1P, by itself, creates a new T1P or raises the existing
%     singleton*.
%
%     H = T1P returns the handle to a new T1P or the handle to
%     the existing singleton*.
%
%     T1P('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in T1P.M with the given input arguments.
%
%     T1P('Property','Value',...) creates a new T1P or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before T1P_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to T1P_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help T1P

% Last Modified by GUIDE v2.5 08-Feb-2016 00:18:12

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',  @T1P_OpeningFcn, ...
                  'gui_OutputFcn',   @T1P_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before T1P is made visible.
function T1P_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin   command line arguments to T1P (see VARARGIN)
background = imread('radiocomunicaciones.jpg');
axis off
imshow(background)
% Choose default command line output for T1P
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes T1P wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = T1P_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in ejercicios.
function ejercicios_Callback(hObject, eventdata, handles)
% hObject handle to ejercicios (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns ejercicios contents as
cell array
% contents{get(hObject,'Value')} returns selected item from ejercicios

% --- Executes during object creation, after setting all properties.
function ejercicios_CreateFcn(hObject, eventdata, handles)
% hObject handle to ejercicios (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in ir.
function ir_Callback(hObject, eventdata, handles)
close(T1P)
T1P1
clc
```

T1P1

```
function varargout = T1P1(varargin)
% T1P1 MATLAB code for T1P1.fig
%   T1P1, by itself, creates a new T1P1 or raises the existing
%   singleton*.
%
%   H = T1P1 returns the handle to a new T1P1 or the handle to
%   the existing singleton*.
%
%   T1P1('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in T1P1.M with the given input arguments.
%
%   T1P1('Property','Value',...) creates a new T1P1 or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before T1P1_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to T1P1_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help T1P1

% Last Modified by GUIDE v2.5 07-Feb-2016 23:37:21

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',  @T1P1_OpeningFcn, ...
                  'gui_OutputFcn',  @T1P1_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before T1P1 is made visible.
function T1P1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin   command line arguments to T1P1 (see VARARGIN)
axis off
global intentos
intentos = 0;
% Choose default command line output for T1P1
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes TlPl wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = TlPl_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function a_Callback(hObject, eventdata, handles)
% hObject handle to a (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of a as text
% str2double(get(hObject,'String')) returns contents of a as a double

% --- Executes during object creation, after setting all properties.
function a_CreateFcn(hObject, eventdata, handles)
% hObject handle to a (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function b_Callback(hObject, eventdata, handles)
% hObject handle to b (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of b as text
% str2double(get(hObject,'String')) returns contents of b as a double

% --- Executes during object creation, after setting all properties.
function b_CreateFcn(hObject, eventdata, handles)
% hObject handle to b (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function c_Callback(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of c as text
%       str2double(get(hObject,'String')) returns contents of c as a double

% --- Executes during object creation, after setting all properties.
function c_CreateFcn(hObject, eventdata, handles)
% hObject    handle to c (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function d_Callback(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of d as text
%       str2double(get(hObject,'String')) returns contents of d as a double

% --- Executes during object creation, after setting all properties.
function d_CreateFcn(hObject, eventdata, handles)
% hObject    handle to d (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function e_Callback(hObject, eventdata, handles)
% hObject    handle to e (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of e as text
% str2double(get(hObject,'String')) returns contents of e as a double

% --- Executes during object creation, after setting all properties.
function e_CreateFcn(hObject, eventdata, handles)
% hObject handle to e (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function f_Callback(hObject, eventdata, handles)
% hObject handle to f (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of f as text
% str2double(get(hObject,'String')) returns contents of f as a double

% --- Executes during object creation, after setting all properties.
function f_CreateFcn(hObject, eventdata, handles)
% hObject handle to f (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject handle to edit7 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
% str2double(get(hObject,'String')) returns contents of edit7 as a double

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit7 (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% hObject      handle to edit8 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%         str2double(get(hObject,'String')) returns contents of edit8 as a double

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit8 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit9_Callback(hObject, eventdata, handles)
% hObject      handle to edit9 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%         str2double(get(hObject,'String')) returns contents of edit9 as a double

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit9 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit10_Callback(hObject, eventdata, handles)
% hObject      handle to edit10 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%         str2double(get(hObject,'String')) returns contents of edit10 as a double

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit10 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit11_Callback(hObject, eventdata, handles)
% hObject      handle to edit11 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%         str2double(get(hObject,'String')) returns contents of edit11 as a double

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit11 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit12_Callback(hObject, eventdata, handles)
% hObject      handle to edit12 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%         str2double(get(hObject,'String')) returns contents of edit12 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function g_Callback(hObject, eventdata, handles)
% hObject    handle to g (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of g as text
%         str2double(get(hObject,'String')) returns contents of g as a double

% --- Executes during object creation, after setting all properties.
function g_CreateFcn(hObject, eventdata, handles)
% hObject    handle to g (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%         str2double(get(hObject,'String')) returns contents of edit14 as a double

% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in corregir.
function corregir_Callback(hObject, eventdata, handles)
Sol=str2double(get(handles.a,'string'));
if Sol > 22 && Sol < 22.5
    set(handles.a,'FontSize',12,'string','22.3','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.a,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.b,'string'));
if Sol > 11 && Sol < 11.2
    set(handles.b,'FontSize',12,'string','11.14','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.b,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.c,'string'));
if Sol > 21.5 && Sol < 22
    set(handles.c,'FontSize',12,'string','21.76','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.c,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.d,'string'));
if Sol > 30.5 && Sol < 31
    set(handles.d,'FontSize',12,'string','30.88','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.d,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.e,'string'));
if Sol > 33.3 && Sol < 33.6
    set(handles.e,'FontSize',12,'string','33.44','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.e,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.f,'string'));
if Sol > 19.8 && Sol < 20.05
    set(handles.f,'FontSize',12,'string','19.95','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.f,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.g,'string'));
if Sol > 4.95 && Sol < 5.1
    set(handles.g,'FontSize',12,'string','5.01','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.g,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.h,'string'));
if Sol > 784 && Sol < 804
    set(handles.h,'FontSize',12,'string','794','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.h,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.i,'string'));
```

```
if Sol > 141 && Sol < 141.5
    set(handles.i,'FontSize',12,'string','141.25','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.i,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.j,'string'));
if Sol > 6.1 && Sol < 6.5
    set(handles.j,'FontSize',12,'string','6.31','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.j,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.k,'string'));
if Sol > 6.5 && Sol < 7.2
    set(handles.k,'FontSize',12,'string','7','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.k,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.l,'string'));
if Sol > 66 && Sol < 68
    set(handles.l,'FontSize',12,'string','67','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.l,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.m,'string'));
if Sol > 102 && Sol < 104
    set(handles.m,'FontSize',12,'string','103','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.m,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
Sol=str2double(get(handles.n,'string'));
if Sol > -45 && Sol < -43
    set(handles.n,'FontSize',12,'string','-44','ForegroundColor','[0 1 0]', ...
        'FontWeight','bold');
else
    set(handles.n,'FontSize',12,'ForegroundColor','[1 0 0]', 'FontWeight','bold');
end
% hObject      handle to corregir (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% --- Executes on button press in soluciones.
function soluciones_Callback(hObject, eventdata, handles)
global intentos
intentos = intentos + 1;
if intentos == 1
    warndlg('Intenta resolverlo por ti mismo antes de pulsar esta casilla','No te rindas')
elseif intentos == 2
    warndlg('Un último intento por ti mismo antes de rendirte','No te rindas')
elseif intentos == 3
    T1P1S
    intentos = 0;
end
```

```
function edit15_Callback(hObject, eventdata, handles)
% hObject      handle to edit15 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%         str2double(get(hObject,'String')) returns contents of edit15 as a double

% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit15 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit16_Callback(hObject, eventdata, handles)
% hObject      handle to edit16 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit16 as text
%         str2double(get(hObject,'String')) returns contents of edit16 as a double

% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit16 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit17_Callback(hObject, eventdata, handles)
% hObject      handle to edit17 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit17 as text
%         str2double(get(hObject,'String')) returns contents of edit17 as a double

% --- Executes during object creation, after setting all properties.
```

```
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit18 as text
%         str2double(get(hObject,'String')) returns contents of edit18 as a double

% --- Executes during object creation, after setting all properties.
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit19 as text
%         str2double(get(hObject,'String')) returns contents of edit19 as a double

% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit20_Callback(hObject, eventdata, handles)
% hObject      handle to edit20 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit20 as text
%         str2double(get(hObject,'String')) returns contents of edit20 as a double

% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit20 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function h_Callback(hObject, eventdata, handles)
% hObject      handle to h (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of h as text
%         str2double(get(hObject,'String')) returns contents of h as a double

% --- Executes during object creation, after setting all properties.
function h_CreateFcn(hObject, eventdata, handles)
% hObject      handle to h (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function i_Callback(hObject, eventdata, handles)
% hObject      handle to i (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of i as text
%         str2double(get(hObject,'String')) returns contents of i as a double
```

```
% --- Executes during object creation, after setting all properties.
function i_CreateFcn(hObject, eventdata, handles)
% hObject    handle to i (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function j_Callback(hObject, eventdata, handles)
% hObject    handle to j (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of j as text
%         str2double(get(hObject,'String')) returns contents of j as a double

% --- Executes during object creation, after setting all properties.
function j_CreateFcn(hObject, eventdata, handles)
% hObject    handle to j (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function l_Callback(hObject, eventdata, handles)
% hObject    handle to l (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of l as text
%         str2double(get(hObject,'String')) returns contents of l as a double

% --- Executes during object creation, after setting all properties.
function l_CreateFcn(hObject, eventdata, handles)
% hObject    handle to l (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

end

```
function edit25_Callback(hObject, eventdata, handles)
% hObject      handle to l (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of l as text
%         str2double(get(hObject,'String')) returns contents of l as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit25_CreateFcn(hObject, eventdata, handles)
% hObject      handle to l (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function m_Callback(hObject, eventdata, handles)
% hObject      handle to m (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of m as text
%         str2double(get(hObject,'String')) returns contents of m as a double
```

```
% --- Executes during object creation, after setting all properties.
function m_CreateFcn(hObject, eventdata, handles)
% hObject      handle to m (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function n_Callback(hObject, eventdata, handles)
% hObject      handle to n (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of n as text
```

```
%          str2double(get(hObject,'String')) returns contents of n as a double

% --- Executes during object creation, after setting all properties.
function n_CreateFcn(hObject, eventdata, handles)
% hObject    handle to n (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function k_Callback(hObject, eventdata, handles)
% hObject    handle to k (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of k as text
%         str2double(get(hObject,'String')) returns contents of k as a double

% --- Executes during object creation, after setting all properties.
function k_CreateFcn(hObject, eventdata, handles)
% hObject    handle to k (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

T1C

```
function varargout = T1C(varargin)
% T1C MATLAB code for T1C.fig
%     T1C, by itself, creates a new T1C or raises the existing
%     singleton*.
%
%     H = T1C returns the handle to a new T1C or the handle to
%     the existing singleton*.
%
%     T1C('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in T1C.M with the given input arguments.
%
%     T1C('Property','Value',...) creates a new T1C or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before T1C_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to T1C_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help T1C

% Last Modified by GUIDE v2.5 03-Feb-2016 22:34:00

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',   @T1C_OpeningFcn, ...
                  'gui_OutputFcn',    @T1C_OutputFcn, ...
                  'gui_LayoutFcn',    [] , ...
                  'gui_Callback',     []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before T1C is made visible.
function T1C_OpeningFcn(hObject, eventdata, handles, varargin)

axis off;

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes T1C wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = T1C_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in siguiente.
function siguiente_Callback(hObject, eventdata, handles)
set(handles.r1,'Value',0);
set(handles.r2,'Value',0);
set(handles.r3,'Value',0);
set(handles.r4,'Value',0);
set(handles.r1,'Enable','on');
set(handles.r2,'Enable','on');
set(handles.r3,'Enable','on');
set(handles.r4,'Enable','on');
global num
num = get(handles.numero,'string');
num = str2double(num);
num=num+1;
set(handles.numero,'String',num);
if num>=1 && num<=4
    set(handles.siguiente,'Enable','Inactive');
else
    set(handles.siguiente,'Enable','on');
end
if num == 1
    %Poner las preguntas
    set(handles.pregunta,'String','¿Que tipos de modo de explotación existen?');
    set(handles.r1,'String','Simplex');
    set(handles.r2,'String','Duplex');
    set(handles.r3,'String','Semiduplex');
    set(handles.r4,'string','Todas son correctas');
elseif num == 2
    set(handles.pregunta,'String','¿Cuántas bandas de frecuencia existen?');
    set(handles.r1,'String','8');
    set(handles.r2,'String','10');
    set(handles.r3,'String','9');
    set(handles.r4,'string','7');
elseif num == 3
    set(handles.pregunta,'String','¿Cuál no es un modo de propagación de las ondas?');
    set(handles.r1,'String','De superficie');
    set(handles.r2,'String','Ionosférica');
    set(handles.r3,'String','Extraterrestre');
    set(handles.r4,'string','Espacial');
elseif num == 4
    set(handles.pregunta,'String','¿Qué fenómenos físicos condicionan la propagación?');
    set(handles.r1,'String','Reflexión, refracción, difracción, absorción y expulsión');
    set(handles.r2,'String','Reflexión, refracción, difracción, dispersión y absorción');
    set(handles.r3,'String','Reflexión, refracción, multifracción y absorción');
    set(handles.r4,'string','Ninguna es correcta');
```

```

else
    set(handles.pregunta, 'String', 'Ha terminado el Test. Presione finalizar');
    set(handles.r1, 'String', ' ');
    set(handles.r2, 'String', ' ');
    set(handles.r3, 'String', ' ');
    set(handles.r4, 'string', ' ');
end

% --- Executes on button press in fin.
function fin_Callback(hObject, eventdata, handles)
fin = questdlg('¿Desea repetir el test?', 'Ha terminado...', 'Si', 'No', 'No');
if strcmp(fin, 'No')
    close T1C
    clc
else
    close T1C
    T1C
end

% --- Executes on button press in r1.
function r1_Callback(hObject, eventdata, handles)
global num
valor1 = get(hObject, 'Value');
if valor1 == 1
    set(handles.r2, 'Value', 0);
    set(handles.r3, 'Value', 0);
    set(handles.r4, 'Value', 0);
    if num > 0 && num < 5

        end
end

end

function r2_Callback(hObject, eventdata, handles)
global num
valor2 = get(hObject, 'Value');
if valor2 == 1
    set(handles.r1, 'Value', 0);
    set(handles.r3, 'Value', 0);
    set(handles.r4, 'Value', 0);
    if num > 0 && num < 5
        if num == 4
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end

end

function r3_Callback(hObject, eventdata, handles)
global num
valor3 = get(hObject, 'Value');
if valor3 == 1
    set(handles.r2, 'Value', 0);
    set(handles.r1, 'Value', 0);
    set(handles.r4, 'Value', 0);
    if num > 0 && num < 5
        if num == 2

```

```
        set(handles.siguiente, 'Enable', 'on');
    elseif num == 3
        set(handles.siguiente, 'Enable', 'on');
    else
        set(handles.siguiente, 'Enable', 'inactive');
    end
end
end

function r4_Callback(hObject, eventdata, handles)
global num
valor4 = get(hObject, 'Value');
if valor4 == 1
    set(handles.r2, 'Value', 0);
    set(handles.r3, 'Value', 0);
    set(handles.r1, 'Value', 0);
    if num > 0 && num < 5
        if num == 1
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end
end
```

T2P

```
function varargout = T2P(varargin)
% T2P MATLAB code for T2P.fig
%
%   T2P, by itself, creates a new T2P or raises the existing
%   singleton*.
%
%   H = T2P returns the handle to a new T2P or the handle to
%   the existing singleton*.
%
%   T2P('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in T2P.M with the given input arguments.
%
%   T2P('Property','Value',...) creates a new T2P or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before T2P_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to T2P_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help T2P

% Last Modified by GUIDE v2.5 16-Feb-2016 19:06:05

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @T2P_OpeningFcn, ...
                  'gui_OutputFcn',  @T2P_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before T2P is made visible.
function T2P_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to T2P (see VARARGIN)
background=imread('radiocomunicaciones.jpg');
axis off
imshow(background);
% Choose default command line output for T2P
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes T2P wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = T2P_OutputFcn(hObject, eventdata, handles)
% varargout    cell array for returning output args (see VARARGOUT);
% hObject      handle to figure
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton3.

% --- Executes on button press in mod.
function mod_Callback(hObject, eventdata, handles)
close(T2P)
Modulaciones

% --- Executes on button press in cuestiones.
function cuestiones_Callback(hObject, eventdata, handles)
T2Pp

% --- Executes on selection change in modulacion.
function modulacion_Callback(hObject, eventdata, handles)
% hObject      handle to modulacion (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns modulacion contents as
cell array
%           contents{get(hObject,'Value')} returns selected item from modulacion

% --- Executes during object creation, after setting all properties.
function modulacion_CreateFcn(hObject, eventdata, handles)
% hObject      handle to modulacion (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in ir.
function ir_Callback(hObject, eventdata, handles)
```

```
ir = get(handles.modulacion, 'Value');  
switch ir  
    case 2  
        mod_ana  
        close(T2P)  
    case 3  
        close(T2P)  
        ASK  
    case 4  
        close(T2P)  
        PSK  
    case 5  
        close(T2P)  
        FSK  
    case 6  
        msgbox('En proceso de desarrollo');  
end  
clc
```

Mod_ana

```
function varargout = mod_ana(varargin)
% MOD_ANA MATLAB code for mod_ana.fig
%   MOD_ANA, by itself, creates a new MOD_ANA or raises the existing
%   singleton*.
%
%   H = MOD_ANA returns the handle to a new MOD_ANA or the handle to
%   the existing singleton*.
%
%   MOD_ANA('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in MOD_ANA.M with the given input arguments.
%
%   MOD_ANA('Property','Value',...) creates a new MOD_ANA or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before mod_ana_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to mod_ana_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help mod_ana

% Last Modified by GUIDE v2.5 02-Mar-2016 20:47:53

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @mod_ana_OpeningFcn, ...
                  'gui_OutputFcn',  @mod_ana_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before mod_ana is made visible.
function mod_ana_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin   command line arguments to mod_ana (see VARARGIN)

% Choose default command line output for mod_ana
handles.output = hObject;

% Update handles structure
```

```
guidata(hObject, handles);

% UIWAIT makes mod_ana wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = mod_ana_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on slider movement.
function frecuencia_Callback(hObject, eventdata, handles)
hz = get(hObject, 'Value')+25;
set(handles.hz, 'String', hz);

% --- Executes during object creation, after setting all properties.
function frecuencia_CreateFcn(hObject, eventdata, handles)
% hObject handle to frecuencia (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject, 'BackgroundColor'),
get(0, 'defaultUicontrolBackgroundColor'))
set(hObject, 'BackgroundColor', [.9 .9 .9]);
end

% --- Executes on button press in actualizar.
function actualizar_Callback(hObject, eventdata, handles)
% Grafica 2
S2 = get(handles.am2, 'value');
if S2 == 1
Ac = 1;
a = 1;
fc = get(handles.frecuencia, 'Value');
fs = 100*3;
t = 0:1/fs:1;
c_t = Ac*cos(2*pi*fc*t);
m_t = 2*sin(2*pi*3*t + pi/4);
e_t = Ac*(1+a*m_t);
s_t = e_t.*c_t;
s_t=s_t/Ac;
axes(handles.axes2)
plot(t,s_t,t,e_t,'--r',t,-(e_t),'--k');
legend('Señal modulada','Señal portadora')
grid on;
title('AM')
xlabel('Tiempo(s)')
e_t = Ac*(1+a*m_t);
```

```
s_t = e_t.*c_t;
s_t=s_t/Ac;
S = fft(s_t)/length(s_t);
k = 0:length(S)/2-1;
fplot = k*fs/length(s_t);
Splot = S(1:end/2);
Splot(2:end) = 2*Splot(2:end);
axes(handles.axes3)
x=get(handles.frecuencia, 'Value');
x=1.5*x;
plot(fplot,abs(Splot));
axis([0 x 0 2]);
xlabel('Frecuencia (Hz)')
grid on
title('AM CONVENCIONAL')
```

```
else
```

```
S2 = get(handles.fm2, 'Value');
if S2 == 1
    fc = get(handles.frecuencia, 'Value');
    fs = 40*fc;
    t = 0:1/(40*fc):1;
    m_t = 2*sin(2*pi*3*t);
    kf = fc/4;
    y_fm = fmmmod(m_t,fc,fs,kf);
    axes(handles.axes2)
    plot(t,y_fm,t,m_t,'--r');
    legend('Señal modulada','Señal portadora')
    grid on;
    title('FM')
    % frec.
    S = fft(y_fm)/length(y_fm);
    k = 0:length(S)/2-1;
    fplot = k*fs/length(y_fm);
    Splot = S(1:end/2);
    Splot(2:end) = 2*Splot(2:end);
    axes(handles.axes3)
    plot(fplot,abs(Splot));
    axis([0 100 0 0.8]);
    xlabel('Frecuencia (Hz)')
    grid on
    title('FM')
```

```
else
```

```
S2 = get(handles.pm2, 'Value');
if S2 == 1
    fc = get(handles.frecuencia, 'Value')+25;
    fs = 40*fc;
    t = 0:1/fs:1-1/fs;
    m_t = 2*sin(2*pi*3*t);
    Ap = pi;
    y_pm = pmmod(m_t,fc,fs,Ap);
    axes(handles.axes2)
    plot(t,y_pm,t,m_t,'--r');
    axis([0 fc/75 -2 2])
    legend('Señal modulada','Señal portadora')
    grid on;
    title('PM')
    % frec.
    S = fft(y_pm)/length(y_pm);
    k = 0:length(S)/2-1;
    fplot = k*fs/length(y_pm);
```

```

        Splot = S(1:end/2);
        Splot(2:end) = 2*Splot(2:end);
        axes(handles.axes3)
        plot(fplot,abs(Splot));
        axis([0 115 0 0.5]);
        xlabel('Frecuencia (Hz)')
        grid on
        title('PM')
    else
        S2 = get(handles.dbl2, 'Value');
        if S2 == 1
            Ac = 1;
            a = 1;
            fc = get(handles.frecuencia, 'Value');
            fs = 100*3;
            t = 0:1/fs:1;
            c_t = Ac*cos(2*pi*fc*t);
            m_t = 2*sin(2*pi*3*t + pi/4);
            e_t = Ac*(1+a*m_t);
            s_t = e_t.*c_t;
            s_t=s_t/Ac;
            axes(handles.axes2)
            plot(t,s_t,t,e_t,'--r',t,-(e_t),'--k');
            legend('Señal modulada','Señal portadora')
            grid on;
            title('AM')
            xlabel('Tiempo(s)')
            s_t = m_t.*c_t;
            s_t=s_t/Ac;
            S = fft(s_t)/length(s_t);
            k = 0:length(S)/2-1;
            fplot = k*fs/length(s_t);
            Splot = S(1:end/2);
            Splot(2:end) = 2*Splot(2:end);
            axes(handles.axes3)
            x=get(handles.frecuencia, 'Value');
            x=1.5*x;
            plot(fplot,abs(Splot));
            axis([0 x 0 2]);
            title('DBL')
            xlabel('Frecuencia (Hz)')
            grid on
        end
    end
end
end
end
end

```

```

% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
clc
close(mod_ana)
mod_ana

```

```

% --- Executes on button press in am2.
function am2_Callback(hObject, eventdata, handles)
S=get(hObject, 'Value');
if S == 1
    set(handles.dbl2, 'Value', 0);
    set(handles.fm2, 'Value', 0);

```

```
        set(handles.pm2, 'Value', 0);
end

% --- Executes on button press in fm2.
function fm2_Callback(hObject, eventdata, handles)
S=get(hObject, 'Value');
if S == 1
    set(handles.dbl2, 'Value', 0);
    set(handles.am2, 'Value', 0);
    set(handles.pm2, 'Value', 0);
end

% --- Executes on button press in pm2.
function pm2_Callback(hObject, eventdata, handles)
S=get(hObject, 'Value');
if S == 1
    set(handles.dbl2, 'Value', 0);
    set(handles.am2, 'Value', 0);
    set(handles.fm2, 'Value', 0);
end

% --- Executes on button press in dbl2.
function dbl2_Callback(hObject, eventdata, handles)
S=get(hObject, 'Value');
if S == 1
    set(handles.am2, 'Value', 0);
    set(handles.fm2, 'Value', 0);
    set(handles.pm2, 'Value', 0);
end
```

ASK

```
function varargout = ASK(varargin)
% ASK MATLAB code for ASK.fig
%     ASK, by itself, creates a new ASK or raises the existing
%     singleton*.
%
%     H = ASK returns the handle to a new ASK or the handle to
%     the existing singleton*.
%
%     ASK('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in ASK.M with the given input arguments.
%
%     ASK('Property','Value',...) creates a new ASK or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before ASK_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to ASK_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ASK

% Last Modified by GUIDE v2.5 17-Feb-2016 22:25:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @ASK_OpeningFcn, ...
                  'gui_OutputFcn',  @ASK_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before ASK is made visible.
function ASK_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to ASK (see VARARGIN)
set(handles.dos,'Value',1)
% Choose default command line output for ASK
handles.output = hObject;

% Update handles structure
```

```
guidata(hObject, handles);

% UIWAIT makes ASK wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = ASK_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in ask_.
function ask__Callback(hObject, eventdata, handles)
global c
global signal
ask=c.*signal;
axes(handles.axes3)
plot(ask);
axis([1 512 -8 8])
grid on
title('Señal modulada');
ppp=gca;
ppp.XTick = [0 32 64 96 128 160 192 224 256 288 320 352 384 416 448 480 512];
ppp.XTickLabel = [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16];
ppp.YTick = [-7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7];

% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
close ASK
ASK

% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject handle to axes1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes1

% --- Executes on slider movement.
function fc_Callback(hObject, eventdata, handles)
f = 1/11;
t=1:512;
global c
c = sin(2*pi*f*t);
axes(handles.axes1)
plot(t,c);
axis([1 512 -1 1])
title('Señal portadora');
ppp=gca;
```

```

ppp.XTick = [0 32 64 96 128 160 192 224 256 288 320 352 384 416 448 480 512];
ppp.XTickLabel = [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16];
xlabel('Periodos de Símbolo (Ts)')

% --- Executes during object creation, after setting all properties.
function fc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on button press in gb.
function gb_Callback(hObject, eventdata, handles)
global signal
chip=32;
total=512;
no_symbols=ceil(total/chip);
symbols=rand(1,no_symbols);
n_s = get(handles.tres,'Value');
if n_s == 1
    symbols(find(symbols>=0.75))=3;
    symbols(find(symbols>=0.25 & symbols<0.5))=7;
    symbols(find(symbols>=0.5 & symbols<0.75))=3;
    symbols(find(symbols<0.25))=0;
else
    n_s = get(handles.cuatro,'Value');
    if n_s == 1
        symbols(find(symbols >= 0.825))=7;
        symbols(find(symbols >= 0.75 & symbols < 0.825))=6;
        symbols(find(symbols >= 0.625 & symbols < 0.75))=5;
        symbols(find(symbols >= 0.5 & symbols < 0.625))=4;
        symbols(find(symbols >= 0.375 & symbols < 0.5))=3;
        symbols(find(symbols >= 0.25 & symbols < 0.375))=2;
        symbols(find(symbols >= 0.125 & symbols < 0.25))=1;
        symbols(find(symbols < 0.125))=0;
    else
        symbols(find(symbols>=0.5))=4;
        symbols(find(symbols<0.5))=0;
    end
end

signal= zeros(1,total);
for i = 1:chip
    signal(i:chip:end)=symbols;
end

axes(handles.axes2)
plot(signal);
axis([1 512 0 8])
grid on
title('Simbolos');
ppp=gca;

```

```
ppp.XTick = [0 32 64 96 128 160 192 224 256 288 320 352 384 416 448 480 512];
ppp.XTickLabel = [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16];
ppp.YTick = [0 1 2 3 4 5 6 7];
ppp.YTickLabel = [ ' ' ]
```

```
% --- Executes on button press in dos.
```

```
function dos_Callback(hObject, eventdata, handles)
ac = get(hObject, 'Value');
if ac == 1
    set(handles.tres, 'Value', 0)
    set(handles.cuatro, 'Value', 0)
elseif ac == 0
    set(hObject, 'Value', 1)
end
```

```
% --- Executes on button press in tres.
```

```
function tres_Callback(hObject, eventdata, handles)
ac = get(hObject, 'Value');
if ac == 1
    set(handles.dos, 'Value', 0)
    set(handles.cuatro, 'Value', 0)
elseif ac == 0
    set(hObject, 'Value', 1)
end
```

```
% --- Executes on button press in cuatro.
```

```
function cuatro_Callback(hObject, eventdata, handles)
ac = get(hObject, 'Value');
if ac == 1
    set(handles.tres, 'Value', 0)
    set(handles.dos, 'Value', 0)
elseif ac == 0
    set(hObject, 'Value', 1)
end
```

FSK

```
function varargout = FSK(varargin)
% FSK MATLAB code for FSK.fig
%     FSK, by itself, creates a new FSK or raises the existing
%     singleton*.
%
%     H = FSK returns the handle to a new FSK or the handle to
%     the existing singleton*.
%
%     FSK('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in FSK.M with the given input arguments.
%
%     FSK('Property','Value',...) creates a new FSK or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before FSK_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to FSK_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help FSK

% Last Modified by GUIDE v2.5 17-Feb-2016 22:44:41

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',  @FSK_OpeningFcn, ...
                  'gui_OutputFcn',   @FSK_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before FSK is made visible.
function FSK_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to FSK (see VARARGIN)
set(handles.dos,'Value',1)
% Choose default command line output for FSK
handles.output = hObject;

% Update handles structure
```

```
guidata(hObject, handles);

% UIWAIT makes FSK wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = FSK_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in fsk_.
function fsk__Callback(hObject, eventdata, handles)
global c
axes(handles.axes3)
plot(c)
axis([1 512 -1 1])
grid on
title('Señal Modulada')
ppp=gca;
ppp.XTick = [0 32 64 96 128 160 192 224 256 288 320 352 384 416 448 480 512];
ppp.XTickLabel = [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16];

% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
close FSK
clear all
FSK

% --- Executes on button press in senal.
function senal_Callback(hObject, eventdata, handles)
global c
global f
global t
f = 1/32;
t=1:512;
c = cos(2*pi*f*t);
axes(handles.axes1)
plot(t,c)
axis([1 512 -1 1])
grid on
title('Señal Portadora')
xlabel('Periodos de símbolos (Ts)')
ppp=gca;
ppp.XTick = [0 32 64 96 128 160 192 224 256 288 320 352 384 416 448 480 512];
ppp.XTickLabel = [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16];

% --- Executes on button press in bits.
function bits_Callback(hObject, eventdata, handles)
global c
f = 1/32;
```

```

t=1:512;
chip=32;
total=512;
no_symbols=ceil(total/chip);
symbols=rand(1,no_symbols);
n_s = get(handles.tres, 'Value');
if n_s == 1
    symbols(find(symbols>=0.75))=8;
    symbols(find(symbols>=0.25 & symbols<0.5))=4;
    symbols(find(symbols>=0.5 & symbols<0.75))=2;
    symbols(find(symbols<0.25))=1;
else
    n_s = get(handles.cuatro, 'Value');
    if n_s == 1
        symbols(find(symbols >= 0.825))=8;
        symbols(find(symbols >= 0.75 & symbols < 0.825))=7;
        symbols(find(symbols >= 0.625 & symbols < 0.75))=6;
        symbols(find(symbols >= 0.5 & symbols < 0.625))=5;
        symbols(find(symbols >= 0.375 & symbols < 0.5))=4;
        symbols(find(symbols >= 0.25 & symbols < 0.375))=3;
        symbols(find(symbols >= 0.125 & symbols < 0.25))=2;
        symbols(find(symbols < 0.125))=1;
    else
        symbols(find(symbols>=0.5))=8;
        symbols(find(symbols<0.5))=2;
    end
end
fh= zeros(1,total);

for i = 1:chip
    fh(i:chip:end)=symbols;
end
axes(handles.axes2)
plot(fh)
axis([1 512 0 8.2])
grid on
c = sin(2*pi*f*t.*fh);
title('Simbolos')
ppp=gca;
ppp.XTick = [0 32 64 96 128 160 192 224 256 288 320 352 384 416 448 480 512];
ppp.XTickLabel = [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16];
ppp.YTick = [0 1 2 3 4 5 6 7 8 9 10];
ppp.YTickLabel = [' '];

% --- Executes on button press in dos.
function dos_Callback(hObject, eventdata, handles)
ac = get(hObject, 'Value');
if ac == 1
    set(handles.tres, 'Value', 0)
    set(handles.cuatro, 'Value', 0)
elseif ac == 0
    set(hObject, 'Value', 1)
end

% --- Executes on button press in tres.
function tres_Callback(hObject, eventdata, handles)
ac = get(hObject, 'Value');
if ac == 1
    set(handles.dos, 'Value', 0)

```

```
        set(handles.cuatro, 'Value', 0)
elseif ac == 0
    set(hObject, 'Value', 1)
end

% --- Executes on button press in cuatro.
function cuatro_Callback(hObject, eventdata, handles)
ac = get(hObject, 'Value');
if ac == 1
    set(handles.tres, 'Value', 0)
    set(handles.dos, 'Value', 0)
elseif ac == 0
    set(hObject, 'Value', 1)
end
```

PSK

```
function varargout = PSK(varargin)
% PSK MATLAB code for PSK.fig
%     PSK, by itself, creates a new PSK or raises the existing
%     singleton*.
%
%     H = PSK returns the handle to a new PSK or the handle to
%     the existing singleton*.
%
%     PSK('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in PSK.M with the given input arguments.
%
%     PSK('Property','Value',...) creates a new PSK or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before PSK_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to PSK_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help PSK

% Last Modified by GUIDE v2.5 17-Feb-2016 23:05:48

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @PSK_OpeningFcn, ...
                  'gui_OutputFcn',  @PSK_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before PSK is made visible.
function PSK_OpeningFcn(hObject, ~, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to PSK (see VARARGIN)
set(handles.dos,'Value',1)
% Choose default command line output for PSK
handles.output = hObject;

% Update handles structure
```

```
guidata(hObject, handles);

% UIWAIT makes PSK wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = PSK_OutputFcn(~, ~, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in psk_.
function psk__Callback(~, ~, handles)
global c
axes(handles.axes3)
plot(c)
axis([1 512 -1 1])
grid on
title('Señal modulada')
ppp=gca;
ppp.XTick = [0 32 64 96 128 160 192 224 256 288 320 352 384 416 448 480 512];
ppp.XTickLabel = [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16];

% --- Executes on button press in reset.
function reset_Callback(~, ~, ~)
close PSK
PSK

% --- Executes on button press in signal.
function signal_Callback(~, ~, handles)
global c
global f
global t
f = 1/32;
t=1:512;
c = cos(2*pi*f*t);
axes(handles.axes1)
plot(t,c)
axis([1 512 -1 1])
grid on
title('Señal portadora')
xlabel('Periodos de símbolo (Ts)')
ppp=gca;
ppp.XTick = [0 32 64 96 128 160 192 224 256 288 320 352 384 416 448 480 512];
ppp.XTickLabel = [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16];

% --- Executes on button press in bits.
function bits_Callback(~, ~, handles)
global c
global f
t=1:512;
chip=32;
total=512;
no_symbols=ceil(total/chip);
```

```

symbols=rand(1,no_symbols);
n_s = get(handles.tres, 'Value');
if n_s == 1
    symbols(find(symbols>=0.75))=2*pi;
    symbols(find(symbols>=0.25 & symbols<0.5))=pi;
    symbols(find(symbols>=0.5 & symbols<0.75))=1.5*pi;
    symbols(find(symbols<0.25))=0.5*pi;
else
    n_s = get(handles.cuatro, 'Value');
    if n_s == 1
        symbols(find(symbols >= 0.825))=2*pi;
        symbols(find(symbols >= 0.75 & symbols < 0.825))=pi/4;
        symbols(find(symbols >= 0.625 & symbols < 0.75))=pi/2;
        symbols(find(symbols >= 0.5 & symbols < 0.625))=3*pi/4;
        symbols(find(symbols >= 0.375 & symbols < 0.5))=pi;
        symbols(find(symbols >= 0.25 & symbols < 0.375))=5*pi/4;
        symbols(find(symbols >= 0.125 & symbols < 0.25))=3*pi/2;
        symbols(find(symbols < 0.125))=7*pi/4;
    else
        symbols(find(symbols>=0.5))=2*pi;
        symbols(find(symbols<0.5))=pi;
    end
end
ph= zeros(1,total);

for i = 1:chip
    ph(i:chip:end)=symbols;
end
axes(handles.axes2)
plot(ph)
axis([1 512 0 2*pi+0.2])
grid on
ph(find(ph==1))=pi/2;
c = sin(2*pi*f*t+ph);
title('Símbolos')
ppp=gca;
ppp.XTick = [0 32 64 96 128 160 192 224 256 288 320 352 384 416 448 480 512];
ppp.XTickLabel = [0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16];
ppp.YTick = [0 pi/4 pi/2 3*pi/4 pi 5*pi/4 3*pi/2 7*pi/4 2*pi];
ppp.YTickLabel = [' ']; %#ok<NBRAK>

% --- Executes on button press in dos.
function dos_Callback(hObject, ~, handles)
ac = get(hObject, 'Value');
if ac == 1
    set(handles.tres, 'Value', 0)
    set(handles.cuatro, 'Value', 0)
elseif ac == 0
    set(hObject, 'Value', 1)
end

% --- Executes on button press in tres.
function tres_Callback(hObject, ~, handles) %#ok<DEFNU>
ac = get(hObject, 'Value');
if ac == 1
    set(handles.dos, 'Value', 0)
    set(handles.cuatro, 'Value', 0)
elseif ac == 0

```

```
        set(hObject,'Value',1)
end

% --- Executes on button press in cuatro.
function cuatro_Callback(hObject, ~, handles) %#ok<DEFNU>
ac = get(hObject,'Value');
if ac == 1
    set(handles.tres,'Value',0)
    set(handles.dos,'Value',0)
elseif ac == 0
    set(hObject,'Value',1)
end
```

T2C

```
function varargout = T2C(varargin)
% T2C MATLAB code for T2C.fig
%     T2C, by itself, creates a new T2C or raises the existing
%     singleton*.
%
%     H = T2C returns the handle to a new T2C or the handle to
%     the existing singleton*.
%
%     T2C('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in T2C.M with the given input arguments.
%
%     T2C('Property','Value',...) creates a new T2C or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before T2C_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to T2C_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help T2C

% Last Modified by GUIDE v2.5 01-Mar-2016 20:11:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @T2C_OpeningFcn, ...
                  'gui_OutputFcn',  @T2C_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before T2C is made visible.
function T2C_OpeningFcn(hObject, eventdata, handles, varargin)

axis off;

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes T2C wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = T2C_OutputFcn(hObject, eventdata, handles)
% varargout    cell array for returning output args (see VARARGOUT);
% hObject     handle to figure
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in siguiente.
function siguiente_Callback(hObject, eventdata, handles)
set(handles.r1, 'Value', 0);
set(handles.r2, 'Value', 0);
set(handles.r3, 'Value', 0);
set(handles.r4, 'Value', 0);
set(handles.r1, 'Enable', 'on');
set(handles.r2, 'Enable', 'on');
set(handles.r3, 'Enable', 'on');
set(handles.r4, 'Enable', 'on');
global num
num = get(handles.numero, 'string');
num = str2double(num);
num=num+1;
set(handles.numero, 'String', num);
if num>=1 && num<=8
    set(handles.siguiente, 'Enable', 'Inactive');
else
    set(handles.siguiente, 'Enable', 'on');
end
if num == 1
    %Poner las preguntas
    set(handles.pregunta, 'String', 'La modulación es la modificación de alguna de
las características de una señal de información en función de una portadora');
    set(handles.r1, 'String', 'Falso');
    set(handles.r2, 'String', 'Verdadero');
    set(handles.r3, 'String', ' ');
    set(handles.r4, 'string', ' ');
elseif num == 2
    set(handles.pregunta, 'String', '¿Cual de las siguientes no es una modulación
digital?');
    set(handles.r1, 'String', 'OFDM');
    set(handles.r2, 'String', 'QAM');
    set(handles.r3, 'String', 'ASK');
    set(handles.r4, 'string', 'AM');
elseif num == 3
    set(handles.pregunta, 'String', 'Con AM convencional se desperdicia potencia,
¿cómo se puede solucionar?');
    set(handles.r1, 'String', 'Transmitiendo con más potencia');
    set(handles.r2, 'String', 'Suprimiendo la componente de la portadora');
    set(handles.r3, 'String', 'Suprimiendo la componente de la información');
    set(handles.r4, 'string', 'Ninguna de las anteriores es correcta');
elseif num == 4
    set(handles.pregunta, 'String', 'Las modulaciones angulares permiten discriminar
más fácilmente el ruido y las interferencias que las de amplitud');
    set(handles.r1, 'String', 'Falso');
    set(handles.r2, 'String', 'Verdadero');
    set(handles.r3, 'String', ' ');
    set(handles.r4, 'string', ' ');
```

```
elseif num == 5
    set(handles.pregunta,'String','Las señales PM y FM sólo se pueden distinguir
');
    set(handles.r1,'String','Porque las señales PM siempre se envían con más
potencia que las FM');
    set(handles.r2,'String','Porque las señales FM suelen tener mayor amplitud');
    set(handles.r3,'String','Cuando se conoce m(t)');
    set(handles.r4,'string','Cuando se conoce la fuente de transmisión');
elseif num == 6
    set(handles.pregunta,'String','El objetivo de las modulaciones digitales es el
de convertir los bits en señales adecuadas para su transmisión');
    set(handles.r1,'String','Verdadero');
    set(handles.r2,'String','Falso');
    set(handles.r3,'String',' ');
    set(handles.r4,'string',' ');
elseif num == 7
    set(handles.pregunta,'String','¿Cuál es una modulación por desplazamiento en
fase?');
    set(handles.r1,'String','ASK');
    set(handles.r2,'String','OFDM');
    set(handles.r3,'String','PSK');
    set(handles.r4,'string','QAM');
elseif num == 8
    set(handles.pregunta,'String','¿Cuál de los siguientes sistemas no utiliza
OFDM?');
    set(handles.r1,'String','La televisión digital terrestre ');
    set(handles.r2,'String','La radio digital de baja frecuencia DRM ');
    set(handles.r3,'String','El sistema de transmisión inalámbrica de datos WiMAX
');
    set(handles.r4,'string','El sistema de telefonía móvil 3.8G-HSPA+');
else
    set(handles.pregunta,'String','Ha terminado el Test. Presione finalizar');
    set(handles.r1,'String',' ');
    set(handles.r2,'String',' ');
    set(handles.r3,'String',' ');
    set(handles.r4,'string',' ');
end

% --- Executes on button press in fin.
function fin_Callback(hObject, eventdata, handles)
fin = questdlg('¿Desea repetir el test?','Ha terminado...','Si','No','No');
if strcmp(fin,'No')
    close T2C
    clc
    clear all
else
    close T2C
    T2C
end

% --- Executes on button press in r1.
function r1_Callback(hObject, eventdata, handles)
global num
valor1 = get(hObject,'Value');
if valor1 == 1
    set(handles.r2,'Value',0);
    set(handles.r3,'Value',0);
    set(handles.r4,'Value',0);
    if num > 0 && num < 9
        if num == 1
```

```
        set(handles.siguiente, 'Enable', 'on');
    elseif num == 6
        set(handles.siguiente, 'Enable', 'on');
    else
        set(handles.siguiente, 'Enable', 'inactive');
    end
end
end
```

```
function r2_Callback(hObject, eventdata, handles)
global num
valor2 = get(hObject, 'Value');
if valor2 == 1
    set(handles.r1, 'Value', 0);
    set(handles.r3, 'Value', 0);
    set(handles.r4, 'Value', 0);
    if num > 0 && num < 9
        if num == 3
            set(handles.siguiente, 'Enable', 'on');
        elseif num == 4
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end
end
```

```
function r3_Callback(hObject, eventdata, handles)
global num
valor3 = get(hObject, 'Value');
if valor3 == 1
    set(handles.r2, 'Value', 0);
    set(handles.r1, 'Value', 0);
    set(handles.r4, 'Value', 0);
    if num > 0 && num < 9
        if num == 5
            set(handles.siguiente, 'Enable', 'on');
        elseif num == 7
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end
end
```

```
function r4_Callback(hObject, eventdata, handles)
global num
valor4 = get(hObject, 'Value');
if valor4 == 1
    set(handles.r2, 'Value', 0);
    set(handles.r3, 'Value', 0);
    set(handles.r1, 'Value', 0);
    if num > 0 && num < 9
        if num == 2
            set(handles.siguiente, 'Enable', 'on');
        elseif num == 8
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end
end
```

Antennarad

```
function varargout = antennarad(varargin)
% ANTENNARAD MATLAB code for antennarad.fig
%   ANTENNARAD, by itself, creates a new ANTENNARAD or raises the existing
%   singleton*.
%
%   H = ANTENNARAD returns the handle to a new ANTENNARAD or the handle to
%   the existing singleton*.
%
%   ANTENNARAD('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in ANTENNARAD.M with the given input arguments.
%
%   ANTENNARAD('Property','Value',...) creates a new ANTENNARAD or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before antennarad_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to antennarad_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help antennarad

% Last Modified by GUIDE v2.5 02-Mar-2016 22:04:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @antennarad_OpeningFcn, ...
                  'gui_OutputFcn',  @antennarad_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before antennarad is made visible.
function antennarad_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to antennarad (see VARARGIN)
set(handles.dosd,'Value',1)
set(handles.tresd,'Value',0)
set(handles.azimuth,'Value',1)
set(handles.mhz,'Value',1)
% Choose default command line output for antennarad
```

```
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes antennarad wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = antennarad_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in dosd.
function dosd_Callback(hObject, eventdata, handles)
active = get(hObject, 'Value');
if active == 1
    set(handles.tresd, 'Value', 0)
    set(handles.azimuth, 'Enable', 'on', 'Value', 1)
    set(handles.elevation, 'Enable', 'on', 'Value', 0)
    set(handles.angle, 'Enable', 'on')
elseif active == 0
    set(handles.tresd, 'Value', 1)
    set(handles.azimuth, 'Enable', 'inactive', 'Value', 0)
    set(handles.elevation, 'Enable', 'inactive', 'Value', 0)
    set(handles.angle, 'Enable', 'inactive', 'Value', 0)
end

% --- Executes on button press in tresd.
function tresd_Callback(hObject, eventdata, handles)
active = get(hObject, 'Value');
if active == 1
    set(handles.dosd, 'Value', 0)
    set(handles.azimuth, 'Enable', 'inactive', 'Value', 0)
    set(handles.elevation, 'Enable', 'inactive', 'Value', 0)
    set(handles.angle, 'Enable', 'inactive', 'Value', 0)
elseif active == 0
    set(handles.dosd, 'Value', 1)
    set(handles.azimuth, 'Enable', 'on', 'Value', 1)
    set(handles.elevation, 'Enable', 'on', 'Value', 0)
    set(handles.angle, 'Enable', 'on')
end

% --- Executes on button press in azimuth.
function azimuth_Callback(hObject, eventdata, handles)
active = get(hObject, 'Value');
if active == 1
    set(handles.elevation, 'Value', 0)
elseif active == 0
    set(handles.elevation, 'Value', 1)
end
```

```
% --- Executes on button press in elevation.
function elevation_Callback(hObject, eventdata, handles)
active = get(hObject, 'Value');
if active == 1
    set(handles.azimuth, 'Value', 0)
elseif active == 0
    set(handles.azimuth, 'Value', 1)
end

% --- Executes on button press in actualizar.
function actualizar_Callback(hObject, eventdata, handles)
%frecuencia
f = str2double(get(handles.f, 'String'));
if f > 0
    value = get(handles.ghz, 'Value');
    if value == 1
        f = f * 1e9;
    elseif value == 0
        f = f * 1e6;
    end

%establecer antenna
sp = spiralArchimedean('Turns', 3);

rf=reflector('Exciter', sp)

%Diagrama de radiación
value = get(handles.tresd, 'Value');
%2D
if value == 0
    dim = get(handles.elevation, 'Value');
    if dim == 0
        %coger el angulo
        ang = get(handles.angle, 'Value');
        if ang == 1
            angle = str2double(get(handles.angulo, 'String'));
            figure;
            patternAzimuth(rf, f, [0 angle])
        else
            figure;
            patternAzimuth(rf, f);
        end
    elseif dim == 1
        %coger el angulo
        ang = get(handles.angle, 'Value');
        if ang == 1
            angle = str2double(get(handles.angulo, 'String'));
            figure;
            patternElevation(rf, f, [0 angle])
        else
            figure;
            patternElevation(rf, f);
        end
    end
end
%3D
elseif value == 1
    figure;
```

```
        pattern(rf,f)
    end
else
    errordlg('Introduce una frecuencia válida','Error');
end

% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
close(antennarad)
antennarad

function f_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function f_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in mhz.
function mhz_Callback(hObject, eventdata, handles)
active = get(hObject,'Value');
if active == 1
    set(handles.ghz,'Value',0)
elseif active == 0
    set(handles.ghz,'Value',1)
end

% --- Executes on button press in ghz.
function ghz_Callback(hObject, eventdata, handles)
active = get(hObject,'Value');
if active == 1
    set(handles.mhz,'Value',0)
elseif active == 0
    set(handles.mhz,'Value',1)
end

% --- Executes on button press in angle.
function angle_Callback(hObject, eventdata, handles)
value = get(hObject,'Value');
if value == 1
    set(handles.angulo,'Enable','on')
elseif value == 0
    set(handles.angulo,'Enable','inactive')
end

function angulo_Callback(hObject, eventdata, handles)
```

```
% --- Executes during object creation, after setting all properties.
function angulo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in info.
function info_Callback(hObject, eventdata, handles)
msgbox('Esta es una antena de espiral arquimediana con un reflector, simulando así
una antena direccional.','Información')
```

T3C

```
function varargout = T3C(varargin)
% T3C MATLAB code for T3C.fig
%   T3C, by itself, creates a new T3C or raises the existing
%   singleton*.
%
%   H = T3C returns the handle to a new T3C or the handle to
%   the existing singleton*.
%
%   T3C('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in T3C.M with the given input arguments.
%
%   T3C('Property','Value',...) creates a new T3C or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before T3C_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to T3C_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help T3C

% Last Modified by GUIDE v2.5 02-Mar-2016 23:04:53

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @T3C_OpeningFcn, ...
                  'gui_OutputFcn',  @T3C_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before T3C is made visible.
function T3C_OpeningFcn(hObject, eventdata, handles, varargin)

axis off;

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes T3C wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = T3C_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in siguiente.
function siguiente_Callback(hObject, eventdata, handles)
set(handles.r1,'Value',0);
set(handles.r2,'Value',0);
set(handles.r3,'Value',0);
set(handles.r4,'Value',0);
set(handles.r1,'Enable','on');
set(handles.r2,'Enable','on');
set(handles.r3,'Enable','on');
set(handles.r4,'Enable','on');
global num
num = get(handles.numero,'string');
num = str2double(num);
num=num+1;
set(handles.numero,'String',num);
if num>=1 && num<=8
    set(handles.siguiente,'Enable','Inactive');
else
    set(handles.siguiente,'Enable','on');
end
if num == 1
    %Poner las preguntas
    set(handles.pregunta,'String','El campo eléctrico y el campo magnético son:');
    set(handles.r1,'String','Ortogonales');
    set(handles.r2,'String','Paralelos');
    set(handles.r3,'String','Oblicuos');
    set(handles.r4,'string','Ninguna de las anteriores');
elseif num == 2
    set(handles.pregunta,'String','Una antena tiene:');
    set(handles.r1,'String','Ganancia directiva');
    set(handles.r2,'String','Ganancia de potencia');
    set(handles.r3,'String','Las anteriores son correctas');
    set(handles.r4,'string','Ninguna de las anteriores');
elseif num == 3
    set(handles.pregunta,'String','¿Cuál de los siguientes es un tipo de diagrama de radiación?');
    set(handles.r1,'String','Omnidireccional');
    set(handles.r2,'String','Isotrópico');
    set(handles.r3,'String','Pincel');
    set(handles.r4,'string','Todas son correctas');
elseif num == 4
    set(handles.pregunta,'String','¿Cómo se denomina el lóbulo que contiene la dirección de la máxima radiación?');
    set(handles.r1,'String','Primario');
    set(handles.r2,'String','Principal');
    set(handles.r3,'String','Máximo');
    set(handles.r4,'string','Ninguna de las anteriores');
elseif num == 5
```

```
set(handles.pregunta, 'String', 'Medios guiados son aquellos de los que se vale
la tecnología de las telecomunicaciones para guiar las señales portadoras de
información, usando medios materiales y finitos');
set(handles.r1, 'String', 'Verdadero');
set(handles.r2, 'String', 'Falso');
set(handles.r3, 'String', ' ');
set(handles.r4, 'string', ' ');
elseif num == 6
    set(handles.pregunta, 'String', '¿Qué es un array?');
    set(handles.r1, 'String', 'Una antena direccional');
    set(handles.r2, 'String', 'Una antena en un buque');
    set(handles.r3, 'String', 'Un conjunto de antenas');
    set(handles.r4, 'string', 'Ninguna es correcta');
elseif num == 7
    set(handles.pregunta, 'String', 'El ancho de banda a -3 dB se establece
entre:');
    set(handles.r1, 'String', 'Puntos de potencia triple');
    set(handles.r2, 'String', 'Puntos de potencia mitad');
    set(handles.r3, 'String', 'Puntos con treita veces la pontencia');
    set(handles.r4, 'string', 'Puntos con seis veces la potencia');
elseif num == 8
    set(handles.pregunta, 'String', 'Los cables coaxiales estan formados por dos
conductores concéntricos de ');
    set(handles.r1, 'String', 'Plata');
    set(handles.r2, 'String', 'Oro');
    set(handles.r3, 'String', 'Aluminio');
    set(handles.r4, 'string', 'Cobre');
else
    set(handles.pregunta, 'String', 'Ha terminado el Test. Presione finalizar');
    set(handles.r1, 'String', ' ');
    set(handles.r2, 'String', ' ');
    set(handles.r3, 'String', ' ');
    set(handles.r4, 'string', ' ');
end

% --- Executes on button press in fin.
function fin_Callback(hObject, eventdata, handles)
fin = questdlg('¿Desea repetir el test?', 'Ha terminado...', 'Si', 'No', 'No');
if strcmp(fin, 'No')
    close T3C
    clc
else
    close T3C
    T3C
end

% --- Executes on button press in r1.
function r1_Callback(hObject, eventdata, handles)
global num
valor1 = get(hObject, 'Value');
if valor1 == 1
    set(handles.r2, 'Value', 0);
    set(handles.r3, 'Value', 0);
    set(handles.r4, 'Value', 0);
    if num > 0 && num < 9
        if num == 1
            set(handles.siguiente, 'Enable', 'on');
        elseif num == 5
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
```

```
        end
    end
end

function r2_Callback(hObject, eventdata, handles)
global num
valor2 = get(hObject, 'Value');
if valor2 == 1
    set(handles.r1, 'Value', 0);
    set(handles.r3, 'Value', 0);
    set(handles.r4, 'Value', 0);
    if num > 0 && num < 9
        if num == 4
            set(handles.siguiente, 'Enable', 'on');
        elseif num == 7
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end
end

function r3_Callback(hObject, eventdata, handles)
global num
valor3 = get(hObject, 'Value');
if valor3 == 1
    set(handles.r2, 'Value', 0);
    set(handles.r1, 'Value', 0);
    set(handles.r4, 'Value', 0);
    if num > 0 && num < 9
        if num == 2
            set(handles.siguiente, 'Enable', 'on');
        elseif num == 6
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end
end

function r4_Callback(hObject, eventdata, handles)
global num
valor4 = get(hObject, 'Value');
if valor4 == 1
    set(handles.r2, 'Value', 0);
    set(handles.r3, 'Value', 0);
    set(handles.r1, 'Value', 0);
    if num > 0 && num < 9
        if num == 3
            set(handles.siguiente, 'Enable', 'on');
        elseif num == 8
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end
end
```

T4P

```
function varargout = T4P(varargin)
% T4P MATLAB code for T4P.fig
%     T4P, by itself, creates a new T4P or raises the existing
%     singleton*.
%
%     H = T4P returns the handle to a new T4P or the handle to
%     the existing singleton*.
%
%     T4P('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in T4P.M with the given input arguments.
%
%     T4P('Property','Value',...) creates a new T4P or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before T4P_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to T4P_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help T4P

% Last Modified by GUIDE v2.5 13-Feb-2016 17:38:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',  @T4P_OpeningFcn, ...
                  'gui_OutputFcn',   @T4P_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before T4P is made visible.
function T4P_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to T4P (see VARARGIN)
axis off
fondo = imread('radiocomunicaciones.jpg');
imshow(fondo);
% Choose default command line output for T4P
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes T4P wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = T4P_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in ejercicio.
function ejercicio_Callback(hObject, eventdata, handles)
% hObject handle to ejercicio (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns ejercicio contents as
cell array
% contents{get(hObject,'Value')} returns selected item from ejercicio

% --- Executes during object creation, after setting all properties.
function ejercicio_CreateFcn(hObject, eventdata, handles)
% hObject handle to ejercicio (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in ir.
function ir_Callback(hObject, eventdata, handles)
ej = get(handles.ejercicio,'Value');
switch ej
case 2
T4P1
case 3
ond_sup
case 4
Fresnel
end
clc
close(T4P)
```

T4P1

```
function varargout = T4P1(varargin)
% T4P1 MATLAB code for T4P1.fig
%   T4P1, by itself, creates a new T4P1 or raises the existing
%   singleton*.
%
%   H = T4P1 returns the handle to a new T4P1 or the handle to
%   the existing singleton*.
%
%   T4P1('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in T4P1.M with the given input arguments.
%
%   T4P1('Property','Value',...) creates a new T4P1 or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before T4P1_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to T4P1_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help T4P1

% Last Modified by GUIDE v2.5 13-Feb-2016 18:24:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @T4P1_OpeningFcn, ...
                  'gui_OutputFcn',  @T4P1_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before T4P1 is made visible.
function T4P1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to T4P1 (see VARARGIN)
title('Distancia de visibilidad entre dos antenas')
background = imread('T4P1.png');
imshow(background);
% Choose default command line output for T4P1
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes T4Pl wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = T4Pl_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function D_Callback(hObject, eventdata, handles)
% hObject handle to D (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of D as text
% str2double(get(hObject,'String')) returns contents of D as a double

% --- Executes during object creation, after setting all properties.
function D_CreateFcn(hObject, eventdata, handles)
% hObject handle to D (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function rx_Callback(hObject, eventdata, handles)
% hObject handle to rx (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rx as text
% str2double(get(hObject,'String')) returns contents of rx as a double

% --- Executes during object creation, after setting all properties.
function rx_CreateFcn(hObject, eventdata, handles)
% hObject handle to rx (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%     See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function tx_Callback(hObject, eventdata, handles)
% hObject    handle to tx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of tx as text
%         str2double(get(hObject,'String')) returns contents of tx as a double

% --- Executes during object creation, after setting all properties.
function tx_CreateFcn(hObject, eventdata, handles)
% hObject    handle to tx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%     See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
global C
C = get(hObject,'Value');
switch C
    case 2
        set(handles.D,'Enable','inactive','String','', 'BackgroundColor',[0.83 0.82
0.78]);
        set(handles.tx,'Enable','on','String','', 'BackgroundColor',[1 1 1]);
        set(handles.rx,'Enable','on','String','', 'BackgroundColor',[1 1 1]);

    case 3
        set(handles.D,'Enable','on','String','', 'BackgroundColor',[1 1 1]);
        set(handles.tx,'Enable','inactive','String','', 'BackgroundColor',[0.83
0.82 0.78]);
        set(handles.rx,'Enable','on','String','', 'BackgroundColor',[1 1 1])
    case 4
        set(handles.D,'Enable','on','String','', 'BackgroundColor',[1 1 1]);
        set(handles.tx,'Enable','on','String','', 'BackgroundColor',[1 1 1]);
        set(handles.rx,'Enable','inactive','String','', 'BackgroundColor',[0.83
0.82 0.78])
end

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in calcular.
function calcular_Callback(hObject, eventdata, handles)
global C
if C == 2
    tx=str2double(get(handles.tx,'String'));
    rx=str2double(get(handles.rx,'String'));
    D=((2*6371*tx/1000)^0.5)+((2*6371*rx/1000)^0.5);
    set(handles.D,'String',D);
elseif C == 3
    D=str2double(get(handles.D,'String'));
    rx=str2double(get(handles.rx,'String'));
    tx=((D*1000-((2*6371000*rx)^0.5))^2)/(2*6371000);
    set(handles.tx,'String',tx);
elseif C == 4
    tx = str2double(get(handles.tx,'String'));
    D = str2double(get(handles.D,'String'));
    rx = ((D*1000-((2*6371000*tx)^0.5))^2)/(2*6371000);
    set(handles.rx,'String',rx);
end

% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
set(handles.tx,'String',' ','Enable','inactive')
set(handles.rx,'String',' ','Enable','inactive')
set(handles.D,'String',' ','Enable','inactive')
set(handles.popupmenu1,'Value',1)
```

Ond_sup

```
function varargout = ond_sup(varargin)
% OND_SUP MATLAB code for ond_sup.fig
%   OND_SUP, by itself, creates a new OND_SUP or raises the existing
%   singleton*.
%
%   H = OND_SUP returns the handle to a new OND_SUP or the handle to
%   the existing singleton*.
%
%   OND_SUP('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in OND_SUP.M with the given input arguments.
%
%   OND_SUP('Property','Value',...) creates a new OND_SUP or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before ond_sup_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to ond_sup_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ond_sup

% Last Modified by GUIDE v2.5 02-Mar-2016 19:25:57

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @ond_sup_OpeningFcn, ...
                  'gui_OutputFcn',  @ond_sup_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before ond_sup is made visible.
function ond_sup_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ond_sup (see VARARGIN)

% Choose default command line output for ond_sup
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes ond_sup wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = ond_sup_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function pt_Callback(hObject, eventdata, handles)
% hObject handle to pt (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of pt as text
% str2double(get(hObject,'String')) returns contents of pt as a double

% --- Executes during object creation, after setting all properties.
function pt_CreateFcn(hObject, eventdata, handles)
% hObject handle to pt (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in calcular.
function calcular_Callback(hObject, eventdata, handles)
num_ter = str2double(inputdlg('Número de terrenos'));
num = num_ter + 1;
m_dis = zeros(1,num);
M_dis = zeros(1,num);
Dis = 0;
pot = str2double(get(handles.pot,'String'));
poten = strcat([num2str(pot)],{'KW'});
text(0.1,2.15,poten)
for n = 2 : num
    nu = n-1;
    Tn = strcat({'Longitud del terreno '},{num2str(nu)},{ ' en km.'});
    dis = str2double(inputdlg(Tn));
    if dis > 0
        m_dis(1,n) = dis;
        Dis = Dis + dis;
        M_dis(1,n) = Dis;
    else
```

```
        errordlg('Introduce un valor correcto');
        n = n - 1;
    end
end
axes(handles.axes1)
axis([0 M_dis(1,n) 0 3])
ppp=gca;
ppp.XTick = M_dis;
ppp.YTick = [2];
ppp.YTickLabel = ['Tx']
xlabel('Distancia en km')
for t = 2:num
    d1 = M_dis(1,t);
    d2 = M_dis(1,t-1);
    te = strcat('T',num2str(t-1));
    text((d1+d2)/2,0.5,te);
end
abs = 2*num_ter-1;
M_abs = zeros(1,abs);
cont = 0;
con = 1;
casitotal = 0;
nume = 1;
numer = 1;
for a = 1:abs
    if cont == 2
        cont = 0;
        con = con + 1;
    end
    if numer == 2
        nume = nume + 1;
        numer = 0;
    end
    abb = strcat({'Introduce el valor de la intensidad de campo para el terreno '}, {num2str(nume)}, {' a una distandcia de '}, {num2str(M_dis(1,con+1))}, {' km'});
    cont = cont + 1;
    numer = numer + 1;
    ab = str2double(inputdlg(abb));
    M_abs(1,a) = ab;
    if cont == 1
        casitotal = casitotal + ab;
    elseif cont == 2
        casitotal = casitotal - ab;
    end
end
total = casitotal + 10*log10(pot)
set(handles.resultado,'String',total)

% --- Executes on button press in reset.
function reset_Callback(~, ~, ~)
close(ond_sup)
ond_sup
```

Fresnel

```
function varargout = Fresnel(varargin)
% FRESNEL MATLAB code for Fresnel.fig
%     FRESNEL, by itself, creates a new FRESNEL or raises the existing
%     singleton*.
%
%     H = FRESNEL returns the handle to a new FRESNEL or the handle to
%     the existing singleton*.
%
%     FRESNEL('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in FRESNEL.M with the given input arguments.
%
%     FRESNEL('Property','Value',...) creates a new FRESNEL or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before Fresnel_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to Fresnel_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Fresnel

% Last Modified by GUIDE v2.5 19-Feb-2016 08:00:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @Fresnel_OpeningFcn, ...
                  'gui_OutputFcn',  @Fresnel_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Fresnel is made visible.
function Fresnel_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to Fresnel (see VARARGIN)
imshow(imread('fresnell.jpg'));
set(handles.hz,'Value',1);
% Choose default command line output for Fresnel
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Fresnel wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = Fresnel_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function ht_Callback(hObject, eventdata, handles)
% hObject handle to ht (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ht as text
% str2double(get(hObject,'String')) returns contents of ht as a double

% --- Executes during object creation, after setting all properties.
function ht_CreateFcn(hObject, eventdata, handles)
% hObject handle to ht (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function hr_Callback(hObject, eventdata, handles)
% hObject handle to hr (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of hr as text
% str2double(get(hObject,'String')) returns contents of hr as a double

% --- Executes during object creation, after setting all properties.
function hr_CreateFcn(hObject, eventdata, handles)
% hObject handle to hr (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function d2_Callback(hObject, eventdata, handles)
% hObject      handle to d2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of d2 as text
%         str2double(get(hObject,'String')) returns contents of d2 as a double

% --- Executes during object creation, after setting all properties.
function d2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to d2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function d1_Callback(hObject, eventdata, handles)
% hObject      handle to d1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of d1 as text
%         str2double(get(hObject,'String')) returns contents of d1 as a double

% --- Executes during object creation, after setting all properties.
function d1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to d1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function dt_Callback(hObject, eventdata, handles)
% hObject      handle to dt (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of dt as text
%         str2double(get(hObject,'String')) returns contents of dt as a double

% --- Executes during object creation, after setting all properties.
function dt_CreateFcn(hObject, eventdata, handles)
% hObject      handle to dt (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function hobst_Callback(hObject, eventdata, handles)
% hObject      handle to hobst (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of hobst as text
%         str2double(get(hObject,'String')) returns contents of hobst as a double

% --- Executes during object creation, after setting all properties.
function hobst_CreateFcn(hObject, eventdata, handles)
% hObject      handle to hobst (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function f_Callback(hObject, eventdata, handles)
% hObject      handle to f (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of f as text
%         str2double(get(hObject,'String')) returns contents of f as a double

% --- Executes during object creation, after setting all properties.
function f_CreateFcn(hObject, eventdata, handles)
% hObject      handle to f (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles      empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
```

```
%      See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),  
get(0,'defaultUicontrolBackgroundColor'))  
    set(hObject,'BackgroundColor','white');  
end
```

```
% --- Executes on button press in hz.
```

```
function hz_Callback(hObject, eventdata, handles)  
fr = get(hObject,'Value');  
if fr == 1  
    set(handles.gh, 'Value',0);  
    set(handles.kh, 'Value',0);  
    set(handles.mh, 'Value',0);  
elseif fr == 0  
    set(hObject,'Value',1);  
end
```

```
% --- Executes on button press in kh.
```

```
function kh_Callback(hObject, eventdata, handles)  
fr = get(hObject,'Value');  
if fr == 1  
    set(handles.hz, 'Value',0);  
    set(handles.gh, 'Value',0);  
    set(handles.mh, 'Value',0);  
elseif fr == 0  
    set(hObject,'Value',1);  
end
```

```
% --- Executes on button press in mh.
```

```
function mh_Callback(hObject, eventdata, handles)  
fr = get(hObject,'Value');  
if fr == 1  
    set(handles.hz, 'Value',0);  
    set(handles.kh, 'Value',0);  
    set(handles.gh, 'Value',0);  
elseif fr == 0  
    set(hObject,'Value',1);  
end
```

```
% --- Executes on button press in gh.
```

```
function gh_Callback(hObject, eventdata, handles)  
fr = get(hObject,'Value');  
if fr == 1  
    set(handles.hz, 'Value',0);  
    set(handles.kh, 'Value',0);  
    set(handles.mh, 'Value',0);  
elseif fr == 0  
    set(hObject,'Value',1);  
end
```

```
% --- Executes on selection change in calcula.
```

```
function calcula_Callback(hObject, eventdata, handles)
```

```
imshow(imread('fresnell.jpg'));
cal = get(hObject,'Value');
if cal == 2
    set(handles.hr,'Style','text')
    set(handles.ht,'Style','edit')
    set(handles.hobst,'Style','edit')
    set(handles.dl,'Style','edit')
    set(handles.f,'Style','edit')
    set(handles.dt,'Style','edit')
elseif cal == 3
    set(handles.hr,'Style','edit')
    set(handles.ht,'Style','edit')
    set(handles.hobst,'Style','text')
    set(handles.dl,'Style','edit')
    set(handles.f,'Style','edit')
    set(handles.dt,'Style','edit')
elseif cal == 4
    set(handles.hr,'Style','edit')
    set(handles.ht,'Style','edit')
    set(handles.hobst,'Style','edit')
    set(handles.dl,'Style','edit')
    set(handles.f,'Style','text')
    set(handles.dt,'Style','edit')
elseif cal == 5
    set(handles.hr,'Style','edit')
    set(handles.ht,'Style','edit')
    set(handles.hobst,'Style','edit')
    set(handles.dl,'Style','edit')
    set(handles.f,'Style','edit')
    set(handles.dt,'Style','edit')
end

% --- Executes during object creation, after setting all properties.
function calcula_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function k__Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function k__CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in k.
function k_Callback(hObject, ~, handles)
k = get(hObject,'Value');
set(handles.G,'Enable','off')
set(handles.k_,'Enable','on');
if k == 1
    set(handles.g_,'Value',0);
elseif k == 0
```

```

        set(hObject, 'Value', 1);
end

% --- Executes on button press in g_.
function g__Callback(hObject, ~, handles)
g = get(hObject, 'Value');
set(handles.k_, 'Enable', 'off')
set(handles.G, 'Enable', 'on');
if g == 1
    set(handles.k, 'Value', 0);
elseif g == 0
    set(handles.g_, 'Value', 1);
end

% --- Executes on button press in calculos.
function calculos_Callback(~, ~, handles)
calculo = get(handles.calcula, 'Value');
if calculo == 2
    %Variables
    ht = str2double(get(handles.ht, 'String'));
    d1 = str2double(get(handles.d1, 'String'));
    d1 = d1*1000;
    D = str2double(get(handles.dt, 'String'));
    D = D*1000;
    h_obst = str2double(get(handles.hobst, 'String'));
    d2 = D - d1;

    %calculamos la k
    ge = get(handles.g_, 'Value');
    if ge == 1
        G = str2double(get(handles.G, 'String'));
        k = ((1e6)/6371)/(((1e6)/6371)+G);
    else
        k = str2double(get(handles.k_, 'String'));
    end

    %Calculamos Lambda
    f = str2double(get(handles.f, 'String'));
    fr = get(handles.kh, 'value');
    if fr == 1
        f = f*1000;
    else
        fr = get(handles.mh, 'value');
        if fr == 1
            f = f*1e6;
        else
            fr = get(handles.gh, 'value');
            if fr == 1
                f = f*1e9;
            end
        end
    end
    lambda = (3e8)/f;

    %operaciones
    r = sqrt(lambda*d1*d2/D);
    h0 = d1*d2/(2*k*6371000);
    y_d1 = h0 + h_obst + r;

```

```
m = (y_d1 - ht)/(d1);
hr = D*m + ht;

%resultados
set(handles.hr, 'String', hr)
imshow(imread('fresnels.jpg'));

elseif calculo == 3
    %Variables
    ht = str2double(get(handles.ht, 'String'));
    hr = str2double(get(handles.hr, 'String'));
    d1 = str2double(get(handles.d1, 'String'));
    d1 = d1*1000;
    D = str2double(get(handles.dt, 'String'));
    D = D*1000;
    d2 = (D - d1)/1000;
    d2 = d2*1000;
    %calculo de lambda
    f = str2double(get(handles.f, 'String'));
    fr = get(handles.kh, 'value');
    if fr == 1
        f = f*1000;
    else
        fr = get(handles.mh, 'value');
        if fr == 1
            f = f*1e6;
        else
            fr = get(handles.gh, 'value');
            if fr == 1
                f = f*1e9;
            end
        end
    end
    lambda = (3e8)/f;

    %calculo de la k
    ge = get(handles.g_, 'Value');
    if ge == 1
        G = str2double(get(handles.G, 'String'));
        k = ((1e6)/6371)/(((1e6)/6371)+G);
    else
        k = str2double(get(handles.k_, 'String'));
    end

    %operaciones
    r = sqrt(lambda*d1*d2/D);
    h0 = d1*d2/(2*k*6371000);
    m = (hr-ht)/D;
    y_d1 = m*d1+ht;
    h_obst = y_d1-h0-r;

    %resultados
    imshow(imread('fresnels.jpg'));
    set(handles.hobst, 'String', h_obst, 'FontSize', 14)
elseif calculo == 4
    %Variables
    ht = str2double(get(handles.ht, 'String'));
    hr = str2double(get(handles.hr, 'String'));
    d1 = str2double(get(handles.d1, 'String'));
    d1 = d1*1000;
    D = str2double(get(handles.dt, 'String'));
```

```

D = D*1000;
d2 = D - d1;
h_obst = str2double(get(handles.hobst, 'String'));
%Calculamos la k
ge = get(handles.g_, 'Value');
if ge == 1
    G = str2double(get(handles.G, 'String'));
    k = ((1e6)/6371)/(((1e6)/6371)+G);
else
    k = str2double(get(handles.k_, 'String'));
end

%operaciones
h0 = d1*d2/(2*k*6371000);
m = (hr-ht)/D;
y_d1 = m*d1+ht;
r = y_d1 - h0 - h_obst;
lambda = r*r*D/(d1*d2);
f = (3e8)/lambda
if f < 1e3
    f = f / 1e3;
    set(handles.hz, 'Value', 1)
    set(handles.kh, 'Value', 0)
    set(handles.mh, 'Value', 0)
    set(handles.gh, 'Value', 0)
else
    if f < 1e6
        f = f / 1e6;
        set(handles.f, 'String', f, 'FontSize', 14)
        set(handles.hz, 'Value', 0)
        set(handles.kh, 'Value', 1)
        set(handles.mh, 'Value', 0)
        set(handles.gh, 'Value', 0)
    else
        if f < 1e9
            f = f / 1e9;
            set(handles.f, 'String', f, 'FontSize', 14)
            set(handles.hz, 'Value', 0)
            set(handles.kh, 'Value', 0)
            set(handles.mh, 'Value', 1)
            set(handles.gh, 'Value', 0)
        else
            if f >= 1e9
                f = f / 1e9;
                set(handles.f, 'String', f, 'FontSize', 14)
                set(handles.hz, 'Value', 0)
                set(handles.kh, 'Value', 0)
                set(handles.mh, 'Value', 0)
                set(handles.gh, 'Value', 1)
            end
        end
    end
end
end
end
imshow(imread('fresnels.jpg'));

elseif calculo == 5
    %Variables
    ht = str2double(get(handles.ht, 'String'));
    hr = str2double(get(handles.hr, 'String'));
    d1 = str2double(get(handles.d1, 'String'));
    d1 = d1*1000;

```

```
D = str2double(get(handles.dt, 'String'));
D = D*1000;
d2 = D - d1;
h_obst = str2double(get(handles.hobst, 'String'));
%Calculamos Lambda
f = str2double(get(handles.f, 'String'));
fr = get(handles.kh, 'value');
if fr == 1
    f = f*1000;
else
    fr = get(handles.mh, 'value');
    if fr == 1
        f = f*1e6;
    else
        fr = get(handles.gh, 'value');
        if fr == 1
            f = f*1e9;
        end
    end
end
%Calculamos la k
ge = get(handles.g_, 'Value');
if ge == 1
    G = str2double(get(handles.G, 'String'));
    k = ((1e6)/6371)/(((1e6)/6371)+G);
    h0 = d1*(D-d1)/(2*k*6371000);
else
    k = str2double(get(handles.k_, 'String'));
    h0 = d1*(D-d1)/(2*k*6371000);
end

%operaciones
r = sqrt((3e8)*d1*d2/(D*f));
m=(hr - ht)/D;
Y_R = m * d1 + ht;
YR = h0 + h_obst + r;

%resultado
if Y_R>=YR
    imshow(imread('fresnels.jpg'));
elseif Y_R<YR
    imshow(imread('fresneln.jpg'));
end
end

function reset_Callback(~, ~, ~)
close(Fresnel)
Fresnel
```

T4C

```
function varargout = T4C(varargin)
% T4C MATLAB code for T4C.fig
%     T4C, by itself, creates a new T4C or raises the existing
%     singleton*.
%
%     H = T4C returns the handle to a new T4C or the handle to
%     the existing singleton*.
%
%     T4C('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in T4C.M with the given input arguments.
%
%     T4C('Property','Value',...) creates a new T4C or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before T4C_OpeningFcn gets called. An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to T4C_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help T4C

% Last Modified by GUIDE v2.5 04-Mar-2016 02:30:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @T4C_OpeningFcn, ...
                  'gui_OutputFcn',  @T4C_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before T4C is made visible.
function T4C_OpeningFcn(hObject, eventdata, handles, varargin)

axis off;

handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes T4C wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = T4C_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in siguiente.
function siguiente_Callback(hObject, eventdata, handles)
set(handles.r1,'Value',0);
set(handles.r2,'Value',0);
set(handles.r3,'Value',0);
set(handles.r4,'Value',0);
set(handles.r1,'Enable','on');
set(handles.r2,'Enable','on');
set(handles.r3,'Enable','on');
set(handles.r4,'Enable','on');
global num
num = get(handles.numero,'string');
num = str2double(num);
num=num+1;
set(handles.numero,'String',num);
if num>=1 && num<=8
    set(handles.siguiente,'Enable','Inactive');
else
    set(handles.siguiente,'Enable','on');
end
if num == 1
    %Poner las preguntas
    set(handles.pregunta,'String','La onda de superficie se propaga siguiendo la
superficie terrestre hasta 1000 km como máximo');
    set(handles.r1,'String','Verdadero');
    set(handles.r2,'String','Falso');
    set(handles.r3,'String',' ');
    set(handles.r4,'string',' ');
elseif num == 2
    set(handles.pregunta,'String','El efecto guía onda:');
    set(handles.r1,'String','Se da a frecuencias muy altas');
    set(handles.r2,'String','Se da a frecuencias medias');
    set(handles.r3,'String','Se da a frecuencias muy bajas');
    set(handles.r4,'string','No depende de la frecuencia');
elseif num == 3
    set(handles.pregunta,'String','¿De qué no dependen las ondas de superficie?');
    set(handles.r1,'String','Naturaleza del suelo');
    set(handles.r2,'String','Fase inicial');
    set(handles.r3,'String','Frecuencia de transmisión');
    set(handles.r4,'string','Potencia transmitida');
elseif num == 4
    set(handles.pregunta,'String','En una onda de superficie, la absorción es
mayor en polarización vertical que en polarización horizontal');
    set(handles.r1,'String','Verdadero');
    set(handles.r2,'String','Falso');
    set(handles.r3,'String',' ');
    set(handles.r4,'string',' ');
elseif num == 5
```

```

        set(handles.pregunta, 'String', 'Las ondas ionosféricas van propagándose a
indices de refracción menores');
        set(handles.r1, 'String', 'Verdadero');
        set(handles.r2, 'String', 'Falso');
        set(handles.r3, 'String', ' ');
        set(handles.r4, 'string', ' ');
elseif num == 6
    set(handles.pregunta, 'String', 'La frecuencia que en condiciones de incidencia
vertical se refleja en la parte más ionizada de una capa es:');
    set(handles.r1, 'String', 'Frecuencia máxima');
    set(handles.r2, 'String', 'Frecuencia de capa');
    set(handles.r3, 'String', 'Frecuencia crítica');
    set(handles.r4, 'string', 'Frecuencia normal');
elseif num == 7
    set(handles.pregunta, 'String', 'Para una determinada frecuencia, la mínima
distancia contada a partir del transmisor para que la onda retorne por efecto de
la ionosfera es:');
    set(handles.r1, 'String', 'Distancia máxima');
    set(handles.r2, 'String', 'Distancia mínima');
    set(handles.r3, 'String', 'Distancia de onda');
    set(handles.r4, 'string', 'Distancia de salto');
elseif num == 8
    set(handles.pregunta, 'String', 'La onda espacial es la única que interesa
en:');
    set(handles.r1, 'String', 'EBL');
    set(handles.r2, 'String', 'HF');
    set(handles.r3, 'String', 'UHF');
    set(handles.r4, 'string', 'MF');
else
    set(handles.pregunta, 'String', 'Ha terminado el Test. Presione finalizar');
    set(handles.r1, 'String', ' ');
    set(handles.r2, 'String', ' ');
    set(handles.r3, 'String', ' ');
    set(handles.r4, 'string', ' ');
end

% --- Executes on button press in fin.
function fin_Callback(hObject, eventdata, handles)
fin = questdlg('¿Desea repetir el test?', 'Ha terminado...', 'Si', 'No', 'No');
if strcmp(fin, 'No')
    close T4C
    clc
else
    close T4C
    T3C
end

% --- Executes on button press in r1.
function r1_Callback(hObject, eventdata, handles)
global num
valor1 = get(hObject, 'Value');
if valor1 == 1
    set(handles.r2, 'Value', 0);
    set(handles.r3, 'Value', 0);
    set(handles.r4, 'Value', 0);
    if num > 0 && num < 9
        if num == 1
            set(handles.siguiente, 'Enable', 'on');
        elseif num == 5
            set(handles.siguiente, 'Enable', 'on');

```

```
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end
```

```
function r2_Callback(hObject, eventdata, handles)
global num
valor2 = get(hObject, 'Value');
if valor2 == 1
    set(handles.r1, 'Value', 0);
    set(handles.r3, 'Value', 0);
    set(handles.r4, 'Value', 0);
    if num > 0 && num < 9
        if num == 4
            set(handles.siguiente, 'Enable', 'on');
        elseif num == 3
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end
```

```
function r3_Callback(hObject, eventdata, handles)
global num
valor3 = get(hObject, 'Value');
if valor3 == 1
    set(handles.r2, 'Value', 0);
    set(handles.r1, 'Value', 0);
    set(handles.r4, 'Value', 0);
    if num > 0 && num < 9
        if num == 8
            set(handles.siguiente, 'Enable', 'on');
        elseif num == 6
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end
```

```
function r4_Callback(hObject, eventdata, handles)
global num
valor4 = get(hObject, 'Value');
if valor4 == 1
    set(handles.r2, 'Value', 0);
    set(handles.r3, 'Value', 0);
    set(handles.r1, 'Value', 0);
    if num > 0 && num < 9
        if num == 7
            set(handles.siguiente, 'Enable', 'on');
        elseif num == 2
            set(handles.siguiente, 'Enable', 'on');
        else
            set(handles.siguiente, 'Enable', 'inactive');
        end
    end
end
```

ANEXO II: CURVAS DE PROPAGACIÓN DE ONDAS DE SUPERFICIE

FIGURA 1

Curvas de propagación de la onda de superficie; agua del mar, salinidad baja, $\sigma = 1 \text{ S/m}$, $\epsilon = 80$

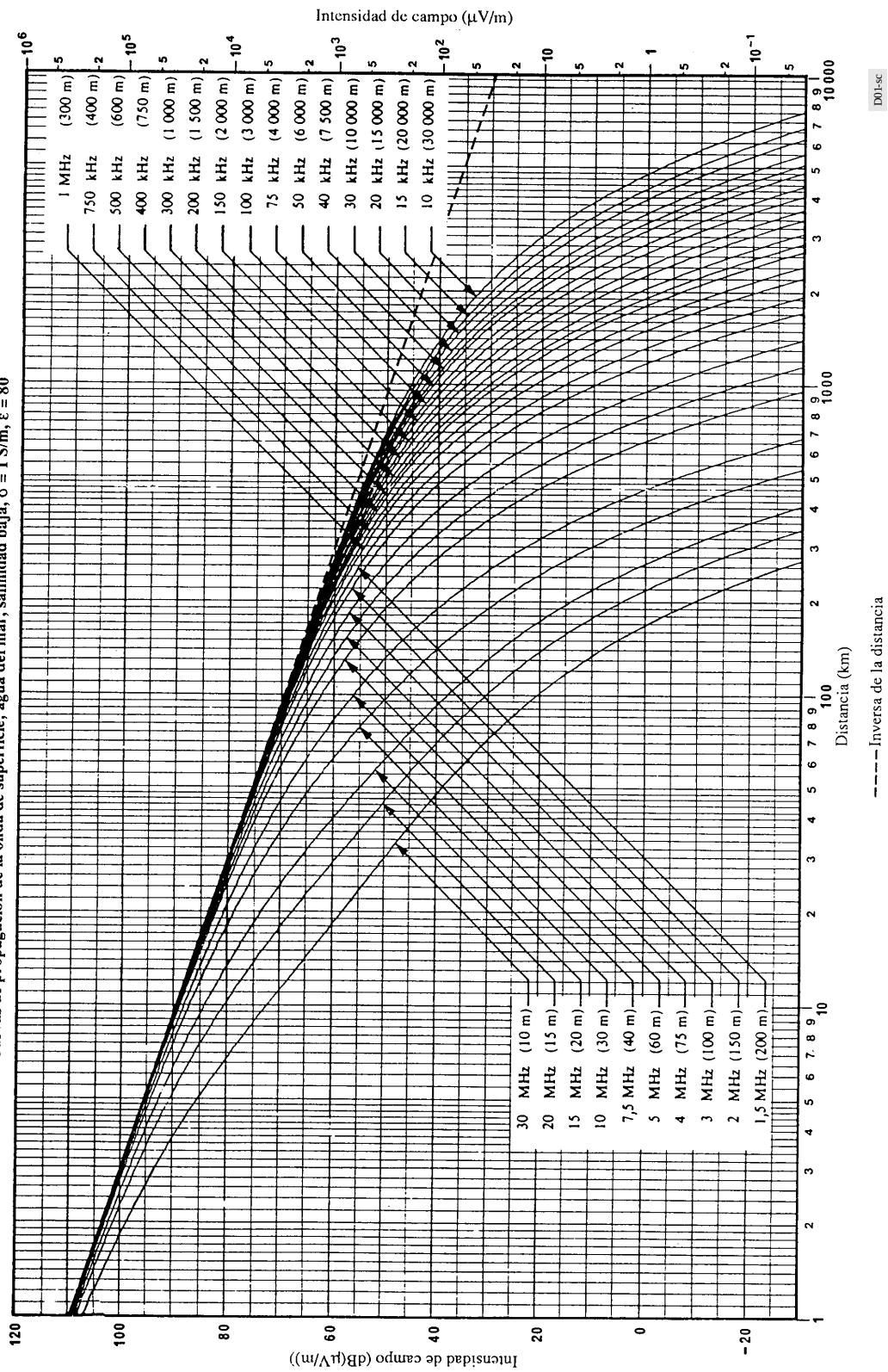


FIGURA 2

Curvas de propagación de la onda de superficie; agua del mar, salinidad media, $\sigma = 5 \text{ S/m}$, $\epsilon = 70$

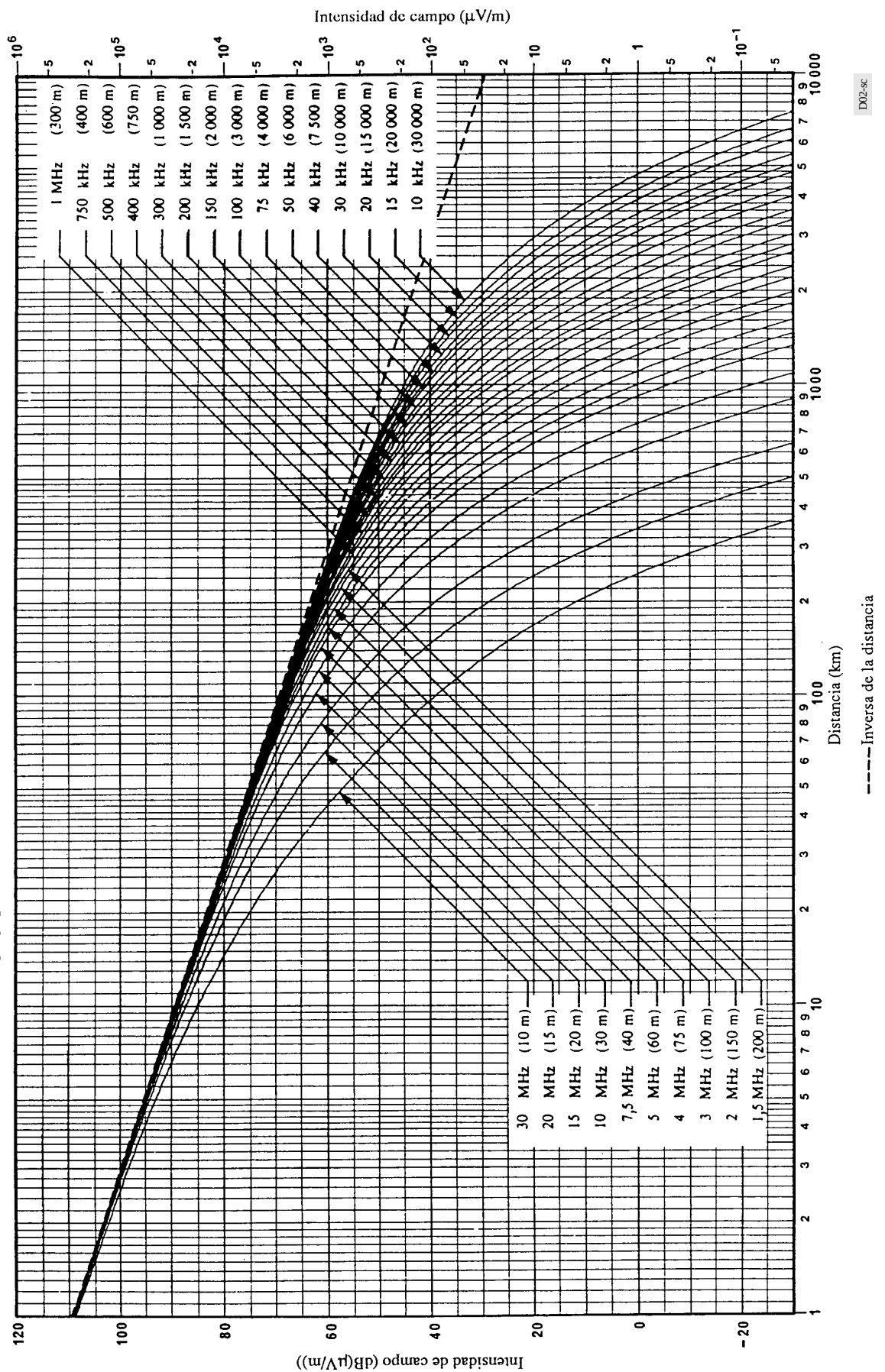
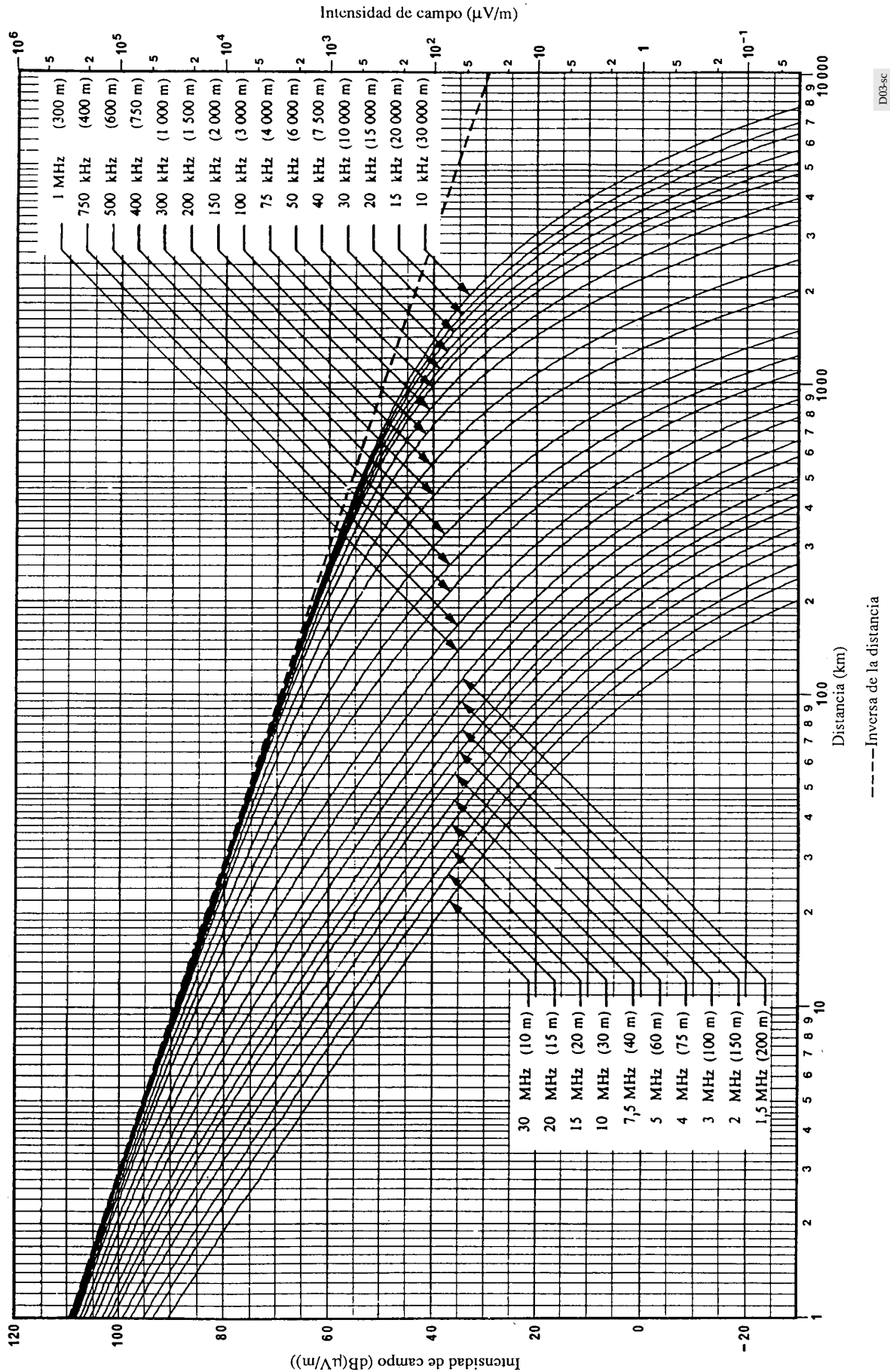


FIGURA 3

Curvas de propagación de la onda de superficie; agua dulce, $\sigma = 3 \times 10^{-3}$ S/m, $\epsilon = 80$



D03-sc

FIGURA 4

Curvas de propagación de la onda de superficie; tierra, $\sigma = 3 \times 10^{-2} \text{ S/m}$, $\epsilon = 40$

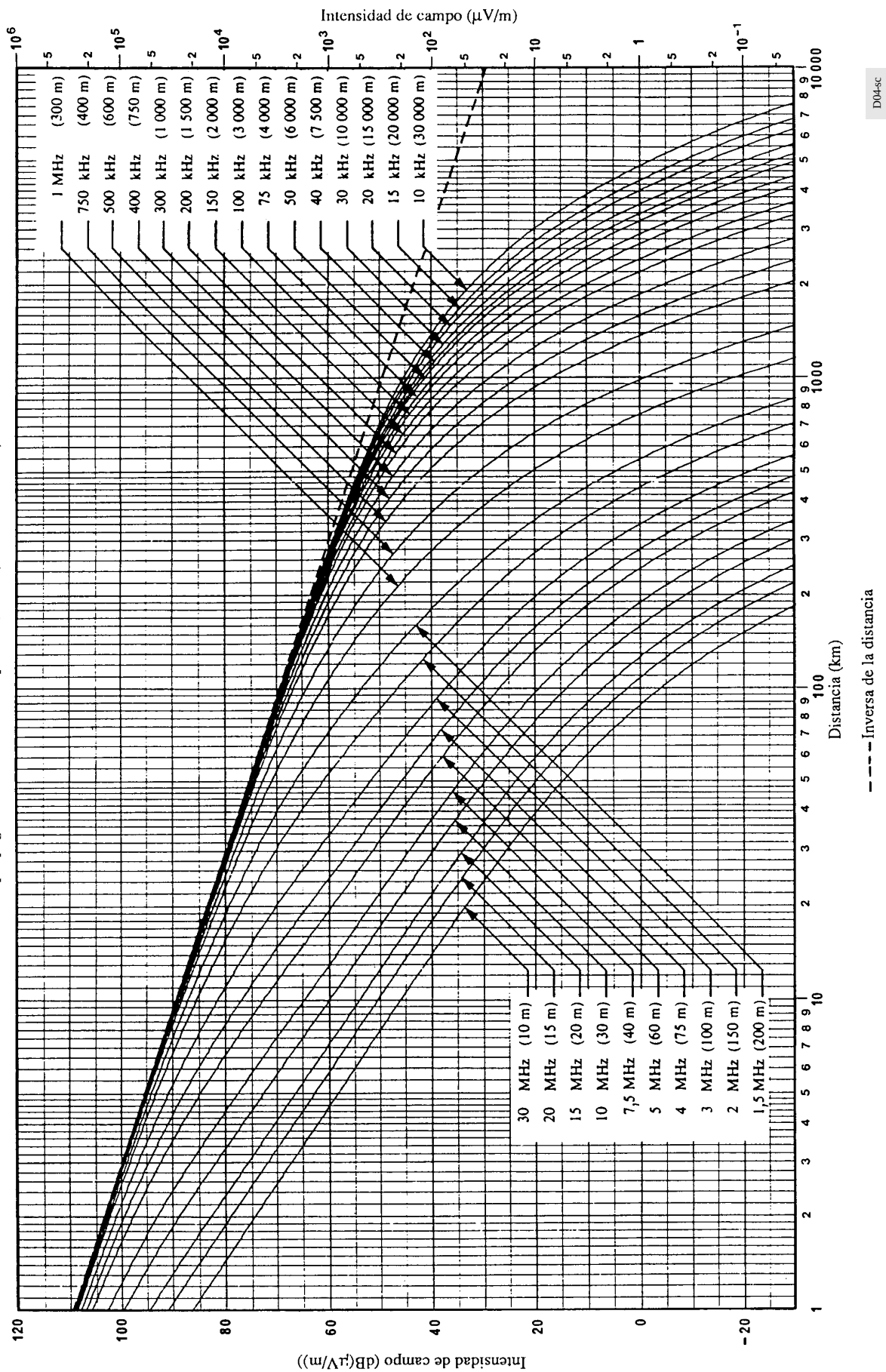


FIGURA 5

Curvas de propagación de la onda de superficie; tierra húmeda, $\sigma = 10^{-2}$ S/m, $\epsilon = 30$

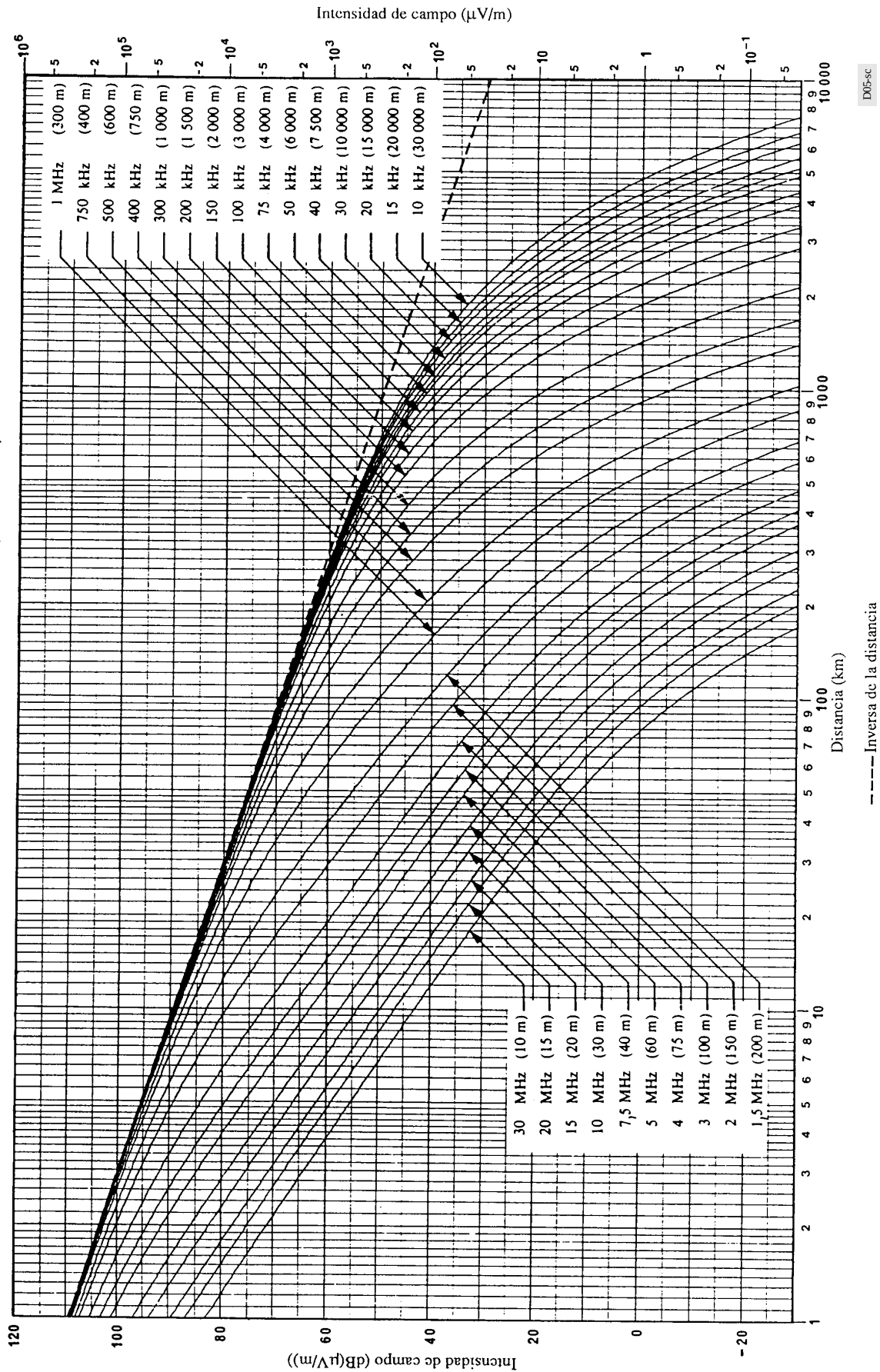
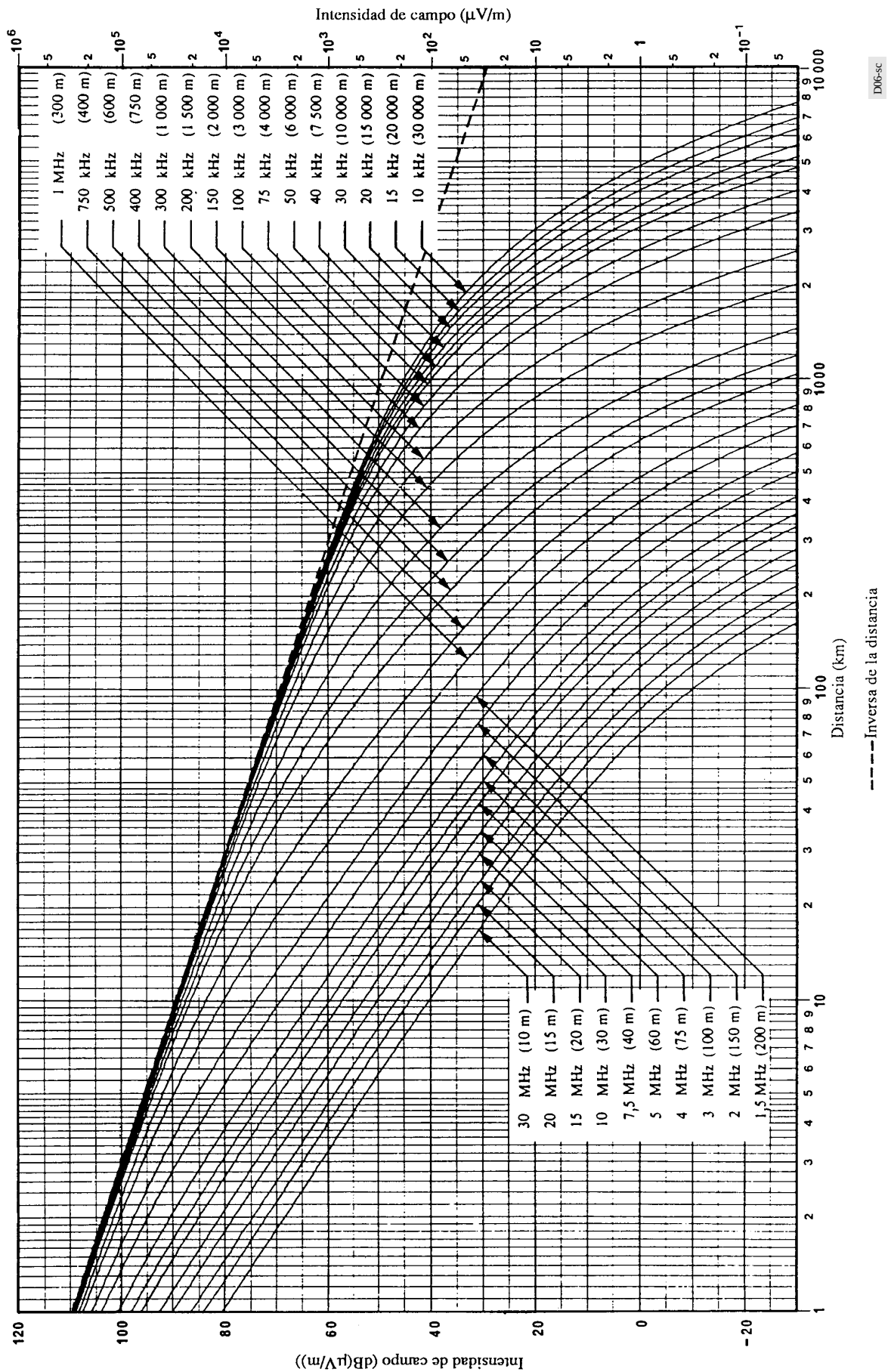


FIGURA 6

Curvas de propagación de la onda de superficie; tierra, $\sigma = 3 \times 10^{-3} \text{ S/m}$, $\epsilon = 22$



D06-sec

FIGURA 7
Curvas de propagación de la onda de superficie; tierra moderadamente seca, $\sigma = 10^{-3}$ S/m, $\epsilon = 15$

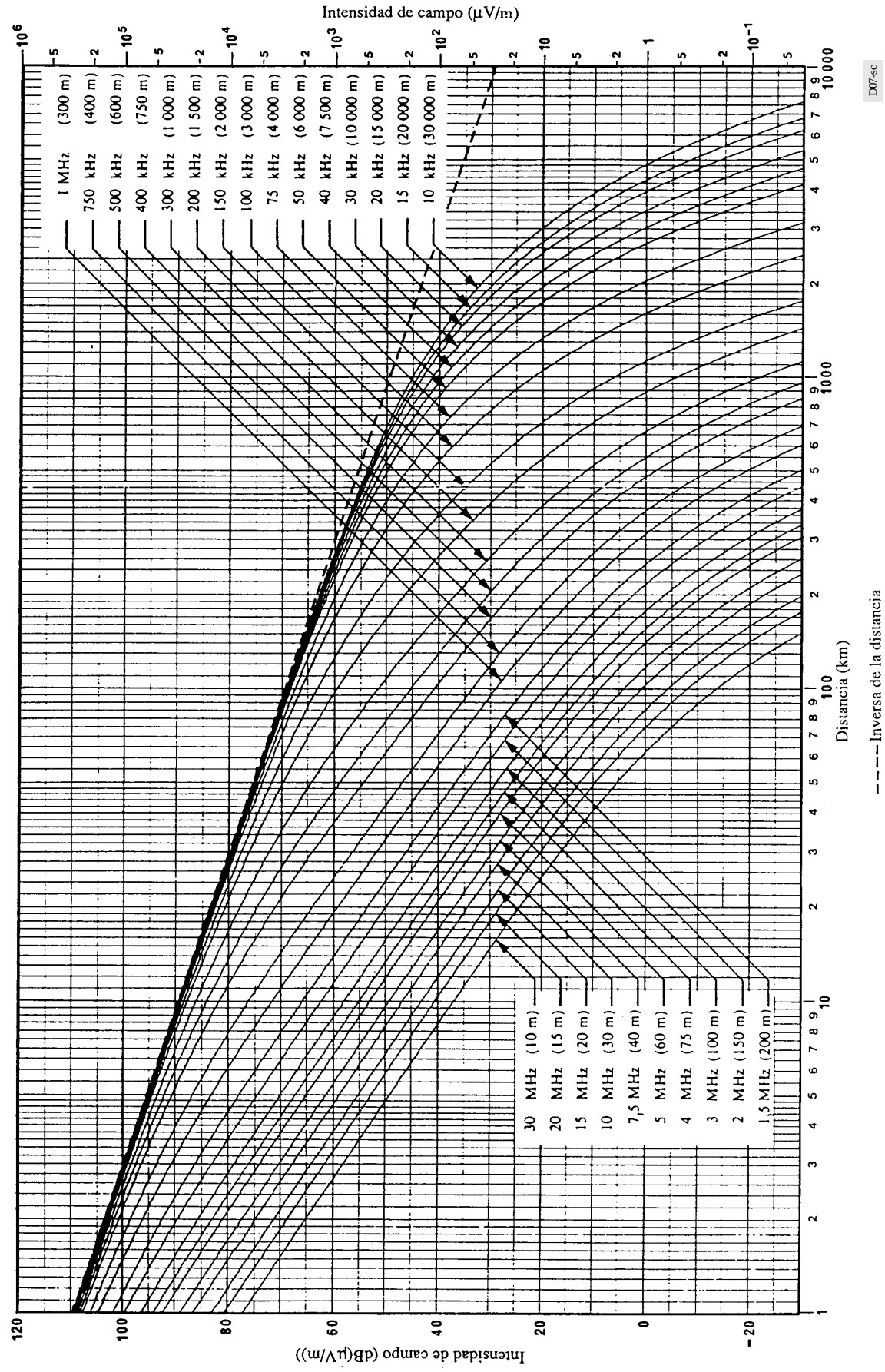


FIGURA 8
Curvas de propagación de la onda de superficie; tierra seca, $\sigma = 3 \times 10^{-4}$ S/m, $\epsilon = 7$

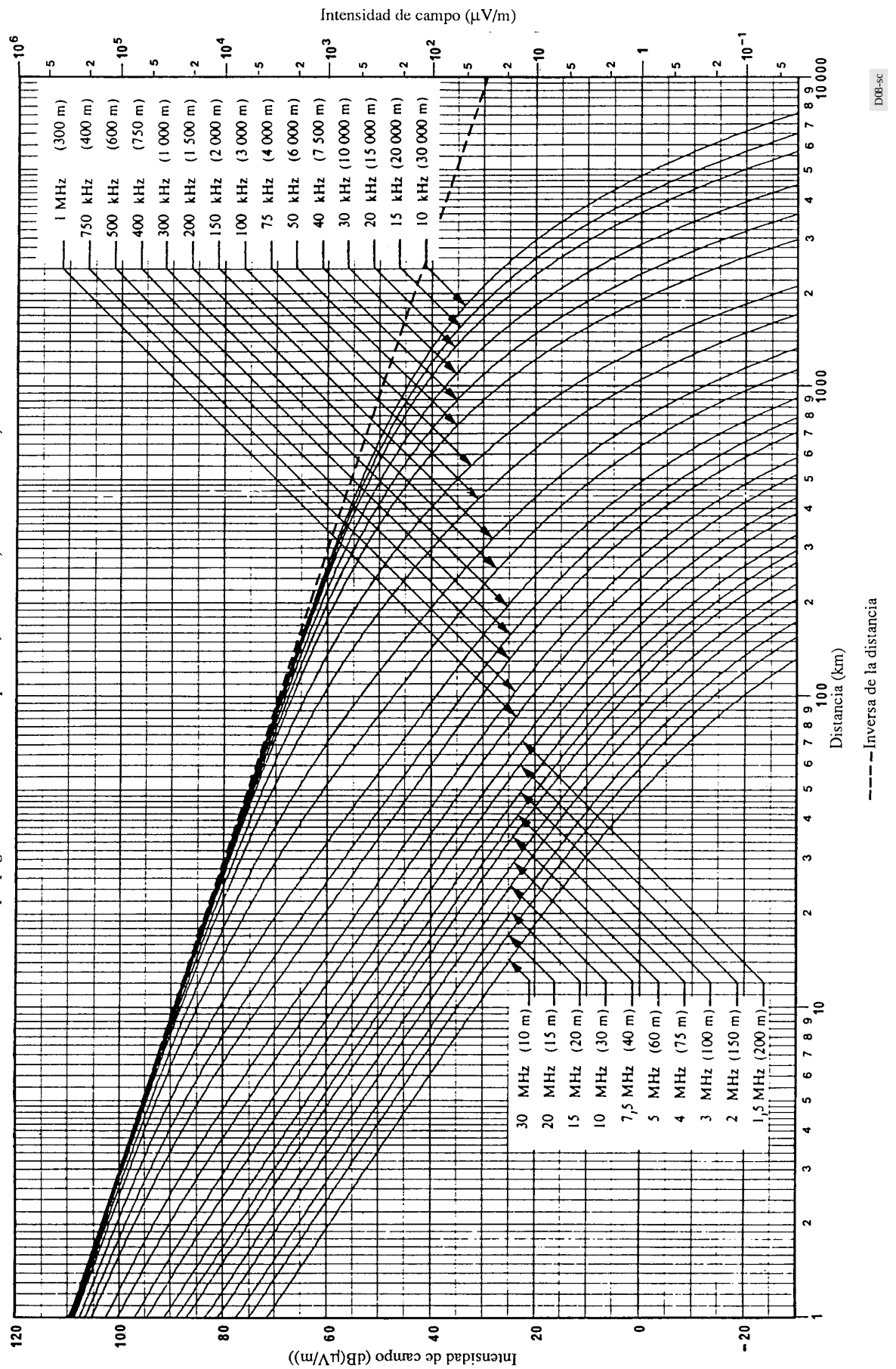


FIGURA 9

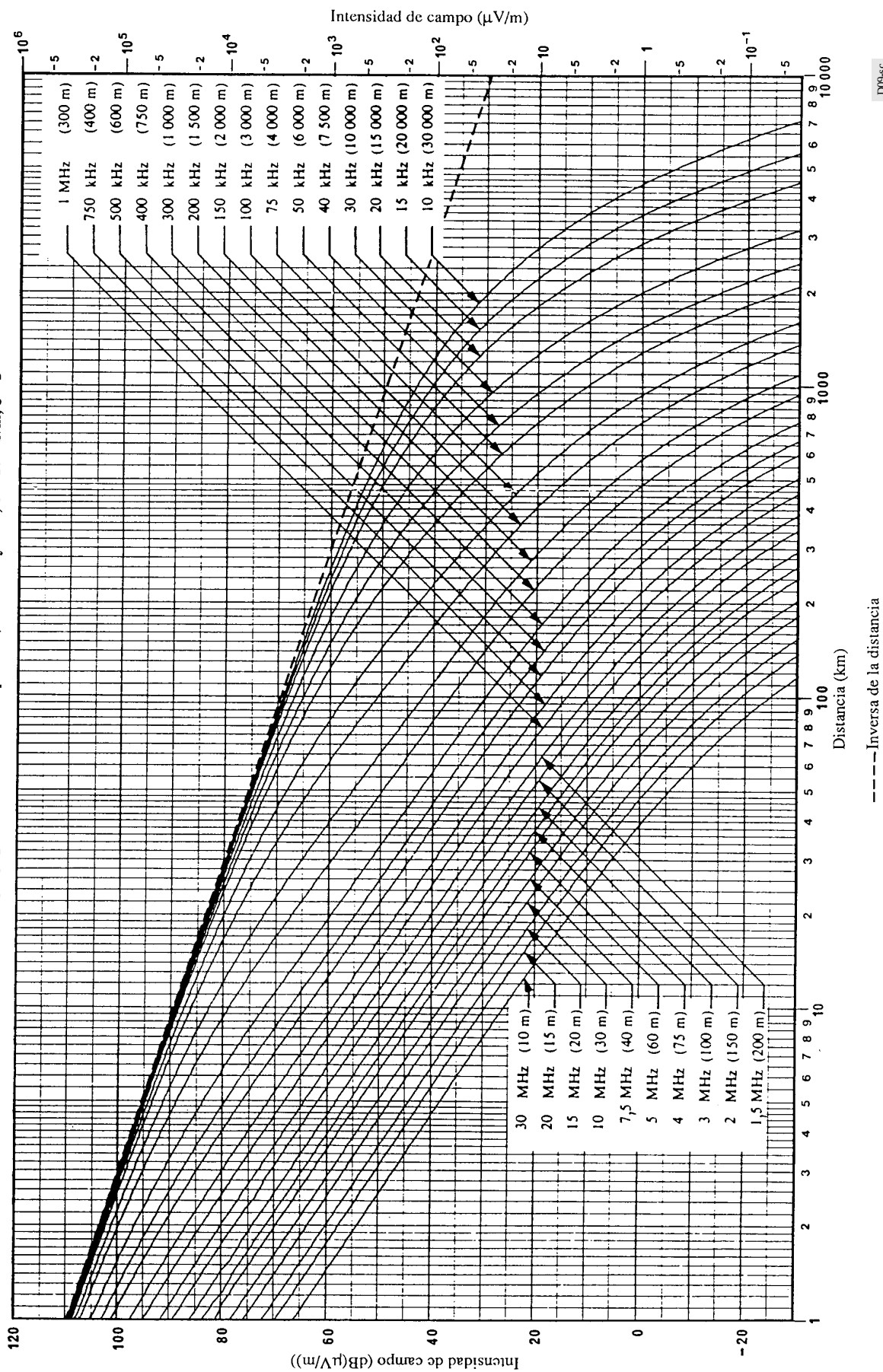
Curvas de propagación de la onda de superficie; tierra muy seca, $\sigma = 10^{-4}$ S/m, $\epsilon = 3$ 

FIGURA 10

Curvas de propagación de la onda de superficie; hielo de agua dulce, -1°C , $\sigma = 3 \times 10^{-5} \text{ S/m}$, $\epsilon = 3$

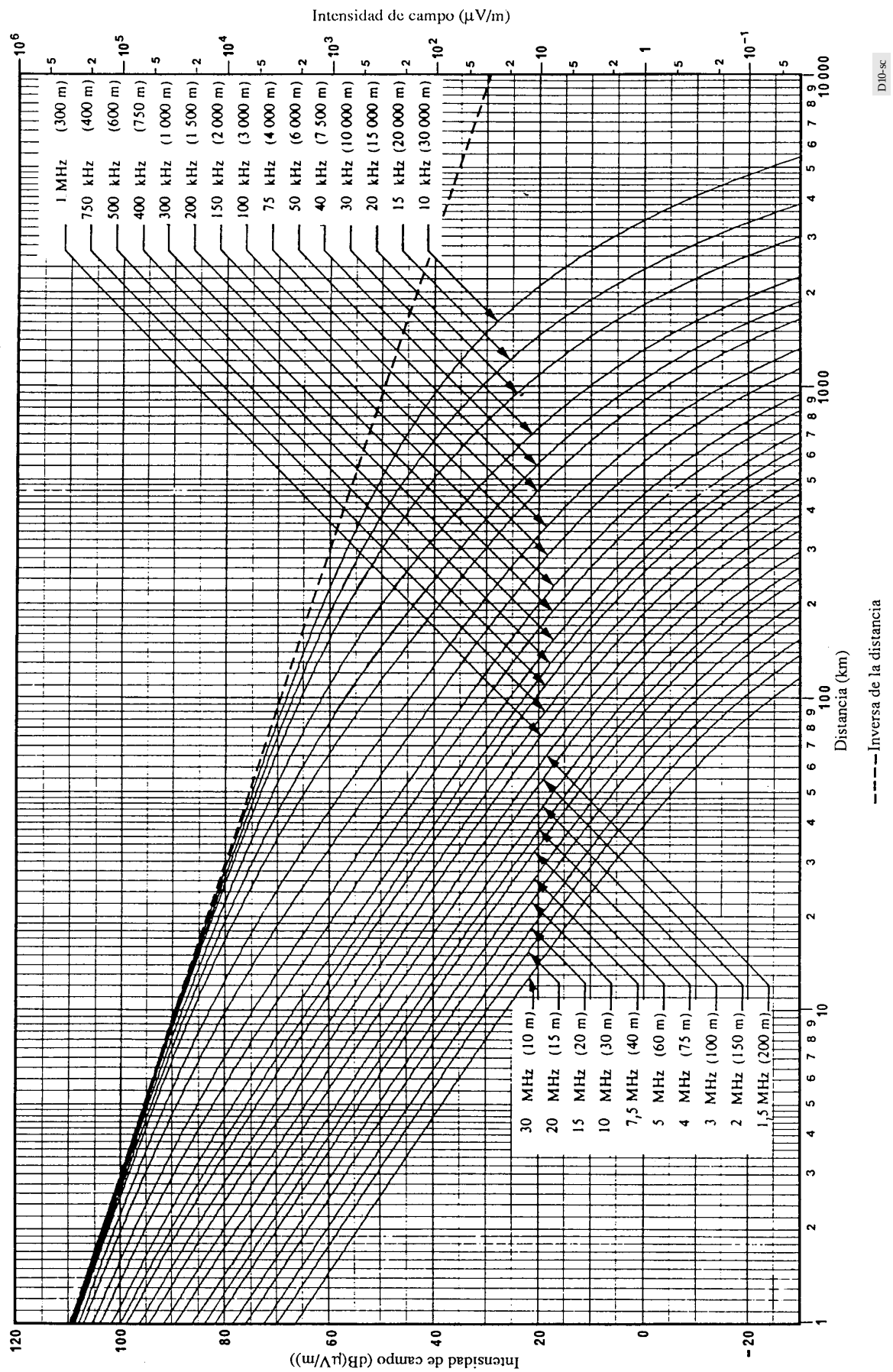


FIGURA 11

Curvas de propagación de la onda de superficie; hielo de agua dulce, -10°C , $\sigma = -10^{-5}\text{ S/m}$, $\epsilon = 3$

