"Responsabilidad con pensamiento positivo"

UNIVERSIDAD TECNOLÓGICA ISRAEL

TRABAJO DE TITULACIÓN EN OPCIÓN AL GRADO DE:

INGENIERO EN ELECTRÓNICA DIGITAL Y TELECOMUNICACIONES

TEMA:

"DESARROLLO DE UNA APLICACIÓN INFORMÁTICA, PARA EL ANÁLISIS Y BALANCE DE UN RADIO ENLACE, BAJO EL SOFTWARE MATLAB"

AUTOR

JULIO PEÑAHERRERA VICHICELA

TUTORES

MSc. Elizabeth Patricia Carrillo Armendáriz

Mg. Edgar Emanuel González Malla

AÑO: 2019

AGRADECIMIENTO

Agradezco a Dios por su protección, cuidado y por haberme dado el entendimiento necesario y la oportunidad para alcanzar esta nueva meta, a los educadores por inculcar sus conocimientos diariamente en las aulas; a mis amigos que hicieron de la universidad un lugar donde compartir buenos y malos momentos; a mi esposa Carlita, por el apoyo incondicional en cada instante, a mis hijos Matías, Thani y Sayani, por su motivación para ser un mejor ejemplo hacia ellos, a mis padres y hermanos César, Christian, Margarita y Abigail, por sus consejos a lo largo de este tiempo de formación. Mi más sincero agradecimiento por el apoyo y la ayuda que recibí de cada uno de ustedes para culminar mi carrera.

DEDICATORIA

A mi esposa e hijos por su cariño y confianza, a mis padres por su enseñanza de valores y buenos hábitos que han sido importantes para tener una buena formación como ser humano y como un buen profesional. A mis hermanos por motivarme a seguir siempre adelante a pesar de los obstáculos presentados en este camino.

CERTIFICACIÓN DEL TUTOR

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación certifico:

Que el trabajo de titulación "DESARROLLO DE UNA APLICACIÓN INFORMÁTICA, PARA EL ANÁLISIS Y BALANCE DE UN RADIO ENLACE, BAJO EL SOFTWARE MATLAB", presentado por el Sr. Julio Peñaherrera Vichicela, estudiante de la carrera de Electrónica Digital y Telecomunicaciones, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

Quito D.M. Agosto del 2019

TUTOR

……..

MSc. Elizabeth Patricia Carrillo Armendáriz

UNIVERSIDAD TECNOLÓGICA ISRAEL

APROBACIÓN DEL TUTOR

En mi calidad de tutor del componente práctico certifico:

Que el trabajo de titulación "DESARROLLO DE UNA APLICACIÓN INFORMÁTICA, PARA EL ANÁLISIS Y BALANCE DE UN RADIO ENLACE, BAJO EL SOFTWARE MATLAB", presentado por el Sr. Julio Peñaherrera Vichicela, estudiante de la carrera de Electrónica Digital y Telecomunicaciones, reúne los requisitos y méritos suficientes para ser sometido a la evaluación del Tribunal de Grado, que se designe, para su correspondiente estudio y calificación.

Quito D.M. Agosto del 2019

TUTOR

……..

Mg. Edgar Emanuel González Malla

DECLARACIÓN

Yo, JULIO PEÑAHERRERA VICHICELA, declaro que el presente trabajo es de mi autoría, que no ha sido presentado para ningún otra calificación o grado profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Tecnológica Israel, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido en la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

JULIO PEÑAHERRERA

APROBACIÓN DEL TRIBUNAL DE GRADO

Proyecto de aprobación de acuerdo con el Reglamento de Títulos y Grados de la Facultad de Ciencias de la Ingeniería de la Universidad Tecnológica Israel.

Quito, septiembre 2019

Para constancia firman:

TRIBUNAL DE GRADO

F ……..…….
PRESIDENTE

F ……..… **F……**
VOCAL **VOCAL**

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ECUACIONES

RESUMEN

El presente trabajo muestra el diseño y la implementación de un software de análisis y predicción de pérdidas de señal de radioenlaces, mediante el modelo de propagación Okumura - Hata, que ofrece las características que se acoplan a ambientes urbanos, para la programación y ejecución se utiliza como compilador al programa MATLAB R2015a con licencia estudiantil. Se define para esta aplicación la recomendación de la UIT P.529-3 que establece los datos necesarios para los modelos de propagación.

Los parámetros que se ingresan deben ser validados para transmisión y recepción, para el modelo de predicción escogido se determina que, la altura de la antena que transmite debe estar entre 30 a 200 metros, y la antena receptora de 1 a 10 metros, el rango de frecuencia aplicable es de 150 MHz y 1500 MHz. Los resultados que se muestran después de ingresar la geolocalización o coordenadas de los puntos transmisor (TX) y receptor (RX) son la distancia entre ellos, expresada en kilómetros, la predicción de la intensidad del campo y las pérdidas de la potencia entre las frecuencias del enlace. Con los valores obtenidos se podrá variar los parámetros del radioenlace para obtener un mejor desempeño. Para el análisis de resultados se realizó simulaciones con puntos ubicados dentro de la ciudad de Quito, cargando mapas SRTM dentro de la interfaz gráfica de usuario implementada.

PALABRAS CLAVE: RADIOENLACE, COBERTURA, FRECUENCIA, MODELO DE PROPAGACIÓN, OKUMURA-HATA, PÉRDIDAS, GANANCIA.

ABSTRACT

The present work is focus on the design and implementation of a software analysis and prediction of radio signal loss links, based on the Okumura - Hata propagation model, which offers the characteristics that are coupled to urban environments, for programming and execution was used MATLAB R2015a software with student license as a compiler. The ITU P.529-3 recommendation is defined for this application,

The parameters that are entered must be validated for transmission and reception, for the prediction model chosen, it is determined that the height of the transmitting antenna must be between 30 to 200 meters, and the receiving antenna of 1 to 10 meters, the Applicable frequency range is 150MHz and 1500MHz. The results shown after entering the geolocation or coordinates of the transmitter (TX) and receiver (RX) points are the distance between them, expressed in kilometers, the prediction of the field strength or the power losses between the link frequencies. With the values obtained, the parameters of the radio link can be varied so that it can have an optimal performance. For the analysis of the results, simulations were performed with points located within the city of Quito, loading SRTM maps within the implemented graphical user interface.

KEY WORDS: RADIO LINK, COVERAGE, FREQUENCY, PROPAGATION MODEL, OKUMURA-HATA, LOSSES, WINNING.

INTRODUCCIÓN

Las telecomunicaciones en el Ecuador se inician con la interconexión entre las ciudades de Quito y Guayaquil en el año de 1884, por la compañía a cargo, ALL-AMERCIAN CABLES INC. Esta misma empresa se encargaría de algunos de los países con presencia en la costa del Pacífico, tales como: Colombia, Ecuador, Perú y Chile. En el año de 1900 a través de la radiotelegrafía se logra la comunicación entre las ciudades de Quito y Guayaquil; entre 1958 y 1963, se crea y se automatiza la empresa "Telégrafos del Ecuador" entre estas dos ciudades. (NIETO, 1990)

En el año de 1970 se inicia la explotación de los servicios de telecomunicaciones Télex y Telegrafía Pública. En el año de 1971 a través de un arrendamiento de 4 canales al país de Colombia, se tiene acceso a la red INTELSAT con lo cual, Ecuador tiene acceso a la comunicación con los demás países del mundo. Luego de que la empresa Mitsubishi Corporación del Japón, construyera la estación terrena y entre en funcionamiento, (NIETO, 1990)

En el año de 1972, se decreta la Ley Básica de Telecomunicaciones por medio de un decreto del Gobierno Nacional, y se crea el Instituto Ecuatoriano de Telecomunicaciones (IETEL), con lo que se inicia regulación, planificación y control a todos los sistemas de telecomunicaciones. Con el paso de los años, los servicios de telecomunicaciones fueron adquiriendo un gran valor como factor de producción y de mucho beneficio para la sociedad debido a su gran demanda, constituyéndose en un pilar para el desarrollo de la economía ecuatoriana. (NIETO, 1990)

En la actualidad en la Constitución del Ecuador, (Art. 313), se considera como un sector estratégico a las telecomunicaciones. Al espectro radioeléctrico como un recurso natural de propiedad del estado (Art. 408). Todas las personas tienen el derecho al uso de las frecuencias del espectro radioeléctrico para gestionar servicios de telecomunicaciones y explotación de

redes inalámbricas bajo un control y asignación equitativa por parte del estado tomando en cuenta el interés colectivo (Art. 16 y 17). (ARCOTEL, 2015)

El organismo que regula a nivel internaciones los estándares anexado a la ONU, es la UIT, del cual el Ecuador forma parte y recibe todas las recomendaciones para poder desarrollar, regular y explotar las telecomunicaciones, como por ejemplo para la elaboración de un radioenlace. La UIT, emite las recomendaciones recogidas en las Conferencias Mundiales de Radiocomunicaciones, donde se describe todos los elementos que se deben considerar para su construcción, además de los modelos matemáticos que deben usarse con las características para una predicción de trayecto punto a punto. (ITU, 2015)

Con estos antecedentes, se considera el desarrollo de una aplicación informática, para el balance de un radioenlace, Okumura – Hata, que posee cálculos matemáticos con más consideraciones en la predicción de pérdidas en ambientes urbanos.

Utilizando MATLAB, como herramienta para el desarrollo de cálculos matemáticos del modelo Okumura-Hata, para cada punto de terreno seleccionado y la lectura de mapas en el formato de imagen. Se propone realizar la programación de las fórmulas para el cálculo y el ingreso de datos y se obtendrá una estimación aproximada de la pérdida de potencia de un radioenlace.

Para el desarrollo del análisis de las pérdidas en la transmisión, se tomarán en cuenta los siguientes parámetros: Ganancia de la antena de transmisión y de recepción, la frecuencia de transmisión, la distancia en kilómetros que existe desde punto que transmite Tx y el que recibe Rx, la altura de las antenas, potencia de transmisión, pérdidas en la transmisión y recepción, el umbral de recepción (sensibilidad del receptor), Zona de Fresnel y Perfil del terreno.

Además, los tipos de pérdidas en la transmisión a tomarse en cuenta son los siguientes: Pérdidas en el cable, acoples y desvanecimiento (factor de rigurosidad, factor climático y factor de confiabilidad), pérdida en espacio libre basadas en el modelo Okumura- Hata.

Con el desarrollo de una aplicación para el balance de un radioenlace, a través de del sistema algebraico computacional de matrices MATLAB, se la realizará en escenarios

simulados por el usuario, a través de interfaces gráficas, donde se ingresarán las ubicaciones geográficas en la ciudad de Quito, de los puntos que conforman el radioenlace. Además, se deberán ingresar los parámetros de pérdida y ganancia considerados en los elementos de transmisión y recepción.

Antecedentes de la situación objeto de estudio.

Desde los experimentos e invenciones realizadas por Marconi, entre los años 1940, la tecnología del desarrollo de métodos inalámbricos nombrados como elementos radiantes de hilo, que utilizaban frecuencias entre 50 y 100 kHz para transmitir, esto causaba que las antenas sean de menor tamaño en relación a las longitudes de onda. Después surgió tríodo de De Forest, se usaban frecuencias entre 100 kHz y algunos MHz, con antenas del mismo tamaño de las ondulaciones. Luego de la Segunda Guerra Mundial se desarrollan otros elementos para irradiar así se tiene los que reflejan las ondas, las guías de ondas, bocinas, entre otros.

Para realizar una transmisión de comunicaciones móviles, es necesario tomar en cuenta su coeficiente de calidad de la comunicación, debido a que este, depende de factores como el ambiente y los obstáculos, que se encuentre en el trayecto de la comunicación y los factores climáticos, estos pueden ser edificios, montañas, túneles, lluvia, etc., Para este análisis es importante realizar una cuidadosa planificación de estas características para optimizar recursos, economizar gastos y brindar un mejor servicio.

Una aplicación de análisis de propagación, puede ayudar a evitar, estos factores perjudiciales para los sistemas de comunicaciones móviles, los cuales usan frecuencias muy altas en el espectro, el cual se encuentra congestionado de bajas frecuencias.

Existen simuladores para radioenlaces como Radio Mobile, que es un software de libre descarga, utilizado para simulación, creado por Roger Coudé, basado en el algoritmo de Longley-Rice. Este programa, utiliza datos digitales de elevación del terreno para generar la simulación del trayecto entre un emisor y un receptor, junto a otros datos relativos al entorno y

a las características técnicas de los equipos, sirven para alimentar un modelo de propagación de las ondas de radio para la predicción de área de propagación de los sistemas de radiocomunicaciones en una frecuencia comprendida entre los 20 MHz y los 20 GHz. (BROWN, 2006)

La mayor parte de estudios y de desarrollos de aplicaciones están en universidades locales, utilizan Radio Mobile por ser una aplicación de libre acceso, para realizar simulaciones y proyectos.

En la investigación realizada se pudo observar, que existen aplicaciones basadas en MATLAB, por ejemplo, en la Universidad Carlos III de Madrid, se utiliza una aplicación para graficar determinadas coberturas en redes VSAT. (Álvarez, 2014)

En la Escuela Politécnica del Litoral (ESPOL), se tiene una aplicación para graficar el enlace Wimax de dos puntos en la ciudad de Manta. (Bravo, 2008)

En la Universidad de las Fuerzas Armadas- ESPE, se encuentra un desarrollo de software para el análisis de modelos de propagación en las bandas 850 y 1900 MHz. Por lo que no se ha encontrado una aplicación basada en Matlab para el análisis y balance de un radioenlace con la utilización basado en los cálculos matemáticos de Okumura – Hata, en las bandas entre 150 MHz y 1500 MHz. (BUSTAMANTE, 2007)

La Escuela Politécnica del Chimborazo (ESPOCH), se ha realizado un modelo estadístico de propagación para la banda de radiodifusión FM, aplicado en la ciudad de Riobamba. (Gabriela Noemi Nartínez Jara, 2016)

Un relevamiento de simuladores 3G – UMTS, donde se generan resultados gráficos con el uso de Matlab y Octave, se encuentra publicado por el Instituto de Computación de la Facultad de Ingeniería de la Universidad de la República en Montevideo Uruguay. (Raúl Hartmam, 2008)

Existe un Estudio y Propuesta de un modelo de Propagación para las señales de TV digital en la ciudad de Lima, en un software desarrollado en Matlab para una estimación de coberturas. (Herrera, 2008)

Existen otros programas de simulación de libre acceso en línea, como Xirio online, y otros programas de pago, que tiene acuerdos con los fabricantes de antenas y equipos de radioenlaces como por ejemplo Pathloss, RadioGIS, ATDI ICS Telecom, NETBEANS IDE, APPSERV, Linkplanner posee su versión de prueba por tiempo limitado, que luego solicita la suscripción con costo y por cada licencia básica tiene un valor desde los USD $ 350 hasta la licencia completa con un valor de $5000 anuales.

Planteamiento del Problema

Debido al crecimiento de edificaciones en la parte urbana de las ciudades se ha visto la necesidad del estudio sobre las comunicaciones inalámbricas y su demanda en esto ambientes urbanos. Radio Mobile utiliza el modelo matemático de propagación ITM, Irregular Terrain Model (Longley-Rice), para la predicción de la propagación de un radioenlace en ambientes abiertos.

Este modelo se aplica a sistemas punto a punto en un rango de frecuencias de los 20 MHz a los 20 GHz, sobre diferentes tipos de terreno utilizando la misma lógica que el modelo 2 rayos, las pérdidas por transmisión son predichas, usando la geometría de propagación según el perfil del terreno y la refractividad con la tropósfera.

La desventaja de este modelo hace que, la aplicación de Radio Mobile no tenga previsto, la determinación de correcciones debido a factores ambientales en las proximidades del receptor, no considera el efecto de la multitrayectoria de las ondas electromagnéticas, causado por obstáculos como edificios y árboles por lo que muestra una debilidad al realizar predicciones en ambientes urbanos.

Otro inconveniente de Radio Mobile es que, a pesar de ser de distribución gratuita para radioaficionados y al uso humanitario, no tienen su código accesible, lo que imposibilita estudiar cómo funciona o llevar a cabo modificaciones, solamente trabaja con ciertos parámetros que fueron tomados en cuenta según la realidad de cada región o país del desarrollador en el cual se encuentra ubicado. Por este motivo, al realizar una simulación para la orografía del Ecuador, no se encuentra con la realidad. (VE2DBE, 1997)

Link Planner utiliza para su predicción la recomendación UIT-R 12 P530 y el modelo Vigants Barnett que, toma en cuenta los factores climáticos y ambientales propios de Estados Unidos. Este software es propietario y tiene un período de prueba gratis, después de cuál se debe adquirir la versión completa y solo se utilizan equipos propietarios de la marca Cambium Networks. Al utilizar una versión de prueba las simulaciones están limitadas por día, si superamos dicha cantidad deberemos esperar 24 horas y si no las modificamos en tres días, estas simulaciones se borran. Esta limitación también afecta a la elección de diferentes antenas y materiales de uso real en radioenlaces.

La herramienta informática a diseñarse, al igual que las actuales herramientas de planificación, utiliza mapas digitales del terreno, que se basan en sistemas de información geográfica (Global Information System GIS), para poder realizar los cálculos de propagación y también permite plantear en un futuro, la actualización para mejorar e incrementar la potencialidad del software, que servirá al usuario, para puedan aumentar sus conocimientos en el área de sistemas de radiocomunicación de una manera práctica.

Al realizar un cálculo de un radioenlace, es muy importante la simulación a través de un modelo matemático que sea diferente a los expuestos anteriormente debido a que se realizará una predicción de pérdidas mediante un modelo matemático que se adapte a la ciudad de Quito. Para esto se debe tomar en cuenta, el perfil del terreno de la zona a modelar y la presencia de obstáculos como edificaciones, árboles, etc.

Justificación

Al realizar el desarrollo de una aplicación para analizar la propagación de ondas para el balance de un radioenlace, se busca optimizar los recursos económicos ya que, con la ayuda del cálculo de los equipos de un radioenlace, se aproxima mucho a la realidad y a sus características. Podrá ser una herramienta de código abierto, para quien desee utilizarla, además de poner en práctica los conocimientos adquiridos por estudiantes de ingeniería en telecomunicaciones en la asignatura de transmisión y recepción de señales radioeléctricas.

Objetivo General

Desarrollar una aplicación para el análisis de la propagación de ondas, para el balance de un radioenlace, mediante el software MATLAB.

Objetivos Específicos

Obtener conocimientos de programación en MATLAB para poder codificar funciones de diferentes propiedades como pueden ser el pedir parámetros por pantalla, que estos se guarden, mostrar resultados y codificar operaciones matemáticas según el modelo Friis.

Diseñar la aplicación, mediante el desarrollo de una interfaz gráfica utilizado el software MATLAB licenciado, que permita simular en función de una señal de entrada, potencia de transmisión, ganancia de antenas, pérdidas en la transmisión y obtener una señal resultante de recepción con el uso del modelo matemático de espacio libre de Friis.

Realizar un código que implemente el modelo de predicción Okumura – Hata, utilizando parámetros que configura el usuario.

Ampliar el código para que pueda ser lo más realista posible, en este caso con la obtención del perfil del terreno de forma real solo indicando las posiciones en las que se encuentra la antena transmisora y la receptora.

Implementar la aplicación desarrollada en un computador de prueba con sistema operativo comercial como Windows.

Evaluar mediante 2 simulaciones en la aplicación, el funcionamiento de la aplicación implementada para el balance de un radioenlace comprendido entre 150 MHz y 1500 MHz.

Alcance

El desarrollo de una aplicación informática, para el balance de un radioenlace, Okumura – Hata se lo realizará en escenarios para la ciudad de Quito, el modelo de propagación que se aplica, posee cálculos matemáticos con más consideraciones en su desarrollo, como la reducción en la señal a nivel de la azotea, debido a las columnas de los edificios, por las superficies de las terrazas y alturas de los edificios, además las pérdidas por difracción de la señal que viaja desde las azoteas de los edificios hasta las calles para ambientes urbano, urbano denso y rural.

MATLAB, se encuentra lista para el desarrollo de cálculos matemáticos del modelo Okumura-Hata para cada punto de terreno seleccionado y la lectura de mapas en el formato de imagen. Con la programación de las fórmulas para el cálculo y el ingreso de datos, se obtendrá una estimación aproximada de la pérdida de potencia de un radioenlace.

Para el desarrollo del análisis del balance de transmisión, se tomarán en cuenta los siguientes parámetros:

- Frecuencia de transmisión
- Ganancia de las antenas
- Umbral de recepción
- Distancia entre el transmisor Tx y el receptor Rx
- Altura de las antenas
- Potencia de transmisión

- Pérdidas en la transmisión
- Zona de Fresnel
- Perfil de terreno

Los tipos de pérdidas en la transmisión a tomarse en cuenta son los siguientes:

- Pérdidas por cables y acoples en el transmisor y receptor
- Pérdidas por desvanecimiento (factor climático, rigurosidad y de confiabilidad)
- Pérdida en espacio libre para el modelo de propagación escogido Okumura- Hata.

Con el desarrollo de una aplicación para el balance de un radioenlace, a través de del sistema algebraico computacional de matrices MATLAB, se la realizará en escenarios simulados por el usuario para la ciudad de Quito, a través de una interfaz gráfica, donde se ingresarán las ubicaciones geográficas de los puntos que conforman el radioenlace. Se utilizará el mapa SRTM, donde se encuentra la ciudad de Quito, para que sean procesado en la aplicación y se muestre el perfil topográfico. Se deberán ingresar los parámetros de pérdida y ganancia considerados en los elementos de transmisión y recepción.

A través de las fórmulas se procederá al análisis de matrices que se aplican en MATLAB, se obtendrá los resultados de potencias de transmisión y recepción, las gráficas de la línea de vista y los umbrales de la primera zona de Fresnel, resultados de la sensibilidad de recepción con la recomendación UIT.530. Posteriormente y tendrá la posibilidad de analizar el balance de radioenlaces, con un diseño de la interfaz gráfica.

Esta aplicación solo será considerada para el análisis para el balance de radioenlaces, donde se tomará en cuenta también mapas de conductividad bajo el modelo matemático Friis,

De tal forma al concluir el presente trabajo de titulación se presentará:

- Diseño, elaboración y evaluación del funcionamiento de una aplicación informática, para el análisis y balance de un radioenlace punto a punto, en un rango de frecuencia de 150 MHz y 1500 MHz para la ciudad de Quito, con la utilización de antena tipo directiva o Yagui.
- Un Manual de usuario dentro del capítulo de Resultados.
- Análisis de resultados con de dos radioenlaces.

Descripción de los Capítulos

CAPÍTULO 1

Este capítulo contiene el estado del arte y la documentación científica con conceptos y definiciones básicas para un diseño de un radioenlace, como pérdidas, ganancias y potencias.

CAPÍTULO 2

En este capítulo se indica las Metodologías de Investigación utilizadas para el diseño e implementación de la interfaz gráfica para el programa que permitirá analizar los radioenlaces, con la utilización del modelo de propagación de Okumura - Hata.

CAPÍTULO 3

En este capítulo se muestra la propuesta para obtener los resultados propuestos en la implementación, la descripción de los requerimientos técnicos mínimos de instalación del software propuesto, el cronograma y costo del desarrollo. También se describen el esquema y el diagrama de flujo con el que se elaborarán las pantallas que permitirán ingresar los datos y mostrar los resultados del perfil topográfico del mapa con los puntos del enlace, así como las zonas de Fresnel y valores del enlace.

CAPÍTULO 4

En este capítulo se describen el diseño, y desarrollo de la programación de los formularios de ingreso de datos, cálculo de pérdidas y potencias, resultado gráfico de resultados, con la discusión e interpretación de las simulaciones realizadas.

Finalmente se presentan las conclusiones, recomendaciones del desarrollo de esta aplicación, trabajos futuros y anexos.

- Manual de Usuario ver ANEXO 1
- Código desarrollado para la aplicación ver ANEXO 2

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

1. Conceptos relevantes para el estudio y análisis de transmisión y recepción de señales

Al realizar el estudio y análisis de un radioenlace, es necesario realizar la simulación a través de un modelo matemático. Para esto se debe tomar en cuenta ciertos factores que intervienen en dicho modelo, como la distancia entre el transmisor y receptor, los tipos de antenas y su altura, la frecuencia con la que se realizará la transmisión, las pérdidas y ganancias que se presentan, así como el tipo de clima. Los tipos de pérdidas y de ganancias considerados para el cálculo de radioenlaces se detallan a continuación. (Tomasi, 2003)

1.1. Ganancia

El término como se indica, es todo valor que aumente la potencia de un sistema de radioenlace, esta ganancia es de dos tipos:

Ganancia directiva: La ganancia directiva es una característica de la antena que nos describe cómo ésta reparte en el espacio la potencia que se le entrega. Es la división de la cantidad de potencia que se irradia en un direccionamiento único entre la densidad de potencia irradiada al punto en común realizada con la misma antena de referencia que normalmente se la toma respecto a una antena isotrópica, se supone que las dos antenas estén irradiando el mismo flujo de potencial. (Tomasi, 2003)

$$D = \frac{P}{P_{ref}} \qquad (1)$$

Ecuación. 1. 1 Ganancia antena directiva

Fuente: (Tomasi, 2003)

Ganancia de potencia: Es el resultante de la ganancia directiva con la excepción que se usa toda la potencia que energizará a la antena sin considerar la existencia de pérdidas.

$$Ap = Dn \qquad (2)$$

Ecuación. 1. 2 Ganancia de potencia

Fuente: (Tomasi, 2003)

1.2. *"Potencia isotrópica irradiada equivalente (PIRE)".*

La PIRE es la intensidad de energía semejante a la que debe difundir una "*antena isotrópica"* para que dicha intensidad tenga el mismo nivel de otra antena en la dirección elegida en una ubicación determinada.

Para ejemplificar, si una antena que transmite tiene un rendimiento de 10, la antena irradiará 10 veces más respecto a una antena isotrópica.

Para el cálculo del valor de PIRE, se multiplica el valor de la potencia con la ganancia relacionada con la antena isotrópica y se define como la potencia equivalente de transmisión con la ecuación:

$$\text{PIRE} = \mathrm{P_{rad}}\,\mathrm{D_t} \qquad (3)$$

Ecuación. 1. 3 Potencia isotrópica irradiada equivalente

Fuente: (Tomasi, 2003)

Donde Prad es la potencia radiada (watts), Dt es la ganancia directiva. (Tomasi, 2003)

1.3. Pérdidas por espacio libre

Es la que produce una onda electromagnética cuando se propaga en el espacio vacío, y su aumento es directamente proporcional a la distancia y la frecuencia. Para muchos fines la pérdida en la trayectoria por encima de los 10 GHz se puede considerar como pérdida de espacio libre (Tomasi, 2003):

$$Lbf = 32{,}4 + 20 \log f\,(MHz) + 20 \log D\,(Km) \;\; |dB| \qquad (4)$$

Ecuación. 1. 4 Pérdidas por espacio libre

Fuente: (Tomasi, 2003)

1.4. ERP (Effective Radiated Power): potencia efectiva de radiación

Se denomina como potencia efectiva de radiación a la suma de la ganancia de la antena con la potencia de transmisión en un escenario ideal sin que se tome en cuenta las pérdidas del sistema. Las características de ERP son la polarización y la refractividad. (Salazar, 2019)

Polarización: puede ser polarización vertical u horizontal pero este modelo asume que tanto el transmisor como receptor tienen la misma polarización.

Refractividad: determinan el "beding" o curvatura de las ondas de radio, durante una transmisión, también se le puede incluir la curvatura efectiva de la tierra, normalmente 4/3=1.333. (Salazar, 2019)

Se puede especificar de 3 formas distintas de refractividad:

1. Se introduce el valor de refractividad directamente, normalmente durante el rango de 250 a 400 correspondiente a la curvatura de la tierra 1.232 a 1.767.
2. Una curvatura efectiva de la tierra de 4/3 = 1.333 refractividad de superficie de valor aproximadamente 301 Unidades de n.
3. Longley y Rice recomienda un valor para condiciones atmosféricas promedio. Se dice que la onda está en condiciones de k = 4/3, que es el valor para una atmósfera estándar, ya que de acuerdo a valores experimentales se encontró que éste era el valor medio. (Salazar, 2019)

$$N_b = 179.3 * \ln\left[\frac{1}{0.046665}\left(1 - \frac{1}{k}\right)\right] \qquad (5)$$

Ecuación. 1. 5 de Refractividad

Fuente: (Salazar, 2019)

1.5. Fenómenos que afectan la propagación de señales

Un presupuesto de potencia para un enlace punto a punto se define como el cálculo de ganancias y pérdidas desde el radio del transmisor o fuente de la señal que emite la onda electromagnética, a través de cables, conectores o el espacio libre hacia el receptor. (Salazar, 2019)

1.5.1 Pérdidas por alimentación

Las pérdidas en el cable, conectores y diversidad, se las conoce como pérdidas por alimentación. Las pérdidas por acoples, por lo general se toma en cuenta a un conector en cada extremo de la guía de onda, en este caso se utilizará cable coaxial para las simulaciones. El valor aproximado para el cálculo de la pérdida por distancia de cable se la realizará tomado en cuenta la altura de la torre con el aumento de 20m. (Morocho, 2011)

Las pérdidas por diversidad se relacionan con la frecuencia que se realizará la transmisión como se muestra a través del siguiente cuadro de referencia.

Tabla 1.1 Pérdidas por Alimentación

PÉRDIDAS POR ALIMENTACIÓN

Aliment.	Banda de operación GHz	Atenuación Específica dB/100 m	Pérdida por diversidad dB	Pérdida por acoples dB
coaxial	Hasta 0,9	3,00	2	1,2
	0,9 - 1,5	4,80		
	1,5 - 1,9	5,00		
	1,9 - 2,2	5,40		
	2,2 - 2,4	5,80		
Guía Onda	2,4 - 3,1	1,40	4	0,6
	3,1 - 4,4	2,10		
	4,4 - 6,2	3,60		

Fuente: (Morocho, 2011)

1.5.2 Atenuación

Fenómeno que tiene por defecto la pérdida de energía de una señal debido a la distancia, esto ocurre mientras más se aleje la onda de la fuente, las ondas se alejan cada vez más entre sí, por lo tanto, la cantidad de ondas por unidad de área es menor, por lo cual se puede decir que no se pierde o se disipa la potencia, la onda solo se distribuye sobre un área mayor disminuyendo la densidad de potencia conforme incrementa la distancia y se lo llama atenuación de la onda. Por tal razón, para que la señal llegue con la suficiente energía es necesario el uso de amplificadores o repetidores. Además, la atenuación se incrementa a medida que se incrementa la frecuencia, temperatura y el tiempo. (Salazar, 2019)

1.5.4 Difracción

Es la distribución de la energía en el frente de onda que viaja por el borde de un obstáculo, dicho fenómeno es el responsable de permitir que una onda se propague en torno a las esquinas de tal objeto, como lo describió Huygens todo punto que esté delante de la onda se puede indicar que es un nuevo origen secundario de ondas, tal como se aprecia en la Figura 1.1, las anulaciones de cada ondulación no es total y se realiza en el borde del material, permitiendo que las nuevas ondulaciones se filtren en los bordes del objeto y en la parte donde no existen anulaciones se llama zona de sombra. (Tomasi, 2003)

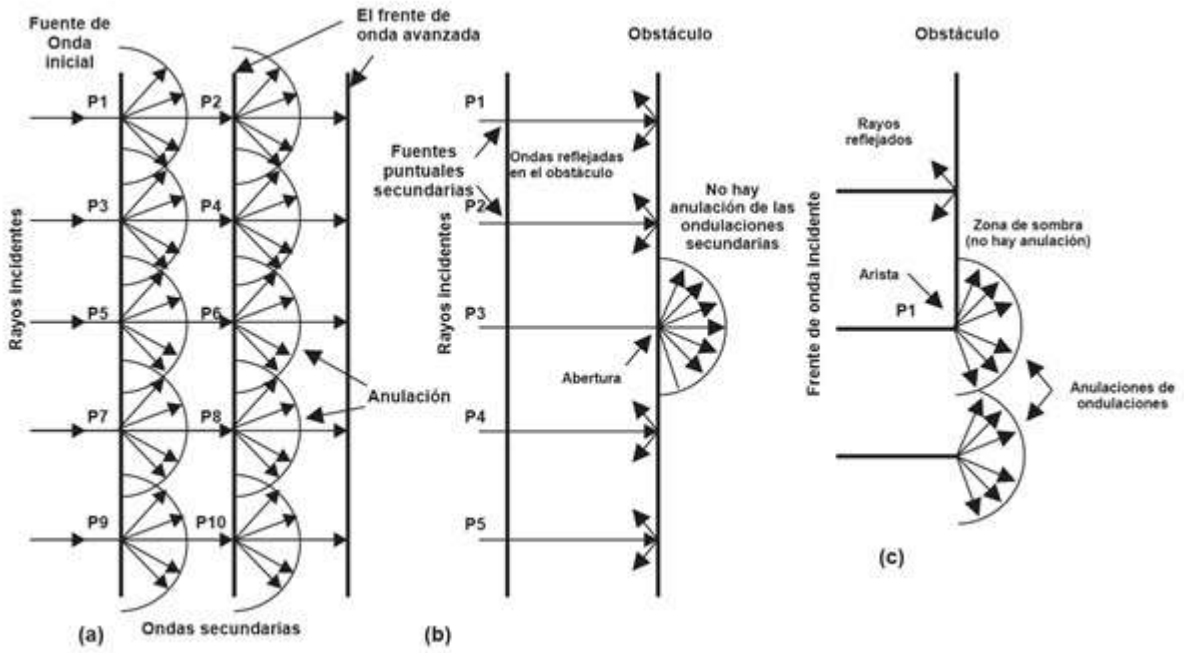


Figura. 1. 1 Difracción de ondas electromagnéticas

Fuente, (Tomasi, 2003)

1.5.5 Refracción y Reflexión

La refracción es un fenómeno que se presenta cuando una onda se encuentra con un obstáculo en su trayectoria, por lo que la onda cambiara de dirección al ser reflejada en un plano perpendicular a la superficie con la cual colisionó, este fenómeno se presenta cuando existe un cambio o la separación entre dos medios físicos por donde viaja la onda. (Tomasi, 2003)

La reflexión se puede causar por diferentes efectos emitidos o producidos por fuente de sonido que se reflejan en las paredes y los objetos a su alrededor.

Este principio de reflexión es muy utilizado para dirigir la radiación y concentrarla a un receptor, un ejemplo se puede ver en la Figura 1.2, en la cual se muestra a la onda incidente pasar por un obstáculo y dividirse en una onda reflejada y en una onda refractada. (Tomasi, 2003)

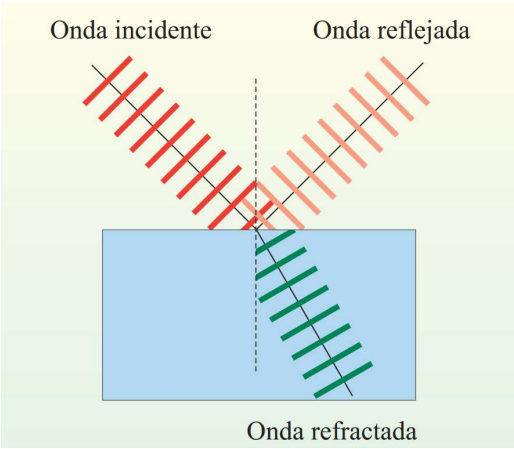


Figura. 1. 2 Modo de reflexión.

Fuente: (Tomasi, 2003)

1.5.6 Margen de desvanecimiento

Se define como una pérdida adicional que se debe tomar en cuenta en las condiciones previamente consideradas en las pérdidas de transmisión, este término se aplica a variables que afectan a los cambios en la pérdida de trayectoria entre el transmisor en una estación, y su receptor normal en otra estación. En el margen de desvanecimiento se están considerando las pérdidas intermitentes en la intensidad de la señal provocadas por perturbaciones meteorológicas, como la lluvia, nieve, trayectos múltiples de transmisión y por la superficie irregular de la Tierra que afectan la propagación de las ondas electromagnéticas. El margen de desvanecimiento se incluye en la ecuación de ganancia para considerar características no ideales y no tan predecibles en la propagación de ondas de radio. (Tomasi, 2003)

Además, el margen de desvanecimiento nos permite tener en cuenta los objetivos de confiabilidad del sistema. El cálculo de este margen se lo puede obtener de una forma teórica con la siguiente ecuación:

$$F_m = 30\log(d) + 10\log(6 * A * B * f) - 10\log(1 - R) - 70 \quad (6)$$

Ecuación. 1. 6 Ecuación de desvanecimiento

Fuente: (Tomasi, 2003)

Donde:

F_m = Margen de desvanecimiento [dB]

$1 - R$ =0.00001 (objetivo de confiabilidad del enlace)

d = Distancia del transmisor al objetivo [Km]

A = Factor de rugosidad:

- 4 si el terreno es plano o agua.
- 1 para un terreno promedio.
- 0.25 para un terreno rugoso. (Tomasi, 2003)

B = Factor climático:

- 0.5 zonas calientes y húmedas.
- 0.25 zonas intermedias.
- 0.125 para áreas montañosas o muy secas.

f = Frecuencia [GHz]. (Tomasi, 2003)

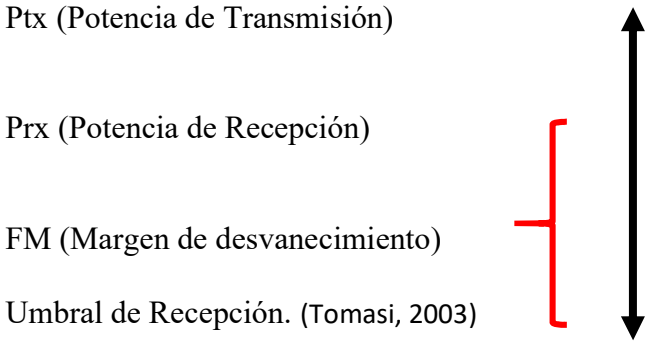
También puede se puede calcular el margen de desvanecimiento real, dado por diferencia del valor absoluto del umbral de recepción y el valor absoluto de la potencia de recepción:

$$FM = |Umbral\ de\ recepción| - |\ Potencia\ de\ recepción| \quad (7)$$

Ecuación. 1. 7 Ecuación de desvanecimiento Real

Fuente: (Tomasi, 2003)

Esto se puede ilustrar en la siguiente escala:



1.6. Zona de Fresnel

Se denomina zona de Fresnel a la cantidad de espacio existente entre el punto de emisión de un flujo electromagnético y un punto de recepción, de tal manera que esta cantidad de espacio tenga un desfase de ondas que no alcance más de 180 grados. (Tomasi, 2003)

La noción de zonas de Fresnel es muy útil en las transmisiones radioeléctricas punto a punto, para las cuales un trayecto sin obstáculos, o con ellos, tienen una influencia determinante al establecer el margen sobre obstáculos que se calcula con relación al radio de la primera zona de Fresnel. (Tomasi, 2003)

La primera zona de Fresnel, es un elipsoide de revolución, cuyos puntos focales están en los extremos del tramo donde se ubican las antenas, un ejemplo se puede observar en la Figura 1.3.

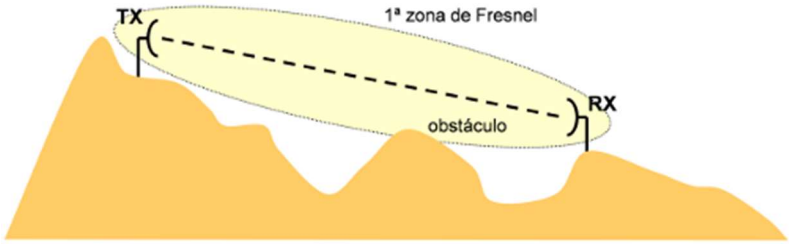


Figura. 1. 3 Gráfica de la primera zona de Fresnel.

Fuente: (Tomasi, 2003)

Para realizar el cálculo de la primera zona de Fresnel, se utiliza la siguiente expresión:

$$R = 17.32\sqrt{\frac{d1 * d2}{d * f}} \qquad (8)$$

Ecuación. 1. 8 Ecuación primera zona de Fresnel

Fuente: (Castro, 2017)

Donde:

R = Radio en metros.

f = frecuencia utilizada en GHz. (Castro, 2017)

d1 = distancia al obstáculo desde el transmisor (km).

d2 = distancia entre el receptor y el obstáculo (km).

d = distancia total entre el emisor y el receptor (km)

Para realizar el cálculo de libertad de la primera zona de Fresnel también se utiliza la siguiente fórmula:

$$F1 = 548\sqrt{\frac{d1 * d2}{f * d}} \quad (9)$$

Ecuación. 1. 9 Ecuación 2 primera zona de Fresnel

Fuente: (Quinzo, 2012)

Donde:

F1 = Radio de la primera Zona de Fresnel (m).

f = frecuencia utilizada en MHz.

d1 = distancia a un extremo del trayecto (m).

d2 = distancia entre el receptor y el obstáculo (m).

d = distancia total entre el emisor y el receptor (m). (Quinzo, 2012)

1.7. Sensibilidad del Receptor

También se lo llama umbral de recepción, es el valor mínimo de señal de radio frecuencia que debe ser detectado a su ingreso al receptor, con el que aún producirá una señal de información demodulada que pueda ser útil. (Tomasi, 2003)

La calidad de la señal que se recibe, es la relación de señal a ruido y la potencia de la señal que se emite y si esta se puede utilizar o no. Como un concepto más amplio se puede decir que la sensibilidad identifica el valor mínimo de potencia necesario para decodificar bits lógicos y

lograr alcanzar una cierta tasa de bits. En esta se puede observar que la tasa de transmisión depende de la sensibilidad de la tarjeta; la recepción del radio microonda es mejor mientras más baja es la sensibilidad. El valor del Umbral de Recepción típico se encuentra entre -70 y -80 dB (Tomasi, 2003)

1.8. Medios de propagación

Se los puede clasificar de la siguiente manera:

Ondas mecánicas: son perturbaciones que viajan por un material o a su vez es una sustancia que se transportan a través de un medio, un ejemplo claro de estas ondas, son: las cuerdas, barras, sonidos y fluidos. A continuación, se muestra en la Figura 1.4 un ejemplo de onda mecánica que es generada y propagad en un fluido. (Tomasi, 2003)

Figura. 1. 4 Ejemplo de onda mecánica

Fuente: (Tomasi, 2003)

Ondas electromagnéticas: son aquellas que son producidas por un campo eléctrico o lo que es similar a una región o espacio que exista electricidad, a estas las podemos evidenciar en la radio y TV, en ondas microondas y rayos X. Un ejemplo de onda electromagnética se observa en la Figura 1.5, donde se observa a una antena dipolo que emite una señal generada, su campo eléctrico, su campo magnético y la dirección de propagación de esta señal. (Tomasi, 2003)

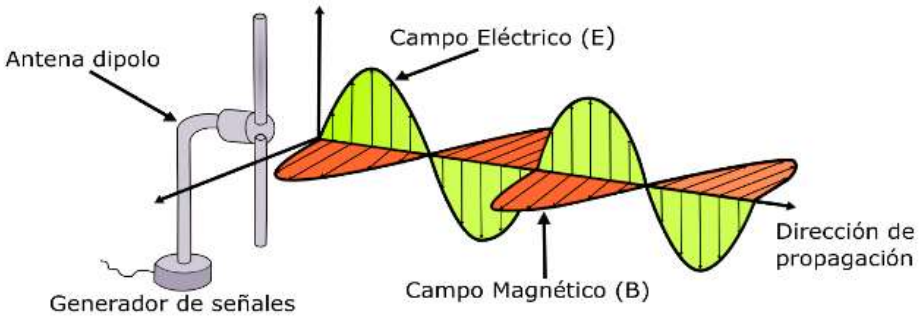


Figura. 1. 5 Modelo de onda electromagnética.

Fuente: (Tomasi, 2003)

1.9. Modelo de Propagación

Un modelo de propagación es una teoría estadística, que se basa en un conjunto de diagramas, algoritmos, operaciones sistemáticas, que permite realizar un cálculo o representaciones de un radioenlace sobre un ambiente determinado. Los modelos se clasifican en empíricos o estadísticos, teóricos o determinísticos, o en una combinación de estos, llamados semiempíricos.

La Unión Internacional de Telecomunicaciones (UIT), como organismo especializado de las Naciones Unidas, para las tecnologías de información y comunicación (TIC), realiza las recomendaciones para normalizar, estandarizar a los procedimientos y manuales como en este caso para el sector de Radiocomunicaciones, realiza recomendaciones de predicción mediante modelos de propagación. (Salazar, 2019)

1.10.1 Modelo de propagación espacio libre o Modelo de Friis

El modelo de Friis es aplicable para los enlaces en los cuales exista una clara línea de vista entre el transmisor y el receptor, este modelo es utilizado en los enlaces de microondas. También para la detección en las pérdidas en los enlaces satelitales, ya que este modelo describe y predice

los cambios en la potencia respecto a la distancia entre el transmisor y receptor y la frecuencia de operación. (Salazar, 2019)

La potencia recibida por la antena receptora en función de la distancia entre el transmisor y la misma viene dada por la ecuación 9.

$$P_r(d) = \frac{P_t * G_t * G_r * \lambda^2}{(4\pi)^2 * d^2 * L} \quad (10)$$

Ecuación. 1. 10 Potencia recibida modelo de Friis

Fuente: (Salazar, 2019)

Donde:

P_r: potencia recibida en Watts

P_t: potencia transmitida en Watts

G_t: ganancia de la antena transmisora

G_r: ganancia de la antena receptora

d: distancia entre las antenas

L: pérdidas del sistema no está relacionado con la propagación

λ: longitud de onda

Otro parámetro a tomar en cuenta es la ganancia de transmisión expresada por la ecuación:

$$G = \frac{4\pi * A_e}{\lambda^2} \quad (11)$$

Ecuación. 1. 11 Ganancia de transmisión Modelo de Friis

Fuente: (Salazar, 2019)

Donde A_e es la apertura efectiva la cual está relacionada con el tamaño físico de la antena y λ es la relación entre la velocidad de la luz y la frecuencia. (Salazar, 2019)

1.10.2 Modelo de Dos Rayos Reflexión Terrestre

En una transmisión con línea de vista directa no siempre es en línea recta ya que la mayoría de las ondas se reflejan en la tierra, lo cual da lugar a diferentes modos de propagación conocidos como multitrayectoria. El modelo de dos rayos toma en cuenta las reflexiones con la tierra, lo cual permite obtener valores aproximados a la realidad de la potencia recibida por el receptor a una cierta distancia d y se puede expresar con la ecuación 9, la descripción del modelo se puede observar en la Figura 1. 6. Donde se puede observar la relación que se tiene entre la altura del transmisor y la altura del receptor con respecto a la distancia entre estos y los ángulos de reflexión relacionados con la curvatura de la tierra. (Salazar, 2019)

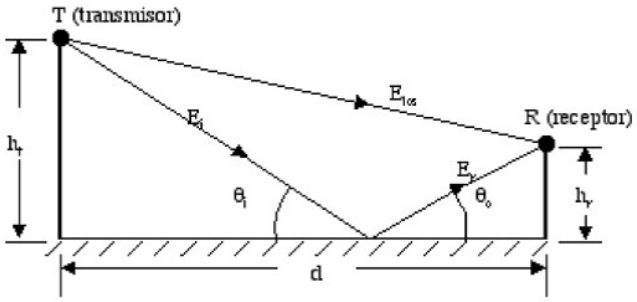


Figura. 1. 6 Modelo de dos rayos terrestres.

Fuente: (Salazar, 2019)

$$P_r(d) = \frac{P_t * G_t * G_r * h_t^2 * h_r^2}{d^4} \qquad (12)$$

Ecuación. 1. 12 Potencia de recepción del Modelo de Dos Rayos Reflexión Terrestre

Fuente: (Salazar, 2019)

Donde:

P_r: potencia recibida en Watts

P_t: potencia transmitida en Watts

G_t: ganancia de la antena transmisora

G_r: ganancia de la antena receptora

h_t: altura de la antena transmisora

h_r: altura de la antena receptora

L: pérdidas del sistema no está relacionado con la propagación

d: distancia entre las antenas

Cabe destacar que el modelo de dos rayos es utilizado cuando la distancia entre las antenas es relativamente grande ya que cuando esta distancia es pequeña los resultados son inexactos. (Salazar, 2019)

1.10.3 Modelo de Okumura

El modelo de Okumura es utilizado principalmente en transmisiones en zonas urbanas, en las cuales sus frecuencias de transmisión estén dentro del rango de 150 a 3000 MHz, es decir dentro de las bandas VHF y UHF. La distancia entre las antenas transmisora y receptora debe estar entre 30 a 1000 m. En la Figura 1.7 se puede observar las curvas de atenuación relativa al espacio libre en función a la distancia presentadas por Okumura, las cuales son utilizadas como referencia. Las pérdidas en este modelo se calculan a través de la siguiente ecuación: (Salazar, 2019)

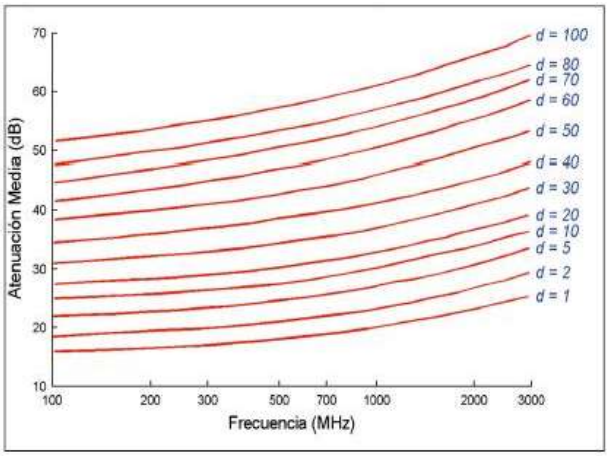


Figura. 1. 7 Curvas de atenuación relativa de Okumura.

Fuente: (Salazar, 2019)

$$L50(dB) = L_f + A_{mu}(f,d) - G(h_{tx}) - G(h_{rx}) - G_{area} \quad (13)$$

Ecuación. 1. 13 Pérdidas del Modelo de Okumura

Fuente: (Salazar, 2019)

Donde:

$L50(dB)$: atenuación mediana por trayectoria

L_f: atenuación de espacio libre

$A_{mu}(f,d)$: atenuación relativa promedio (tomada de las curvas)

$G(h_{tx})$: Ganancia de la altura de la antena transmisora

$G(h_{rx})$: Ganancia de la altura de la antena receptora

G_{area}: Ganancia debido al tipo de ambiente

Okumura descubrió que la $G(h_{tx})$ oscila en un índice de 20 dB/década y que la $G(h_{rx})$ oscila en un índice de 10 dB/década h<<3m. Por lo que las ganancias del transmisor y el receptor en función de las alturas vienen dadas por: (Salazar, 2019)

$$G(h_{tx}) = 20 * \log\left(\frac{h_{tx}}{200}\right), \text{ 30m} < h_{tx} < \text{1000 m} \quad (14)$$

Ecuación. 1. 14 Ganancia del transmisor en función de la altura

Fuente: (Salazar, 2019)

$$G(h_{rx}) = 10 * \log\left(\frac{h_{rx}}{3}\right), \qquad h_{tx} < 3 \text{ m} \quad (15)$$

Ecuación. 1. 15 Ganancia del receptor en función de la altura

Fuente: (Salazar, 2019)

$$G(h_{rx}) = 20 * \log\left(\frac{h_{rx}}{3}\right), \qquad 3\text{m} < h_{tx} < 10\text{m} \quad (16)$$

Ecuación. 1. 16 Ganancia del receptor en función de la altura con oscilación

Fuente: (Salazar, 2019)

1.10.4 Modelo Okumura-Hata

El modelo de Okumura-Hata es utilizado para las frecuencias en el rango de 150 a 1500 MHz, este modelo es uno de los más empleados para la predicción de pérdida de propagación en áreas urbanas.

Con el objetivo de hacer que este método fuera más fácil de aplicar, Hata (Hata, 1980) estableció una serie de relaciones numéricas que describen el método gráfico propuesto por Okumura. Dichas expresiones de carácter empírico, son conocidas bajo el nombre de modelo de Okumura-Hata, también llamado modelo de Hata. (XIRIOonline, 2019)

El principal resultado que proporciona el modelo es el valor mediano de la pérdida básica de propagación, en función de la frecuencia, la distancia, y las alturas de las antenas de la estación transmisora y la estación receptora, pero éste modelo no incluye ninguno de los factores de corrección por tipo de trayecto, los cuales sí están en el modelo de Okumura, las ecuaciones propuestas por Hata tienen un importante valor práctico. (XIRIOonline, 2019)

El modelo de Okumura-Hata establece lo siguiente:

•Frecuencia: 150 <f< 1500 MHz

•Altura de la antena transmisoras: 30 <hTx <200 m

•Altura de la antena receptora: 1 <hRx< 10 m (XIRIOonline, 2019)

Las pérdidas en un área urbana en este modelo se calculan mediante la siguiente expresión:

$$L_b = 69.55 + 26.16 * \log(f) - 13.82\log(h_{Tx}) - a(h_{Rx}) + (44.9 - 6.55\log(h_{Tx}))\log(d) \quad (17)$$

Ecuación. 1. 17 Pérdida de área Urbana

Fuente: (XIRIOonline, 2019)

El término $a(h_{Rx})$ es conocido como factor de corrección el cual se obtiene de:

$$a(h_{Rx}) = (1.1\log(f) - 0.7) * h_{Rx} - (1.56\log(f) - 0.8) \quad (18)$$

Ecuación. 1. 18 Factor de corrección

Fuente: (XIRIOonline, 2019)

El factor de corrección en ciudades grandes viene dado por la expresión:

$$a(h_m) = \begin{cases} 8.29(log1.54h_m)^2 - 1.1 & f \leq 200MHz \\ 3.2(log11.75h_m)^2 - 4.97 & f \geq 400MHz \end{cases} \quad (19)$$

Ecuación. 1. 19 Factor de corrección en ciudades grandes

Fuente: (XIRIOonline, 2019)

La pérdida en un área suburbana se calcula con la siguiente ecuación:

$$L_b = L_b(urbana) - 2\left[lo\left(\frac{f}{28}\right)\right]^2 - 5.4 \quad (20)$$

Ecuación. 1. 20 Pérdida de áreas Suburbanas

Fuente: (XIRIOonline, 2019)

Las pérdidas en áreas rurales se calculan mediante la expresión:

$$L_b = L_b(urbana) - 4.78(f)^2 + 18.33logf - 40.94 \quad (21)$$

Ecuación. 1. 21 Pérdida de áreas Rurales

Fuente: (XIRIOonline, 2019)

El modelo de Okumura-Hata tiene un mejor desempeño en áreas urbanas y suburbanas, pero no es así para zonas rurales, debido a que este modelo no toma en cuenta la ondulación del terreno ni la densidad de urbanización en la que se encuentre el trayecto. (XIRIOonline, 2019)

1.10.5 Modelo de Walfish Bertoni

El principal parámetro utilizado en el modelo de Walfish Bertoni es la difracción, la cual ayuda a calcular la potencia media recibida por el receptor cuando este se encuentra a nivel del suelo. El modelo además toma en cuenta el efecto producido por la altura de los edificios y por los techos de los mismos. (Salazar, 2019)

Este modelo trabaja con frecuencias en el rango de 300 a 3000 MHz, una distancia entre las antenas de 200 a 5000 m. Las pérdidas por trayectoria en este modelo se calculan mediante la ecuación siguiente:

$$S = P_0 * Q^2 * P_1 \quad (22)$$

Ecuación. 1. 22 Pérdidas por trayectoria Walfish Bertoni

Fuente: (Salazar, 2019)

Donde:

Q^2: reducción de la señal a nivel de los techos.

P_1: pérdidas por difracción en la señal.

P_0: pérdida en espacio libre de las antenas isotrópicas.

$$P_0 = \left(\frac{\lambda}{4\pi . R}\right)^2 \quad (23)$$

Ecuación. 1. 23 Pérdida en espacio libre, antena isotrópica

Fuente: (Salazar, 2019)

1.10.6 Modelo de Longley-Rice

Es un modelo utilizado en sistemas punto a punto los cuales estén en el rango de frecuencias de VHF y EHF. Los datos utilizados para la especificación de este modelo se basan en frecuencias entre 20 MHz y 20 GHz, distancias entre 1 a 2000 Km, las antenas están comprendidas entre las alturas de 0.5 y 3000m y por último las antenas tienen polarización vertical y horizontal. (Salazar, 2019)

Para predecir la potencia de la señal dentro del "horizonte" *line of sight* (LOS) se utiliza principalmente el modelo de reflexión terrestre de 2 rayos como se puede observar en la Figura 1.8. (Salazar, 2019)

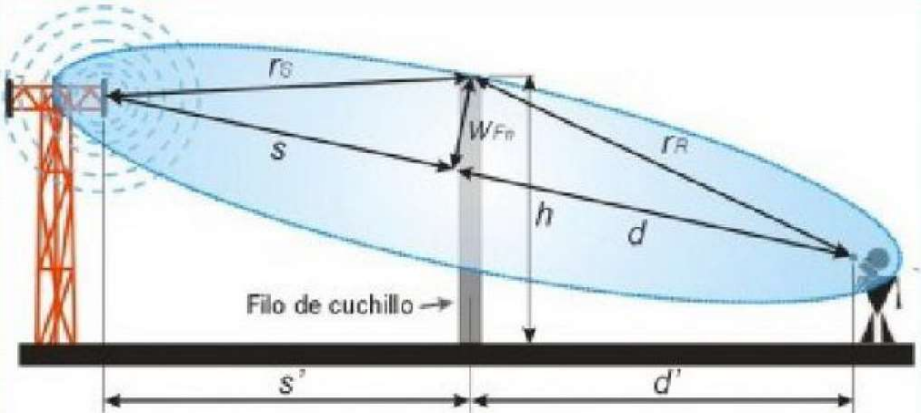


Figura. 1. 8 Fenómeno filo de cuchillo.

Fuente: (Salazar, 2019)

Además, utiliza un modelo estadístico que toma muchos parámetros de para el cálculo de pérdidas:

- Altura media del terreno (ondulación)

- Refracción de la tropósfera
- Perfiles del terreno
- Conductividad y permisividad del suelo
- Ángulos de elevación

1.10. Mapas SRTM

Es un proyecto de la NASA, *Shuttle Terrain Radar Mapping Mission* (SRTM) el cual provee una serie de datos los cuales con una altitud de precisión de 3 segundos de arco (100m).

Los mapas pueden ser utilizados con todas las informaciones de imágenes topografías incluso de carreteras o imágenes satelitales. Estos mapas se los pueden descargar gratuitamente en las páginas oficiales, como por ejemplo la siguiente figura.

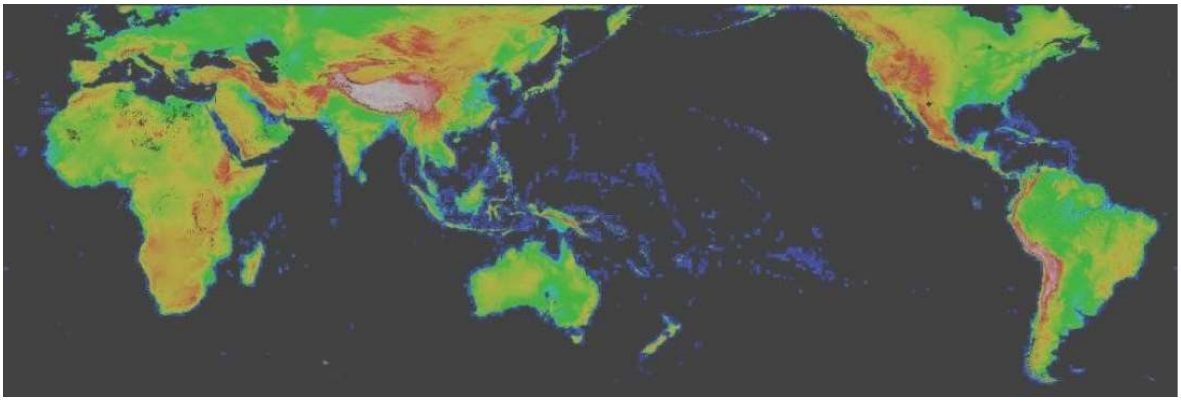


Figura. 1. 9 Elevación SRTM del mapa mundial.

Fuente: Elaborado por el autor

CAPÍTULO 2

MARCO METODOLÓGICO

2.1. Métodos de investigación

2.1.1 Método inductivo

Este método nos permite observar los problemas que se pueden presentar al momento de diseñar los elementos que conforman un radioenlace y las características que deben ser optimizadas y las condiciones que a continuación se expondrán para optimizar según las características de estos, generando una simulación que ayudará a la elección de equipos e infraestructura. Forma parte de pruebas reales de enlaces con el objetivo de buscar una proyección de una solución al diseño para el problema planteado.

El método es utilizado para presentar el problema de forma clara y que actualmente existe con el uso de herramientas para la predicción de pérdidas de potencia en radioenlaces y que se espera que sea base para modificaciones futuras que generará una mejor adaptación de esta predicción al entorno real.

2.1.2 Método deductivo

A través del método deductivo se presentará el análisis con el ingreso de los valores variables de los elementos que confirman un radioenlace a través de ecuaciones que previamente se determinarán para que se apliquen y calculen automáticamente, demostrando de esta manera

los resultados que se desean conocer para que de esta manera se pueda proponer una solución que permita establecer una comunicación óptima.

2.1.3 Método analítico sintético

Mediante este método, se muestra el problema que existe en al realizar la simulación de un radioenlace en busca de un resultado adecuado. Esto permitirá proyectar una solución útil que se acerque a la realidad a las características de los elementos que lo conforman.

2.2. Tipos de investigación

2.2.1 Investigación bibliográfica

Se utiliza este tipo de investigación debido a que el tema propuesto se enfocado en la tecnología, por lo tanto, las investigaciones que se han realizado anteriormente se enfocan en generar una proyección para dar la mejor solución al problema.

Esto permite buscar una alternativa que sea acorde a las necesidades que presenta al diseñar un radioenlace, al iniciar con desarrollos similares que puedan mostrar este tipo de soluciones.

2.2.2 Investigación explicativa

Se describe como se inicia el problema, sus razones, y muestra los fundamentos de esta necesidad que genera al diseñar un radioenlace. De esa manera buscar la simulación y las características de las pérdidas más adecuada que garantice la solución al problema planteado.

2.3. Definición de la metodología de trabajo

En este proyecto se aplica un método de investigación cualitativo y otro método proyectivo. Al utilizar el método cuantitativo, se utilizará funciones y cálculos numéricos para

el análisis de datos con el objetivo de obtener los resultados que cumplan con los requerimientos necesarios de las pérdidas de potencia entre dos puntos y que puedan aplicarse en el balance de un radioenlace.

La investigación aplicada para este trabajo es de tipo proyectiva, ya que, a través de conocimientos previos, se propone el desarrollo de una aplicación informática para el análisis de un radioenlace, mediante el software Matlab, que posee herramientas matemáticas para la obtención de resultados mediante interfaces gráficas.

Para realizar las actividades a través de una planificación metodológica, se cumplirá en Ocho (8) fases, como se muestran en la figura 2.1

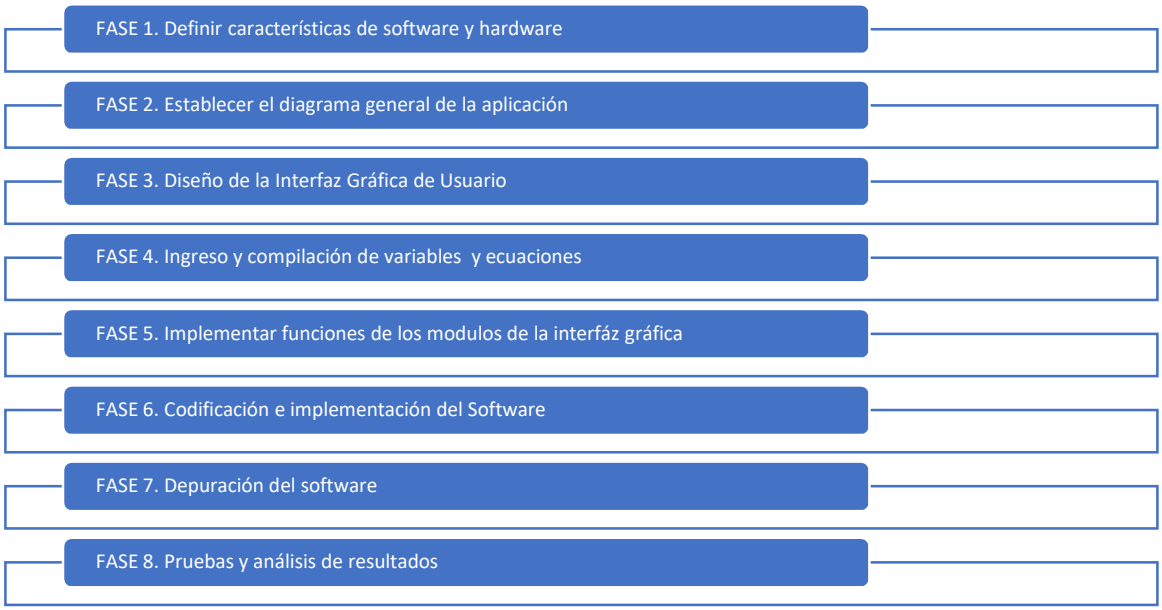


Figura 2.1 Fases para la implementación del programa.

Fuente: Elaborado por el autor

Características de Software y Hardware

En la primera parte se definirán las características de software y hardware, para esto se definirá que versión de Matlab utilizar y características del computador en el que se va compilar el programa.

Diagrama general de la aplicación

En la segunda fase se establecerá el diagrama general de la aplicación con las características que debe tener y en que unidades va trabajar.

Diseño de la interfaz gráfica de usuario.

En la tercera fase, se conformará el diseño de la interfaz gráfica de usuario, donde se definirá 3 pantallas principales para el ingreso y muestra de resultados;

Compilación de variables y ecuaciones

En la cuarta fase, se realizará la compilación de las variables y ecuaciones en el software, aplicando los modelos de propagación definidos en el alcance del proyecto.

Implementación de los módulos y funciones

En la quinta fase, se implementará los módulos y funciones de las interfaces gráficas necesarias.

Codificación e implementación del Software

En la sexta fase, se establecerá la codificación e implementación del software, tomando los datos establecidos por el usuario.

Depuración del Software

En la séptima fase, se ejecutará la depuración del software para que no existan errores de compilación y que los resultados que se obtengan, sean coherentes con los esperados.

Pruebas y análisis de resultados

En la octava y última fase se llevará a cabo las pruebas y análisis de resultados del software, para esto se probará dos radioenlaces con puntos conocidos dentro de la ciudad de Quito para evaluar si verdaderamente se tiene resultados reales.

CAPÍTULO 3

PROPUESTA

Después de describir los métodos y tipos de investigación, en esta sección se explicará los esquemas, flujos y procesos a realizarse en Matlab, tanto para transformaciones de unidades, el cálculo de la potencia de recepción en el modelo de Okumura – Hata, pérdidas en la transmisión, la obtención de la representación en dos dimensiones de las elevaciones del terreno usando los mapas SRTM y la gráfica de las zonas de Fresnel.

La elección de la utilización de Matlab se enfoca en las ventajas de ser una herramienta muy útil para el desarrollo de simuladores, al analizar datos a través de fórmulas y matrices matemáticas, además por ser un software licenciado que cuenta con el soporte en línea y los ejemplos e investigaciones realizadas por los usuarios de este software, garantizan que las operaciones matemáticas funcionen de manera correcta y sean verificadas continuamente.

3.1 Esquema de implementación

Para poder realizar el diseño de la aplicación se pensó en la facilidad que debe brindar la interfaz para el usuario, la usabilidad que tenga debe estar clara, sencilla y breve para que sin necesidad que el usuario revise todo el manual, pueda ejecutar una simulación, para que se transforme en una aplicación intuitiva para el usuario.

El diseño se planteó con la ejecución de 3 pantallas principales, la primera es una pantalla que permite ingresar las coordenadas de ubicación de los emplazamientos del transmisor y

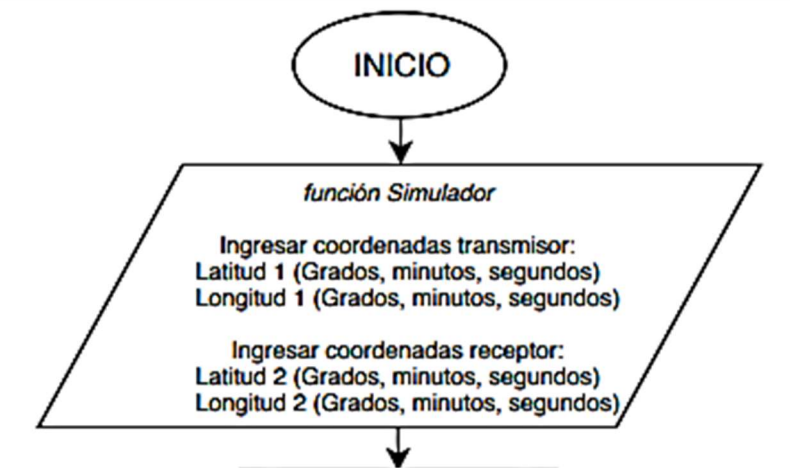
receptor, esta pantalla también ya permite mostrar el resultado de la distancia entre los dos puntos.

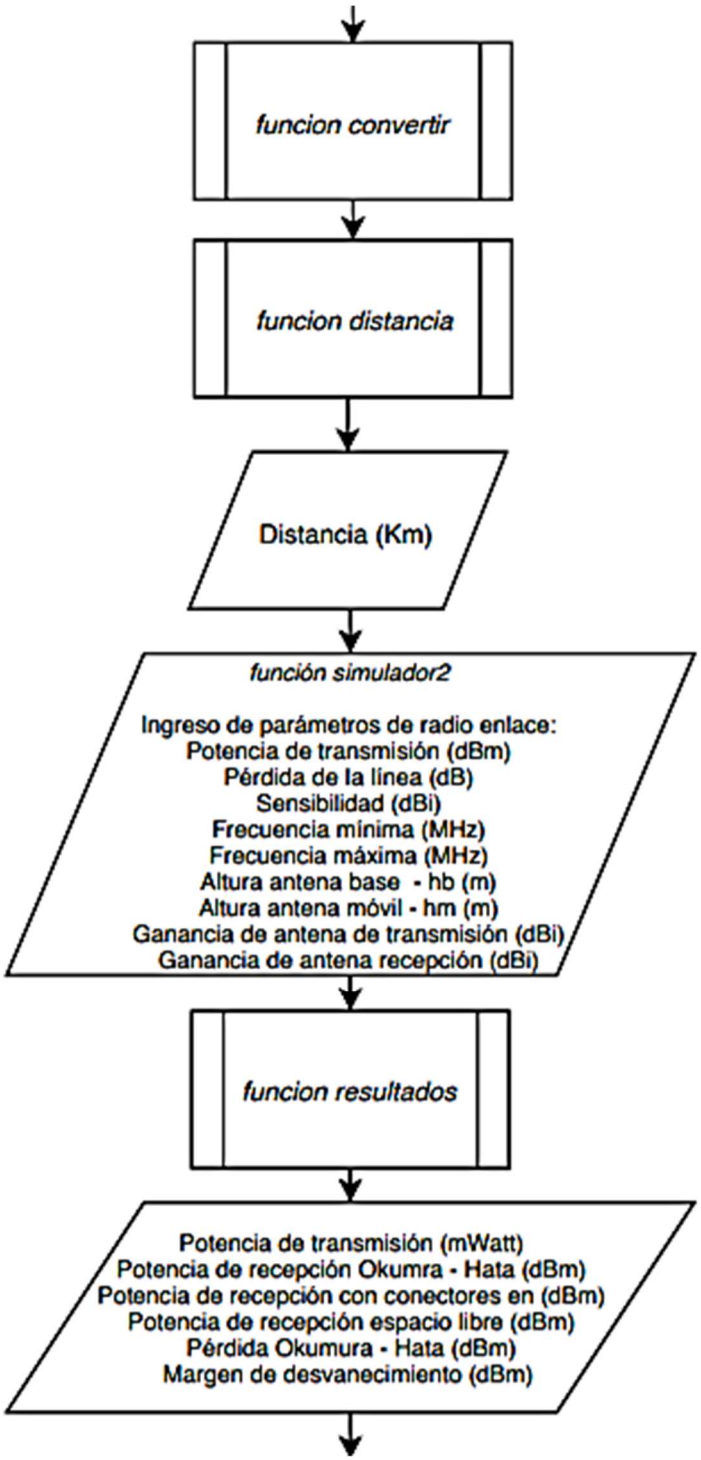
La segunda pantalla va a permitir ingresar los parámetros de radiación de los transmisores y de las antenas, las alturas de ubicación de las antenas, la sensibilidad y ganancias de los equipos transmisores y receptores, esta segunda pantalla ya muestra resultados de pérdidas, margen de desvanecimiento y cálculos de potencia aplicando el modelo de Okumura-Hata.

Finalmente, la tercera pantalla, permite cargar el fichero que contenga el mapa SRTM y de acuerdo a eso muestra un perfil topográfico con las zonas de Fresnel para poder conocer si existe una correcta transmisión o si no hay un enlace bueno debido a las interferencias.

3.2 Módulos de implementación

Para poder presentar el diseño y la implementación que se llevó acabo para el programa, a continuación, se muestran los módulos a través de diagramas de flujo, el primero con la descripción del funcionamiento general y los siguientes de las funciones específicas que permiten llegar a cumplir con el objetivo, mediante esta metodología de diagramas de flujo se podrá tener una visión más amplia de su funcionamiento y se muestra en la Figura 3.1.




funcion convertir
funcion distancia
Distancia (Km)
función simulador2
Ingreso de parámetros de radio enlace:
Potencia de transmisión (dBm)
Pérdida de la línea (dB)
Sensibilidad (dBi)
Frecuencia mínima (MHz)
Frecuencia máxima (MHz)
Altura antena base - hb (m)
Altura antena móvil - hm (m)
Ganancia de antena de transmisión (dBi)
Ganancia de antena recepción (dBi)
funcion resultados
Potencia de transmisión (mWatt)
Potencia de recepción Okumra - Hata (dBm)
Potencia de recepción con conectores en (dBm)
Potencia de recepción espacio libre (dBm)
Pérdida Okumura - Hata (dBm)
Margen de desvanecimiento (dBm)

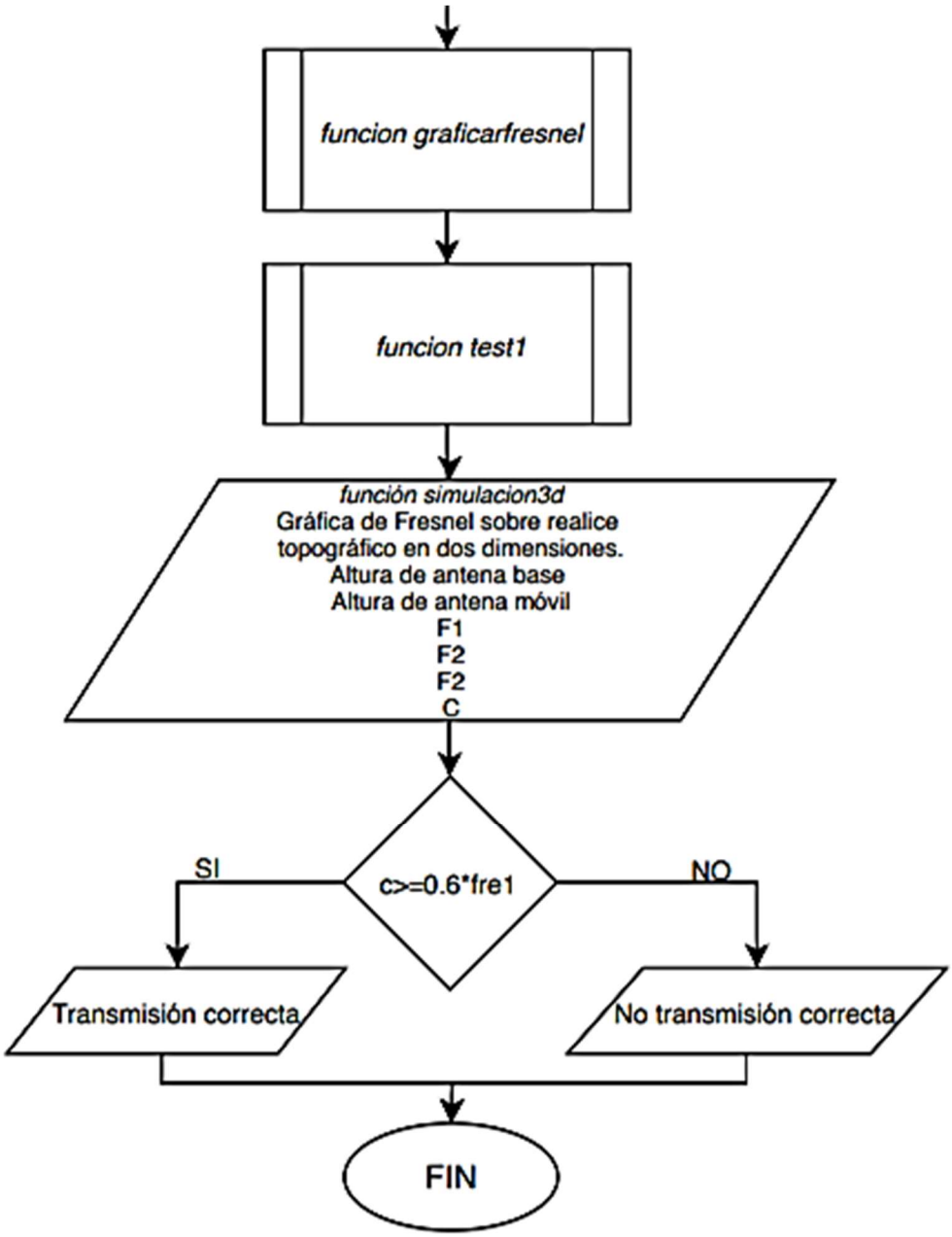


Figura. 3. 1 Diagrama de flujo general del programa.

Fuente: Elaborado por el autor

En la figura 3.2 se muestra el flujo de la función *convertir,* la cual realiza la transformación de la latitud y longitud del formato de grados, minutos y segundos, a decimales; esto permite facilitar los cálculos posteriores.

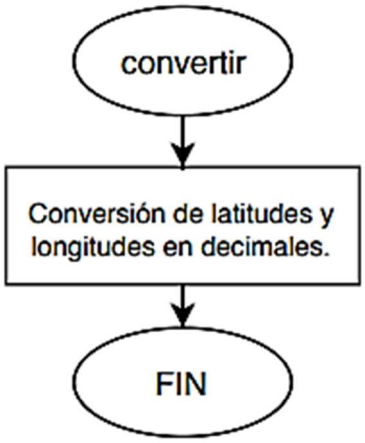


Figura. 3. 2 Función de conversión de unidades de longitud y latitud a decimal.

Fuente: Elaborado por el autor

Una vez obtenido los datos de la longitud y latitud en decimales, se procede a realizar el cálculo de la distancia entre los dos puntos, los cuales son expresados en kilómetros, como se muestra en flujo de la Figura 3.3.

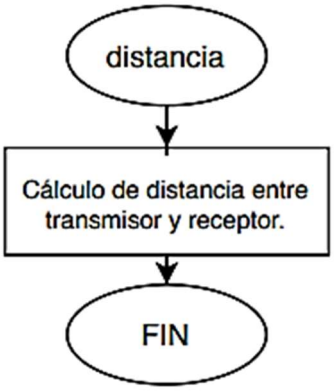


Figura. 3. 3 Función de cálculo de la distancia entre P1 transmisor y P2 receptor.

Fuente: Elaborado por el autor

La siguiente Figura 3.4, muestra el flujo para la función denominada *resultados* es la encargada de realizar los cálculos de las potencias de recepción en los modelos de espacio libre y Okumura – Hata, además de las pérdidas aplicando el modelo Okumura – Hata y el margen de desvanecimiento.

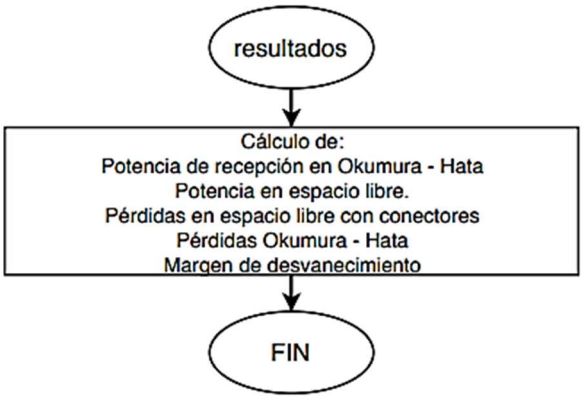


Figura. 3. 4 Función para cálculo de potencias de recepción en modelo de espacio libre y Okumura - Hata.

Fuente: Elaborado por el autor

La función *test1* obtiene una matriz que define el mapa SRTM la cual tendrá el conjunto de coordenadas de longitud y latitud con sus alturas correspondientes. Posteriormente, compara los dos puntos ingresados por el usuario con las coordenadas de la matriz y los ubica en el mapa SRTM, para después obtener el perfil topográfico y llevarlo a dos dimensiones, como se muestra en flujo de a Figura 3.5

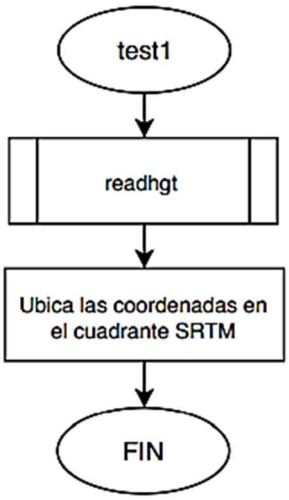


Figura. 3. 5 Función para ubicar puntos P1 transmisor y P2 receptor en mapa SRTM.

Fuente: Elaborado por el autor

El flujo para la función *readhgt* se muestra en la Figura 3.6, esta función es la encargada de la lectura de los mapas SRTM con extensión hgt. La cual se la encuentra en la biblioteca de intercambio de archivos de internet de la página oficial de Matlab.

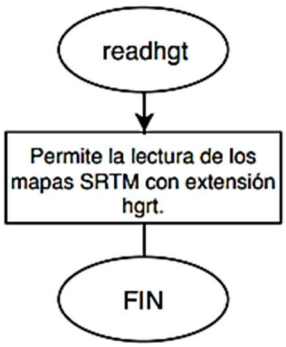


Figura. 3. 6 Función para lectura de mapas SRTM.

Fuente: Elaborado por el autor

3.2.1 Mapa SRTM Quito

El modelo de elevación SRTM para la ciudad de Quito está en los siguientes rangos indicado en la Tabla 3.1:

Tabla. 3 1 Datos SRTM para la ciudad de Quito.

Datos SRTM para la ciudad de Quito	
Resolución: "3 seconds of an arc"	
Rango de latitud	S01 a N00
Rango de longitud	O079 a O078
Nombre del archive	S01W079

Fuente: Elaborado por el autor

El mapa de la Figura 4.18, muestra todo el cuadrante que contiene la zona de la ciudad de Quito se puede observar que la altura máxima es aproximadamente 5870 metros, la ciudad se encuentra a menos altura en la parte superior del mapa en la siguiente Figura:

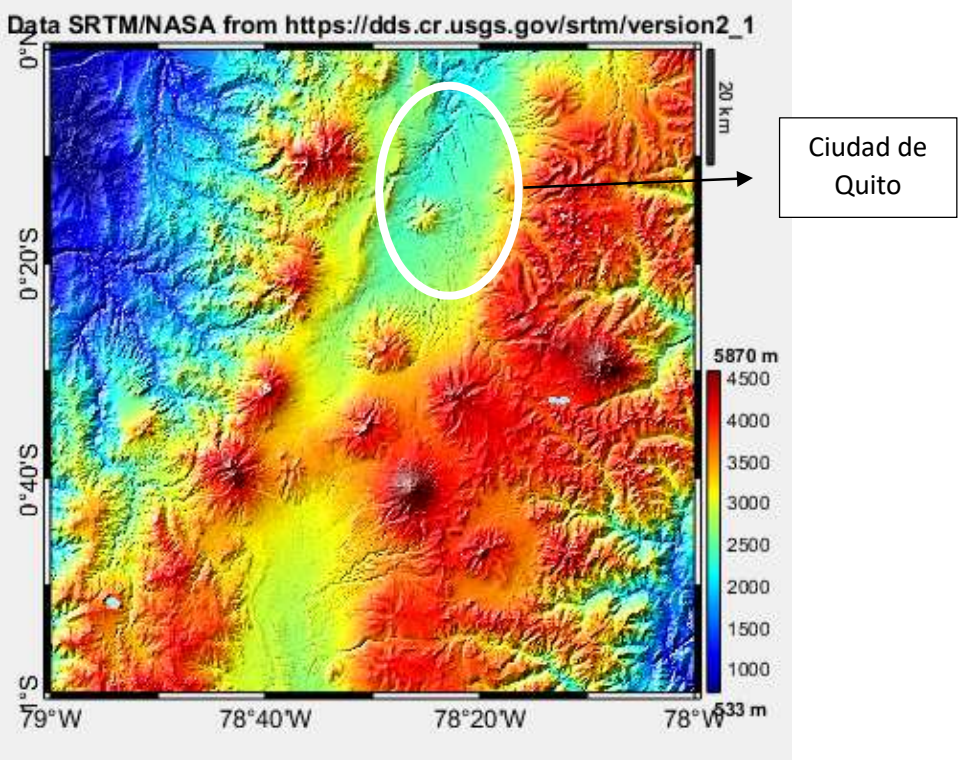


Figura. 3. 7 Mapa SRTM del cuadrante donde se ubica Quito.

Fuente: Elaborado por el autor

3.3 Aspectos técnicos

Para realizar la implementación de la aplicación se utilizará el software MATLAB con la versión 15a, de licencia estudiantil, el cual se instalará en una computadora portátil. Los requerimientos mínimos de hardware para la instalación de este software son:

- Procesador: AMD o Intel x86 – 64 soporte instrucciones AVX2.
- Memoria RAM: 1.00 GB mínimo.
- Espacio de Disco Duro: 4 a 6 GB.
- Sistema Operativo: Windows 7 de 32 o 64 Bits / Linux Kernel 2.6 o superior.

En el proceso de instalación, se debe tomar en cuenta la elección del complemento (*toolbox*) para la creación de interfaces gráficas GUI.

3.4 Análisis de costos y tiempo del proyecto.

El tiempo para realizar la aplicación se proyectó en 281 días como se muestra en la siguiente planificación:

Id	Duración	Comienzo	Fin	Predecesoras
1	281 días	mar 8/1/19	mié 30/10/19	
2	283,5 días	mar 8/1/19	jue 31/10/19	
3	0,83 días	mar 8/1/19	mar 8/1/19	
4	14,5 días	mar 22/1/19	mar 5/2/19	
5	7 días	mar 22/1/19	mar 29/1/19	
6	7,5 días	mar 29/1/19	mar 5/2/19	5
7	256,83 días	mar 5/2/19	jue 31/10/19	
8	112,83 días	mar 5/2/19	lun 3/6/19	
9	62,17 días	lun 4/3/19	mar 7/5/19	6
10	26,17 días	mar 7/5/19	lun 3/6/19	6
11	171,5 días	lun 1/4/19	jue 26/9/19	
12	20 días	lun 3/6/19	lun 24/6/19	10
13	60 días	lun 24/6/19	lun 26/8/19	12
14	30,83 días	mar 30/7/19	vie 30/8/19	12
15	20 días	vie 30/8/19	vie 20/9/19	14
16	87,5 días	jue 1/8/19	jue 31/10/19	
17	15 días	vie 20/9/19	lun 7/10/19	15
18	20 días	vie 20/9/19	vie 11/10/19	15
19	24,83 días	vie 20/9/19	mié 16/10/19	15
20	28,83 días	mié 2/10/19	jue 31/10/19	15
21				
22	281 días	mié 9/1/19	jue 31/10/19	

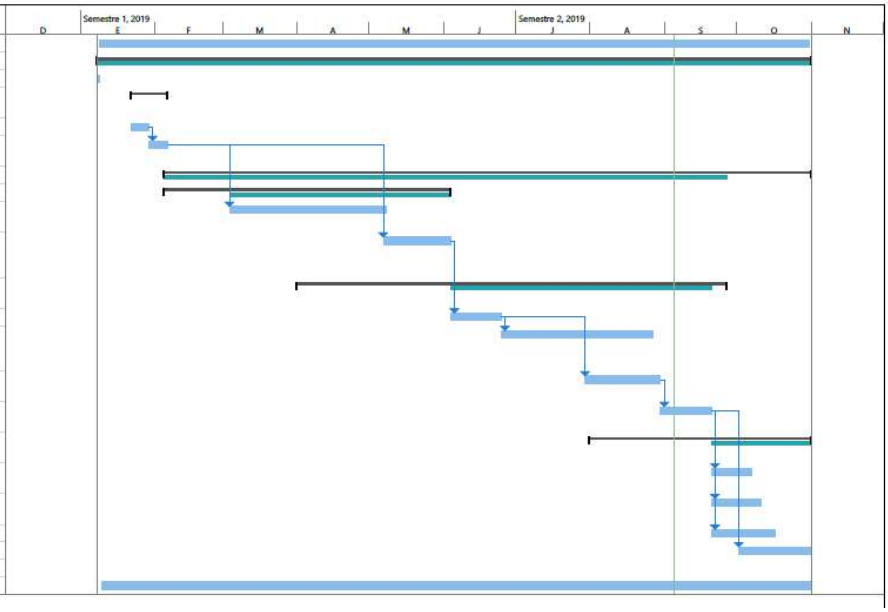


Figura. 3. 8 Cronograma de actividades

Fuente: Elaborado por el autor

Al proponer un tiempo estimado se puede calcular el valor del recurso humano utilizado para este desarrollo. Tomando en cuenta que una sola persona utilizará 4 horas diarias de lunes a viernes por 281 días se obtiene el tiempo total utilizado:

Figura. 3. 9 Presupuesto

Fuente: Elaborado por el autor

3.5 Ventajas

Las ventajas principales de este desarrollo son:

- Elaborado en un software licenciado que cuenta con código abierto en una herramienta que utilizan ingenieros y científicos de todo el mundo para crear nuevos productos y avances.
- Es una base investigativa para adicionar otros modelos de propagación.
- Es una herramienta para la comprobación y ayuda de aprendizaje estudiantil.
- Se encuentra elaborado en un modelo de predicción adaptado para zonas urbanas como la ciudad de Quito.

CAPÍTULO 4

IMPLEMENTACIÓN

A continuación, se presentan los resultados obtenidos de la realización del presente trabajo. En primera instancia, se presentará las pantallas realizadas en el GUIDE de Matlab que componen la aplicación diseñada. Seguidamente, se efectuará un manual de usuario y posteriormente con base a la ejecución de las coordenadas de un punto real en Quito se realizará un ejemplo y se explicarán los resultados obtenidos.

4.1. Desarrollo

Para la elaboración del software se realiza el diseño con base al flujo indicado en el esquema de la propuesta. En primer lugar, se mostrará la pantalla con el ingreso de las coordenadas geográficas de los puntos de transmisión y recepción que pertenecen la ciudad de Quito.

En esta pantalla se escoge el mapa SRTM de la ciudad, para obtener la gráfica y la distancia en kilómetros, aplicando la fórmula de Haversine.

Para continuar se dará un click en el cuadro de siguiente, donde aparecerá el segundo formulario, en el cual se debe ingresar los parámetros de ganancia y pérdida en el transmisor y receptor, dando como resultado el cálculo de las pérdidas ideales y las pérdidas teóricas aplicando el modelo de propagación propuesto de Okumura – Hata.

Por último, en la pantalla fina se mostrará el perfil topográfico con la representación de Zona de Fresnel y los valores resultantes de los cálculos de distancia, pérdidas y potencias.

4.2. Implementación

A continuación, se describe la programación y los formularios tanto para el ingreso de datos, desarrollo de la implementación y la obtención de resultados.

4.2.1 Ingreso de latitud y longitud de los puntos P1 transmisor y P2 receptor, y transformación de unidades

Los datos ingresados por el usuario, son leídos en la función *Simulador* mediante el comando *get* y transformados a valor numérico con str2num, para poder trabajar con estos datos. Este código se muestra a continuación en la figura 4.1

```
% --- EJECUTA AL PRESIONAR EL BOTÓN sim.
% PUNTO 1
%GRADOS, MINUTOS Y SEGUNDOS DE LATITUD
grados1=str2num(get(handles.grado1,'string'));
mins1=str2num(get(handles.min1,'string'));
segs1=str2num(get(handles.seg1,'string'));
signo=get(handles.Sur1,'value');
.
.
.
%GRADOS, MINUTOS Y SEGUNDOS DE LONGITUD
grados4=str2num(get(handles.grado4,'string'));
mins4=str2num(get(handles.min4,'string'));
segs4=str2num(get(handles.seg4,'string'));
signo3=get(handles.oeste2,'value');
.
```

Figura. 4. 1 Sección de código para obtener los datos de las coordenadas ingresadas por el usuario.

Fuente: Elaborado por el autor

En el código que se muestra en la Figura 4.2, indica la secuencia luego del ingreso de datos de las longitudes y latitudes en el formato de grados, minutos y segundos, correspondientes a los puntos donde se ubicarán las antenas de transmisión y recepción, es necesario realizar una transformación de unidades a decimales. Esto es lo que se efectúa en el siguiente código, que se encuentra en la función *convertir*.

```
function decimal=convertir(grado,min,seg)
%TRANSFORMA GRAD,MIN Y SEG A DECIMAL
segundo=seg/3600;
minuto=min/60;
decimal=(grado+minuto+segundo)*pi/180;
```

Figura. 4. 2 Sección del código de transformación de formato a decimal de las coordenadas ingresadas.

Fuente: Elaborado por el autor

En método para el procedimiento aplicado para la transformación a grados, minutos y segundos se muestra en la Figura 4.3:

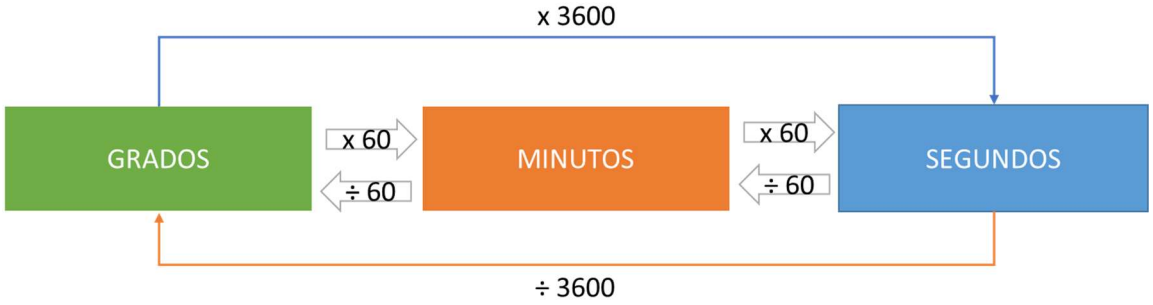


Figura. 4. 3 Método de transformación de coordenadas a formato decimal.

Fuente: Elaborado por el autor

La conversión es empleada en la función *Simulador* se muestra en la Figura 4.4 y se utiliza para la obtención de las longitudes y latitudes de los puntos P1 transmisor y P2 receptor.

```
.
.
.
if signo1==1
    longitud1=-1*convertir(grados2,mins2,segs2);
    long1=longitud1*180/pi
else
    longitud1=convertir(grados2,mins2,segs2);
    long1=longitud1*180/pi
end

.
.
.
if signo2==1
    latitud2=-1*convertir(grados3,mins3,segs3);
    lati2=latitud2*180/pi
else
    latitud2=convertir(grados3,mins3,segs3);
    lati2=latitud2*180/pi
end
```

Figura. 4. 4 Uso de la función convertir con las coordenadas ingresadas por el usuario.

Fuente: Elaborado por el autor

4.2.2 Cálculo de la distancia entre los puntos de transmisión y recepción

Conociendo las coordenadas geográficas en decimales procedemos a calcular la distancia entre la antena de transmisión y recepción. Se utilizará la fórmula del Haversine para de esta manera tomar en cuenta la curvatura terrestre. Este código se encuentra en la función denominada *distancia*. Su código se muestra en la Figura 4.5.

```
function d=distancia(longitud1,latitud1,longitud2,latitud2)
%CALCULA LA DISTANCIA CON LA FÓRMULA DE HAVERSINE
r=6370;
lat=latitud2-latitud1;
lon=longitud2-longitud1;
a=sind(lat/2)^2+cosd(latitud1)*cosd(latitud2)*(sind(lon/2))^2;
d=2*r*asind(sqrt(a));
```

Figura. 4. 5 Código de la función para calcular la distancia entre los dos puntos.

Fuente: Elaborado por el autor

Las fórmulas para el cálculo de las distancias que se encuentran implementadas son las siguientes.

$$d = 2 * R * arcsen(\sqrt{a}) \quad (24)$$

Ecuación. 4. 1 Cálculo de distancia Haversine

Fuente: (Bucheli, 2017)

$$a = sen^2\left(\frac{\Delta lat}{2}\right) + \cos(latitud\ 1) * \cos(latitud\ 2) * sen\left(\frac{\Delta long}{2}\right) \quad (25)$$

Ecuación. 4. 2 Cálculo de variable (a)

Fuente: (Bucheli, 2017)

Donde:

$R = radio\ de\ la\ Tierra$

$\Delta lat = latitud\ 2 - latitud\ 1$

$$\Delta long = longitud\ 2 - longitud\ 1$$

En la Figura 4.6 se muestra la descripción de la función distancia es llamada en la función *Simulador*, e imprime en la ventana mediante el comando *set*.

```
%DISTANCIA ENTRE LOS PUNTOS
global distancias64
distancias64=distancia(longitud1,latitud1,longitud2,latitud2);
set(handles.dis,'string',val);
```

Figura. 4. 6 Llamada a la función distancia para el cálculo de la separación en Km entre transmisor y receptor.

Fuente: Elaborado por el autor

4.2.3 Formulario de ingreso de datos de georreferencia

La primera pantalla que se representa en la Figura 4.7, nos solicita el ingreso de los datos de las coordenadas; en formato grados, minutos y segundos; del punto 1 transmisor y puntos 2 receptor las cuales corresponderían a las posiciones geográficas en las que se ubicarían las antenas de transmisión y recepción.

Figura. 4. 7 Ingreso de coordenadas y cálculo de distancia.

Fuente: Elaborado por el autor

A continuación, en la figura 4.8, se detalla el formulario de resultados que nos proporciona la distancia en Km que existen entre estos dos puntos geográficos. Además, se muestra una imagen del modelo de elevación tridimensional de la Tierra correspondiente al cuadrante donde se encuentran las coordenadas ingresadas; este modelo es obtenido de los mapas libres de la misión *Shuttle Radar Topography Mission* (SRTM) realizada por la NASA.

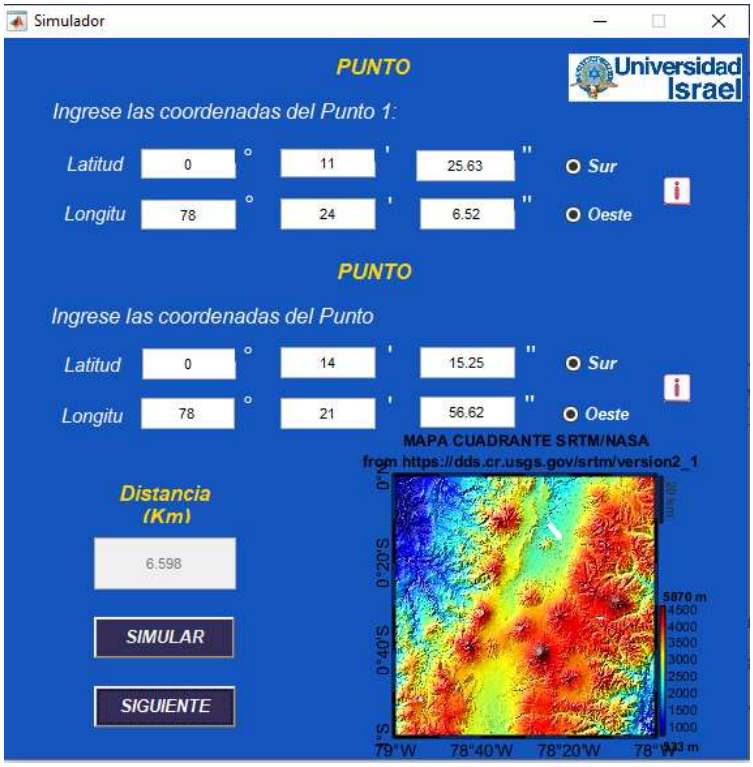


Figura. 4. 8 Resultado del ingreso de coordenadas y cálculo de distancia.

Fuente: Elaborado por el autor

4.2.4 Ingreso de los parámetros del radioenlace y cálculos realizados

El programa solicitará ciertas características del radioenlace para realizar cálculos de la potencia de recepción en espacio libre y con el modelo de Okumura – Hata, las pérdidas de recepción y el margen de desvanecimiento. Esto se encuentra en la función *simulador2*.

Los parámetros que deben ser ingresados son: la potencia de transmisión (dBm), la pérdida de la línea que es la suma de las pérdidas en el cable, en los acoples (dB), la sensibilidad del

receptor, la frecuencia máxima y mínima de transmisión, la altura de la antena de la estación base (transmisor, hb); la altura de la antena de la estación receptora, hm); y por último las ganancias en dBi de la antena de transmisión y recepción respectivamente.

A continuación, en la Figura 4.9, se muestra la sección de la función *simulador2* donde se leen los datos ingresados.

```
.
potedb=str2num(get(handles.potdb,'string'));%POTENCIA DE TRANSMISIÓN dBm
g1=str2num(get(handles.ga1,'string'));%GANANCIA DE ANTENA DE RECEPCIÓN dBi
g2=str2num(get(handles.ga2,'string'));%GANANCIA ANTENA DE TRANSMISIÓN dBi
f1=str2num(get(handles.fmin,'string'));%FRECUENCIA MÍNIMA MHz
f2=str2num(get(handles.fmax,'string'));%FRECUENCIA MÁXIMA MHz
lin=str2num(get(handles.linea,'string'));%PERDIDA DE LA LÍNEA
sensa=str2num(get(handles.sensi,'string'));%SENSIBILIDAD dBi
hb=str2num(get(handles.edit31,'string'));%ALTURA BASE
hm=str2num(get(handles.edit32,'string'));%ALTURA RECEPTOR
.
```

Figura. 4. 9 Sección de código que obtiene los parámetros de radiación ingresados.

Fuente: Elaborado por el autor

Con los cálculos realizados se elabora la función *resultados*, y su código se muestra en la Figura 4.10, donde las ecuaciones implementadas para el cálculo de las potencias de recepción y las pérdidas en los modelos de espacio libre y Okumura – Hata, se encuentran en la sección 1.10.4 del capítulo 1. El código se encuentra en la siguiente imagen.

```
function [salida2 salida potredb potredb1
salida3]=resultados(potedb,g1,g2,d,f,lin,sensa,hm,hb)

%PROGRAMA CALCULOS PANTALLA SIMULADOR 2

potredb1=potedb+g1+g2-32.44-20*log10(d)-20*log10(f);%ESPACIO LIBRE
a=(1.1*log10(f)-0.7)*hm-(1.56*log10(f)-0.8);%CONSTANTE
potredb=potedb+g1+g2-69.55+26.16*log10(f)-13.82*log10(hb)-a+(44.9-
6.55*log10(hb))*log10(d)%MODELO OKUMURA HATA, POTENCIA DE RECEPCION

lb=69.55+26.16*log10(f)-13.82*log10(hb)-a+(44.9-
6.55*log10(hb))*log10(d);%PERDIDA
salida=potredb1-lin;%CON CONECTORES

salida3=salida+sensa;%MARGEN
salida2=lb;%PERDIDA
```

Figura. 4. 10 Código de la función que calcula las potencia de transmisión, las pérdidas y el margen de desvanecimiento.

Fuente: Elaborado por el autor

La impresión de los datos en la ventana se realiza en la función *simulador2*, con el código que se indica a continuación en la Figura 4.11, en donde se transforma los datos de numérico a un arreglo de caracteres mediante *num2st*.

```
global  frecuencia1000 distancias64 hb hm
frecuencia1000=(f2+f1)/2;%FRECUENCIA MEDIA
[salida2 salida res potredb1
salida3]=resultados(potedb,g1,g2,distancias64,frecuencia1000,lin,sensa,hm,
hb)%LLAMA A LA FUNCION RESULTADOS

%MODELO OKUMURA HATA, POTENCIA DE RECEPCIÓN
val=num2str(res);
set(handles.res,'string',val);

%POTENCIA DE TRANSMISIÓN mWatt
potencia=10^(potedb/10);
val2=num2str(potencia);
set(handles.po,'string',val2);

%POTENCIA DE RECEPCIÓN CON CONECTORES
val3=num2str(salida);
set(handles.salid,'string',val3);

%PÉRDIDA OKUMURA-HATA
val4=num2str(salida2);
set(handles.mar,'string',val4);

%POTENCIA DE RECEPCIÓN ESPACIO LIBRE
val5=num2str(potredb1);
set(handles.edit33,'string',val5);

%MARGEN DE DESVANECIMIENTO
val6=num2str(salida3);
set(handles.edit34,'string',val6);
```

Figura. 4. 11 Llamada a la función e ingreso de parámetros.

Fuente: Elaborado por el autor

4.2.5 Formulario de ingreso de valores de los elementos del radioenlace

La segunda pantalla corresponde a los datos del sistema que se deben ingresar tanto en el emisor como en el transmisor y son: la potencia de transmisión en dBm del radioenlace; las pérdidas de la línea en dB provocada por el cable, las cavidades y los conectores empleados; la sensibilidad del receptor; las frecuencias mínima y máxima en MHz; la altura de la antena de la estación base (transmisor, hb); la altura de la antena de la estación receptora (hm); y por último las ganancias en dBi de la antena de transmisión y recepción respectivamente, como se muestra en la Figura 4.11. Para este modelo de predicción se toma en cuenta el valor de pérdidas por alimentación de 3.2 dB, expresado en la tabla 1 en el capítulo 1. Este valor corresponde a la Frecuencia del modelo Okumura – Hata, entre 150 a 1500 MHz.

Como resultado de esta pantalla se mostrarán los valores de: la potencia de transmisión en mWatt; la potencia de recepción y las pérdidas en dBm con el modelo de Okumura - Hata; la potencia en dBm de recepción con conectores; la potencia de recepción en dBm espacio libre; y el margen de desvanecimiento. También se agrega una imagen del patrón de radiación de una antena tipo Yagui.

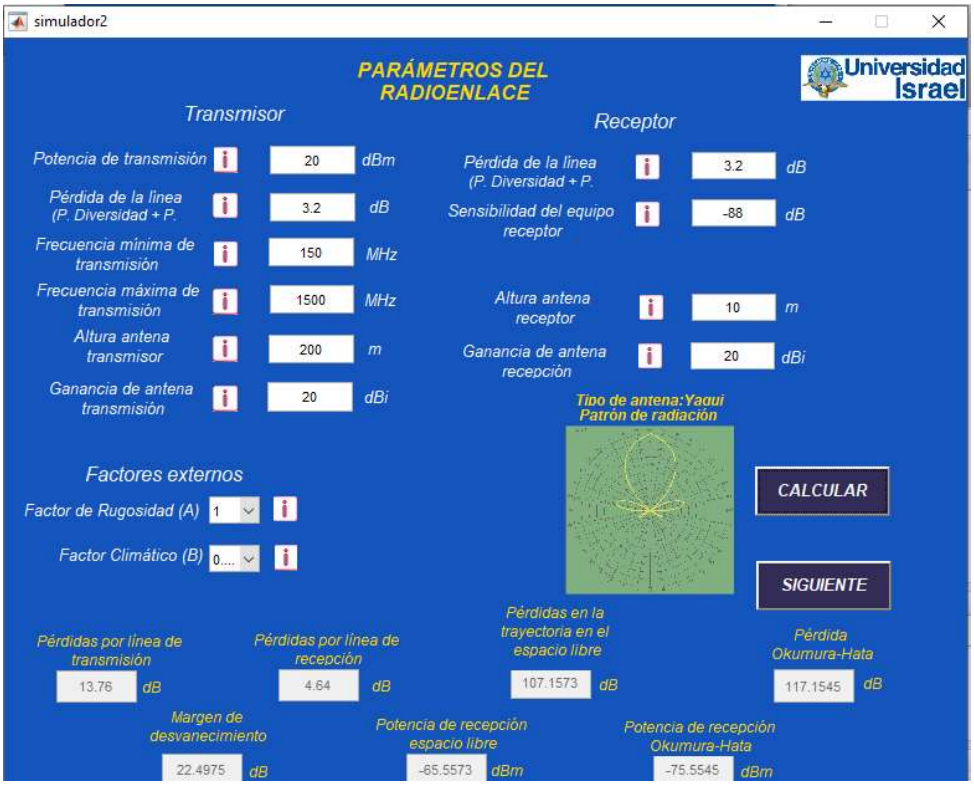


Figura. 4. 12 Llamada a la función e ingreso de parámetros.

Fuente: Elaborado por el autor

4.2.6 Gráfica de zonas de Fresnel y simulación del radioenlace en el mapa de relieve

La función encargada de graficar las zonas de Fresnel tiene el nombre de *graficarfresnel* este brinda los datos de las frecuencias de Fresnel, la distancia de la línea de vista directa al punto máximo del objeto y los vectores de las frecuencias para realizar el gráfico.

A continuación, en la Figura 4.13, se presenta el código de la función *graficarfresnel*.

```
function
[fre1,fre2,fre3,gf3,gf2,gf,x,y,c]=graficarfresnel(f,d,d1,d2,hm,hb,hs,posy1
,posy2)

%f ES LA FRECUENCIA MEDIA
%d DISTANCIA ENTRE P1 Y P2
%d1 DISTANCIA DE P1 AL OBSTÁCULO
%d2 DISTANCIA DEL OBSTÁCULO A P2
%hm ALTURA DEL RECEPTOR
%hb ALTURA DE LA BASE
%hs ALTURA DEL OBSTÁCULO

%CÁLCULO VALORES DE FRESNEL
fre1=548*sqrt((d1*d2)/(f*d));%ENÉSIMA ZONA DE FRESNEL
fre2=sqrt(2)*fre1;
fre3=sqrt(2)*fre2;

%DISTANCIA DE PUNTO ALTO DE OBSTÁCULO A LINEA DIRECTA
c=d1*((hb+posy2)-(hm+posy1))/d+(hm+posy1)-hs;

%CÁLCULO PARA GRÁFICA DE FRESNEL
d=floor(d);
x=0:0.1:d;
gf=548*sqrt((x.*(d-x))/(f*d));
gf2=sqrt(2)*gf;
gf3=sqrt(2)*gf2;
pendiente=((hb+posy2)-(hm+posy1))/(d);
y=pendiente.*(x)+(hm+posy1);
```

Figura. 4. 13 Código de la función para el cálculo de las zonas de Fresnel.

Fuente: Elaborado por el autor

Las ecuaciones empleadas para el cálculo de las zonas de Fresnel se encuentran en la sección 1.6 del Capítulo 1 correspondiente a la Fundamentación Teórica.

Para la realización de la gráfica de simulación se hizo uso de un programa encontrado en Internet en la biblioteca de intercambio de archivos de programadores en Matlab. Esta función se denomina *readhgt* que permitirá leer los mapas SRTM cuya extensión es. hgt; este programa no será explicado solamente será referenciado.

Mediante la Figura 4.14, se muestra la sección para la función programada llamada *test1,* que es la encargada de proporcionar los datos que permiten graficar el relieve en dos dimensiones del cuadrante SRTM. Además, esta utiliza la función *dem* que permite dar diferentes colores según la altitud de las elevaciones.

Esta función utiliza las variables globales de longitud y latitud del punto 1 transmisor y punto 2 receptor, los cuales compara con todos los puntos del mapa SRTM (conjunto de coordenadas) con el objeto de ubicarlos exactamente en el mapa, el cual es leído con la función *readhgt*.

```
global lati1 long1 lati2 long2 distancias64
X=readhgt();
lat=X.lat;
lon=X.lon;
SRTM=X.z;
[m n]=size(SRTM);
```

Figura. 4. 14 Sección de código donde se trabaja con la función readhgt.

Fuente: Elaborado por el autor

En la Figura 4.15 se encuentra el código que realiza la comparación para cada coordenada del punto 1 transmisor y 2 receptor, y es el siguiente:

```
%% Comparacion de datos
latIn1=lati1;
lonIn1=long1;
latIn2=lati2;
lonIn2=long2;
%redondeo a 4 cifras decimales de datos obtenidos
latIn1=round(latIn1,4);
lonIn1=round(lonIn1,4);
latIn2=round(latIn2,4);
lonIn2=round(lonIn2,4);
%Evaluacion longitud1
vectlon1=lonIn1*ones(1,length(lon));%creo matrices de 1
resta1=lon-vectlon1;
[minVectlon1, Ilonmin1]=min(abs(resta1));%Ilonmin1 es la posicion
%Evaluacion latitud1
vectlat1=latIn1*ones(length(lat),1);%creo matrices de 1
resta2=lat-vectlat1;
[minVectlat1, Ilatmin1]=min(abs(resta2));%Ilatmin1 es la posicion
%Evaluacion longitud2
vectlon2=lonIn2*ones(1,length(lon));%creo matrices de 1
resta3=lon-vectlon2;
[minVectlon2, Ilonmin2]=min(abs(resta3));%Ilonmin1 es la posicion
%Evaluacion latitud2
```

Figura. 4. 15 Sección de código que compara las coordenadas ingresadas con la matriz representativa del cuadrante SRTM.

Fuente: Elaborado por el autor

Para facilitar la ubicación de los puntos, se transforma el mapa leído a una red de puntos en 3D, y posteriormente se realiza una interpolación como se muestra en el código de la Figura 4.16.

```
.
%% Grafico
[x,y]=meshgrid(linspace(lon(1,1),lon(1,length(lon)),n),linspace(lat(1,1),l
at(length(lat),1),m));
h=interp2(x,y,single(SRTM),lon,lat,'spline');
x1=linspace(0,120,n);
.
```

Figura. 4. 16 Sección de código de la interpolación de los puntos.

Fuente: Elaborado por el autor

Además, se transforma el dominio de puntos a un dominio de distancia en km. Con esto se logra obtener la altura de los puntos y su ubicación en el cuadrante SRTM. Para poder graficar en 2D, se toma los puntos de la altura en función de la latitud, por otro lado, con el objetivo de suavizar la gráfica, se utiliza un filtro Gauss, el cual permite ver la superficie sin muchas imperfecciones. La sección de este proceso se muestra en la Figura 4.17.

```
%FILTRO
[h1,h2]=max(h,[],2);
g=gausswin(30);
g=g/sum(g);
y3=conv(h1,g,'same');
posy1=matrizMaxi(posx1);
posy2=matrizMaxi(IPos2);
% auxMax=max(matrizMaxi(posx1:IPos2))
[ValorPMax, IVpmax]=max(matrizMaxi(posx1:IPos2));
IVpmax=length(matrizMaxi(1:posx1))+IVpmax;
IVpmax=x1(IVpmax);
```

Figura. 4. 17 Filtro para suavizado de curva del relieve topográfico.

Fuente: Elaborado por el autor

4.2.7 Formulario de resultados

En la Figura 4.18 se encuentra la pantalla que muestra una simulación aproximada del sistema, donde se puede apreciar el gráfico de las zonas de Fresnel del radioenlace entre las antenas de transmisión y recepción, ubicadas en una representación gráfica en dos dimensiones de las elevaciones del terreno tomando como referencia el nivel del mar. Además, nos proporciona resultados de las zonas de Fresnel.

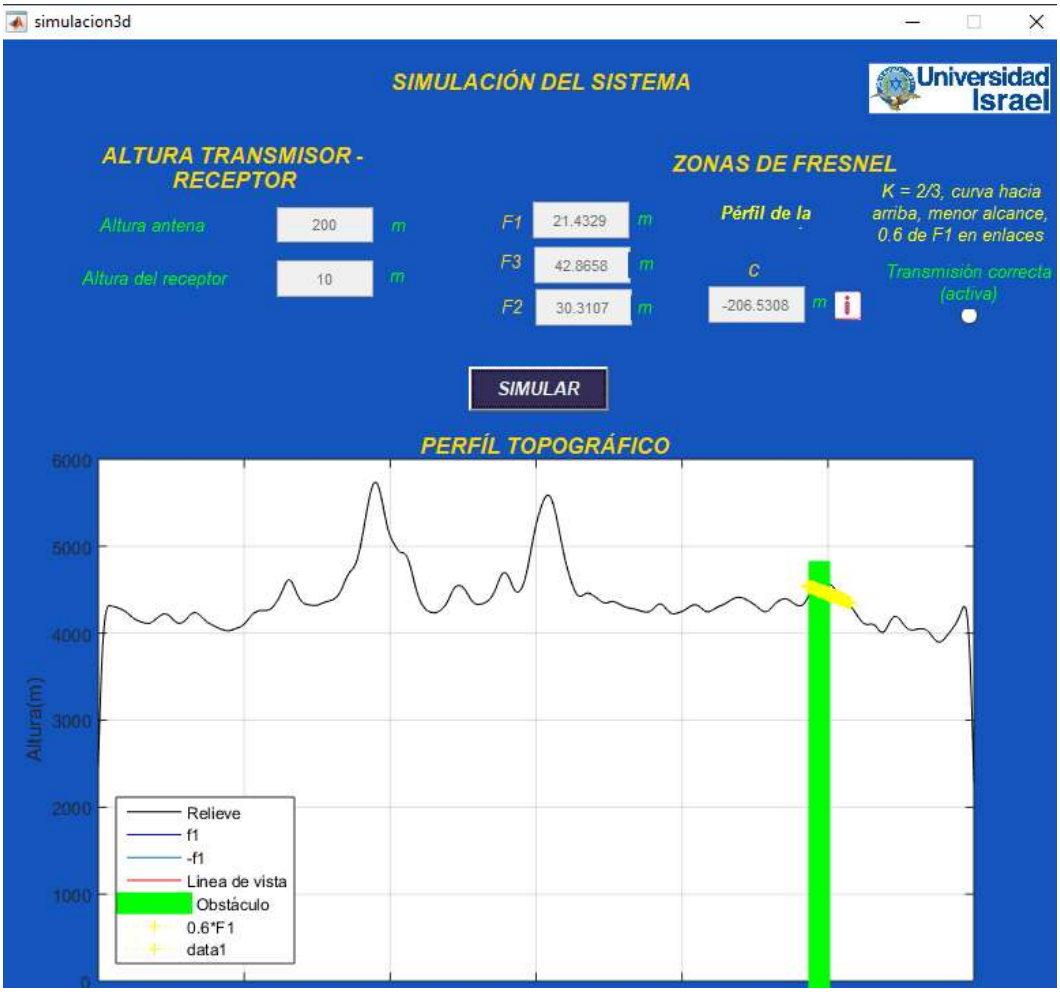


Figura. 4. 18 Perfil topográfico con las zonas de Fresnel.

Fuente: Elaborado por el autor

4.3. Pruebas de funcionamiento

Para ejemplificar se presentará la Pantalla 1 de la aplicación, llamada Simulador. En esta pantalla se procede a introducir coordenadas de dos puntos en formato GMS (Grados, minutos y segundos). El Punto 1 corresponde a la coordenada donde se ubicará el transmisor del radioenlace, el cual, en este caso se ubica en el sector de Solanda, ubicado en el Sur de Quito. Por otro lado, el Punto 2 corresponde a la coordenada donde se ubicará el receptor del radioenlace. el cual, en este caso se ubica en el sector del Condado ubicado en el Norte de Quito como se visualiza en la Figura 4.19.

PUNTO 1 TRANSMISOR
Latitud: S 0°, 15´, 50.619”
Longitud: O 78°, 32´, 36.66”

PUNTO 2 RECEPTOR
Latitud: S 0°, 5´, 45.914”
Longitud: O 78°, 31´, 10.497”

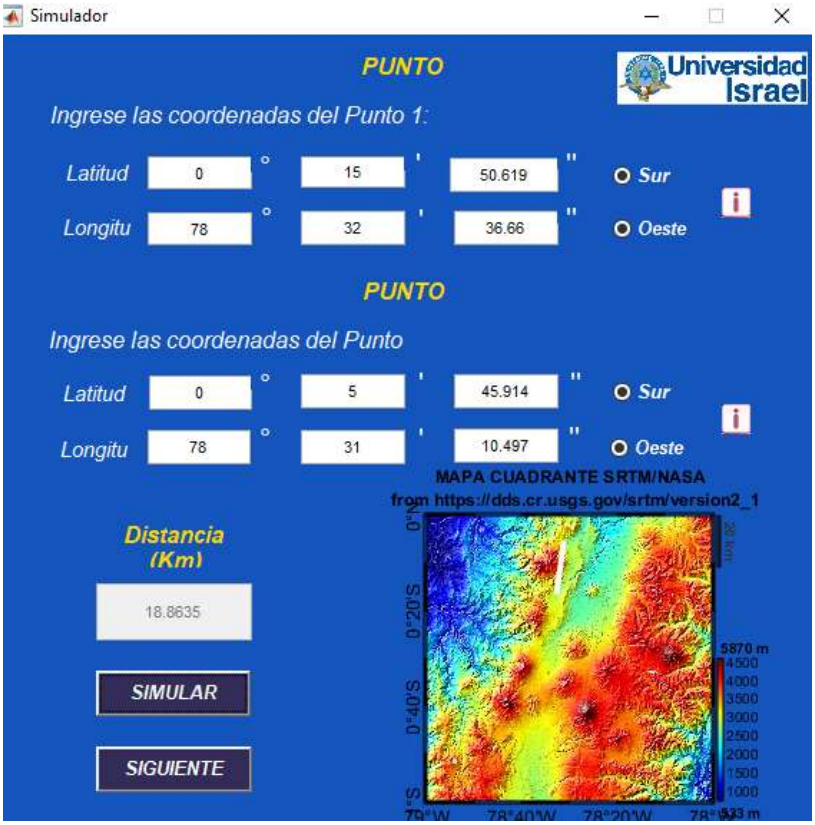


Figura. 4. 19 Pantalla de ingreso de coordenadas geográficas de Punto 1 transmisor y Punto 2 receptor.

Fuente: Elaborado por el autor

En esta pantalla, el programa determina la distancia entre los dos puntos, la cual en este caso es de 18.8635 km. Además, se solicitará al usuario que se seleccione el archivo. hgt el cual corresponde al mapa SRTM del cuadrante en el que se encuentran las coordenadas de los dos puntos. Este mapa se muestra en la parte inferior derecha, donde se puede apreciar que los colores más cálidos corresponden a alturas más altas y los colores más fríos a las zonas más bajas. Cabe recalcar que, para el presente ejemplo como se verifica en la Figura 4.20, solo la parte superior del mapa SRTM contiene la ciudad de Quito.

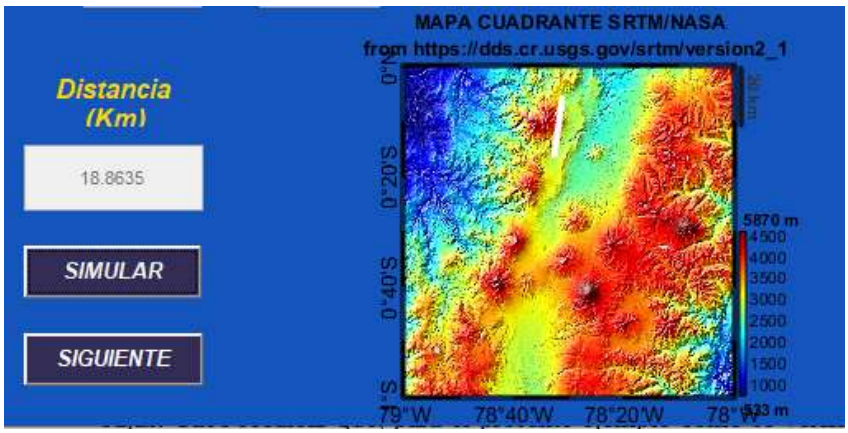


Figura. 4. 20 Resultado de la distancia entre transmisor y receptor.

Fuente: Elaborado por el autor

En la Figura 4.21 se encuentra la comprobación de los datos ingresados en Google Maps con la distancia 18.87 Km.

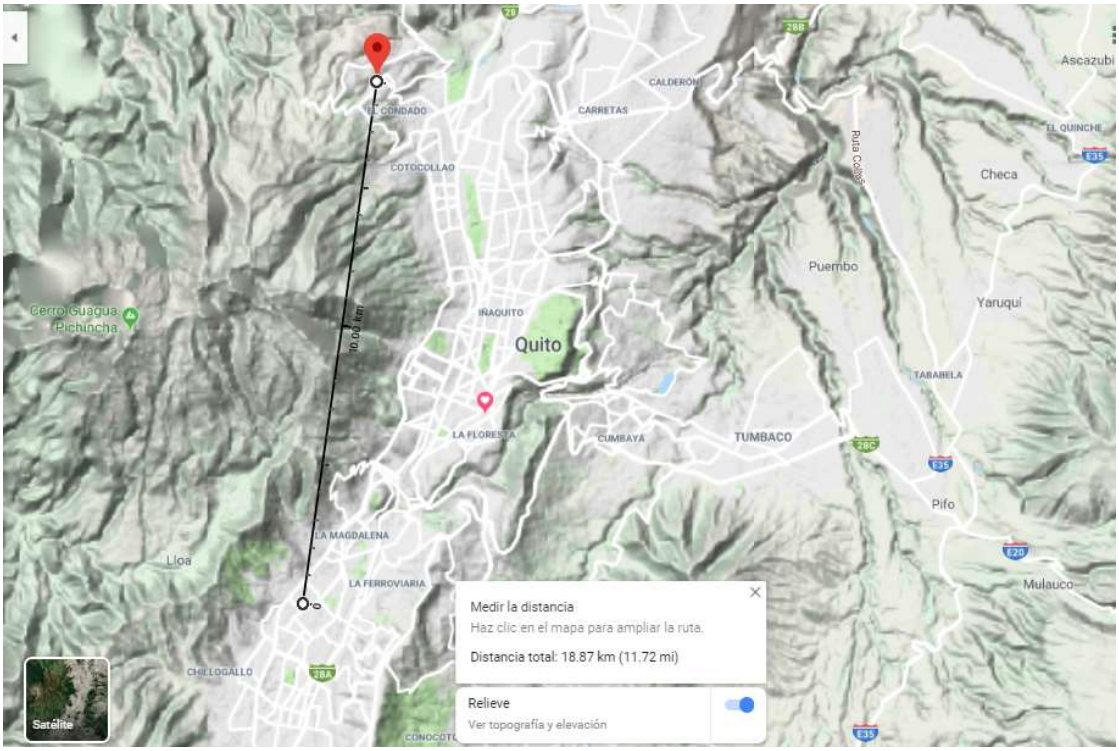


Figura. 4. 21 Puntos de radioenlace en Google Maps.

Fuente: Elaborado por el autor

De igual manera en la Figura 4.22, se comprueban los datos ingresados en Radio Mobile:

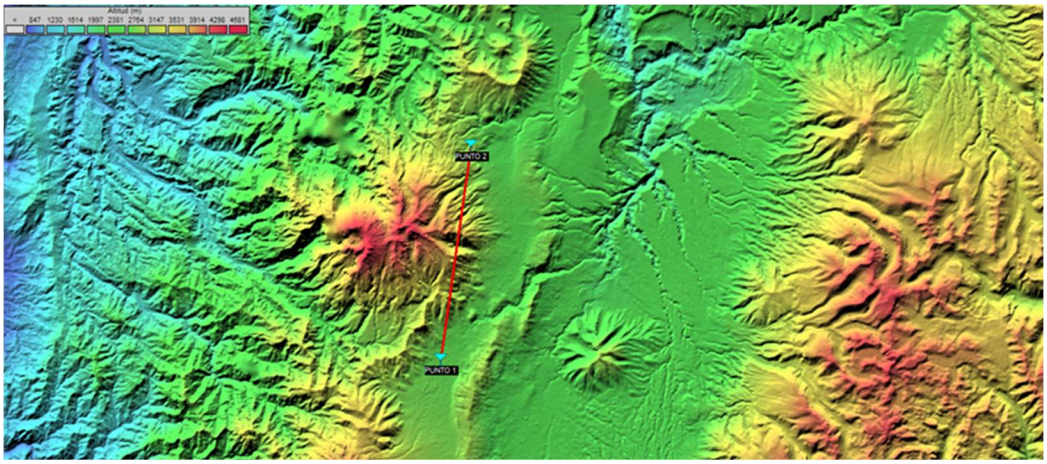


Figura. 4. 22 Puntos de radioenlace en Radio Mobile

Fuente: Elaborado por el autor

Con los datos obtenidos se puede realizar la siguiente Tabla 4.1, en la cual se compara y se demuestra que las distancias son similares.

Tabla. 4 1 Comparación de la distancia calculada entre los simuladores del enlace 1

SIMULADOR	DISTANCIA CALCULADA [Km]
Matlab (Figura. 4.5)	18.8635
Google Maps (Figura. 4.6)	18.87
Radio Mobile (Figura. 4.7)	18.85

Fuente: Elaborado por el autor

Una vez dado clic en el botón siguiente, se abrirá otra interfaz llamada “simulador2”, en la cual se deberá ingresar parámetros del radioenlace, como se observa en la Figura 4.23.

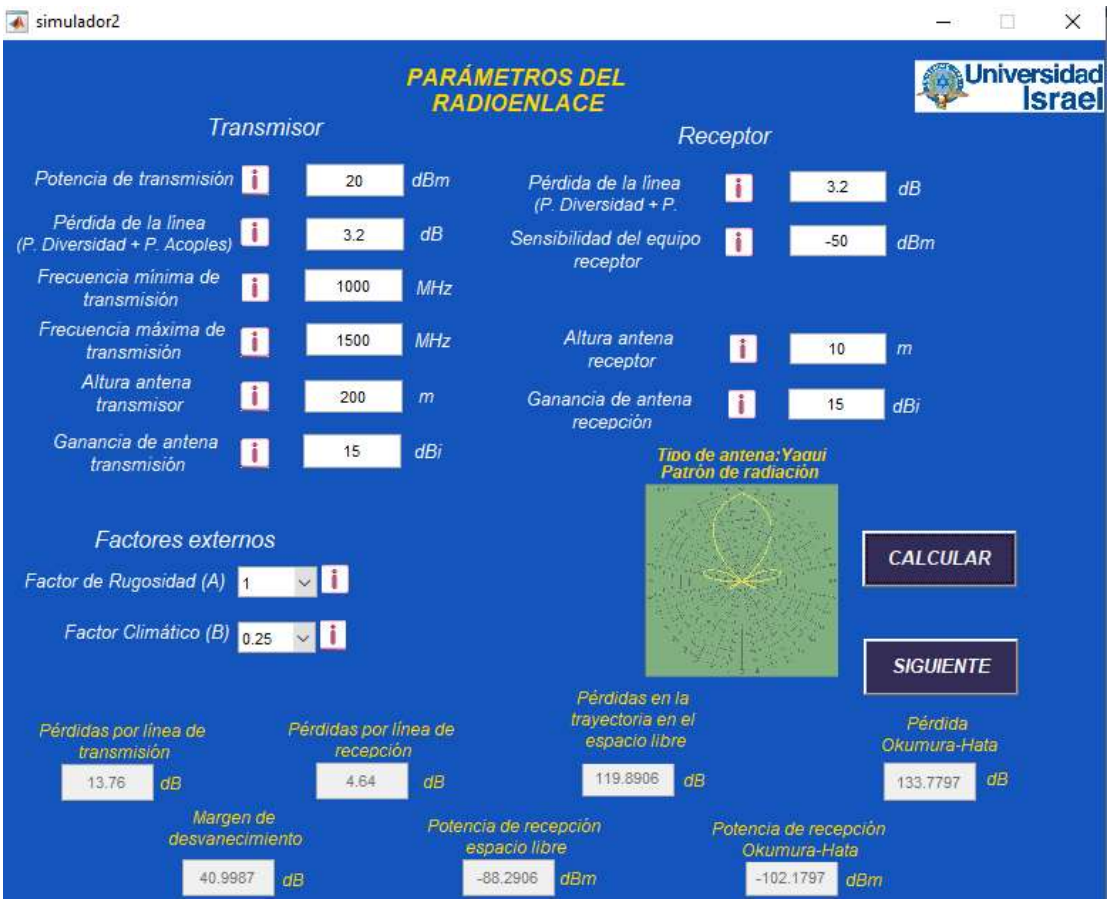


Figura. 4. 23 Pantalla para el ingreso de parámetros del radioenlace.

Fuente: Elaborado por el autor

En la parte inferior de la pantalla, se muestran los parámetros del radioenlace, que de acuerdo con los datos ingresados se puede analizar los resultados:

- *Potencia de recepción en espacio libre (dBm):* Este parámetro corresponde a la potencia de recepción sin la aplicación de ningún modelo de predicción, lo cual permite al usuario tener una referencia de la potencia de la señal en esta unidad.

- *Potencia de recepción Okumura Hata (dBm):* De acuerdo con los parámetros ingresados, se utiliza el modelo Okumura Hata para obtener la potencia de recepción, en este caso su valor fue de 44.6797 dBm

- *Patrón de radiación:* Dado que la antena es tipo Yagui, se obtiene un patrón de radicación Directivo.

Por último, los resultados se visualizan como en la Figura 4.24 en la interfaz llamada "*simulador3d*":

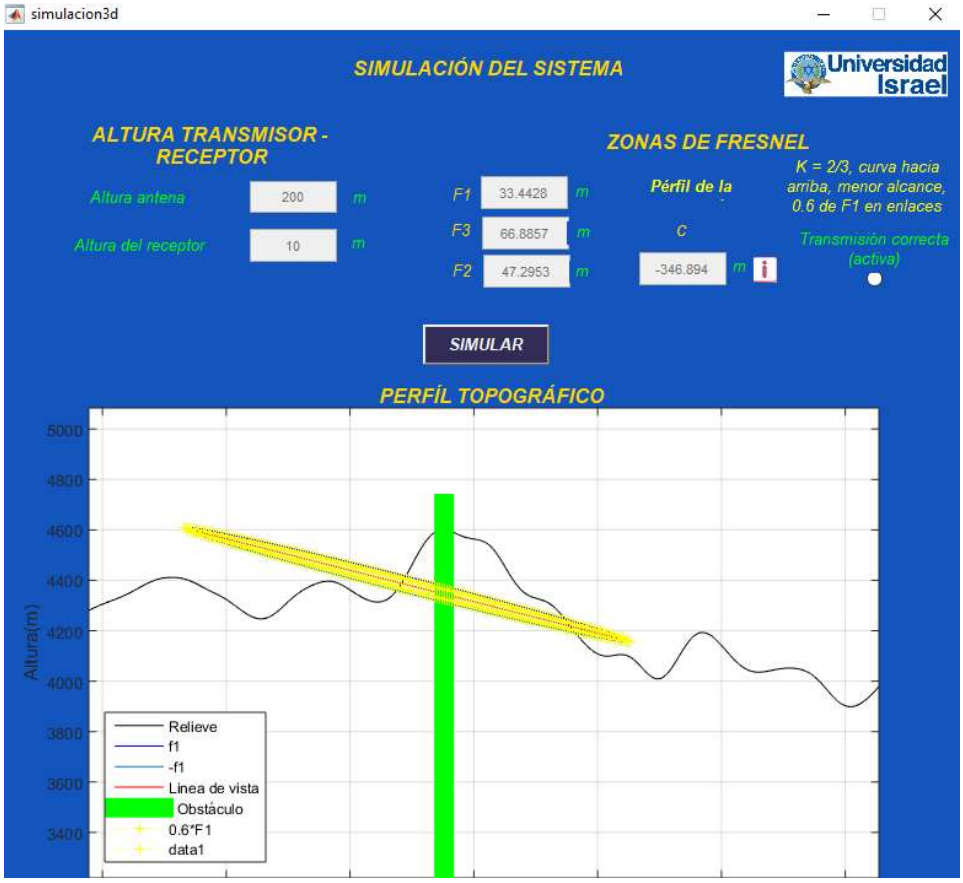


Figura. 4. 24 Pantalla de resultados

Fuente: Elaborado por el autor

Se puede apreciar que entre el Punto 1 transmisor y el Punto 2 receptor existe aproximadamente una distancia de 18.8 km, donde el primer punto se ubica a una altura de 200 metros de la superficie (correspondiente al parámetro de la altura de transmisor) y el segundo punto a una altura de 10 m (correspondiente al parámetro de la altura del receptor). Se observa que una montaña interrumpe la línea de vista entre el transmisor y el receptor, lo cual afecta la

zona de Fresnel y provoca una pérdida de potencia de recepción de la señal. La zona de Fresnel para el siguiente caso tendrá los parámetros que se muestran en la figura 4.25:

Figura. 4. 25 Valores de los parámetros de Fresnel.

Fuente: Elaborado por el autor

En la parte superior derecha se observa que el Radio Button: Transmisión correcta (activa) que indica si la Transmisión se ha realizado correctamente, no está activo, como se observa en la figura 4.25, dado que existe un obstáculo entre los dos puntos. Esto se puede comprobar con el valor de "c", mostrado en la misma figura, el cual corresponde al valor de la distancia entre la primera zona de Fresnel y la altura del obstáculo, el cual, en este caso es un valor negativo de -346.894. El signo negativo quiere decir que el obstáculo supero la primera zona de Fresnel, e incluso cruzó la línea de vista, por lo que se infiera que la transmisión es incorrecta. Además, se observa que F3 es mayor que F2 y esta a su vez es mayor que F1, lo cual quiere decir que las zonas de Fresnel van aumentando en distancia a la línea de vista.

Prueba 2:

En la figura 4.26 se observa cómo se configura la interfaz "*Simulador*" con las siguientes coordenadas:

PUNTO 1 TRANSMISOR

Latitud: S 0°, 9′, 52.3"

Longitud: O 78°, 28′, 34.5"

PUNTO 2 RECEPTOR

Latitud: S 0°, 14′, 35.9”

Longitud: O 78°, 30′, 0.1”

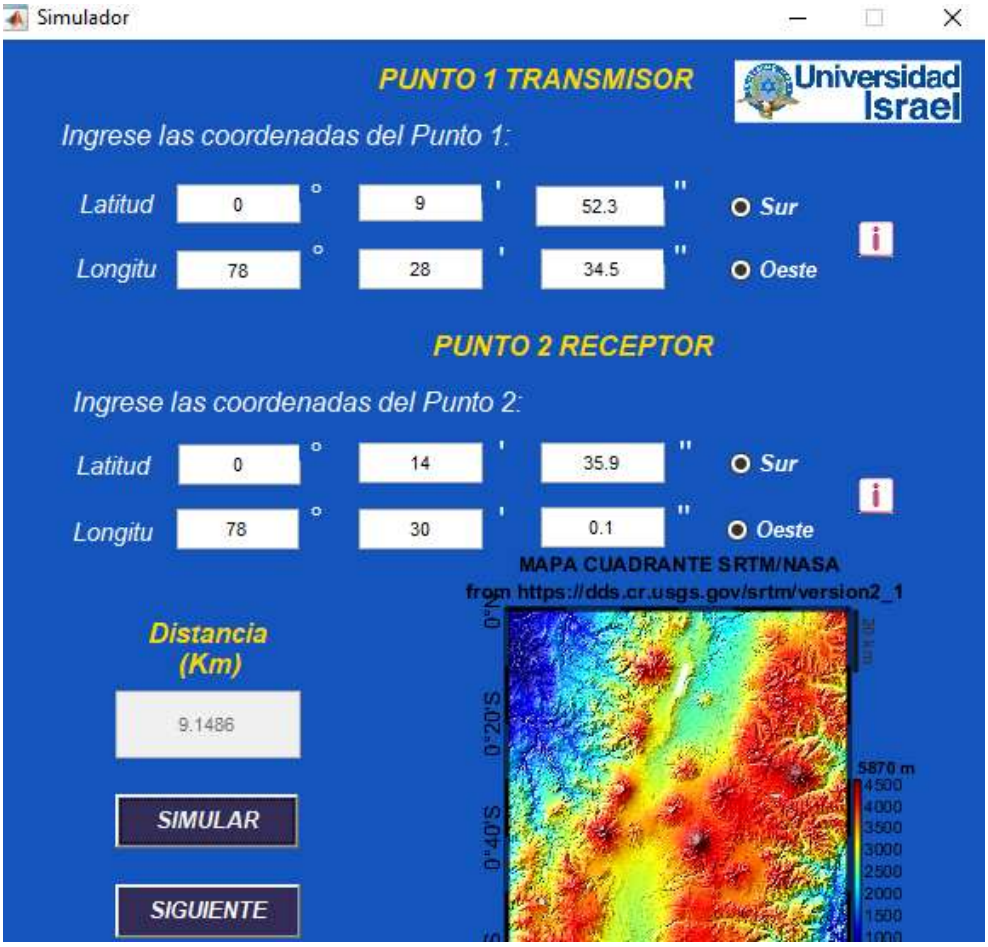


Figura. 4. 26 Ingreso de coordenadas para prueba 2.

Fuente: Elaborado por el autor

En la figura 4.27 se muestra la distancia en este caso es 9.1486 km. Similar al caso anterior, las coordenadas corresponden a un mapa SRTM donde se encuentra la ciudad de Quito. Para esta prueba, el Punto 1 transmisor corresponde al sector del Inca, mientras que el Punto 2 receptor corresponde al sector de la Loma de Puengasí

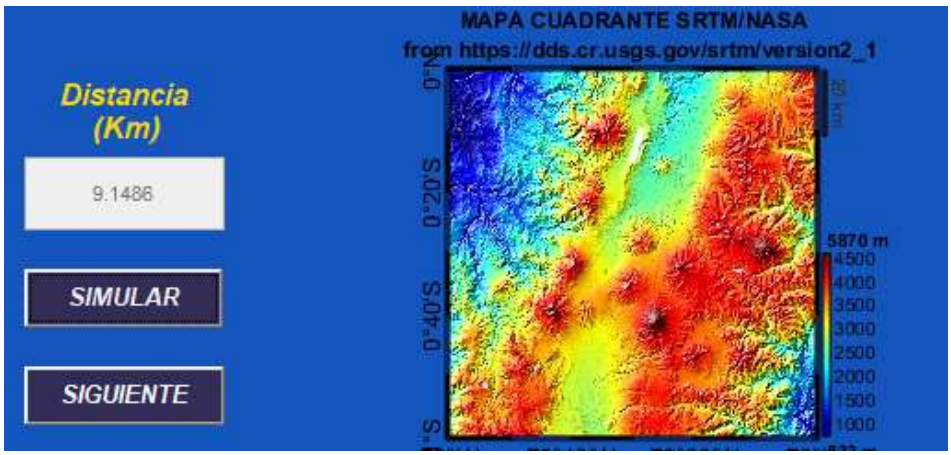


Figura. 4. 27 Resultado de la distancia entre transmisor y receptor.

Fuente: Elaborado por el autor

En la figura 4.28 se observa la comprobación de los datos ingresados en Google Maps:

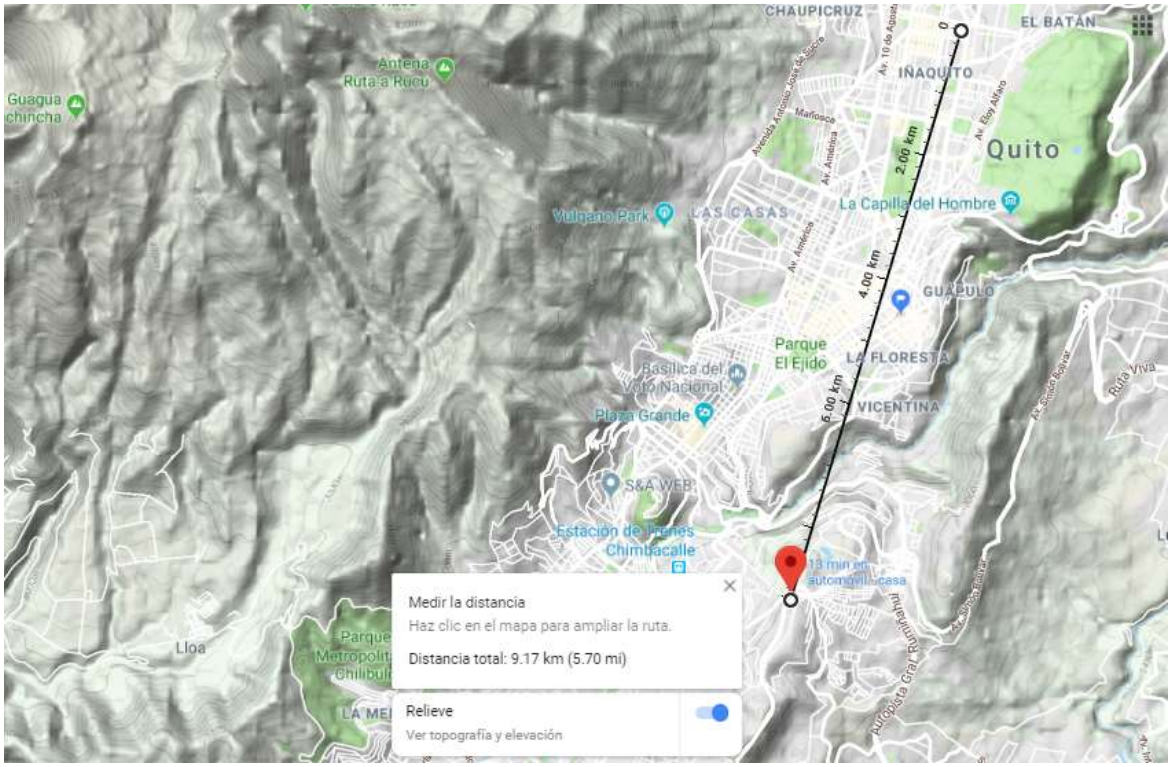


Figura. 4. 28 Puntos de radioenlace para prueba 2 en Google Maps.

Fuente: Elaborado por el autor

La distancia obtenida en Google Maps, es de 9.17 km, la cual es muy aproximada a los datos obtenidos en la Pantalla 1. Se observa que en el primer punto existe un leve relieve, en el

cual se ubica la antena de transmisión, por otro lado, en el segundo punto se encuentra en una zona de menor altura, lo cual es el escenario ideal para una buena transmisión.

Se comprueban los datos ingresados en Radio Mobile, como se observa en la Figura 4.29

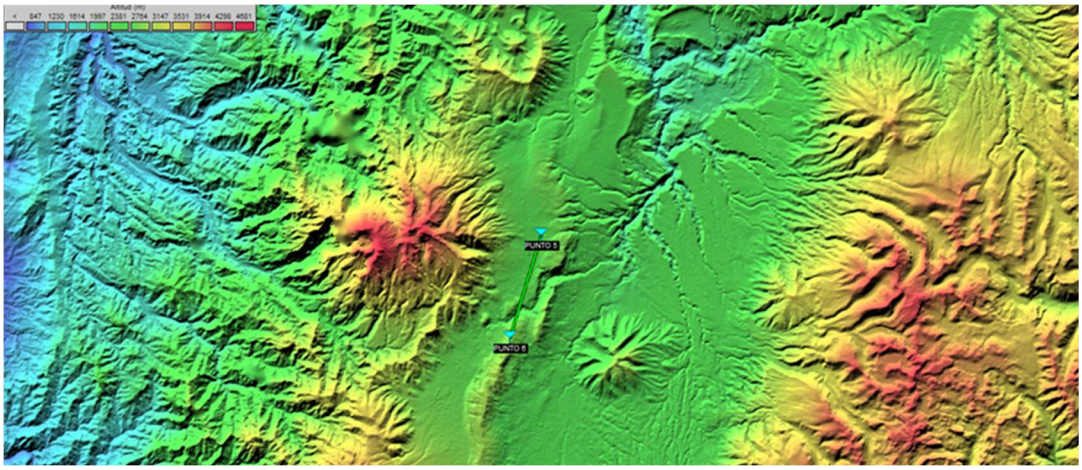


Figura. 4. 29 Puntos de radioenlace para prueba 2 en Radio Mobile.

Fuente: Elaborado por el autor

Con los datos obtenidos se puede realizar la siguiente Tabla 4.2, en la cual se compara y se demuestra que las distancias son similares.

Tabla. 4 2 Comparación de la distancia calculada entre los simuladores del enlace 2

SIMULADOR	DISTANCIA CALCULADA [Km]
Matlab (Figura. 4.5)	9.1486
Google Maps (Figura. 4.6)	9.17
Radio Mobile (Figura. 4.7)	9.14

Fuente: Elaborado por el autor

Una vez dado click en el botón simular se abrirá otra interfaz llamada “simulador2”, en la cual se deberá ingresar parámetros de la antena, tal y como se ve en la Figura 4.30:

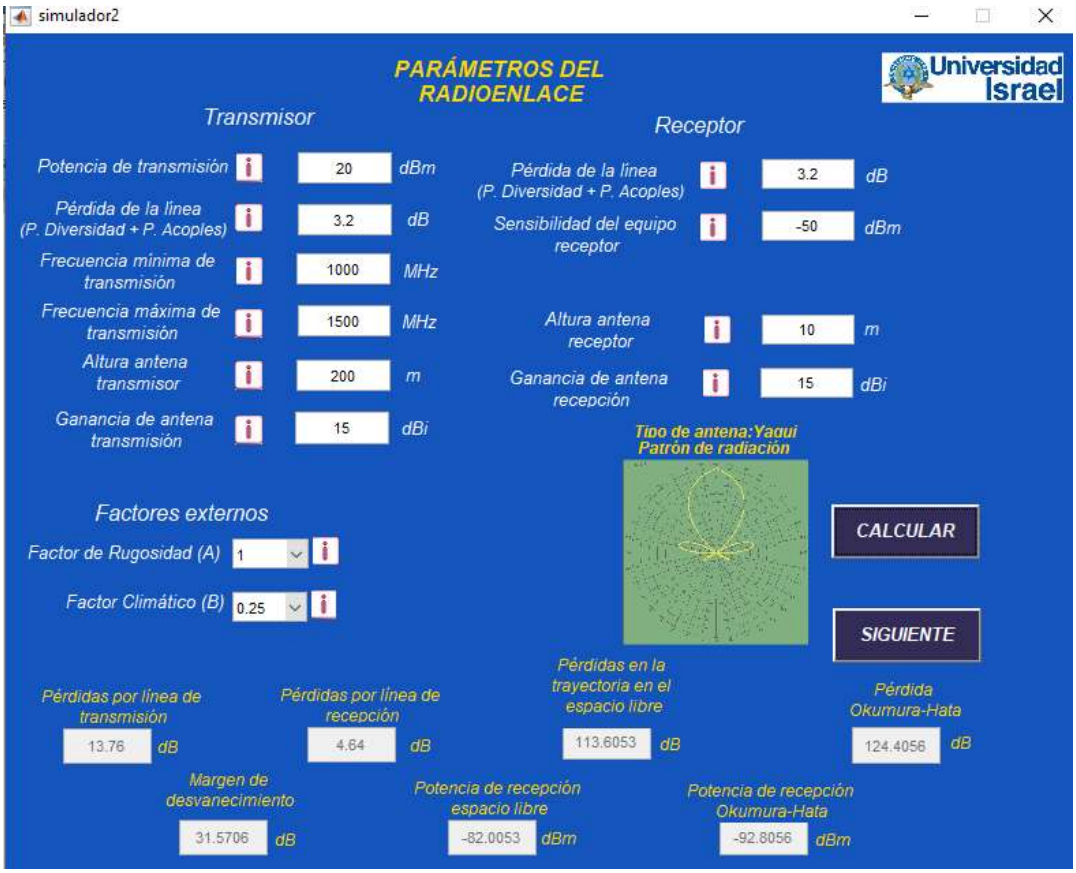


Figura. 4. 30 Ingreso de parámetros de la antena para prueba 2.

Fuente: Elaborado por el autor

Similar al caso anterior, algunos de los datos obtenidos son muy similares, dado que las características de la antena son las mismas, sin embargo, otros datos si varían porque su cálculo depende de la distancia entre los dos puntos del radioenlace.

Por último, se tienen los siguientes resultados, los cuales se visualizan en la interfaz llamada "*simulador3d*" de la Figura 4.31:

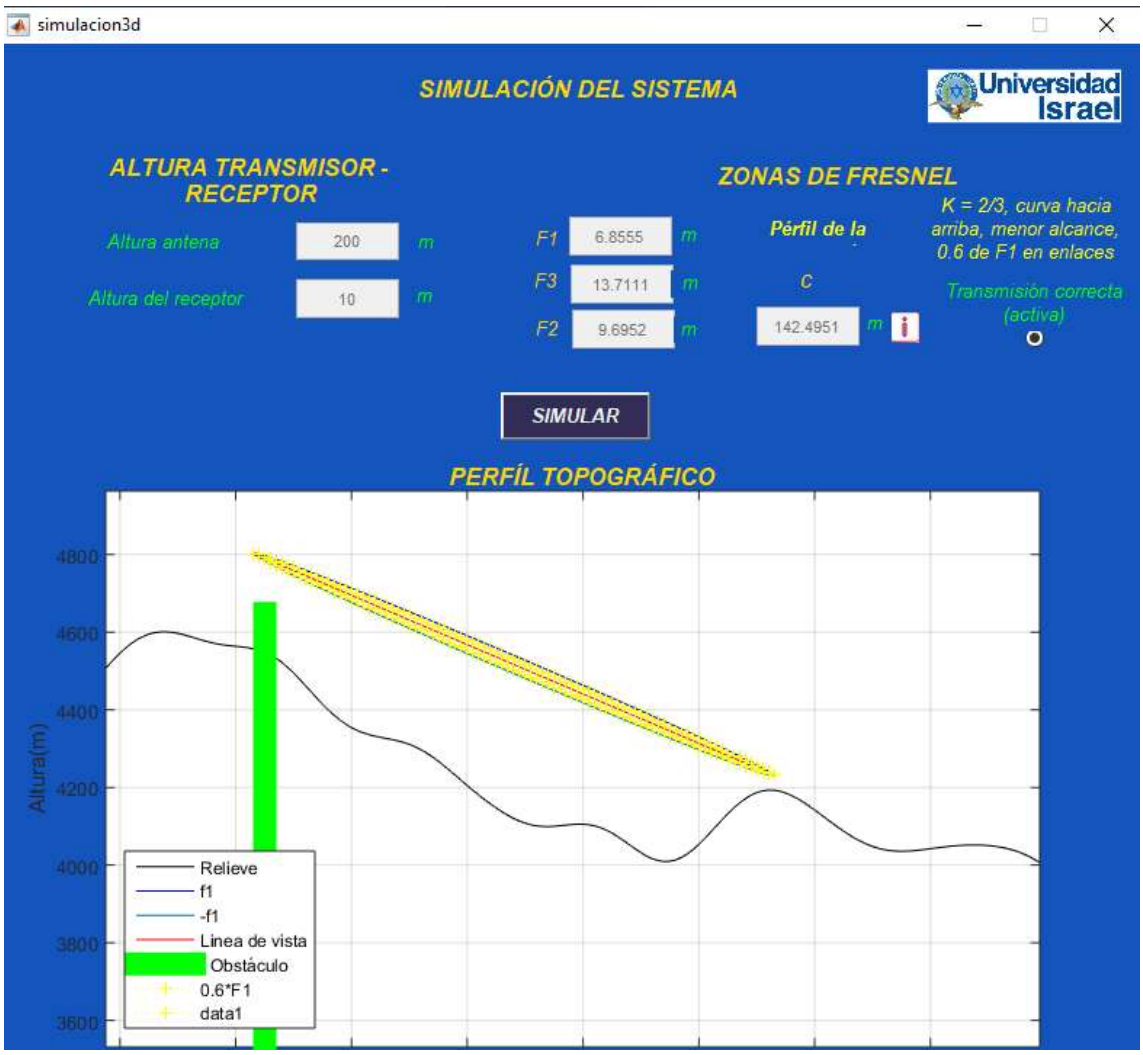


Figura. 4. 31 Resultados obtenidos prueba 2

Fuente: Elaborado por el autor

En este caso, la señal se transmite sin problema hasta llegar al final, por lo cual se considera que la transmisión es correcta debido a que no hay ningún obstáculo por la señal, el indicador Transmisión correcta se encuentra activo, como se visualiza en la Figura 4.32. Para este caso la zona de Fresnel tendrá los siguientes parámetros:

Figura. 4. 32 Parámetros zona de Fresnel prueba 2.

Fuente: Elaborado por el autor

Se observa que el valor de "c", en este caso es positivo lo cual quiere decir que no sobrepasa la primera Zona de Fresnel, y además se observa que el *Radio Button*: Transmisión correcta (activa) que indica si la Transmisión se ha realizado correctamente, si está activo, esto se debe a que no existe un obstáculo entre los dos puntos, y se logrará una transmisión sin mayores pérdidas.

4.4. Análisis de Resultados

Para analizar los resultados se calculará la simulación anterior de forma matemática y se comparará los resultados obtenidos:

4.4.1 Cálculo matemático

Datos Generales

Parámetro	Punto A	Punto B
Frecuencia Media	1250 MHz	
Distancia	9.1486 Km	
Pérdida Guía de Onda	4.8 dB/100	4.8 dB/100 m
Ganancia de Antena	15 dBi	15 dBi
Pérdidas Filtros + Diversidad	3.2 dB	3.2 dB
Altura de Antena	200 m	10 m
Umbral de Rx	-50 dBm	-50 dBm
Potencia de Tx	100 mW	---------

Factor de Rugosidad del Medio		**Factor Climático del Medio**	
Terreno Normal	1	Región Sierra, clima normal	0.25

Cálculo de pérdidas

Pérdidas por alimentación Punto A

Distancia total = Altura antena + 20 m = 220 m

100m - 4.80
220 m x

Pérdida Cable = 10.56 dB

Pérdidas por alimentación Punto A = Pérdidas Cable + Pérdidas acoples y diversidad

Pérdidas por alimentación Punto A La= 10.56 + 3.2 = **13.76 dB**

Pérdidas por alimentación Punto B

Distancia total = Altura antena + 20 m = 30 m

100m - 4.80
30 m x

Pérdida Cable = 1.44 dB

Pérdidas por alimentación Punto B = Pérdidas Cable + Pérdidas acoples y diversidad

Pérdidas por alimentación Punto B Lb = 1.44 + 3.2 = **4.64 dB**

Pérdida en el Espacio Libre (Lb)

$$Lp\ (dB) = 32{,}4 + 20\log\ f(MHz) + 20\log\ D\ (Km)$$

$$Lp\ (dB) = 32{,}4 + 20\log\ (1250\ MHz) + 20\log\ 9.1486\ (Km)$$

$$\boldsymbol{Lp\ (dB) = 113.565\ dB}$$

Pérdida en el Espacio Modelo Okumura - Hata (LbOH)

Parámetro a(hRx)

$$a(h_{Rx}) = (1.1\log(f) - 0.7) * h_{Rx} - (1.56\log(f) - 0.8)$$

$$a(h_{Rx}) = (1.1\log(1250) - 0.7) * 10 - (1.56\log(1250) - 0.8)$$

$$\boldsymbol{a(h_{Rx}) = 23.0348}$$

$$L_b = 69.55 + 26.16 * \log(f) - 13.82\log(h_{Tx}) - a(h_{Rx}) + (44.9 - 6.55\log(h_{Tx}))\log(d)$$

$$L_b = 69.55 + 26.16 * \log(1250\ MHz) - 13.82\log(200) - 23.0348 + (44.9 - 6.55\log(200))\log(9.1486\ Km)$$

$$\boldsymbol{L_b = 124.092\ dB}$$

Potencia de recepción solo espacio libre

$$PR_x = PT_x + G_{atx} + G_{arx} - L_a - L_b - L_{FbOH}$$

$$PR_x = 20\ dBm + 15\ dB + 15dB - 13.76\ dB - 4.64\ dB - 113.565\ dB$$

$$\boldsymbol{PR_x = -81.965\ dBm}$$

Potencia de recepción Okumura – Hata

$$PR_x = PT_x + G_{atx} + G_{arx} - L_a - L_b - L_{Fb}$$

$$PR_x = 20\ dBm + 15\ dB + 15dB - 13.76\ dB - 4.64\ dB - 124.092\ dB$$

$$\boldsymbol{PR_{xO-H} = -92.492\ dBm}$$

Margen de desvanecimiento Teórico

$$F_m = 30\log(d) + 10\log(6 * A * B * f) - 10\log(1 - R) - 70$$

$$F_m = 30\log(9.1486) + 10\log(6 * 0.25 * 0.25 * 1250) - 10\log(1 - 0.9999) - 70$$

$$\boldsymbol{F_m = 25,55\ dB}$$

Margen de desvanecimiento Real

$$FM_r = |Umbral\ de\ recepción| - |\ Potencia\ de\ recepción|$$

$$FM_r = |50| - |81{,}965|$$

$$\boldsymbol{FM_r = -31.965\ dB}$$

Con los valores calculados y los indicados en el software para los mismos parámetros, se los puede comparar y observar que los valores son similares en la tabla 4.3:

Tabla. 4 3 Comparación de Resultados

	Valores Aplicación	Valores Calculados
Pérdidas Espacio libre	113,6053 dB	113,565 dB
Pérdidas Okumura - Hata	124,4056 dB	124,092 dB
Potencia Recepción Espacio Libre	-82,0053 dBm	-81,965 dBm
Potencia Recepción Okumura - Hata	-92,8056 dBm	-92,492 dBm
Margen de Desvanecimiento	25,55 dB	25,55 dB

Fuente: Elaborado por el autor

CONCLUSIONES

El uso de los simuladores informáticos para radioenlaces comúnmente conocidos como Radio Mobile, Link Planner, se orientan solamente en modelos de marcas de antenas propietarias o en modelos de propagación en zonas rurales, de manera que la simulación con la aplicación elaborada en Matlab licenciado en un entorno Windows, es más precisa en sus cálculos realizados en zonas urbanas mediante un patrón de radiación de una antena en común, sin atarse a una marca en particular.

Algunos trabajos realizados por estudiantes de universidades, los realizan en plataformas de códigos propietarios, sin tener la posibilidad de observar los cálculos que estás realizan para las pérdidas y potencias de recepción de un radioenlace, por lo que las soluciones en código libre como esta implementación, tomarán fuerzas ya que puede incluirse mejoras como la implementación de varios modelos matemáticos para otros tipos de ambientes y zonas climáticas. Al igual que sea considerado por varios fabricantes que tengan sus productos compatibles con los parámetros que se estiman en esta aplicación.

Con los datos obtenidos en la simulación y su análisis se puede determinar que, con los cálculos realizados, se puede obtener la información de los parámetros de los equipos que conforman la trasmisión y recepción del radioenlace, para su balance, optimizando las características de estos para un mejor funcionamiento en zonas urbanas.

El tema de estudio para esta aplicación, se basa en la investigación y análisis técnico de todos los conceptos de ganancias y pérdidas que intervienen en el proceso de transmisión y recepción de un radioenlace incluyendo los modelos de matemático de Friis y de Okumura – Hata, que aportan significativamente al entendimiento de lo propuesto.

Con los datos ingresados se obtiene el perfil topográfico y se ilustra de manera gráfica la zona de Fresnel entre los puntos de transmisión y recepción ubicados en la ciudad de Quito, mostrando así una visualización de análisis complementario para el usuario.

.

RECOMENDACIONES

Se recomienda que los usuarios antes de utilizar el programa verifiquen la guía o manual de usuario que consta en el ANEXO 1, así como también dispongan de un conocimiento básico acerca de radioenlaces, para tener un criterio adecuado al momento de interpretar los resultados.

Se sugiere ingresar frecuencias que no sobrepasen el rango de 150 MHz a 1500 MHz, que son los tipos adecuados para el modelo de propagación de Okumura-Hata. Utilizar ubicaciones que estén dentro del rango perteneciente a la ciudad de Quito e ingresar alturas y potencias coherentes para tener resultados que se aproximen a la realidad.

Se recomienda el uso de esta aplicación como base para trabajos futuros y continuar el desarrollo para otros modelos de propagación que permitan tener un rango más amplio de frecuencias, que se acoplen a otros entornos suburbanos y rurales, zonas de bosques, selvas, etc.

También sería interesante continuar desarrollando estos programas como nuevas herramientas de código abierto en otros entornos y en otros lenguajes de programación como Java, Python, Lua, C++, etc., que inclusive sean multiplataforma y que funcionen en distintos sistemas operativos, esto ayudará básicamente a los estudiantes y profesionales del área de las Telecomunicaciones a disponer de múltiples herramientas que faciliten el análisis de sus proyectos y estudios de campo.

REFERENCIAS BIBLIOGRÁFICAS

- Álvarez, J. H. (24 de Septiembre de 2014). Unversidad Carlos III de Madrid. Obtenido de Sistema Gráfico para: https://e-archivo.uc3m.es/bitstream/handle/10016/26257/TFG_Javier_Hernandez_Alvarez_2014.pdf
- ARCOTEL. (18 de febrero de 2015). ARCOTEL. Obtenido de http://www.arcotel.gob.ec/wp-content/uploads/downloads/2016/01/ley-organica-de-telecomunicaciones.pdf
- Bravo, É. O. (2008). Dspace Espol . Obtenido de https://www.dspace.espol.edu.ec/bitstream/123456789/10963/3/ERIKA%20ORDONEZ%20B.%20TESIS%20FINAL.pdf
- Bucheli, C. S. (Mayo de 2017). Repositorio EPN. Obtenido de https://bibdigital.epn.edu.ec/bitstream/15000/17305/1/CD-7799.pdf
- BUSTAMANTE, I. J. (2007). Repositorio Dspace. Obtenido de http://repositorio.espe.edu.ec/xmlui/bitstream/handle/21000/139/T-ESPE-025041.pdf?sequence=1&isAllowed=y
- Castro, G. A. (13 de Marzo de 2017). Repositorio Universidad Católica de Guayaquil. Obtenido de http://repositorio.ucsg.edu.ec/bitstream/3317/7692/1/T-UCSG-PRE-TEC-ITEL-190.pdf
- Durland D., N. D. (s.f.). “Modelos de Propagación”. Obtenido de Modelos Semiempíricos de propagación: http://durlancito.wixsite.com/propagacion/modelos
- Gabriela Noemi Nartínez Jara, M. D. (2016). dspace.espoch. Obtenido de http://dspace.espoch.edu.ec/bitstream/123456789/5445/1/98T00099.pdf
- Hata, M. (1980). Empirical formula for propagation loss in land mobile radio services. IEEE Transactions on Vehicular Technology, 317-325.
- Herrera, J. R. (2008). Universidad Nacional de Ingenieria. Obtenido de http://cybertesis.uni.edu.pe/bitstream/uni/1373/1/fernandez_hj.pdf
- ITU. (2015). ITU NEWS. Obtenido de http://itunews.itu.int/es/5907-El-origen-de-la-UIT-y-su-importancia-en-la-actualidad.note.aspx
- Morocho, M. V. (Enero de 2011). Radiocomunicaciones. Obtenido de http://www.radiocomunicaciones.net/pdf/planificacion-radioenlace-cartografia-digital.pdf
- NIETO, F. H. (26 de junio de 1990). Repositirio IAEN. Obtenido de IAEN: http://repositorio.iaen.edu.ec/bitstream/24000/4193/1/HIdalgo%20Fernando.pdf

- Quinzo, F. H. (2012). Repositorio Universidad Nacional del Chimborazo. Obtenido de http://dspace.unach.edu.ec/bitstream/51000/749/1/UNACH-EC-IET-2012-0007.pdf
- Raúl Hartmam, S. N. (2008). Facultad de Ingenieria de Uruguay. Obtenido de https://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR0813.pdf
- Remache, P. (Abril de 2015). Estudio y Diseño de un radio enlace para transmisión de datos, e internet en frecuencia libre para la cooperativa indigena "Alfa y Omega" utilizandp equipos Airmax de Ubiquiti. Obtenido de https://bibdigital.epn.edu.ec/bitstream/15000/10776/1/CD-6315.pdf
- Salazar, C. (2019). Cap. 6 Modelos de propagación. Obtenido de Curso de propagación en entornos Urbanos: https://eva.fing.edu.uy/pluginfile.php/63628/mod_resource/content/1/clase_6_Modelos_de_Propagacion.pdf
- Sánchez, J., & Hoy, V. (s.f.). Slide Share. Obtenido de http://www.slidesahre.net/josefranciscos/modelo-de-difraccin-por-filo-de-cuchillo-microondas-uft
- Tomasi, W. (2003). SISTEMAS DE COMUNICACIONES ELECTRONICAS . Phoenix, Arizona , United States of America.
- VE2DBE. (1997). Radio Mobile VE2DBE. Obtenido de https://www.ve2dbe.com/english1.html
- XIRIOonline. (1 de Agosto de 2019). Obtenido de https://www.xirio-online.com/help/es/okumura-hata.htm

- Barcelona, U. A. (04 de 2019). Seañales y Propagacion. Obtenido de https://www.studocu.com/es/u/2444509
- Cid, M. A. (11 de 05 de 2011). Ciencias Ubiobio. Obtenido de http://ciencias.ubiobio.cl/fisica/wiki/uploads/AntonellaCid/F2_14.pdf
- CSIC), C. d. (2009). Proyecto Académico con el Radio Telescopio de NASA en Robledo . Obtenido de http://www.partner.cab.inta-csic.es/index.php?Section=Curso_Fundamentos_Capitulo_4#1.1
- SSR, D. (2007). Tutorial de Radio Mobile . ETSIT-UPM.
- Torres-Papaqui, D. J. (2016). Tipos de Ondas Mecánicas. Obtenido de http://www.astro.ugto.mx/~papaqui/ondasyfluidos/Tema_1.01-Tipos_de_Ondas_Mecanicas.pdf

ANEXOS

ANEXO 1 MANUAL DE USUARIO

MANUAL DE USO DE LA APLICACIÓN

En esta sección se proporcionará un manual de usuario donde se explica el procedimiento a realizarse para el uso de la aplicación. La explicación se realizará en base a dos ejemplos, uno en donde la transmisión se ejecute de manera correcta, y otra donde por la existencia de un obstáculo la transmisión no se realiza satisfactoriamente.

- Como primer paso, se procede a abrir el software Matlab.

- Una vez abierto Matlab se introducen los comandos clear all y clc en el Command Window para eliminar cualquier dato que haya quedado en la memoria de datos temporal como se muestra en la Figura 1.

Figura 1. Command Window con comandos clc y clear all.

- Se digita en el Command Window de Matlab el comando guide, como se observa en la Figura 2.

Figura 2. Command Window con comando guide.

- Se abrirá la pantalla GUIDE Quick Start, en la pestaña Open Existing GUI como se muestra en las imágenes de la Figura 3 "*GUIDE Quick Start*", Figura 4 "*Buscar en GUIDE Quick*

Start”, y Figura 5 “*Ventana para seleccionar y abrir el archivo extensión ..fig*.”, mediante el botón *Browse*… se busca y selecciona el archivo Simulador.fig y click en Abrir.

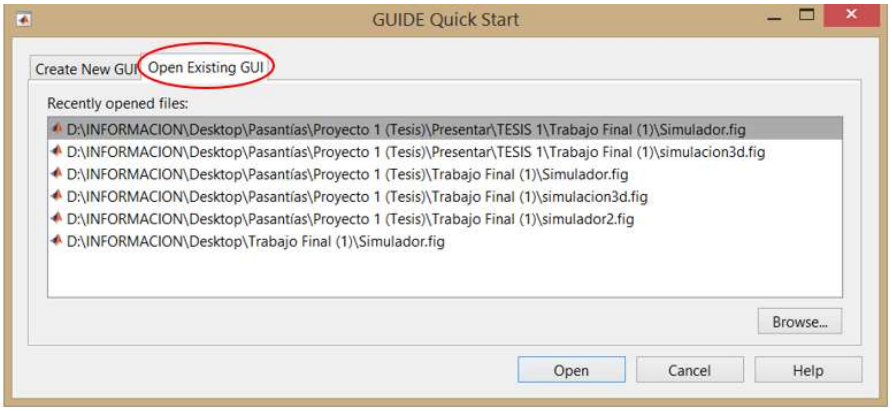


Figura 3. GUIDE Quick Start.

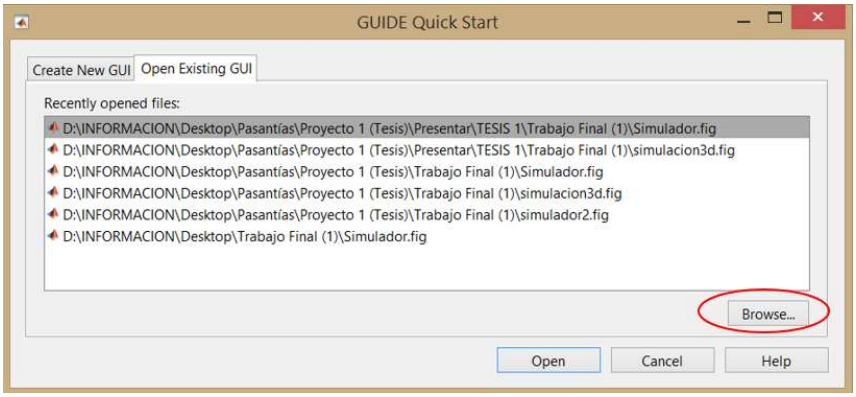


Figura. 4 Buscar en GUIDE Quick Start.

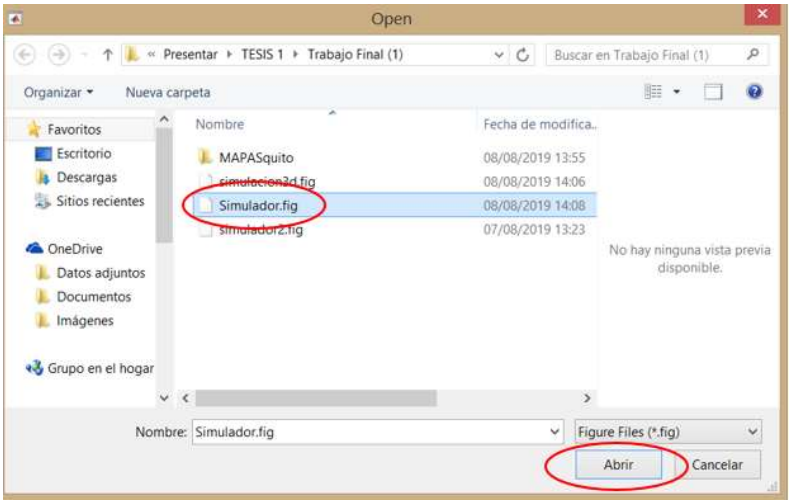


Figura. 5 Ventana para seleccionar y abrir el archivo extensión.fig.

- A continuación, como se observa en la Figura 6, se abrirá la pantalla de edición de *Simulador.fig* y dando click en Run comenzará a ejecutarse la aplicación diseñada.

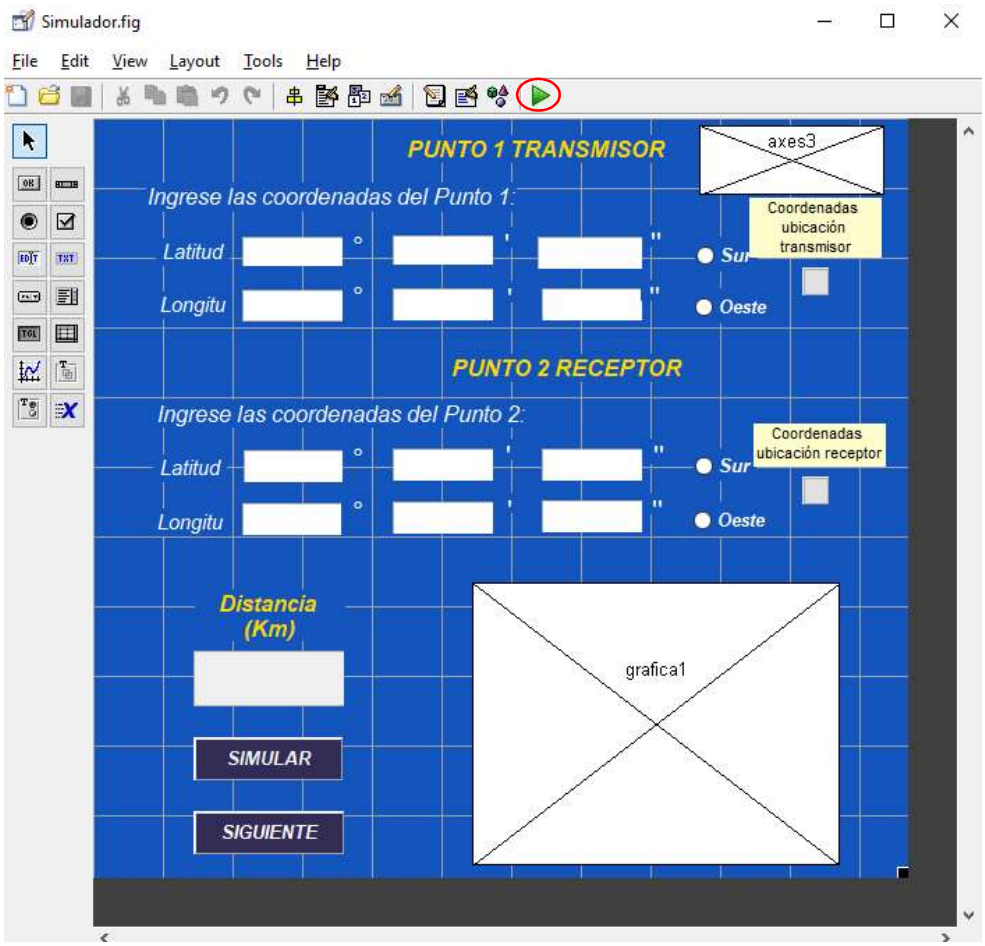


Figura.6 Pantalla de editor gráfico.

ANEXO 2 CÓDIGO DE MATLAB

❖ Código completo función *Simulador*

Contiene el código de la primera pantalla.

```
function varargout = Simulador(varargin)

% SIMULADOR MATLAB code for Simulador.fig
%      SIMULADOR, by itself, creates a new SIMULADOR or raises the existing
%      singleton*.
%
%      H = SIMULADOR returns the handle to a new SIMULADOR or the handle to
%      the existing singleton*.
%
%      SIMULADOR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SIMULADOR.M with the given input
arguments.
%
%      SIMULADOR('Property','Value',...) creates a new SIMULADOR or raises
the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before Simulador_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Simulador_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 Simulador

% Last Modified by GUIDE v2.5 30-Aug-2019 15:56:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Simulador_OpeningFcn, ...
                   'gui_OutputFcn',  @Simulador_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 Simulador is made visible.
function Simulador_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 Simulador (see VARARGIN)

%LOGO UISRAEL
logo=imread('logoU2.jpg');
axes(handles.axes3);
imshow(logo);

%ICONO INFO1
info=imread('info.jpg');
set(handles.info1,'CData',info)

%ICONO INFO1
info=imread('info.jpg');
set(handles.info2,'CData',info)

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

% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = Simulador_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;


% --- EJECUTA AL PRESIONAR EL BOTÓN sim.
function sim_Callback(hObject, eventdata, handles)
global lati1 long1 lati2 long2 Xmat

% PUNTO 1
%GRADOS, MINUTOS Y SEGUNDOS DE LATITUD
grados1=str2num(get(handles.grado1,'string'));
mins1=str2num(get(handles.min1,'string'));
segs1=str2num(get(handles.seg1,'string'));
```

```
signo=get(handles.Sur1,'value');
if signo==1
    latitud1=-1*convertir(grados1,mins1,segs1);
    lati1=latitud1*180/pi
else
    latitud1=convertir(grados1,mins1,segs1);
    lati1=latitud1*180/pi
end

%GRADOS, MINUTOS Y SEGUNDOS DE LONGITUD
grados2=str2num(get(handles.grado2,'string'));
mins2=str2num(get(handles.min2,'string'));
segs2=str2num(get(handles.seg2,'string'));
signo1=get(handles.oeste1,'value');
if signo1==1
    longitud1=-1*convertir(grados2,mins2,segs2);
    long1=longitud1*180/pi
else
    longitud1=convertir(grados2,mins2,segs2);
    long1=longitud1*180/pi
end

% PUNTO 2
%GRADOS, MINUTOS Y SEGUNDOS DE LATITUD
grados3=str2num(get(handles.grado3,'string'));
mins3=str2num(get(handles.min3,'string'));
segs3=str2num(get(handles.seg3,'string'));
signo2=get(handles.sur2,'value');
if signo2==1
    latitud2=-1*convertir(grados3,mins3,segs3);
    lati2=latitud2*180/pi
else
    latitud2=convertir(grados3,mins3,segs3);
    lati2=latitud2*180/pi
end

%GRADOS, MINUTOS Y SEGUNDOS DE LONGITUD
grados4=str2num(get(handles.grado4,'string'));
mins4=str2num(get(handles.min4,'string'));
segs4=str2num(get(handles.seg4,'string'));
signo3=get(handles.oeste2,'value');
if signo3==1
    longitud2=-1*convertir(grados4,mins4,segs4);
    long2=longitud2*180/pi
else
    longitud2=convertir(grados4,mins4,segs4);
    long2=longitud2*180/pi
end

%Ecuación de la recta
pend=(lati2-lati1)./(long2-long1);
brecta=lati1-(pend*long1);

%DISTANCIA ENTRE LOS PUNTOS
global distancias64
```

```
distancias64=distancia(longitud1,latitud1,longitud2,latitud2);%LLAMA A LA
FUNCIÓN DISTANCIA, DISTANCIA ENTRE LOS PUNTOS
val=num2str(distancias64);
set(handles.dis,'string',val);

x=[latitud1 latitud2];
y=[longitud1 longitud2];

axes(handles.grafica1);
X=readhgt('plot');

Xmat=X;

set(handles.btnSiguiente,'visible','on')%BOTON SIGUIENTE VISIBLE

if long2>long1
    t1=long1:(1/1021):long2;
end
if long2<long1
    t1=long2:(1/1201):long1;
end
fun=pend*t1+brecta;
hold on
plot(t1,fun,'-w','LineWidth',3)
zoom on
%plot(handles.grafica1,x,y);
grid on;
hold off


%simulador2();%LLAMA A SIMULADOR 2


function entrada_Callback(hObject, eventdata, handles)
% hObject    handle to entrada (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 entrada as text
%        str2double(get(hObject,'String')) returns contents of entrada as a
double


% --- Executes during object creation, after setting all properties.
function entrada_CreateFcn(hObject, eventdata, handles)
% hObject    handle to entrada (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 salida_Callback(hObject, eventdata, handles)
% hObject    handle to salida (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 salida as text
%        str2double(get(hObject,'String')) returns contents of salida as a
double


% --- Executes during object creation, after setting all properties.
function salida_CreateFcn(hObject, eventdata, handles)
% hObject    handle to salida (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 entrada_Callback(hObject, eventdata, handles)
% hObject    handle to entrada (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 entrada as text
%        str2double(get(hObject,'String')) returns contents of entrada as a
double


% --- Executes during object creation, after setting all properties.
function entrada_CreateFcn(hObject, eventdata, handles)
% hObject    handle to entrada (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 grado1_Callback(hObject, eventdata, handles)
% hObject    handle to grado1 (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 grado1 as text
%        str2double(get(hObject,'String')) returns contents of grado1 as a
double


% --- Executes during object creation, after setting all properties.
function grado1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to grado1 (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 min1_Callback(hObject, eventdata, handles)
% hObject    handle to min1 (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 min1 as text
%        str2double(get(hObject,'String')) returns contents of min1 as a
double


% --- Executes during object creation, after setting all properties.
function min1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to min1 (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 seg1_Callback(hObject, eventdata, handles)
% hObject    handle to seg1 (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 seg1 as text
%        str2double(get(hObject,'String')) returns contents of seg1 as a
double


% --- Executes during object creation, after setting all properties.
function seg1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to seg1 (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 grado2_Callback(hObject, eventdata, handles)
% hObject    handle to grado2 (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 grado2 as text
%        str2double(get(hObject,'String')) returns contents of grado2 as a
double


% --- Executes during object creation, after setting all properties.
function grado2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to grado2 (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 min2_Callback(hObject, eventdata, handles)
% hObject    handle to min2 (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 min2 as text
```

```
%        str2double(get(hObject,'String')) returns contents of min2 as a
double


% --- Executes during object creation, after setting all properties.
function min2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to min2 (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 seg2_Callback(hObject, eventdata, handles)
% hObject    handle to seg2 (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 seg2 as text
%        str2double(get(hObject,'String')) returns contents of seg2 as a
double


% --- Executes during object creation, after setting all properties.
function seg2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to seg2 (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 grado3_Callback(hObject, eventdata, handles)
% hObject    handle to grado3 (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 grado3 as text
%        str2double(get(hObject,'String')) returns contents of grado3 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function grado3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to grado3 (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 min3_Callback(hObject, eventdata, handles)
% hObject    handle to min3 (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 min3 as text
%        str2double(get(hObject,'String')) returns contents of min3 as a
double


% --- Executes during object creation, after setting all properties.
function min3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to min3 (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 seg3_Callback(hObject, eventdata, handles)
% hObject    handle to seg3 (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 seg3 as text
%        str2double(get(hObject,'String')) returns contents of seg3 as a
double


% --- Executes during object creation, after setting all properties.
function seg3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to seg3 (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 grado4_Callback(hObject, eventdata, handles)
% hObject    handle to grado4 (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 grado4 as text
%        str2double(get(hObject,'String')) returns contents of grado4 as a
double


% --- Executes during object creation, after setting all properties.
function grado4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to grado4 (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 min4_Callback(hObject, eventdata, handles)
% hObject    handle to min4 (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 min4 as text
%        str2double(get(hObject,'String')) returns contents of min4 as a
double


% --- Executes during object creation, after setting all properties.
function min4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to min4 (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 seg4_Callback(hObject, eventdata, handles)
% hObject    handle to seg4 (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 seg4 as text
%        str2double(get(hObject,'String')) returns contents of seg4 as a
double


% --- Executes during object creation, after setting all properties.
function seg4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to seg4 (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 dis_Callback(hObject, eventdata, handles)
% hObject    handle to dis (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 dis as text
%        str2double(get(hObject,'String')) returns contents of dis as a
double


% --- Executes during object creation, after setting all properties.
function dis_CreateFcn(hObject, eventdata, handles)
% hObject    handle to dis (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 NS.
function NS_Callback(hObject, eventdata, handles)
% hObject    handle to NS (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 NS contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from NS


% --- Executes during object creation, after setting all properties.
function NS_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NS (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 selection change in NS2.
function NS2_Callback(hObject, eventdata, handles)
% hObject    handle to NS2 (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 NS2 contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from NS2


% --- Executes during object creation, after setting all properties.
function NS2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NS2 (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 selection change in EO.
function EO_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to EO (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 EO contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from EO


% --- Executes during object creation, after setting all properties.
function EO_CreateFcn(hObject, eventdata, handles)
% hObject    handle to EO (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 selection change in EO2.
function EO2_Callback(hObject, eventdata, handles)
% hObject    handle to EO2 (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 EO2 contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from EO2


% --- Executes during object creation, after setting all properties.
function EO2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to EO2 (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 Sur1.
function Sur1_Callback(hObject, eventdata, handles)
% hObject    handle to Sur1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of Sur1


% --- Executes on button press in oeste1.
function oeste1_Callback(hObject, eventdata, handles)
% hObject    handle to oeste1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of oeste1


% --- Executes on button press in sur2.
function sur2_Callback(hObject, eventdata, handles)
% hObject    handle to sur2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of sur2


% --- Executes on button press in oeste2.
function oeste2_Callback(hObject, eventdata, handles)
% hObject    handle to oeste2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of oeste2


% --- Executes on button press in btnSiguiente.
function btnSiguiente_Callback(hObject, eventdata, handles)
% hObject    handle to btnSiguiente (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
simulador2();


% --- Executes on button press in info1.
function info1_Callback(hObject, eventdata, handles)
% hObject    handle to info1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo1,'visible','on')
set (handles.tinfo2,'visible','off')
n=3;
pause(n)
set (handles.tinfo1,'visible','off')


% --- Executes on button press in info2.
function info2_Callback(hObject, eventdata, handles)
% hObject    handle to info2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo2,'visible','on')
set (handles.tinfo1,'visible','off')
n=3;
pause(n)
set (handles.tinfo2,'visible','off')
```

❖ Código completo función *simulador2*

Contiene el código de la segunda pantalla.

```
function varargout = simulador2(varargin)
% SIMULADOR2 MATLAB code for simulador2.fig
%      SIMULADOR2, by itself, creates a new SIMULADOR2 or raises the
existing
%      singleton*.
%
%      H = SIMULADOR2 returns the handle to a new SIMULADOR2 or the handle
to
%      the existing singleton*.
%
%      SIMULADOR2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SIMULADOR2.M with the given input
arguments.
%
%      SIMULADOR2('Property','Value',...) creates a new SIMULADOR2 or
raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before simulador2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to simulador2_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 simulador2

% Last Modified by GUIDE v2.5 03-Sep-2019 15:48:03

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @simulador2_OpeningFcn, ...
                   'gui_OutputFcn',  @simulador2_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 simulador2 is made visible.
function simulador2_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 simulador2 (see VARARGIN)
% Choose default command line output for simulador2

%LOGO UISRAEL
logo=imread('logoU2.jpg');
axes(handles.axes4);
imshow(logo);

%ICONO INFO3
info=imread('info.jpg');
set(handles.info3,'CData',info)

%ICONO INFO5
info=imread('info.jpg');
set(handles.info5,'CData',info)

%ICONO INFO6
info=imread('info.jpg');
set(handles.info6,'CData',info)

%ICONO INFO7
info=imread('info.jpg');
set(handles.info7,'CData',info)

%ICONO INFO8
info=imread('info.jpg');
set(handles.info8,'CData',info)

%ICONO INFO9
info=imread('info.jpg');
set(handles.info9,'CData',info)

%ICONO INFO10
info=imread('info.jpg');
set(handles.info10,'CData',info)
```

```
%ICONO INFO11
info=imread('info.jpg');
set(handles.info11,'CData',info)

%ICONO INFO12
info=imread('info.jpg');
set(handles.info12,'CData',info)

%ICONO INFO13
info=imread('info.jpg');
set(handles.info13,'CData',info)

%ICONO INFO14
info=imread('info.jpg');
set(handles.info14,'CData',info)

%ICONO INFO15
info=imread('info.jpg');
set(handles.info15,'CData',info)

clear val val2 val3 val4 val5 val6 hb hm

set(handles.potdb,'string','');
set(handles.linea,'string','3.2');
set(handles.linea2,'string','3.2');
set(handles.sensi,'string','');
set(handles.fmin,'string','');
set(handles.fmax,'string','');

set(handles.edit31,'string','');
set(handles.edit32,'string','');
set(handles.ga1,'string','');
set(handles.ga2,'string','');
set(handles.atenu2,'string','');
set(handles.res,'string','');
set(handles.mar,'string','');
set(handles.edit33,'string','');
set(handles.edit34,'string','');
set(handles.edit31,'string','');
set(handles.edit32,'string','');


handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes simulador2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = simulador2_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 atenu2_Callback(hObject, eventdata, handles)
% hObject    handle to atenu2 (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 atenu2 as text
%        str2double(get(hObject,'String')) returns contents of atenu2 as a
double


% --- Executes during object creation, after setting all properties.
function atenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to atenu2 (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 potdb_Callback(hObject, eventdata, handles)
% hObject    handle to potdb (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 potdb as text
%        str2double(get(hObject,'String')) returns contents of potdb as a
double


% --- Executes during object creation, after setting all properties.
function potdb_CreateFcn(hObject, eventdata, handles)
% hObject    handle to potdb (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 linea_Callback(hObject, eventdata, handles)
% hObject    handle to linea (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 linea as text
%        str2double(get(hObject,'String')) returns contents of linea as a
double


% --- Executes during object creation, after setting all properties.
function linea_CreateFcn(hObject, eventdata, handles)
% hObject    handle to linea (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 ga1_Callback(hObject, eventdata, handles)
% hObject    handle to ga1 (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 ga1 as text
%        str2double(get(hObject,'String')) returns contents of ga1 as a
double


% --- Executes during object creation, after setting all properties.
function ga1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ga1 (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 ga2_Callback(hObject, eventdata, handles)
% hObject    handle to ga2 (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 ga2 as text
%        str2double(get(hObject,'String')) returns contents of ga2 as a
double


% --- Executes during object creation, after setting all properties.
function ga2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ga2 (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 edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (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 edit21 as text
%        str2double(get(hObject,'String')) returns contents of edit21 as a
double


% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (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


% --- EJECUTA AL PRESIONAR EL CALCULAR.
function calcular_Callback(hObject, eventdata, handles)

set(handles.btnSiguiente,'visible','on')%BOTON SIGUIENTE VISIBLE

potedb=str2num(get(handles.potdb,'string'));%POTENCIA DE TRANSMISIÓN dBm
g1=str2num(get(handles.ga1,'string'));%GANANCIA DE ANTENA DE RECEPCIÓN dBi
g2=str2num(get(handles.ga2,'string'));%GANANCIA ANTENA DE TRANSMISIÓN dBi
f1=str2num(get(handles.fmin,'string'));%FRECUENCIA MÍNIMA MHz
f2=str2num(get(handles.fmax,'string'));%FRECUENCIA MÁXIMA MHz
lin=str2num(get(handles.linea,'string'));%PERDIDA DE LA LÍNEA
lin2=str2num(get(handles.linea2,'string'));%PERDIDA DE LA LÍNEA
sensa=str2num(get(handles.sensi,'string'));%SENSIBILIDAD dBi
hb=str2num(get(handles.edit31,'string'));%ALTURA BASE
hm=str2num(get(handles.edit32,'string'));%ALTURA RECEPTOR

global  frecuencia1000 distancias64 hb hm valA valB
frecuencia1000=(f2+f1)/2;%FRECUENCIA MEDIA

%ATENUACIÓN POR LÍNEA DE TRANSMISIÓN
calcA=(20+hb)*(4.8/100); % Atenuación en cables del alimentador a la antena
de transmision, por defecto 20m
lin=lin+calcA;%(dB)
atenu1=num2str(lin);
set(handles.atenu1,'string',atenu1);

%ATENUACIÓN POR LÍNEA DE RECEPCIÓN
calcA2=(20+hm)*(4.8/100); % Atenuación en cables del alimentador a la
antena de transmision, por defecto 20m
lin2=lin2+calcA2;%(dB)
atenu2=num2str(lin2);
```

```
set(handles.atenu2,'string',atenu2);

valorTotalAt=lin+lin2

[lb potrelb potrepel Mdesv
Pel]=resultados(potedb,g1,g2,distancias64,frecuencia1000,valorTotalAt,sensa
,hm,hb,valA,valB)%LLAMA A LA FUNCION RESULTADOS

%PÉRDIDAS EN ESPACIO LIBRE
valp=num2str(Pel);
set(handles.Pel,'string',valp);

%MARGEN DE DESVANECIMIENTO
val6=num2str(Mdesv);
set(handles.edit34,'string',val6);



%POTENCIA DE TRANSMISIÓN mWatt
% potencia=10^(potedb/10);
% val2=num2str(potencia);
% set(handles.atenu2,'string',val2);

% %POTENCIA DE RECEPCIÓN CON CONECTORES
% val3=num2str(salida);
% set(handles.salid,'string',val3);

%PÉRDIDA OKUMURA-HATA
val4=num2str(lb);
set(handles.mar,'string',val4);

%POTENCIA DE RECEPCIÓN ESPACIO LIBRE
val5=num2str(potrepel);
set(handles.edit33,'string',val5);

%POTENCIA DE RECEPCIÓN OKUMURA - HATA
val7=num2str(potrelb);
set(handles.res,'string',val7);



%MOSTRAR IMAGEN DE YAGUI
antena=imread('yagui.png');
axes(handles.axes3);
imshow(antena);

% simulacion3d();%LLAMA SIMULADOR 3D



function res_Callback(hObject, eventdata, handles)
% hObject    handle to res (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 res as text
%        str2double(get(hObject,'String')) returns contents of res as a
double


% --- Executes during object creation, after setting all properties.
function res_CreateFcn(hObject, eventdata, handles)
% hObject    handle to res (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 fmin_Callback(hObject, eventdata, handles)
% hObject    handle to fmin (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 fmin as text
%        str2double(get(hObject,'String')) returns contents of fmin as a
double


% --- Executes during object creation, after setting all properties.
function fmin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fmin (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 fmax_Callback(hObject, eventdata, handles)
% hObject    handle to fmax (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 fmax as text
```

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


% --- Executes during object creation, after setting all properties.
function fmax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fmax (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 salid_Callback(hObject, eventdata, handles)
% hObject    handle to salid (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 salid as text
%        str2double(get(hObject,'String')) returns contents of salid as a
double


% --- Executes during object creation, after setting all properties.
function salid_CreateFcn(hObject, eventdata, handles)
% hObject    handle to salid (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 mar_Callback(hObject, eventdata, handles)
% hObject    handle to mar (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 mar as text
%        str2double(get(hObject,'String')) returns contents of mar as a
double
```

```
% --- Executes during object creation, after setting all properties.
function mar_CreateFcn(hObject, eventdata, handles)
% hObject    handle to mar (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 sensi_Callback(hObject, eventdata, handles)
% hObject    handle to sensi (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 sensi as text
%        str2double(get(hObject,'String')) returns contents of sensi as a
double


% --- Executes during object creation, after setting all properties.
function sensi_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sensi (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 edit31_Callback(hObject, eventdata, handles)
% hObject    handle to edit31 (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 edit31 as text
%        str2double(get(hObject,'String')) returns contents of edit31 as a
double


% --- Executes during object creation, after setting all properties.
function edit31_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit31 (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 edit32_Callback(hObject, eventdata, handles)
% hObject    handle to edit32 (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 edit32 as text
%        str2double(get(hObject,'String')) returns contents of edit32 as a
double


% --- Executes during object creation, after setting all properties.
function edit32_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit32 (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 edit33_Callback(hObject, eventdata, handles)
% hObject    handle to edit33 (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 edit33 as text
%        str2double(get(hObject,'String')) returns contents of edit33 as a
double


% --- Executes during object creation, after setting all properties.
function edit33_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit33 (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 edit34_Callback(hObject, eventdata, handles)
% hObject    handle to edit34 (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 edit34 as text
%        str2double(get(hObject,'String')) returns contents of edit34 as a
double


% --- Executes during object creation, after setting all properties.
function edit34_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit34 (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 btnSiguiente.
function btnSiguiente_Callback(hObject, eventdata, handles)
% hObject    handle to btnSiguiente (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
simulacion3d();


% --- Executes on button press in btnSiguiente.
function btnSiguiente_Callback(hObject, eventdata, handles)
% hObject    handle to btnSiguiente (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 info3.
function info3_Callback(hObject, eventdata, handles)
% hObject    handle to info3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo3,'visible','on')
set (handles.tinfo5,'visible','off')
set (handles.tinfo6,'visible','off')
```

```
set (handles.tinfo7,'visible','off')
set (handles.tinfo8,'visible','off')
set (handles.tinfo9,'visible','off')
set (handles.tinfo10,'visible','off')
set (handles.tinfo11,'visible','off')
set (handles.tinfo12,'visible','off')
set (handles.tinfo13,'visible','off')
set (handles.tinfo14,'visible','off')
set (handles.tinfo15,'visible','off')
n=3;
pause(n)
set (handles.tinfo3,'visible','off')


% --- Executes on button press in info5.
function info5_Callback(hObject, eventdata, handles)
% hObject    handle to info14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo5,'visible','on')
set (handles.tinfo3,'visible','off')
set (handles.tinfo6,'visible','off')
set (handles.tinfo7,'visible','off')
set (handles.tinfo8,'visible','off')
set (handles.tinfo9,'visible','off')
set (handles.tinfo10,'visible','off')
set (handles.tinfo11,'visible','off')
set (handles.tinfo12,'visible','off')
set (handles.tinfo13,'visible','off')
set (handles.tinfo14,'visible','off')
set (handles.tinfo15,'visible','off')
n=3;
pause(n)
set (handles.tinfo5,'visible','off')

% --- Executes on button press in info6.
function info6_Callback(hObject, eventdata, handles)
% hObject    handle to info11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo6,'visible','on')
set (handles.tinfo3,'visible','off')
set (handles.tinfo5,'visible','off')
set (handles.tinfo7,'visible','off')
set (handles.tinfo8,'visible','off')
set (handles.tinfo9,'visible','off')
set (handles.tinfo10,'visible','off')
set (handles.tinfo11,'visible','off')
set (handles.tinfo12,'visible','off')
set (handles.tinfo13,'visible','off')
set (handles.tinfo14,'visible','off')
set (handles.tinfo15,'visible','off')
n=3;
pause(n)
set (handles.tinfo6,'visible','off')
```

```
% --- Executes on button press in info7.
function info7_Callback(hObject, eventdata, handles)
% hObject    handle to info11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo7,'visible','on')
set (handles.tinfo3,'visible','off')
set (handles.tinfo5,'visible','off')
set (handles.tinfo6,'visible','off')
set (handles.tinfo8,'visible','off')
set (handles.tinfo9,'visible','off')
set (handles.tinfo10,'visible','off')
set (handles.tinfo11,'visible','off')
set (handles.tinfo12,'visible','off')
set (handles.tinfo13,'visible','off')
set (handles.tinfo14,'visible','off')
set (handles.tinfo15,'visible','off')
n=3;
pause(n)
set (handles.tinfo7,'visible','off')


% --- Executes on button press in info8.
function info8_Callback(hObject, eventdata, handles)
% hObject    handle to info14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo8,'visible','on')
set (handles.tinfo3,'visible','off')
set (handles.tinfo5,'visible','off')
set (handles.tinfo6,'visible','off')
set (handles.tinfo7,'visible','off')
set (handles.tinfo9,'visible','off')
set (handles.tinfo10,'visible','off')
set (handles.tinfo11,'visible','off')
set (handles.tinfo12,'visible','off')
set (handles.tinfo13,'visible','off')
set (handles.tinfo14,'visible','off')
set (handles.tinfo15,'visible','off')
n=3;
pause(n)
set (handles.tinfo8,'visible','off')


% --- Executes on button press in info9.
function info9_Callback(hObject, eventdata, handles)
% hObject    handle to info11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo9,'visible','on')
set (handles.tinfo3,'visible','off')
set (handles.tinfo5,'visible','off')
set (handles.tinfo6,'visible','off')
set (handles.tinfo7,'visible','off')
set (handles.tinfo8,'visible','off')
```

```
set (handles.tinfo10,'visible','off')
set (handles.tinfo11,'visible','off')
set (handles.tinfo12,'visible','off')
set (handles.tinfo13,'visible','off')
set (handles.tinfo14,'visible','off')
set (handles.tinfo15,'visible','off')
n=3;
pause(n)
set (handles.tinfo9,'visible','off')


% --- Executes on button press in info10.
function info10_Callback(hObject, eventdata, handles)
% hObject    handle to info14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo10,'visible','on')
set (handles.tinfo3,'visible','off')
set (handles.tinfo5,'visible','off')
set (handles.tinfo6,'visible','off')
set (handles.tinfo7,'visible','off')
set (handles.tinfo8,'visible','off')
set (handles.tinfo9,'visible','off')
set (handles.tinfo11,'visible','off')
set (handles.tinfo12,'visible','off')
set (handles.tinfo13,'visible','off')
set (handles.tinfo14,'visible','off')
set (handles.tinfo15,'visible','off')
n=3;
pause(n)
set (handles.tinfo10,'visible','off')


% --- Executes on button press in info11.
function info11_Callback(hObject, eventdata, handles)
% hObject    handle to info15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo11,'visible','on')
set (handles.tinfo3,'visible','off')
set (handles.tinfo5,'visible','off')
set (handles.tinfo6,'visible','off')
set (handles.tinfo7,'visible','off')
set (handles.tinfo8,'visible','off')
set (handles.tinfo9,'visible','off')
set (handles.tinfo10,'visible','off')
set (handles.tinfo12,'visible','off')
set (handles.tinfo13,'visible','off')
set (handles.tinfo14,'visible','off')
set (handles.tinfo15,'visible','off')
n=3;
pause(n)
set (handles.tinfo11,'visible','off')


% --- Executes on button press in info12.
```

```
function info12_Callback(hObject, eventdata, handles)
% hObject    handle to info14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo12,'visible','on')
set (handles.tinfo3,'visible','off')
set (handles.tinfo5,'visible','off')
set (handles.tinfo6,'visible','off')
set (handles.tinfo7,'visible','off')
set (handles.tinfo8,'visible','off')
set (handles.tinfo9,'visible','off')
set (handles.tinfo10,'visible','off')
set (handles.tinfo11,'visible','off')
set (handles.tinfo13,'visible','off')
set (handles.tinfo14,'visible','off')
set (handles.tinfo15,'visible','off')
n=3;
pause(n)
set (handles.tinfo12,'visible','off')


% --- Executes on button press in info13.
function info13_Callback(hObject, eventdata, handles)
% hObject    handle to info15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo13,'visible','on')
set (handles.tinfo3,'visible','off')
set (handles.tinfo5,'visible','off')
set (handles.tinfo6,'visible','off')
set (handles.tinfo7,'visible','off')
set (handles.tinfo8,'visible','off')
set (handles.tinfo9,'visible','off')
set (handles.tinfo10,'visible','off')
set (handles.tinfo11,'visible','off')
set (handles.tinfo12,'visible','off')
set (handles.tinfo14,'visible','off')
set (handles.tinfo15,'visible','off')
n=3;
pause(n)
set (handles.tinfo13,'visible','off')


% --- Executes during object creation, after setting all properties.
function tinfo5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to tinfo5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



function atenu1_Callback(hObject, eventdata, handles)
% hObject    handle to atenu1 (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 atenu1 as text
%        str2double(get(hObject,'String')) returns contents of atenu1 as a
double


% --- Executes during object creation, after setting all properties.
function atenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to atenu1 (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 Pel_Callback(hObject, eventdata, handles)
% hObject    handle to Pel (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 Pel as text
%        str2double(get(hObject,'String')) returns contents of Pel as a
double


% --- Executes during object creation, after setting all properties.
function Pel_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Pel (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 paraA.
function paraA_Callback(hObject, eventdata, handles)
global valA
valor=get(handles.paraA,'value');
switch valor
    case 2
        valA=4;
    case 3
        valA=1;
```

```
    case 4
        valA=0.25;
    otherwise
        valA=1;
end
% hObject    handle to paraA (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 paraA contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from paraA


% --- Executes during object creation, after setting all properties.
function paraA_CreateFcn(hObject, eventdata, handles)
% hObject    handle to paraA (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 selection change in paraB.
function paraB_Callback(hObject, eventdata, handles)
global valB
valor=get(handles.paraB,'value');
switch valor
    case 2
        valB=1;
    case 3
        valB=0.5;
    case 4
        valB=0.25;
    case 5
        valB=0.125;
    otherwise
        valB=1;
end
% hObject    handle to paraB (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 paraB contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from paraB


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

```
% hObject    handle to paraB (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



function linea2_Callback(hObject, eventdata, handles)
% hObject    handle to linea2 (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 linea2 as text
%        str2double(get(hObject,'String')) returns contents of linea2 as a
double


% --- Executes during object creation, after setting all properties.
function linea2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to linea2 (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 info14.
function info14_Callback(hObject, eventdata, handles)
% hObject    handle to info6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo14,'visible','on')
set (handles.tinfo3,'visible','off')
set (handles.tinfo5,'visible','off')
set (handles.tinfo6,'visible','off')
set (handles.tinfo7,'visible','off')
set (handles.tinfo8,'visible','off')
set (handles.tinfo9,'visible','off')
set (handles.tinfo10,'visible','off')
set (handles.tinfo11,'visible','off')
set (handles.tinfo12,'visible','off')
set (handles.tinfo13,'visible','off')
set (handles.tinfo15,'visible','off')
n=3;
```

```
pause(n)
set (handles.tinfo14,'visible','off')



function edit38_Callback(hObject, eventdata, handles)
% hObject    handle to edit38 (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 edit38 as text
%        str2double(get(hObject,'String')) returns contents of edit38 as a
double


% --- Executes during object creation, after setting all properties.
function edit38_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit38 (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 info15.
function info15_Callback(hObject, eventdata, handles)
% hObject    handle to info4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set (handles.tinfo15,'visible','on')
set (handles.tinfo3,'visible','off')
set (handles.tinfo5,'visible','off')
set (handles.tinfo6,'visible','off')
set (handles.tinfo7,'visible','off')
set (handles.tinfo8,'visible','off')
set (handles.tinfo9,'visible','off')
set (handles.tinfo10,'visible','off')
set (handles.tinfo11,'visible','off')
set (handles.tinfo12,'visible','off')
set (handles.tinfo13,'visible','off')
set (handles.tinfo14,'visible','off')
n=3;
pause(n)
set (handles.tinfo15,'visible','off')
```

❖ Código completo función *simulacion3d*

Contiene el código de la tercera pantalla.

```
function varargout = simulacion3d(varargin)
% SIMULACION3D MATLAB code for simulacion3d.fig
%      SIMULACION3D, by itself, creates a new SIMULACION3D or raises the
existing
%      singleton*.
%
%      H = SIMULACION3D returns the handle to a new SIMULACION3D or the
handle to
%      the existing singleton*.
%
%      SIMULACION3D('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in SIMULACION3D.M with the given input
arguments.
%
%      SIMULACION3D('Property','Value',...) creates a new SIMULACION3D or
raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before simulacion3d_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to simulacion3d_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 simulacion3d

% Last Modified by GUIDE v2.5 20-Aug-2019 20:08:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @simulacion3d_OpeningFcn, ...
                   'gui_OutputFcn',  @simulacion3d_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 simulacion3d is made visible.
function simulacion3d_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 simulacion3d (see VARARGIN)

%LOGO UISRAEL
logo=imread('logoU2.jpg');
axes(handles.axes3);
imshow(logo);

clear val val2 val3 val4 val5 val6

%set(handles.d1,'string','');
%set(handles.hs1,'string','');
set(handles.F1,'string','');
set(handles.F2,'string','');
set(handles.F3,'string','');
set(handles.c1,'string','');
set(handles.edit16,'string','');
set(handles.edit17,'string','');


% Choose default command line output for simulacion3d
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes simulacion3d wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = simulacion3d_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 edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (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 edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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 edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (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 edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (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 edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (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 edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (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 edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (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 edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (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 edit5 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (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 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 hs1_Callback(hObject, eventdata, handles)
% hObject    handle to hs1 (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 hs1 as text
%        str2double(get(hObject,'String')) returns contents of hs1 as a
double


% --- Executes during object creation, after setting all properties.
function hs1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hs1 (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 hrec_Callback(hObject, eventdata, handles)
% hObject    handle to hrec (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 hrec as text
%        str2double(get(hObject,'String')) returns contents of hrec as a
double


% --- Executes during object creation, after setting all properties.
function hrec_CreateFcn(hObject, eventdata, handles)
% hObject    handle to hrec (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 htra_Callback(hObject, eventdata, handles)
% hObject    handle to htra (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 htra as text
%        str2double(get(hObject,'String')) returns contents of htra as a
double


% --- Executes during object creation, after setting all properties.
function htra_CreateFcn(hObject, eventdata, handles)
% hObject    handle to htra (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 F1_Callback(hObject, eventdata, handles)
% hObject    handle to F1 (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 F1 as text
%        str2double(get(hObject,'String')) returns contents of F1 as a
double


% --- Executes during object creation, after setting all properties.
function F1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to F1 (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 F2_Callback(hObject, eventdata, handles)
% hObject    handle to F2 (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 F2 as text
%        str2double(get(hObject,'String')) returns contents of F2 as a
double


% --- Executes during object creation, after setting all properties.
function F2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to F2 (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 F3_Callback(hObject, eventdata, handles)
% hObject    handle to F3 (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 F3 as text
%        str2double(get(hObject,'String')) returns contents of F3 as a
double


% --- Executes during object creation, after setting all properties.
function F3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to F3 (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 pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
global distancias64 frecuencia1000 hm hb
%hminima=str2num(get(handles.hrec,'string'));
%hmaxima=str2num(get(handles.htra,'string'));

%POSICIÓN DEL OBSTÁCULO
[PS1, posx12, posx22, posy1, posy2, IVpmax, ValorPMax, x1, y3]=test1();

d1=IVpmax-posx12;%DISTANCIA DE P1 AL OBSTÁCULO
hs=ValorPMax;%ALTURA DEL OBSTÁCULO
d2=distancias64-d1;% DISTANCIA ENTRE PUNTOS - DISNTACIA DEL OBSTÁCULO

%GRAFICAR FRESNEL
[fre1,fre2,fre3,gf3,gf2,gf,x,y,c]=graficarfresnel(frecuencia1000,distancias
64,d1,d2,hb,hm,hs,posy1,posy2);

%FRECUENCIAS DE FRESNEL
val1=num2str(fre1);
set(handles.F1,'string',val1);
val2=num2str(fre2);
set(handles.F2,'string',val2);
val3=num2str(fre3);
set(handles.F3,'string',val3);

%C FRESNEL
val4=num2str(c);
set(handles.c1,'string',val4);

%ALTURA BASE Y ALTURA RECEPTOR
val5=num2str(hb);
set(handles.edit16,'string',val5);
val6=num2str(hm);
set(handles.edit17,'string',val6);

%TRANSMISIÓN CORRECTA
if c>=0.6*fre1
    set(handles.casos,'value',1);
else
    set(handles.casos,'value',0);
end


%GRÁFA FRESNEL Y MAPA
axes(handles.axes1);

plot(x1,y3,'k')%IMPRESION RELIEVE

hold on;
%imagesc(x,gf+y,fondo);
% MaxEntrePuntos=max()
plot(handles.axes1,x+posx12,gf+y,'b');%MITAD SUPERIOR LOBULO
```

```
% plot(handles.axes1,x,gf2+y);
% plot(handles.axes1,x,gf3+y);
plot(handles.axes1,x+posx12,y-gf);%MITAD INFERIOR LOBULO
% plot(handles.axes1,x,y-gf2);
% plot(handles.axes1,x,y-gf3);
plot(handles.axes1,x+posx12,y,'r');%LINEA DIRECTA LOBULO

P1=[IVpmax 0];P2=[IVpmax hs];
plot([P1(1) P2(1)],[P1(2) P2(2)],'g','linewidth',12);%OBSTACULO

plot(handles.axes1,x+posx12,(0.6*gf+y),'+y:');%AMARILLO SUPERIOR LOBULO
plot(handles.axes1,x+posx12,(+y-0.6*gf),'+y:');%AMARILLO INFERIOR LOBULO

legend(handles.axes1,'Relieve','f1','-f1','Linea de vista','
Obstáculo','0.6*F1')%LEYENDA
xlabel(handles.axes1,'Distancia(Km)');%TITULO X
ylabel(handles.axes1,'Altura(m)');% TITULO Y

zoom on;
%pan on;

hold off;

grid on;


function c1_Callback(hObject, eventdata, handles)
% hObject    handle to c1 (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 c1 as text
%        str2double(get(hObject,'String')) returns contents of c1 as a
double


% --- Executes during object creation, after setting all properties.
function c1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to c1 (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 casos.
function casos_Callback(hObject, eventdata, handles)
% hObject    handle to casos (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of casos



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
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Simulador;
```

❖ Código completo función *resultados*

Contiene el código de la función resultados.

```
function [salida2 salida potredb potredb1
salida3]=resultados(potedb,g1,g2,d,f,lin,sensa,hm,hb)

%PROGRAMA CALCULOS PANTALLA SIMULADOR 2

potredb1=potedb+g1+g2-32.44-20*log10(d)-20*log10(f);%ESPACIO LIBRE

a=(1.1*log10(f)-0.7)*hm-(1.56*log10(f)-0.8);%CONSTANTE
potredb=potedb+g1+g2-69.55+26.16*log10(f)-13.82*log10(hb)-a+(44.9-
6.55*log10(hb))*log10(d)%MODELO OKUMURA HATA, POTENCIA DE RECEPCION

lb=69.55+26.16*log10(f)-13.82*log10(hb)-a+(44.9-
6.55*log10(hb))*log10(d);%PERDIDA DE ESPACIO LIBRE
salida=potredb1-lin;%PERDIDA CON CONECTORES

salida3=salida+sensa;%MARGEN DE DESVANECIMIENTO
salida2=lb;%PERDIDA DE ESPACIO LIBRE
```

❖ Código completo función *convertir*

Contiene el código de la función convertir.

```
function decimal=convertir(grado,min,seg)
%TRANSFORMA GRAD,MIN Y SEG A DECIMAL
segundo=seg/3600;
minuto=min/60;
decimal=(grado+minuto+segundo)*pi/180;
```

❖ Código completo función *distancia*

Contiene el código de la función distancia.

```
function d=distancia(longitud1,latitud1,longitud2,latitud2)
%CALCULA LA DISTANCIA CON LA FÓRMULA DE HAVERSINE
r=6370;
lat=latitud2-latitud1;
lon=longitud2-longitud1;
a=sind(lat/2)^2+cosd(latitud1)*cosd(latitud2)*(sind(lon/2))^2;
d=2*r*asind(sqrt(a));
```

❖ Código completo función *graficarfresnel*

Contiene el código de la función graficar Zona de Fresnel.

```
function
[fre1,fre2,fre3,gf3,gf2,gf,x,y,c]=graficarfresnel(f,d,d1,d2,hm,hb,hs,posy1,
posy2)

%f ES LA FRECUENCIA MEDIA
%d DISTANCIA ENTRE P1 Y P2
%d1 DISTANCIA DE P1 AL OBSTÁCULO
%d2 DISTANCIA DEL OBSTÁCULO A P2
%hm ALTURA DEL RECEPTOR
%hb ALTURA DE LA BASE
%hs ALTURA DEL OBSTÁCULO

%CÁLCULO VALORES DE FRESNEL
fre1=548*sqrt((d1*d2)/(f*d));%ENÉSIMA ZONA DE FRESNEL
fre2=sqrt(2)*fre1;
fre3=sqrt(2)*fre2;

%DISTANCIA DE PUNTO ALTO DE OBSTÁCULO A LINEA DIRECTA
c=d1*((hb+posy2)-(hm+posy1))/d+(hm+posy1)-hs;

%CÁLCULO PARA GRÁFICA DE FRESNEL
d=floor(d);
x=0:0.1:d;
gf=548*sqrt((x.*(d-x))/(f*d));
gf2=sqrt(2)*gf;
gf3=sqrt(2)*gf2;
pendiente=((hb+posy2)-(hm+posy1))/(d);
y=pendiente.*(x)+(hm+posy1);
```

❖ Código completo función *test1*

Contiene el código de la función interpolar los puntos en el mapa.

```
function [PS1, posx12, posx22, posy1, posy2, IVpmax, ValorPMax, x1,
y3]=test1()
global lati1 long1 lati2 long2 distancias64
X=readhgt();
lat=X.lat;
lon=X.lon;
SRTM=X.z;
[m n]=size(SRTM);

%% Comparacion de datos
latIn1=lati1;
lonIn1=long1;
latIn2=lati2;
lonIn2=long2;
%redondeo a 4 cifras decimales de datos obtenidos
latIn1=round(latIn1,4);
lonIn1=round(lonIn1,4);
latIn2=round(latIn2,4);
lonIn2=round(lonIn2,4);

%Evaluacion longitud1
vectlon1=lonIn1*ones(1,length(lon));%creo matrices de 1
resta1=lon-vectlon1;
[minVectlon1, Ilonmin1]=min(abs(resta1));%Ilonmin1 es la posicion

%Evaluacion latitud1
vectlat1=latIn1*ones(length(lat),1);%creo matrices de 1
resta2=lat-vectlat1;
[minVectlat1, Ilatmin1]=min(abs(resta2));%Ilatmin1 es la posicion

%Evaluacion longitud2
vectlon2=lonIn2*ones(1,length(lon));%creo matrices de 1
resta3=lon-vectlon2;
[minVectlon2, Ilonmin2]=min(abs(resta3));%Ilonmin1 es la posicion

%Evaluacion latitud2
vectlat2=latIn2*ones(length(lat),1);%creo matrices de 1
resta4=lat-vectlat2;
[minVectlat2, Ilatmin2]=min(abs(resta4));%Ilatmin1 es la posicion
disp('posiciones')
lim11=Ilonmin1; %Longitud1
posx1=Ilatmin1 %Latitud1
lim12=Ilonmin2;%Longitud2
posx2=Ilatmin2; %Latitud2
%% Grafico

[x,y]=meshgrid(linspace(lon(1,1),lon(1,length(lon)),n),linspace(lat(1,1),la
t(length(lat),1),m));
h=interp2(x,y,single(SRTM),lon,lat,'spline');

x1=linspace(0,120,n);
%Valores en posiciones en el dominio de 0 a 120km
posx12=x1(posx1);
posx22=x1(posx2);
%determinación del punto2
```

```
d=floor(distancias64);
distancia1=posx12+d;
%Encuentro diastnacia 1 en vector de 0 a 120 x1
vectAux=distancia1*ones(1,length(x1));%creo matrices de 1
restaAux=x1-vectAux;
[minPos2, IPos2]=min(abs(restaAux));%Ilatmin1 es la posicion
ValorP2x=x1(IPos2)
posx22=ValorP2x;

%MÁXIMO
matrizMaxi=max(h,[],2);
[M Indice]=max(matrizMaxi(:))

%FILTRO
[h1,h2]=max(h,[],2);
g=gausswin(30);
g=g/sum(g);
y3=conv(h1,g,'same');
%plot(x1,y3,'r')
% hold on
% plot(max(h,[],2),y3)
% hold off;
PS1=matrizMaxi(2);
posy1=matrizMaxi(posx1);
posy2=matrizMaxi(IPos2);
% auxMax=max(matrizMaxi(posx1:IPos2))
[ValorPMax, IVpmax]=max(matrizMaxi(posx1:IPos2));
IVpmax=length(matrizMaxi(1:posx1))+IVpmax;
IVpmax=x1(IVpmax);
% ValorP2y=matrizMaxi(IPos2)
end
```

❖ Código completo función *readhgt*

Contiene el código de la función que grafica el mata SRTM.

```
function varargout = readhgt(varargin)
%READHGT Import/download NASA SRTM data files (.HGT).
%   READHGT(AREA) where AREA is a 4-element vector [LAT1,LAT2,LON1,LON2]
%   downloads the SRTM data and plots a map corresponding to the geographic
%   area defined by latitude and longitude limits (in decimal degrees). If
%   the needed SRTM .hgt files are not found in the current directory (or
%   in the path), they are downloaded from the USGS data server (needs an
%   Internet connection and a companion file "readhgt_srtm_index.txt"). For
%   better plot results, it is recommended to install DEM personal function
%   available at author's Matlab page.
%
%   READHGT(LAT,LON) reads or downloads the SRTM tiles corresponding to LAT
%   and LON (in decimal degrees) coordinates (lower-left corner).
%
%   LAT and/or LON can be vectors: in that case, tiles corresponding to all
%   possible combinations of LAT and LON values will be downloaded, and
%   optional output structure X will have as much elements as tiles.
%
%   READHGT(FILENAME) reads HGT data file FILENAME, must be in the form
%   "[N|S]yy[E|W]xxx.hgt[.zip]", as downloaded from SRTM data servers.
%
%   X=READHGT(...) returns a structure X containing:
%       lat: coordinate vector of latitudes (decimal degree)
%       lon: coordinate vector of longitudes (decimal degree)
%         z: matrix of elevations (meters, INT16 class)
%       hgt: downloaded filename(s)
%
%   X=READHGT(...,'plot') also plots the tile(s).
%
%
%   --- Additionnal options ---
%
%   'tiles'
```

```
%       Imports and plots individual tiles instead of merging them (default
%       behavior if adjoining values of LAT and LON).
%
%   'interp'
%       Linearly interpolates missing data.
%
%   'decim',N
%       Decimates the tiles at 1/N times of the original sampling. Plot
%       is automatically decimated if necessary. Use N=1 to force full
%       resolution (might induces memory issue for large areas).
%
%   'crop'
%       crops the resulting map around existing land (reduces any sea or
%       novalue areas at the borders).
%
%   'crop',[LAT1,lAT2,LON1,LON2]
%       Former syntax that crops the map using latitude/longitude limits.
%       Prefer the new syntax READHGT(AREA).
%
%   'srtm1'
%       Downloads SRTM1 tiles which are 9 times bigger than default SRTM3
%       ! EXPERIMENTAL ! since the used URL seems unofficial.
%       ! Beware with large zones may lead to computer memory issues.
%       ! SRTM1 and SRTM3 tiles hold the same filename, while they have
%       different size. Do not store them in the same directory to avoid
%       errors when merging tiles.
%
%   'srtm3'
%       Forces SRTM3 download for all areas (by default, SRTM1 tiles are
%       downloaded only for USA territory, if exists).
%
%   'outdir',OUTDIR
%       Specifies output directory OUTDIR to write downloaded files and/or
%       to search existing files. Former syntax READHGT(LAT,LON,OUTDIR) also
%       accepted.
%
%   'url',URL
%       Specifies the URL address to find HGT files (default is USGS).
%       Former syntax READHGT(LAT,LON,OUTDIR,URL) still accepted.
%
%   'wget'
%       Will use external command wget to download the files (for Linux and
%       MacOSX systems). For MacOSX, wget must be installed using homebrew,
%       macports, fink or compiling from sources.
%       ! NOTICE ! since 2016, USGS has moved SRTM files to a secured
%       https:// URL. This causes Matlab versions older than 2014b failing
%       dowload the tiles automatically, because of unzip function and
%       certificate problems. This issue can be surrounded using this 'wget'
%       option.
%
%
%   --- Examples ---
%
%   - to plot a map of the Paris region, France (single tile):
%       readhgt(48,2)
%
%   - to plot a map of Flores volcanic island, Indonesia (5 tiles):
```

```
%       readhgt(-9,119:123)
%
%   - to plot a map of the Misti volcano, Peru (SRTM1 cropped tile):
%      readhgt([-16.4,-16.2,-71.5,-71.3],'srtm1','interp')
%
%   - to download SRTM1 data of Cascade Range (27 individual tiles):
%       X=readhgt(40:48,-123:-121,'tiles');
%
%
%   --- Information ---
%
%   - each file corresponds to a tile of 1x1 degree of a square grid
%     1201x1201 of elevation values (SRTM3 = 3 arc-seconds), and for USA
%     territory or when using the 'srtm1' option, at higher resolution
%     3601x3601 grid (SRTM1 = 1 arc-second). Note that SRTM1 and SRTM3
%     files have the same syntax names; only the size differs.
%
%   - elevations are of class INT16: sea level values are 0, unknown values
%     equal -32768 (there is no NaN for INT class), use 'interp' option to
%     fill the gaps.
%
%   - note that borders are included in each tile, so to concatenate tiles
%     you must remove one row/column in the corresponding direction (this
%     is made automatically by READHGT when merging tiles).
%
%   - downloaded file is written in the current directory or optional
%     OUTDIR directory, and it remains there. Take care that mixed SRTM1
%     and SRTM3 files may lead to fail to merge. It is better to use
%     different directories for SRTM1 and SRTM3 (see 'outdir' option).
%
%   - NASA Shuttle Radar Topography Mission [February 11 to 22, 2000]
%     produced a near-global covering on Earth land, but still limited to
%     latitudes from 60S to 60N. Offshore tiles will be output as flat 0
%     value grid.
%
%   - if you look for other global topographic data, take a look to ASTER
%     GDEM, worldwide 1 arc-second resolution (from 83S to 83N):
%     http://gdex.cr.usgs.gov/gdex/ (free registration required)
%
%
%   Author: François Beauducel <beauducel@ipgp.fr>
%       Institut de Physique du Globe de Paris
%
%   References:
%       https://dds.cr.usgs.gov/srtm/version2_1
%
%   Acknowledgments: Yves Gaudemer, Jinkui Zhu, Greg
%
%   Created: 2012-04-22 in Paris, France
%   Updated: 2019-05-20
%   Copyright (c) 2019, François Beauducel, covered by BSD License.
%   All rights reserved.
%
%   Redistribution and use in source and binary forms, with or without
%   modification, are permitted provided that the following conditions are
%   met:
%
```

```
%      * Redistributions of source code must retain the above copyright
%        notice, this list of conditions and the following disclaimer.
%      * Redistributions in binary form must reproduce the above copyright
%        notice, this list of conditions and the following disclaimer in
%        the documentation and/or other materials provided with the
distribution
%
%   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
IS"
%   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE
%   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
%   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
BE
%   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
%   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
%   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS
%   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
%   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
%   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE
%   POSSIBILITY OF SUCH DAMAGE.
fidx = 'readhgt_srtm_index.txt';
% ATTENTION: this file must exist in the Matlab path to use default SRTM3
tiles
% since USGS delivers data continent-by-continent with nominative
directories,
% this index file is needed to know the full path name of each tile.
sz1 = [3601,3601]; % SRTM1 tile size
sz3 = [1201,1201]; % SRTM3 tile size
novalue = intmin('int16'); % -32768
n = 1;
srtm1 = any(strcmpi(varargin,'srtm1'));
if srtm1
    % EXPERIMENTAL: SRTM1 full resolution tiles available here (2016):
    %url = 'http://e4ftl01.cr.usgs.gov/SRTM/SRTMGL1.003/2000.02.11';
    %url =
'http://e4ftl01.cr.usgs.gov/MODV6_Dal_D/SRTM/SRTMGL1.003/2000.02.11';
    url = 'http://rmd.neoknet.com/srtm1';
else
    % official USGS SRTM3 tiles (and SRTM1 for USA):
    url = 'https://dds.cr.usgs.gov/srtm/version2_1';
end
srtm3 = any(strcmpi(varargin,'srtm3'));
makeplot = any(strcmpi(varargin,'plot'));
merge = any(strcmpi(varargin,'merge')); % unused former option but needs to
be considered as valid
tiles = any(strcmpi(varargin,'tiles'));
inter = any(strcmpi(varargin,'interp'));
wget = any(strcmpi(varargin,'wget'));
% --- option: 'crop' or 'crop',[LAT1,LAT2,LON1,LON2]
crop = [];
cropflag = 0;
kcrop = find(strcmpi(varargin,'crop'));
if ~isempty(kcrop)
```

```
    cropflag = 1;
    if (kcrop + 1) <= nargin && isnumeric(varargin{kcrop+1})
        crop = varargin{kcrop+1};
        if any(size(crop) ~= [1,4])
            error('CROP option arguments must be a 1x4 vector.')
        end
        cropflag = 2;
    end
end
% --- option: 'decim',N
decim = 0;
decimflag = 0;
kdecim = find(strcmpi(varargin,'decim'));
if ~isempty(kdecim)
    decimflag = 1;
    if (kdecim + 1) <= nargin && isnumeric(varargin{kdecim+1})
        decim = round(varargin{kdecim+1});
        if ~isscalar(decim) || decim < 1
            error('DECIM option argument must be a positive integer.')
        end
        decimflag = 2;
    end
end
% --- option: 'outdir',OUTDIR
out = '.';
outflag = 0;
koutdir = find(strcmpi(varargin,'outdir'));
if ~isempty(koutdir)
    if (koutdir + 1) <= nargin && ischar(varargin{koutdir+1})
        outflag = 2;
        out = varargin{koutdir+1};
        if ~exist(out,'dir')
            error('OUTDIR is not a valid directory.')
        end
    else
        error('''outdir'' option must be followed by OUTDIR string.')
    end
end
% --- option: 'url',URL
urlflag = 0;
kurl = find(strcmpi(varargin,'url'));
if ~isempty(kurl)
    if (kurl + 1) <= nargin && ischar(varargin{kurl+1})
        urlflag = 2;
        url = varargin{kurl+1};
    else
        error('''url'' option must be followed by URL string.')
    end
end
% needs to count the arguments to allow former syntaxes...
nargs = makeplot + merge + tiles + cropflag + srtm1 + srtm3 + inter ...
    + decimflag + outflag + urlflag + wget;
% syntax READHGT without argument: opens the GUI to select a file
if nargin == nargs
    [filename,pathname] = uigetfile('*.hgt;*.hgt.zip','Select a HGT file');
    f = {[pathname,filename]};
    if filename == 0
```

```
        error('Please select a HGT file or use function arguments.');
    end
end
% syntax READHGT(FILENAME, ...)
if nargin == (1 + nargs) && ischar(varargin{1})
    f = varargin{1};
    if ~exist(f,'file')
        error('FILENAME must be a valid file name')
    end
    [~,filename] = fileparts(f);
    f = {f};
end
if nargin < (2 + nargs) && exist('filename','var')
    lat = str2double(filename(2:3));
    if filename(1) == 'S'
        lat = -lat;
    end
    lon = str2double(filename(5:7));
    if filename(4) == 'W'
        lon = -lon;
    end
else
    if nargin < (2 + nargs)
        crop = varargin{1};
        if ~isnumeric(crop) || any(size(crop) ~= [1,4])
            error('Area must be a 4-element vector [LAT1,LAT2,LON1,LON2].')
        end
        lat = floor(min(crop(1:2))):floor(max(crop(1:2)));
        lon =
floor(min(normlon(crop(3:4)))):floor(max(normlon(crop(3:4))));
        cropflag = 2;
    else
        lat = floor(varargin{1}(:));
        lon = normlon(floor(varargin{2}(:)));   % longitudes are normilized
to -180/+179 interval
    end
    if ~isnumeric(lon) || ~isnumeric(lat) || any(abs(lat) > 60) || any(lon
< -180) || any(lon > 179) || isempty(lat) || isempty(lon)
        error('LAT and LON must be numeric and in valid SRTM interval
(abs(LAT)<60).');
    end
    if ~tiles && (any(diff(lat) ~= 1) || any(diff(lon) ~= 1))
        fprintf('READHGT: Warning! LAT and LON vectors do not define
adjoining tiles. Cannot merge and force TILES option.');
        tiles = 1;
    end
    % former syntax: readhgt(LAT,LON,OUTDIR)
    if nargin > (2 + nargs)
    if ~isempty(varargin{3})
            out = varargin{3};
            if ~exist(varargin{3},'dir')
                error('OUTDIR is not a valid directory.')
            end
        end
    end

    % wget option: automatic setting
```

```
    mrel = version('-release');
    if issorted({mrel,'2014b'}) && ~wget
        wget = 1;
        fprintf('** Warning ** Matlab release %s: ''wget'' option
forced.\n',mrel);
    end

    % if LAT/LON are vectors, NDGRID makes a grid of corresponding tiles
    [lat,lon] = ndgrid(lat,lon);
    f = cell(size(lat));
    for n = 1:numel(f)
        if lat(n) < 0
            slat = sprintf('S%02d',-lat(n));
        else
            slat = sprintf('N%02d',lat(n));
        end
        if lon(n) < 0
            slon = sprintf('W%03d',-lon(n));
        else
            slon = sprintf('E%03d',lon(n));
        end
        f{n} = sprintf('%s/%s%s.hgt',out,slat,slon);

        if ~exist(f{n},'file')
            ff = '';
            % former syntax: readght(LAT,LON,OUTDIR,URL)
            if nargin > (3 + nargs)
                url = varargin{4};
                if ~ischar(url)
                    error('URL must be a string.');
                end
            else
                if srtm1
                    %ff = sprintf('/%s%s.SRTMGL1.hgt.zip',slat,slon);
                    ff = sprintf('/%s%s.hgt.zip',slat,slon);
                else
                    %fsrtm =
sprintf('%s/%s',fileparts(mfilename('fullpath')),fidx);
                    fsrtm = fidx;
                    if exist(fsrtm,'file')
                        fid = fopen(fsrtm,'rt');
                        idx = textscan(fid,'%s');
                        fclose(fid);
                        k =
find(~cellfun('isempty',strfind(idx{1},sprintf('%s%s',slat,slon))));
                        if isempty(k)
                            %fprintf('READHGT: Warning! Cannot find %s tile
in SRTM database. Consider it offshore...\n',ff);
                        else
                            % forcing SRTM3 option: takes the first match
in the list
                            if srtm3
                                ff = idx{1}{k(1)};
                            else
                                ff = idx{1}{k(end)};
                            end
```

```
                        end
                    else
                        error('Cannot find "%s" index file to parse SRTM
database. Please download HGT file manually.',fsrtm);
                    end
                end
            end
            if isempty(ff)
                f{n} = '';
            else
                fprintf('Download %s%s ... ',url,ff);
                f{n} = '';
                try
                    if wget
                        if system('which wget')
                            fprintf(' ** wget binary not found. Cannot
download tiles.\n');
                        else
                            tmp = tempname;
                            mkdir(tmp)
                            ftmp =
sprintf('%s/%s%s.hgt.zip',tmp,slat,slon);
                            [s,w] = system(sprintf('wget -O %s
%s%s',ftmp,url,ff));
                            if s
                                disp(w)
                            end
                            f(n) = unzip(ftmp,out);
                            delete(ftmp)
                        end
                    else
                        f(n) = unzip([url,ff],out);
                    end
                    fprintf('done.\n');
                catch
                    fprintf(' ** tile not found. Considering offshore.\n');
                end
            end
        end
    end
end
% pre-allocates X structure (for each file/tile)
X = repmat(struct('hgt',[],'lat',[],'lon',[]),[n,1]);
if n == 1
    tiles = 0;
end
for n = 1:numel(f)
    % unzips HGT file if needed
    if ~isempty(strfind(f{n},'.zip'))
        X(n).hgt = char(unzip(f{n}));
        funzip = 1;
    else
        X(n).hgt = f{n};
        funzip = 0;
    end
    if srtm1
        sz = sz1;
```

```
    else
        sz = sz3;
    end
    if isempty(f{n})
        % offshore: empty tile...
        X(n).z = [];
    else
        % loads data from HGT file
        fid = fopen(X(n).hgt,'rb','ieee-be');
            X(n).z = fread(fid,'*int16');
        fclose(fid);
        switch numel(X(n).z)
        case prod(sz1)
            % srtm3 option: decimates the tile...
            if srtm3
                z = reshape(X(n).z,sz1);
                X(n).z = z(1:3:end,1:3:end);
                sz = sz3;
            else
                sz = sz1;
            end
        case prod(sz3)
            sz = sz3;
        otherwise
            error('"%s" seems not a regular SRTM data file or is
corrupted.',X(n).hgt);
        end
        X(n).z = rot90(reshape(X(n).z,sz));
        % erases unzipped file if necessary
        if (funzip)
            delete(f{n});
        end
    end
    % builds latitude and longitude coordinates
    X(n).lon = linspace(lon(n),lon(n)+1,sz(2));
    X(n).lat = linspace(lat(n),lat(n)+1,sz(1))';

    % interpolates NaN (if not merged)
    if inter && tiles
        X(n).z = fillgap(X(n).lon,X(n).lat,X(n).z,novalue);
    end
end
if ~tiles
    % NOTE: cannot merge mixted SRTM1 / SRTM3 or discontiguous tiles
    Y.lat = linspace(min(lat(:)),max(lat(:))+1,size(lat,1)*(sz(1)-1)+1)';
    Y.lon = linspace(min(lon(:)),max(lon(:))+1,size(lon,2)*(sz(2)-1)+1);
    Y.z = zeros(length(Y.lat),length(Y.lon),'int16');
    for n = 1:numel(X)
        if ~isempty(X(n).z)
            Y.z((sz(1)-1)*(X(n).lat(1)-Y.lat(1)) + (1:sz(1)),(sz(2)-
1)*(X(n).lon(1)-Y.lon(1)) + (1:sz(2))) = X(n).z;
        end
    end
    if cropflag
        if cropflag == 1 || isempty(crop)
            klat = firstlast(any(Y.z ~= 0 & Y.z ~= novalue,2));
            klon = firstlast(any(Y.z ~= 0 & Y.z ~= novalue,1));
```

```
        else
            crop = [minmax(crop(1:2)),normlon(minmax(crop(3:4)))];
            klat = find(Y.lat >= crop(1) & Y.lat <= crop(2));
            klon = find(Y.lon >= crop(3) & Y.lon <= crop(4));
        end
        Y.lat = Y.lat(klat);
        Y.lon = Y.lon(klon);
        Y.z = Y.z(klat,klon);
    end

    if inter
        Y.z = fillgap(Y.lon,Y.lat,Y.z,novalue);
    end
end
if nargout == 0 || makeplot
    if ~tiles
        fplot(Y.lon,Y.lat,Y.z,decim,url,novalue)
    else
        for n = 1:numel(X)
            fplot(X(n).lon,X(n).lat,X(n).z,decim,url,novalue)
        end
    end
end
if nargout == 3 % for backward compatibility...
    varargout{1} = X(1).lon;
    varargout{2} = X(1).lat;
    varargout{3} = X(1).z;
elseif nargout > 0
    if tiles
        if decim > 1
            for n = 1:numel(X)
                X(n).lon = X(n).lon(1:decim:end);
                X(n).lat = X(n).lat(1:decim:end);
                X(n).z = X(n).z(1:decim:end,1:decim:end);
            end
        end
        varargout{1} = X;
    else
        if decim > 1
            Y.lon = Y.lon(1:decim:end);
            Y.lat = Y.lat(1:decim:end);
            Y.z = Y.z(1:decim:end,1:decim:end);
        end
        varargout{1} = Y;
    end
    if nargout == 2
        varargout{2} = f{1}; % for backward compatibility...
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function fplot(x,y,z,decim,url,novalue)
%FPLOT plot the data using DEM function if exists, or IMAGESC
demoptions = {'latlon','legend','lake','nodecim'};
%figure
if decim
    n = decim;
```

```
else
    n = ceil(sqrt(numel(z))/1201);
end
if n > 1
    x = x(1:n:end);
    y = y(1:n:end);
    z = z(1:n:end,1:n:end);
    fprintf('READHGT: In the figure data has been decimated by a factor of
%d...\n',n);
end
if exist('dem','file')
    dem(x,y,z,demoptions{:})
else
    warning('For better results you might install the function dem.m from
http://www.ipgp.fr/~beaudu/matlab.html#DEM')
    z(z==novalue) = 0;
    imagesc(x,y,z);
    if exist('landcolor','file')
        colormap(landcolor(256).^1.3)
    else
        colormap(jet)
    end
    % aspect ratio (lat/lon) is adjusted with mean latitude
    xyr = cos(mean(y)*pi/180);
    set(gca,'DataAspectRatio',[1,xyr,1])
    orient tall
    axis xy, axis tight
end
title(sprintf('MAPA CUADRANTE SRTM/NASA \n from
%s',url),'FontSize',9,'Interpreter','none')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function y = firstlast(x)
k = find(x);
y = k(1):k(end);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function y = minmax(x)
y = [min(x(:)),max(x(:))];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function z = fillgap(x,y,z,novalue)
% GRIDDATA is not efficient for large arrays, but has great advantage to be
% included in Matlab core functions! To optimize interpolation, we
% reduce the number of relevant data by building a mask of surrounding
% pixels of novalue areas... playing with linear index!
sz = size(z);
k = find(z == novalue);
k(k == 1 | k == numel(z)) = []; % removes first and last index (if exist)
if ~isempty(k)
    [xx,yy] = meshgrid(x,y);
    mask = zeros(sz,'int8');
    k2 = ind90(sz,k); % k2 is linear index in the row order
    % sets to 1 every previous and next index, both in column and row order
    mask([k-1;k+1;ind90(fliplr(sz),[k2-1;k2+1])]) = 1;
    mask(k) = 0; % removes the novalue index
    kb = find(mask); % keeps only border values
```

```
    z(k) = int16(griddata(xx(kb),yy(kb),double(z(kb)),xx(k),yy(k)));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function k2 = ind90(sz,k)
[i,j] = ind2sub(sz,k);
k2 = sub2ind(fliplr(sz),j,i); % switched i and j: k2 is linear index in row
order
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function y = normlon(x)
% normalize longitude between -180 and 180
y = mod(x+180,360) - 180;
```

❖ Código completo función *dem*

```
function varargout=dem(x,y,z,varargin)
%DEM Shaded relief image plot
%
%   DEM(X,Y,Z) plots the Digital Elevation Model defined by X and Y
%   coordinate vectors and elevation matrix Z, as a lighted image using
%   specific "landcolor" and "seacolor" colormaps. DEM uses IMAGESC
%   function which is much faster than SURFL when dealing with large
%   high-resolution DEM. It produces also high-quality and moderate-size
%   Postscript image adapted for publication.
%
%   DEM(X,Y,Z,'Param1',Value1,'Param2',Value2,...) specifies options or
%   parameter/value couple (case insensitive):
%
%   [H,I] = DEM(...); returns graphic handle H and optional illuminated
%   image as I, a MxNx3 matrix (if Z is MxN and DECIM is 1).
%
%   I = DEM(...,'noplot') returns a structure I containing fields x, y, z,
%   and illuminated image .rgb without producing a graph on current figure.
%
%
%   --- Lighting options ---
%
%   'Azimuth',A
%       Light azimuth in degrees clockwise relative to North. Default is
%       A = -45 for a natural northwestern illumination.
%
%   'Contrast',C
%       Light contrast, as the exponent of the gradient value:
%           C = 1 for linear contrast (default),
%           C = 0 to remove lighting,
%           C = 0.5 for moderate lighting,
%           C = 2 or more for strong contrast.
%
%   'LCut',LC
```

```
%       Lighting scale saturation cut with a median-style filter in % of
%       elements, such as LC% of maximum gradient values are ignored:
%           LC = 0.2 is default,
%           LC = 0 for full scale gradient.
%
%   'km'
%       Stands that X and Y coordinates are in km instead of m (default).
%       This allows correct lighting. Ignored if LATLON option is used.
%
%
%   --- Elevation colorscale options ---
%
%   'ZLim',[ZMIN,ZMAX]
%       Fixes min and max elevation values for colormap. Use NaN to keep
%       real min and/or max data values.
%
%   'ZCut',ZC
%       Median-style filter to cut extremes values of Z (in % of elements),
%       such that ZC% of most min/max elevation values are ignored in the
%       colormap application:
%           ZC = 0.5 is default,
%           ZC = 0 for full scale.
%
%
%   --- "No Value" elevation options ---
%
%   'NoValue',NOVALUE
%       Defines the values that will be replaced by NaN. Note that values
%       equal to minimum of Z class are automatically detected as NaN
%       (e.g., -32768 for int16 class).
%
%   'NaNColor',[R,G,B]
%       Sets the RGB color for NaN/NoValue pixels (default is a dark gray).
%       Note that your must specify a valid 3-scalar vector (between 0 and
%       1); color characters like 'w' or 'k' are not allowed, use [1,1,1]
%       or [0,0,0] instead.
%
%   'Interp'
%       Interpolates linearly all NaN values (fills the gaps using linear
%       triangulation), using an optimized algorithm.
%
%
%   --- Colormap options ---
%
%   'LandColor',LMAP
%       Uses LMAP colormap instead of default (landcolor, if exists or
%       jet) for Z > 0 elevations.
%
%   'SeaColor',SMAP
%       Sets the colormap used for Z <= 0 elevations. Default is seacolor
%       (if exists) or single color [0.7,0.9,1] (a light cyan) to simulate
%       sea color.
%
%   'ColorMap',CMAP
%       Uses CMAP colormap for full range of elevations, instead of default
%       land/sea. This option overwrites LANDCOLOR/SEACOLOR options.
%
```

```
%    'Lake'
%        Detects automatically flat areas different from sea level (non-zero
%        elevations) and colors them as lake surfaces.
%
%    'LakeZmin',ZMIN
%        Activates the 'lake' option only above ZMIN elevations. For
%        example, use 'lakezmin',0 to limit lake detection on land.
%
%    'Watermark',N
%        Makes the whole image lighter by a factor of N.
%
%
%    --- Basemap and scale options ---
%
%    'Legend'
%        Adds legends to the right of graph: elevation scale (colorbar)
%        and a distance scale (in km).
%
%    'Cartesian'
%        Plots classic basemap-style axis, considering coordinates X and Y
%        as cartesian in meters. Use parameter "km' for X/Y in km.
%
%    'LatLon'
%        Plots geographic basemap-style axis in deg/min/sec, considering
%        coordinates X as longitude and Y as latitude. Axis aspect ratio
%        will be adjusted to approximatively preserve distances (this is
%        not a real projection!). This overwrites ZRatio option.
%
%    'AxisEqual', 'auto' (default) | 'manual' | 'off'
%        When 'Cartesian' or 'LatLon' option is used, automatic axes scaling
%        is applied to respect data aspect ratio. Default mode is 'auto' and
%        uses AXIS EQUAL and DASPECT functions. The 'manual' mode modifies
%        axes width or height with respect to the paper size in order to
%        produce correct data scaling at print (but not necessarily at
%        screen). The 'off' mode disables any scaling.
%
%    Additionnal options for basemap CARTESIAN or LATLON:
%
%    'BorderWidth',BW
%        Border width of the basemap axis, in % of axis height. Default is
%        BW = 1%.
%
%    'XTick',DX
%    'YTick',DY
%        X and Y Tick length (same unit as X and Y). Default is automatic.
%        Tick labels are every 2 ticks.
%
%    'FontSize',FS
%        Font size for X and Y tick labels. Default is FS = 10.
%
%    'FontBold'
%        Font weight bold for tick labels.
%
%    'Position',P
%        Position of the tick labels: 'southwest' (default), 'southeast',
%        'northwest','northeast'
%
```

```
%
%   --- Decimation options ---
%
%   For optimization purpose, DEM will automatically decimate data to limit
%   to a total of 1500x1500 pixels images. To avoid this, use following
%   options, but be aware that large grids may require huge computer
%   ressources or induce disk swap or memory errors.
%
%   'Decim',N
%       Decimates matrix Z at 1/N times of the original sampling.
%       If N < 0, oversamples at -N rate.
%
%   'NoDecim'
%       Forces full resolution of Z, no decimation (N =1).
%
%
%
%   --- Informations ---
%
%   Colormaps are Mx3 RGB matrix so it is easy to modify saturation
%   (CMAP.^N), set darker (CMAP/N), lighter (1 - 1/N + CMAP/N), inverse
%   it (flipud(CMAP)), etc...
%
%   To get free worldwide topographic data (SRTM), see READHGT function.
%
%   For backward compatibility, the former syntax is still accepted:
%   DEM(X,Y,Z,OPT,CMAP,NOVALUE,SEACOLOR) where OPT = [A,C,LC,ZMIN,ZMAX,ZC],
%   also option aliases DEC, DMS and SCALE, but there is no argument
%   checking. Please prefer the param/value syntax.
%
%   Author: François Beauducel <beauducel@ipgp.fr>
%   Created: 2007-05-17 in Guadeloupe, French West Indies
%   Updated: 2017-03-29
%   History:
%   [2017-03-29] v2.6
%       - fix in 'lakezmin' option (thanks to Mustafa Çomo?lu)
%   [2017-01-09] v2.5
%       - new option 'lakezmin' to limit lake detection
%   [2016-12-21] v2.4
%       - improves the colormap splitting between land and sea
%   [2016-04-19] v2.3
%       - major update (thanks to mas Wiwit)
%   [2016-01-31] v2.2
%       - adds option 'Position' for tick labels
%   [2015-08-22] v2.1
%       - minor fix (former versions of Matlab compatibility)
%   [2015-08-19] v2.0
%       - image is now 100% true color (including the legend colorbar),
%         thus completely independent from the figure colormap
%   [2014-10-14]
%       - 'decim' option allows oversampling (negative value)
%   [2014-06-06]
%       - improves backward compatibility (adds strjoin subfunction)
%   [2014-03-18]
%       - adds new axisequal option
%   [2013-03-11]
%       - new options: 'km', 'watermark', 'fontsize', 'bordersize'
```

```
%       - improve legend colorbar
%       - all options now passed as param/value
%   [2013-01-14]
%       - improved light rendering (using surface normals instead of
gradient)
%       - improved 'lake' detection algorithm
%       - new 'nancolor' option to set NaN color
%       - adds a length scale with 'dec' option
%       - minor code improvements
%   [2013-01-07]
%       - adds 'interp' option (fill the gaps)
%       - adds 'seacolor' colormap for negative elevations (bathymetry)
%   [2013-01-02]
%       - adds a 'lake' option
%       - minor bug correction
%   [2012-09-26]
%       - now accepts row/column vectors for X and/or Y.
%   [2012-05-29]
%       - adds basemap-style axis in decimal or lat/lon modes
%       - adds elevation and distance scales
%   [2012-05-18]
%       - new landcolor.m colormap function
%       - new arguments to control colormap scaling
%       - median-style filters for light and colormap
%   [2012-04-26]
%       - Optimizations: adds a decimation for large DEM grids.
%
%   Copyright (c) 2016, François Beauducel, covered by BSD License.
%   All rights reserved.
%
%   Redistribution and use in source and binary forms, with or without
%   modification, are permitted provided that the following conditions are
%   met:
%
%      * Redistributions of source code must retain the above copyright
%        notice, this list of conditions and the following disclaimer.
%      * Redistributions in binary form must reproduce the above copyright
%        notice, this list of conditions and the following disclaimer in
%        the documentation and/or other materials provided with the
distribution
%
%   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
IS"
%   AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE
%   IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
%   ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS
BE
%   LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
%   CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
%   SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS
%   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
%   CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
%   ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
THE
```

```
%   POSSIBILITY OF SUCH DAMAGE.
if nargin < 3
    error('Not enough input arguments.');
end
holdon = ishold;
degkm = 6378*pi/180; % one latitude degree in km
sea_color = [.7,.9,1]; % default sea color (light cyan)
grey = 0.2*[1,1,1]; % a dark gray
% ------------------------------------------------------------------------
% --- Manage input arguments
% number of arguments param/value
nargs = 0;
if ~isnumeric(x) || ~isnumeric(y) || ~isnumeric(z)
    error('X,Y and Z must be numeric.')
end
if all(size(x) ~= 1) || all(size(y) ~= 1)
    error('X and Y must be vectors, not matrix.')
end
if length(x) ~= size(z,2) || length(y) ~= size(z,1)
    error('If Z has a size of [M,N], X must have a length of N, and Y a
length of M.')
end
% OPTIONS and PARAM/VALUE arguments

% AZIMUTH param/value
[s,az] = checkparam(varargin,'azimuth',@isscalar);
nargs = nargs + 2;
if s==0
    az = -45; % default
end
% ELEVATION param/value
[s,el] = checkparam(varargin,'elevation',@isscalar);
nargs = nargs + 2;
if s==0
    el = 0; % default
end
% CONTRAST param/value
[s,ct] = checkparam(varargin,'contrast',@isscalar);
nargs = nargs + 2;
if s
    ct = abs(ct);
else
    ct = 1; % default
end
% LCUT param/value
[s,lcut] = checkparam(varargin,'lcut',@isperc);
nargs = nargs + 2;
if s==0
    lcut = .2; % default
end
% NOVALUE param/value
[s,novalue] = checkparam(varargin,'novalue',@isscalar);
nargs = nargs + 2;
if s==0
    % default: min value for integer class / NaN for float
    S = whos('z');
    if strfind(S.class,'int')
```

```
        novalue = intmin(S.class);
    else
        novalue = NaN;
    end
end
% NANCOLOR param/value
[s,novalue_color] = checkparam(varargin,'nancolor',@isrgb);
nargs = nargs + 2;
if s==0
    novalue_color = grey; % default
end
% LANDCOLOR param/value
[s,cland] = checkparam(varargin,'landcolor',@isrgb);
nargs = nargs + 2;
if s==0
    % default: landcolor or jet
    if exist('landcolor','file')
        cland = landcolor.^1.3;
    else
        cland = jet(256);
    end
end
% SEACOLOR param/value
[s,csea] = checkparam(varargin,'seacolor',@isrgb);
nargs = nargs + 2;
if s==0
    % default: seacolor or single color
    if exist('seacolor','file')
        csea = seacolor;
    else
        csea = sea_color;
    end
end
% COLORMAP param/value
[s,cmap] = checkparam(varargin,'colormap',@isrgb);
nargs = nargs + 2;
if s
    cland = [];
    csea = [];
else
    % default
    cmap = cland;
end
% ZLIM param/value
[s,zmm] = checkparam(varargin,'zlim',@isvec);
nargs = nargs + 2;
if s
    zmin = min(zmm);
    zmax = max(zmm);
else
    zmin = NaN; % default
    zmax = NaN; % default
end
% ZCUT param/value
[s,zcut] = checkparam(varargin,'zcut',@isperc);
nargs = nargs + 2;
if s==0
```

```
    zcut = .5; % default
end
% ZRATIO param/value
[s,zratio] = checkparam(varargin,'zratio',@isscalar);
nargs = nargs + 2;
if s==0
    zratio = 1; % default
end
% WATERMARK param/value
[s,wmark] = checkparam(varargin,'watermark',@isscalar);
nargs = nargs + 2;
if s
    wmark = abs(wmark);
else
    wmark = 0; % default
end
% DECIM param/value and NODECIM option
[s,decim] = checkparam(varargin,'decim',@isscalar);
if s
    decim = round(decim);
    nargs = nargs + 2;
else
    decim = any(strcmpi(varargin,'nodecim')); % default
    nargs = nargs + 1;
end
% LAKEZMIN param/value option
[s,lakezmin] = checkparam(varargin,'lakezmin',@isscalar);
if s
    lake = 1;
    nargs = nargs + 2;
else
    lake = 0;
    lakezmin = NaN;
end
% FONTSIZE param/value
[s,fs] = checkparam(varargin,'fontsize',@isscalar);
nargs = nargs + 2;
if s==0
    fs = 10; % default
end
% BORDERWIDTH param/value
[s,bw] = checkparam(varargin,'borderwidth',@isperc);
nargs = nargs + 2;
if s==0
    bw = 1; % default
end
% XTICK param/value
[s,ddx] = checkparam(varargin,'xtick',@isscalar);
nargs = nargs + 2;
if s==0
    ddx = 0; % default (automatic)
end
% YTICK param/value
[s,ddy] = checkparam(varargin,'ytick',@isscalar);
nargs = nargs + 2;
if s==0
    ddy = 0; % default (automatic)
```

```
end
% POSITION param/value
[s,tpos] =
checkparam(varargin,'position',@ischar,{'southwest','southeast','northwest'
,'northeast'});
nargs = nargs + 2;
if s==0
    tpos = 'southwest'; % default
end
% AXISEQUAL param/value
[s,axeq] =
checkparam(varargin,'axisequal',@ischar,{'auto','manual','off'});
nargs = nargs + 2;
if s==0 || ~any(strcmpi(axeq,{'manual','off'}))
    axeq = 'auto'; % default (automatic)
end
% CROP param/value
[s,crop] = checkparam(varargin,'crop',@isvec,4);
nargs = nargs + 2;
% options without argument value
km = any(strcmpi(varargin,'km'));
dec = any(strcmpi(varargin,'cartesian') | strcmpi(varargin,'dec'));
dms = any(strcmpi(varargin,'latlon') | strcmpi(varargin,'dms'));
kmscale = any(strcmpi(varargin,'kmscale'));
scale = any(strcmpi(varargin,'legend') | strcmpi(varargin,'scale'));
inter = any(strcmpi(varargin,'interp'));
lake = any(strcmpi(varargin,'lake')) || lake;
fbold = any(strcmpi(varargin,'fontbold'));
noplot = any(strcmpi(varargin,'noplot'));
clines = any(strcmpi(varargin,'contourlines'));
% for backward compatibility (former syntax)...
nargs = nargs + dec + dms + scale + kmscale + inter + lake + km + fbold +
noplot + clines;
if (nargin - nargs) > 3 && ~isempty(varargin{1})
    opt = varargin{1};
    if ~isnumeric(opt)
        error('OPT = [A,C,S,ZMIN,ZMAX,ZCUT] argument must be numeric.');
    end
    if ~isempty(opt)
        az = opt(1);
    end
    if length(opt) > 1
        ct = opt(2);
    end
    if length(opt) > 2
        lcut = opt(3);
    end
    if length(opt) > 4
        zmin = opt(4);
        zmax = opt(5);
    end
    if length(opt) > 5
        zcut = opt(6);
    end
end
if (nargin - nargs) > 4 && ~isempty(varargin{2})
    cmap = varargin{2};
```

```
    csea = [];
end
if (nargin - nargs) > 5 && ~isempty(varargin{3})
    novalue = varargin{3};
end
if (nargin - nargs) > 6 && ~isempty(varargin{4})
    csea = varargin{4};
end
% further test of input arguments
if dms && any(abs(y) > 91)
    error('With LATLON option Y must be in valid latitudes interval
(decimal degrees).')
end
if km
    zratio = 1000;
end
% ---------------------------------------------------------------------
% --- Pre-process DEM data
% crops data if needed
if numel(crop)==4
    fprintf('DEM: crops original data from [%g,%g,%g,%g] to
[%g,%g,%g,%g]...\n', ...
        min(x(:)),max(x(:)),min(y(:)),max(y(:)),crop);
    kx = find(x >= crop(1) & x <= crop(2));
    ky = find(y >= crop(3) & y <= crop(4));
    x = x(kx);
    y = y(ky);
    z = z(ky,kx);
end
% decimates data to avoid disk swap/out of memory...
nmax = 1500;
if decim
    n = decim;
else
    n = ceil(sqrt(numel(z))/nmax);
end
if n > 1
    x = x(1:n:end);
    y = y(1:n:end);
    z = z(1:n:end,1:n:end);
    fprintf('DEM: data has been decimated by a factor of %d...\n',n);
end
z = double(z); % necessary for most of the following calculations...
z(z==novalue) = NaN;
if isempty(csea)
    k = (z~=0 & ~isnan(z));
else
    k = ~isnan(z);
end
if isnan(zmin)
    zmin = nmedian(z(k),zcut/100);
end
if isnan(zmax)
    zmax = nmedian(z(k),1 - zcut/100);
end
dz = zmax - zmin;
if decim && n < 0
```

```
    xi = linspace(x(1),x(end),-n*length(x));
    yi = linspace(y(1),y(end),-n*length(y))';
    [xx,yy] = meshgrid(xi,yi);
    z = interp2(x,y,z,xx,yy,'*cubic');
    x = xi;
    y = yi;
    fprintf('DEM: data has been oversampled by a factor of %d...\n',-n);
end
if inter
    z = fillgap(x,y,z);
end
% ------------------------------------------------------------------------
% --- Process lighting
if dz > 0
    % builds the colormap: concatenates seacolor and landcolor around 0
    % after interpolation to have exactly one color level per meter.
    if ~isempty(csea)
%       l = size(csea,1);
%       if zmin < 0 && zmax > 0
%           r = size(cland,1)*abs(zmin)/zmax/l;
%           cmap =
cat(1,interp1(1:l,csea,linspace(1,l,ceil(l*r)),'*linear'),cland);
        if zmin < 0 && zmax > 0
            lcs = size(csea,1);
            lcl = size(cland,1);
            cmap =
cat(1,interp1(1:lcs,csea,linspace(1,lcl,abs(zmin)+1),'*linear'), ...
                interp1(1:lcl,cland,linspace(1,lcl,abs(zmax)),'*linear'));
        elseif zmax <=0
            cmap = csea;
        end
    end

    % normalisation of Z using CMAP and convertion to RGB
    I = ind2rgb(uint16(round((z - zmin)*(size(cmap,1) - 1)/dz) + 1),cmap);

    if ct > 0
        % computes lighting from elevation gradient
        %[fx,fy] = gradient(z,x,y);
        if dms
            ryz = degkm*1000;
            rxz = degkm*1000*cosd(mean(y));
        else
            rxz = zratio;
            ryz = zratio;
        end
        [xx,yy] = meshgrid(x*rxz,y*ryz);
        [fx,fy,fz] = surfnorm(xx,yy,z);
        [ux,uy,uz] = sph2cart((90-az)*pi/180,el*pi/180,1);
        fxy = fx*ux + fy*uy + fz*uz;
        clear xx yy fx fy fz    % free some memory...

        fxy(isnan(fxy)) = 0;
        % computes maximum absolute gradient (median-style), normalizes,
        % saturates and duplicates in 3-D matrix
        li = 1 - abs(sind(el)); % light amplitude (experimental)
```

```
        r = repmat(max(min(li*fxy/nmedian(abs(fxy),1 - lcut/100),1),-
1),[1,1,3]);
        rp = (1 - abs(r)).^ct;

        % applies contrast using exponent
        I = I.*rp;

        % lighter for positive gradient
        I(r>0) = I(r>0) + (1 - rp(r>0));

    end
    % set novalues / NaN to nancolor
    [i,j] = find(isnan(z));
    if ~isempty(i)

I(sub2ind(size(I),repmat(i,1,3),repmat(j,1,3),repmat(1:3,size(i,1),1))) =
repmat(novalue_color,size(i,1),1);
    end

    % lake option
    if lake
        klake = islake(z);
        if ~isnan(lakezmin)
            klake(z < lakezmin) = false; % removes indexes below ZMIN
        end
    else
        klake = 0;
    end

    % set the seacolor (upper color) for 0 values
    if ~isempty(csea)
        [i,j] = find(z==0 | klake);
        if ~isempty(i)

I(sub2ind(size(I),repmat(i,1,3),repmat(j,1,3),repmat(1:3,size(i,1),1))) =
repmat(csea(end,:),size(i,1),1);
        end
    end
    if wmark
        I = watermark(I,wmark);
    end
    txt = '';

else

    I = repmat(shiftdim(sea_color,-1),size(z));
    cmap = repmat(sea_color,[256,1]);
    txt = 'Mak Byur!';  % Splash !
end
% ------------------------------------------------------------------------
% --- ends the function when 'noplot' option is on
if noplot
    varargout{1} = struct('x',x,'y',y,'z',z,'rgb',I);
    return
end
```

```
% --------------------------------------------------------------------
% --- plots the RGB image
hh = imagesc(x,y,I);
if ~isempty(txt)
    text(mean(x),mean(y),txt,'Color',sea_color/4, ...
        'FontWeight','bold','HorizontalAlignment','center')
end
orient tall; axis xy
if strcmpi(axeq,'auto')
    axis equal
end
axis tight
xlim = [min(x),max(x)];
ylim = [min(y),max(y)];
zlim = [min([z(z(:) ~= novalue);zmin]),max([z(z(:) ~= novalue);zmax])];
if dms
    % approximates X-Y aspect ratio for this latitude (< 20-m precision for
1x1° grid)
    xyr = cos(mean(y)*pi/180);
else
    xyr = 1;
end
bw0 = max(diff(xlim)*xyr,diff(ylim))/100;
bwy = bw*bw0; % Y border width = 1%
bwx = bwy/xyr; % border width (in degree of longitude)
% --------------------------------------------------------------------
% --- Axis basemap style
if dec || dms
    axis off
    if strcmpi(axeq,'manual')
        ppos = get(gcf,'PaperPosition');
        apos = get(gca,'Position');
        xyf =
(xyr*diff(xlim)/apos(3)/ppos(3))/(diff(ylim)/apos(4)/ppos(4));
        if xyf >= 1
            set(gca,'Position',[apos(1),apos(2),apos(3),apos(4)/xyf]);
        else
            set(gca,'Position',[apos(1),apos(2),apos(3)*xyf,apos(4)]);
        end
    end
    if strcmpi(axeq,'auto')
        if diff(xlim)*xyr <= diff(ylim)
            set(gca,'DataAspectRatio',[1,xyr,1])
        else
            set(gca,'DataAspectRatio',[1/xyr,1,1])
        end
    end
    if bw > 0
        % transparent borders
        patch([xlim(1)-bwx,xlim(2)+bwx,xlim(2)+bwx,xlim(1)-bwx],ylim(1) -
bwy*[0,0,1,1],'k','FaceColor','none','clipping','off')
        patch([xlim(1)-bwx,xlim(2)+bwx,xlim(2)+bwx,xlim(1)-bwx],ylim(2) +
bwy*[0,0,1,1],'k','FaceColor','none','clipping','off')
        patch(xlim(1) - bwx*[0,0,1,1],[ylim(1)-
bwy,ylim(2)+bwy,ylim(2)+bwy,ylim(1)-
bwy],'k','FaceColor','none','clipping','off')
```

```
        patch(xlim(2) + bwx*[0,0,1,1],[ylim(1)-
bwy,ylim(2)+bwy,ylim(2)+bwy,ylim(1)-
bwy],'k','FaceColor','none','clipping','off')
    end
    dlon = {'E','W'};
    dlat = {'N','S'};
    if fbold
        fw = 'bold';
    else
        fw = 'normal';
    end

    if ddx == 0
        ddx = dtick(diff(xlim),dms);
    end
    if ddy == 0
        ddy = dtick(diff(ylim),dms);
    end
    xtick = (ddx*ceil(xlim(1)/ddx)):ddx:xlim(2);
    for xt = xtick(1:2:end)
        dt = ddx - max(0,xt + ddx - xlim(2));
        patch(repmat(xt + dt*[0,1,1,0]',[1,2]),[ylim(1) -
bwy*[0,0,1,1];ylim(2) + bwy*[0,0,1,1]]','k','clipping','off')
        if fs > 0
            if ~isempty(regexp(tpos,'north','once'))
                text(xt,ylim(2) +
1.2*bwy,deg2dms(xt,dlon,dec),'FontSize',fs,'FontWeight',fw, ...

'HorizontalAlignment','center','VerticalAlignment','bottom');
            else
                text(xt,ylim(1) -
1.2*bwy,deg2dms(xt,dlon,dec),'FontSize',fs,'FontWeight',fw, ...

'HorizontalAlignment','center','VerticalAlignment','top');
            end
        end
    end
    ytick = (ddy*ceil(ylim(1)/ddy)):ddy:ylim(2);
    for yt = ytick(1:2:end)
        dt = ddy - max(0,yt + ddy - ylim(2));
        patch([xlim(1) - bwx*[0,0,1,1];xlim(2) + bwx*[0,0,1,1]]',repmat(yt
+ dt*[0,1,1,0]',[1,2]),'k','clipping','off')
        if fs > 0
            if ~isempty(regexp(tpos,'east','once'))
                text(xlim(2) +
1.2*bwx,yt,deg2dms(yt,dlat,dec),'FontSize',fs,'FontWeight',fw, ...

'HorizontalAlignment','center','VerticalAlignment','top','rotation',90);
            else
                text(xlim(1) -
1.2*bwx,yt,deg2dms(yt,dlat,dec),'FontSize',fs,'FontWeight',fw, ...

'HorizontalAlignment','center','VerticalAlignment','bottom','rotation',90);
            end
        end
    end
end
```

```
% ------------------------------------------------------------------------
% --- Contour lines
% contour lines
if clines
    dz = diff(zlim);
    % empirical ratio between horizontal extent and elevation interval (dz)
    %rzh =
dz/min(diff(x([1,end]))*cosd(mean(dlat)),diff(y([1,end])))/degkm/4e2;
    dd = dtick(dz);
    dz0 = ceil(zlim(1)/dd)*dd:dd:floor(zlim(2)/dd)*dd;
    dz0(ismember(0,dz0)) = [];  % eliminates 0 value
    dd = dtick(dz/5);
    dz1 = ceil(zlim(1)/dd)*dd:dd:floor(zlim(2)/dd)*dd;
    dz1(ismember(dz1,dz0)) = [];    % eliminates minor ticks in major ticks
    clrgb = .3*ones(1,3);
    hold on
    [~,h] = contour(x,y,z,[0,0,dz1],'-','Color',clrgb);
    set(h,'LineWidth',0.1);
    [cs,h] = contour(x,y,z,[0,0,dz0],'-','Color',clrgb);
    set(h,'LineWidth',1);
    if ~isempty(dz0)% && clineslabel
        clabel(cs,h,dz0,'Color',clrgb,'FontSize',fs/2,'FontWeight','bold',
...
            'LabelSpacing',288,'Margin',fs)
    end
    hold off
end
% ------------------------------------------------------------------------
% --- Scales legend
%wsc = diff(xlim)*0.01;
wsc = bw0;
xsc = xlim(2) + wsc*2 + bwx;
if scale
    if wmark
        cmap = watermark(cmap,wmark);
    end
    % -- elevation scale (colorbar)
    zscale = linspace(zmin,zmax,length(cmap))';
    yscale = linspace(0,diff(ylim)/2,length(cmap));
    ddz = dtick(dz*max(0.5*xyr*diff(xlim)/yscale(end),1));
    ztick = (ddz*ceil(zscale(1)/ddz)):ddz:zscale(end);
    rgbscale = ind2rgb(uint16(round((zscale - zmin)*(size(cmap,1) - 1)/dz)
+ 1),cmap);
    ysc = ylim(1);
    hold on
    imagesc(xsc + wsc*[-1,1]/2,ysc +
yscale,repmat(rgbscale,1,2),'clipping','off');
    patch(xsc + wsc*[-1,1,1,-1],ysc +
yscale(end)*[0,0,1,1],'k','FaceColor','none','Clipping','off')
    text(xsc + 2*wsc + zeros(size(ztick)),ysc + (ztick -
zscale(1))*0.5*diff(ylim)/diff(zscale([1,end])),num2str(ztick'), ...

'HorizontalAlignment','left','VerticalAlignment','middle','FontSize',fs*.75
)
    % indicates min and max Z values
    text(xsc,ysc - bwy/2,sprintf('%g
m',roundsd(zlim(1),3)),'FontWeight','bold', ...
```

```
'HorizontalAlignment','left','VerticalAlignment','top','FontSize',fs*.75)
    text(xsc,ysc + .5*diff(ylim) + bwy/2,sprintf('%g
m',roundsd(zlim(2),3)),'FontWeight','bold', ...

'HorizontalAlignment','left','VerticalAlignment','bottom','FontSize',fs*.75
)

    % frees axes only if not hold on
    if ~holdon
        hold off
    end

end
if scale || kmscale
    % -- distance scale (in km)
    if dms
        fsc = degkm;
    else
        fsc = zratio/1e3;
    end
    dkm = dtick(diff(ylim)*fsc);
    ysc = ylim(2) - 0.5*dkm/fsc;
    if dkm > 1
        skm = sprintf('%g km',dkm);
    else
        skm = sprintf('%g m',dkm*1000);
    end
    if kmscale
        xsc = xlim(1) + wsc*2;
        ysc = ylim(1) + wsc*2;
        patch(xsc + dkm*[0,1,1,0]/fsc,ysc + wsc*[-1,-
1,0,0],'k','FaceColor','w')
        for n = 0:2:(dkm-1)
            patch(xsc + (n + [0,1,1,0])/fsc,ysc + wsc*[-1,-
1,0,0],'k','FaceColor','k')
        end
        text(xsc +
.5*dkm/fsc,ysc,skm,'HorizontalAlignment','center','VerticalAlignment','bott
om', ...
            'Color','k','FontWeight','bold','FontSize',fs)
    else
        patch(xsc + wsc*[-1,-1,0,0],ysc + dkm*0.5*[-1,1,1,-
1]/fsc,'k','FaceColor',grey,'clipping','off')
        text(xsc,ysc,skm,'rotation',-
90,'HorizontalAlignment','center','VerticalAlignment','bottom', ...
            'Color',grey,'FontWeight','bold','FontSize',fs*.75)
    end
end
if nargout > 0
    varargout{1} = hh;
end
if nargout > 1
    varargout{2} = I;
end
if nargout > 2
```

```
    varargout{3} = z;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function y = nmedian(x,n)
%NMEDIAN Generalized median filter
%   NMEDIAN(X,N) sorts elemets of X and returns N-th value (N normalized).
%   So:
%      N = 0 is minimum value
%      N = 0.5 is median value
%      N = 1 is maximum value
if nargin < 2
    n = 0.5;
end
y = sort(x(:));
y = interp1(sort(y),n*(length(y)-1) + 1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function dd = dtick(dlim,deg)
%DTICK Tick intervals
if nargin < 2
    deg = 0;
end
if deg && dlim <= 2/60
    % less than 2 minutes: base 36
    m = 10^floor(log10(dlim*36))/36;
elseif deg && dlim <= 2
    % less than 2 degrees: base 6
    m = 10^floor(log10(dlim*6))/6;
else
    % more than few degrees or not degrees: decimal rules
    m = 10^floor(log10(dlim));
end
p = ceil(dlim/m);
if p <= 1
    dd = .1*m;
elseif p == 2
    dd = .2*m;
elseif p <= 5
    dd = .5*m;
else
    dd = m;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function s = deg2dms(x,ll,dec)
%DEG2DMS Degree/minute/second display
if dec
    s = sprintf('%7.7g',x);
else
    xa = abs(x) + 1/360000;
    %sd = sprintf('%d%c',floor(xa),176);    % ASCII char 176 is the degree
sign
    sd = sprintf('%d°',floor(xa));
    sm = '';
    ss = '';
    if mod(x,1)
```

```
        sm = sprintf('%02d''',floor(mod(60*xa,60)));
        sa = floor(mod(3600*xa,60));
        if sa
            ss = sprintf('%02d"',sa);
        else
            if strcmp(sm,'00''')
                sm = '';
            end
        end
    end
    s = [sd,sm,ss,ll{1+int8(x<0)}];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function z = fillgap(x,y,z)
% GRIDDATA is not efficient for large arrays, but has great advantage to be
% included in Matlab's core functions! To optimize interpolation, we
% reduce the amount of relevant data by building a mask of all surrounding
% pixels of novalue areas... playing with linear index!
sz = size(z);
k = find(isnan(z));
k(k == 1 | k == numel(z)) = []; % removes first and last index (if exist)
if ~isempty(k)
    [xx,yy] = meshgrid(x,y);
    mask = zeros(sz,'int8');
    k2 = ind90(sz,k); % k2 is linear index in the row order
    % sets to 1 every previous and next index, both in column and row order
    mask([k-1;k+1;ind90(fliplr(sz),[k2-1;k2+1])]) = 1;
    mask(k) = 0; % removes the novalue index
    kb = find(mask); % keeps only border values
    z(k) = griddata(xx(kb),yy(kb),z(kb),xx(k),yy(k));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function k2 = ind90(sz,k)
[i,j] = ind2sub(sz,k);
k2 = sub2ind(fliplr(sz),j,i); % switched i and j: k2 is linear index in row
order
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function k = islake(z)
% ISLAKE mask of zero gradient on 3x3 tiles
% We use diff matrix in row and column directions, and shift it to build
% a single vectorized test of surrounding pixels. To do this we must
% concatenate unit vectors in different combinations...
dx = diff(z,1,2);   % differences in X direction
dy = diff(z,1,1);   % differences in Y direction
u1 = ones(size(z,1),1); % row unit vector
u2 = ones(1,size(z,2)); % column unit vector
u2r = u2(2:end);
% index of the tiles center pixel
k = ( ...
    [u2;dy] == 0 & [dy;u2] == 0 & ...
    [u1,dx] == 0 & [dx,u1] == 0 & ...
    [u1,[dx(2:end,:);u2r]] == 0 & [[dx(2:end,:);u2r],u1] == 0 & ...
    [u1,[u2r;dx(1:end-1,:)]] == 0 & [[u2r;dx(1:end-1,:)],u1] == 0 ...
);
```

```
% now extends it to surrounding pixels
k(1:end-1,:) = (k(1:end-1,:) | k(2:end,:));
k(2:end,:) = (k(2:end,:) | k(1:end-1,:));
k(:,1:end-1) = (k(:,1:end-1) | k(:,2:end));
k(:,2:end) = (k(:,2:end) | k(:,1:end-1));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function s = isrgb(x,n)
if nargin < 2
    n = 0;
end
if isnumeric(x) && (n == 1 && all(size(x) == [1,3]) || n == 0 && size(x,2)
== 3) ...
        && all(x(:) >= 0 & x(:) <= 1)
    s = 1;
else
    s = 0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function s = isperc(x)
if isnumeric(x) && isscalar(x) && x >= 0 && x <= 100
    s = 1;
else
    s = 0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function s = isvec(x,n)
if nargin < 2
    n = 2;
end
if isnumeric(x) && numel(x) == n
    s = 1;
else
    s = 0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function y=roundsd(x,n)
og = 10.^(floor(log10(abs(x)) - n + 1));
y = round(x./og).*og;
y(x==0) = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function y = watermark(x,n)
if nargin < 2
    n = 2;
end
if n == 0
    y = x;
else
    y = (x/n + 1 - 1/n);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function [s,v] = checkparam(arg,nam,func,val)
```

```
switch func2str(func)
    case 'isscalar'
        num = 1;
        mes = 'scalar value';
    case 'isperc'
        num = 1;
        mes = 'percentage scalar value';
    case 'isvec'
        num = 1;
        if nargin < 4
            val = 2;
        end
        mes = sprintf('%d-element vector',val);
    case 'isrgb'
        num = 1;
        mes = '[R,G,B] vector with 0.0 to 1.0 values';
    case 'ischar'
        num = 0;
        mes = 'string';
        if nargin > 3
            mes = sprintf('%s (%s)',mes,strjoin(val,' or '));
        end
    otherwise
        num = 1;
        mes = 'value';
end
s = 0;
v = [];
k = find(strcmpi(arg,nam));
if ~isempty(k)
    if (k + 1) <= length(arg) ...
            && (~num || isnumeric(arg{k+1})) ...
            && (nargin < 4 && func(arg{k+1}) ...
                || (nargin > 3 && (strcmp(func2str(func),'ischar') &&
ismember(arg{k+1},val)) ...
                    || strcmp(func2str(func),'isvec') &&
func(arg{k+1},val)))
        v = arg{k+1};
        s = 1;
    else
        error('%s option must be followed by a valid %s.',upper(nam),mes)
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
function s=strjoin(c,d)
%STRJOIN Join cell array of strings
%(this is for Matlab versions < 2013a backward compatibility)
if nargin < 2
    d = '';
end
n = numel(c);
ss = cell(2,n);
ss(1,:) = reshape(c,1,n);
ss(2,1:n-1) = {d};
s = [ss{:}];
```