Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Simulador en Python del procedimiento de predicción de la interferencia entre estaciones a frecuencias superiores a 0,1 GHz según la Rec. ITU-R P.452

Autor: Lázaro Rodríguez Márquez
Tutor: Susana Hornillo Mellado

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Simulador en Python del procedimiento de predicción de la interferencia entre estaciones a frecuencias superiores a 0,1 GHz según la Rec. ITU-R P.452

Autor:
Lázaro Rodríguez Márquez

Tutor:
Susana Hornillo Mellado
Profesor titular

Dpto. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Proyecto Fin de Carrera: Simulador en Python del procedimiento de predicción de la interferencia entre estaciones a frecuencias superiores a 0,1 GHz según la Rec. ITU-R P.452

Autor: Lázaro Rodríguez Márquez

Tutor: Susana Hornillo Mellado

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia y amigos

A mis maestros

Agradecimientos

Me gustaría aprovechar estas líneas para agradecer a todas aquellas personas que de un modo u otro han contribuido en el desarrollo y finalización en este duro y largo camino.

A mi familia, por todo el sacrificio realizado para que yo tuviera la oportunidad de estudiar una carrera universitaria.

A mis amigos, un apoyo incondicional, que, en los momentos mas difíciles en los que ni yo confiaba en mi, me alentaban a continuar sin excepción.

A mis compañeros de clase y estudio, amigos también, por que sin ellos el camino habría sido mucho mas duro.

A mis profesores, por toda la ayuda aportada y por todo lo que me han enseñado.

Mención especial para Susana Hornillo, quien ha compartido todos sus conocimientos y me ha guiado en este proyecto.

Lázaro Rodríguez Márquez

Sevilla, 2020

Resumen

Este proyecto está basado en la Recomendación ITU-R P.452 [1], la cual describe un método de predicción para evaluar la interferencia entre estaciones situadas en la superficie de la tierra, teniendo en cuenta los mecanismos de interferencia en cielo despejado.

Para determinar esta interferencia entre estaciones se elaborá un simulador en Python. Este programa partirá de una seria de parámetros introducidos por el usuario y se encargará de desarrollar el método descrito en la Recomendación.

Abstract

This project is based on Recommendation ITU-R P.452 [1], which describes a prediction method to evaluate interference between stations located on the earth's surface, considering clear sky interference mechanisms.

To determine this interference between stations, a simulator will be developed in Python. This program will start from a series of parameters entered by the user and will oversee developing the method described in the Recommendation.

Índice

Agradecimientos ix

Resumen xi

Abstract xiii

Índice xv

Índice de Tablas xvii

Índice de Figuras xix

Notación xxi

1 Introducción 3

2 Conceptos Básicos 5

2.1 Ondas de radio 5

2.2 Propagación 6

2.3. Atenuaciones 7

2.3.1. Visibilidad directa 7

2.3.2. Difracción 7

2.3.3. Dispersión troposférica 8

2.3.4. Propagación por conductos de superficie 8

2.3.5. Reflexión y refracción en capas elevadas 8

2.3.6. Variación altura-ganancia de la ocupación del suelo 8

2.3.7. Dispersión por hidrometeoros 9

2.4. Recomendación UIT-R P.452-16. Datos necesarios para la formulación de una predicción 9

2.4.1. Datos de partida 9

2.4.2. Predicción de mes más desfavorable 9

2.4.3. Datos radiometeorológicos 10

2.4.4. Radio efectivo de la Tierra 11

2.4.5. Parámetros del análisis del trayecto 11

3 Herramientas de desarrollo 13

3.1. Python 3.8.2 13

3.1.1. Librerías usadas 13

3.2. TkInter 14

3.3. Pycraf 15

3.3.1. Instalación 15

3.3.2. Funcionalidad 15
3.3.3. Datos SRTM 16

4 Funcionamieno del programa 19
4.1. Pestaña principal 19
4.2. Pestaña secundaria o de gráficas 22
4.3. Ventanas emergentes de Warnings 24
4.4. Programa Python 28
4.4.1. Fichero programa.py 28
4.4.2. Fichero menu.py 28
4.4.3. Fichero funciones.py 30

5 Casos prácticos y comparación de resultados 31
5.1. Caso 1 31
5.2. Caso 2 34

6 Conclusiones y futuras lineas de trabajo 39

Apéndice A 41

Código fuente 41

Referencias 61

Índice de Códigos 63

Glosario 65

Índice de Tablas

Tabla 4.1 Alturas y distancias nominales de la ocupación del suelo 21

Índice de Figuras

Figura 2-1. Espectro elecromagnético 5
Figura 2-2. Transmisor isotrópico 6
Figura 2-3. Caminos de propagación 6
a) Rayo directo y reflexión desde la superficie de la tierra. 6
b) Onda ionosférica. 6
c) Onda de superficie. 6
Figura 2-4. Mecanismos de propagación de la interferencia (I) [1]. 7
Figura 2-5. Mecanismos de propagación de la interferencia (II) [1]. 8

Figura 3-1. Ejemplo ventana TkInter 15
Figura 3-2. Ejemplo de uso de Pycraf 16
Figura 3-3. Resultados ejemplo Pycraf 16

Figura 4-1. Ventana principal 20
Figura 4-2. Ayuda para el parámetro "ht" 20
Figura 4-3. Desplegable para activar el cuadro de texto 21
Figura 4-4. Pestaña de gráficas 22
Figura 4-5. Ejemplo de gráficas 23
Figura 4-6. Ventana de selcción de formatode las gráficas 23
Figura 4-7. Warning debido a campos sin rellenar 24
Figura 4-8. Warning debido a los parámetros geográficos 25
Figura 4-9. Warning debido al parámetro "d" fuera de rango 25
Figura 4-10. Warning debido al parámetro "f" fuera de rango 26
Figura 4-11. Warning debido a no elegir una opción en la ocupacion del suelode la estación interferida 26
Figura 4-12. Warning en la pestaña de gráficas (I) 27

Figura 4-13. Warning en la pestaña de gráficas (II) 27
Figura 4-14. Warning en la pestaña de gráficas (III) 28

Figura 5-1. Enlace caso 1 31
Figura 5-2. Perfil del trayecto caso 1 32
Figura 5-3. Datos de entrada para el caso 1 32
Figura 5-4. Resultado caso 1 32
Figura 5-5. Módulo Pycraf para el caso 1 33
Figura 5-6. Resultado de la simulación con el módulo Pycraf para el caso 1 33
Figura 5-7. Gráfica caso 1 34
Figura 5-8. Enlace caso 2 35
Figura 5-9. Perfil del trayecto caso 2 35
Figura 5-10. Datos de entrada para el caso 2 36
Figura 5-11. Resultado caso 2 36
Figura 5-12. Módulo Pycraf para el caso 2 37
Figura 5-13. Resultado de la simulación con el módulo Pycraf para el caso 2 37
Figura 5-14. Gráfica caso 2 38

Notación

EM	Electromagnetismo
p	Porcentaje de tiempo requerido durante el cual no se rebasa la pérdida básica de transmisión
p_w	Pérdidas básicas de transmisión para el porcentaje de tiempo del mes más desfavorable
β_0	Porcentaje de tiempo en el que pueden esperarse sobretasas de la variación de la refracción en la atmosfera inferior
e	Número e
k_{50}	Factor del valor mediano del radio efectivo de la Tierra
k_β	Factor del radio efectivo de la Tierra excedido durante el β_0 % del tiempo
N_0	Refractividad de la superficie a nivel del mar
a_e	Radio efectivo de la Tierra
a_β	Radio efectivo de la Tierra rebasado durante el β_0 % del tiempo
ΔN	Proporción de variación del índice medio de refracción radioeléctrica
$\leq$	Menor o igual
$\geq$	Mayor o igual
$<$	Menor
$>$	Mayor
◦	Grados

1 INTRODUCCIÓN

El mundo de las tecnologías y la telecomunicación ha cambiado de forma radical en los últimos 20 años. Todo empezó con las teorías de James C. Maxwell, quien sentó las bases de la propagación de ondas electromagnéticas. A partir de estos estudios se desarrolló todo lo que hoy conocemos acerca de la transmisión por radio, televisión, telefonía móvil…

Por otro lado, la programación como base para el cálculo y simulación de diferentes entornos puede llegar a ser fundamental para el estudio. En este ámbito debemos mencionar el lenguaje de programación que usaremos, Python, el cual se caracteriza por su simplicidad, rapidez, flexibilidad y por su gran comunidad de desarrolladores.

Es aquí donde unimos ambos conceptos, telecomunicaciones y programación, para realizar nuestro trabajo. Este tratará de simular el proceso detallado en la recomendación ITU-R P.452-16, en concreto el capitulo dedicado a los modelos de propagación en cielo despejado. Una predicción para evaluar la interferencia entre estaciones situadas en la superficie de la Tierra a frecuencias superiores a 0.1 GHz en cielo despejado.

En una propagación con cielo despejado deberemos tener en cuenta:

- La interferencia sufrida por otra antena con visibilidad directa.
- Difracción.
- Dispersión troposférica.
- Interferencia debida a la propagación por conductos y reflexión en las capas.
- Pérdidas adicionales debidas a la ocupación del suelo.

La simulación estará compuesta por un entorno gráfico donde deberemos introducir varios parámetros iniciales para el posterior cálculo de la interferencia.

2 CONCEPTOS BÁSICOS

James Clerk Maxwell, en 1865, introdujo por primera vez el término de electromagnetismo (EM), deduciendo así que la luz estaba compuesta de campos eléctricos y magnéticos que se propagan por el espacio. Varios años después, un científico alemán llamado Hertz descubrió que las ondas de radio tienen una naturaleza similar a las ondas EM, pero son invisibles.

Estas ondas de radio, las cuales corresponden a una determinada banda de frecuencias en el espectro electromagnético (10 KHz – 10 THz), se propagan por el espacio libre desde una antena transmisora a otra antena receptora. En su propagación se ven afectadas por varios fenómenos que atenúan la energía radiada y provoca pérdidas en las transmisiones.

2.1 Ondas de radio

Como hemos comentado anteriormente, las ondas de radio son un tipo de radiación electromagnética que se propagan desde frecuencias de 10 KHz hasta 10 THz. Como todas las ondas electromagnéticas, si viajan por el vacío o por el aire, las ondas de radio viajan a la velocidad de la luz.

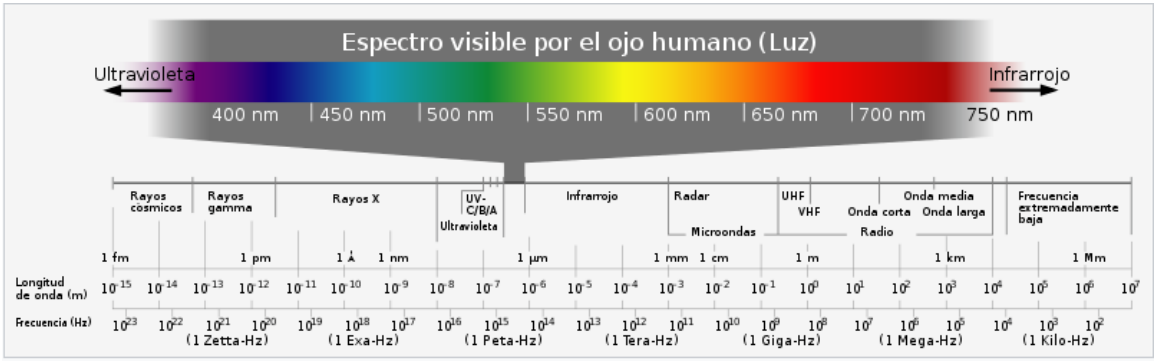


Figura 2.1. Espectro electromagnético.

Las ondas de radio se generan normalmente de forma artificial por un transmisor, aunque hay algunos fenómenos naturales, como los relámpagos, que las crean de forma natural.

Una de las cosas mas interesantes que poseen este tipo de ondas es que tienen características de propagación diferentes en función de la frecuencia usada. Por ejemplo, en determinadas frecuencias, la onda puede difractarse alrededor de obstáculos como montañas y seguir el contorno de la superficie terrestre. Estas son las llamadas ondas de superficie. También pueden refractarse en la ionosfera y alcanzar puntos más allá

del horizonte, recibiendo el nombre de ondas ionosféricas.

2.2 Propagación

En este apartado consideraremos un radiador isotrópico en la parte superior de un mástil, como se muestra en la Figura 2.2. La característica principal de este elemento es su capacidad para radiar en todas las direcciones.

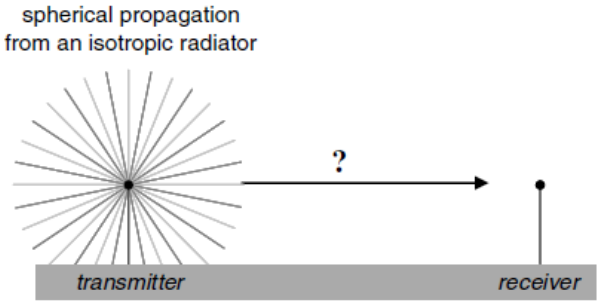


Figura 2.2. Transmisor isotrópico [2]

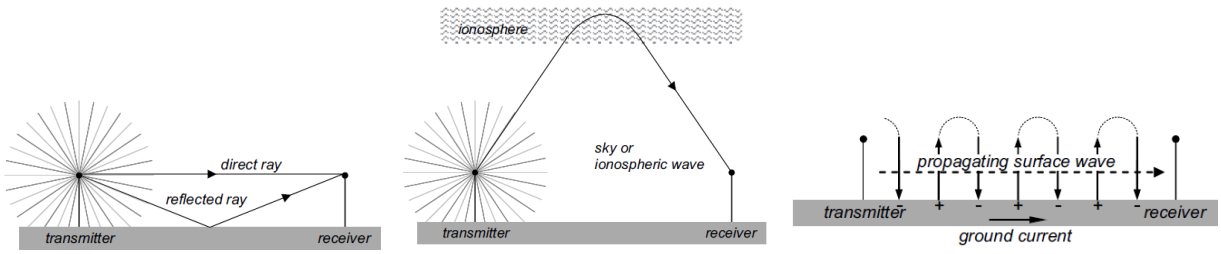
En este escenario tendríamos tres caminos diferentes en los que la señal llegaría a la antena receptora, a través del espacio, cielo y superficie.

En la figura 2.3a tenemos representada la ruta mas rápida que podría tomar la onda hasta llegar al receptor, el rayo directo. También representado en la misma figura, tenemos un segundo camino como resultado de la reflexión desde la superficie de la tierra o de objetos como edificios o vehículos. Es evidente que el receptor y el transmisor deben estar en una visión directa para que estos rayos lleven la señal.

La ionosfera es el conjunto de las capas cargadas que existen sobre la superficie de la Tierra y es la culpable de la existencia de la llamada onda ionosférica. Esta es el resultado de algunos de los rayos proyectados hacia arriba de la figura 2.2 que son refractados a la superficie de la tierra de nuevo y por lo tanto al receptor.

Por último, tenemos la onda de superficie. Este mecanismo de propagación transporta la señal de forma paralela al suelo usando las cargas de la capa superficial de la tierra, como se ilustra en la figura 2.3c.

Los tres mecanismos pueden existir simultáneamente sin problemas, sin embargo, cada mecanismo tiene sus propios rangos mas efectivos de frecuencia. Además, los ejemplos ilustrados están basados en un elemento isotrópico, el cual en la práctica no es ultilizado. Las antenas elegidas estarían diseñadas para optimizar el mecanismo de propagación más apropiado para el rango de frecuencia utilizado.



a) b) c)

Figura 2.3. Caminos de propagación [2].

a) Rayo directo y reflexión desde la superficie de la tierra.

b) Onda ionosférica.

c) Onda de superficie.

2.3. Atenuaciones

A lo largo del trayecto que recorre la onda desde la estación transmisora hasta la receptora, la señal puede sufrir interferencias debida a diversos factores tales como el clima, el porcentaje de tiempo en cuestión, la distancia o la topografía del trayecto. Puede darse un único mecanismo de propagación de la interferencia o varios a la vez. A continuación, presentaremos los principales:

- Visibilidad directa (Figura 2.4).
- Difracción (Figura 2.4).
- Dispersión troposférica (Figura 2.4).
- Propagación por conductos de superficie (Figura 2.5).
- Reflexión y refracción en capas elevadas (Figura 2.5).
- Variación altura-ganancia de la ocupación del suelo.
- Dispersión por hidrometeoros (Figura 2.4)

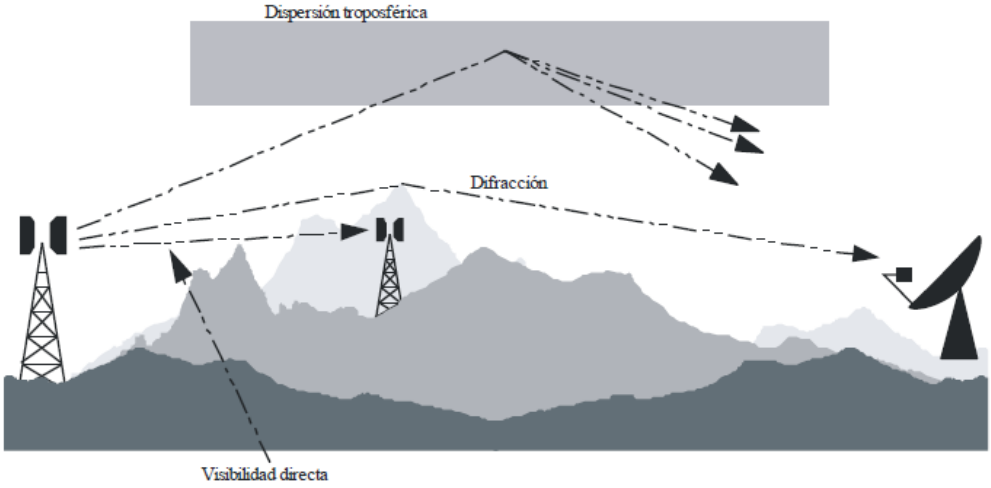


Figura 2.4. Mecanismos de propagación de la interferencia (I) [1].

2.3.1. Visibilidad directa

Este es el mecanismo más directo de transmisión de la interferencia en unas condiciones atmosféricas de equilibrio. No obstante, existe un problema adicional cuando la difracción del subtrayecto produce un ligero aumento del nivel de la señal.

Además, en la mayoría de los trayectos debemos tener en cuenta el fenómeno llamado multitrayecto. Este fenómeno se da cuando las señales de radio llegan a la antena o antenas receptoras por dos o más caminos y en diferentes tiempos y puede provocar problemas en la recepción de la señal, debido a la interacción entre las señales recibidas.

2.3.2. Difracción

La difracción es un término que se atribuye a varios fenómenos que ocurren cuando una onda se encuentra con un obstáculo o una pequeña apertura. La onda, que atraviesa un obstáculo por un orificio pequeño o se desvía por las esquinas, se distorsiona y se propaga en todas las direcciones detrás de dicho obstáculo.

Los efectos de difracción suelen ser dominantes cuando aparecen niveles significativos de la

señal. La capacidad de predicción de la difracción debe ser tal que permita incluir las situaciones de terreno liso, de obstáculos discretos y de terreno irregular.

2.3.3. Dispersión troposférica

En los trayectos en los que el campo de difracción se hace muy débil, como pueden ser trayectos de 100 km o 200 km, la dispersión troposférica define el nivel de fondo de interferencia.

Las ondas de radio, normalmente transmitidas en un haz estrecho y dirigido por encima del horizonte en la dirección de la estación receptora, pasan a través de la troposfera y parte de la energía es rebotada hacia la Tierra de nuevo, permitiendo así que la estación receptora pueda captar la señal.

A menudo, este mecanismo de dispersión será tan pequeño que podamos considerarlo nulo.

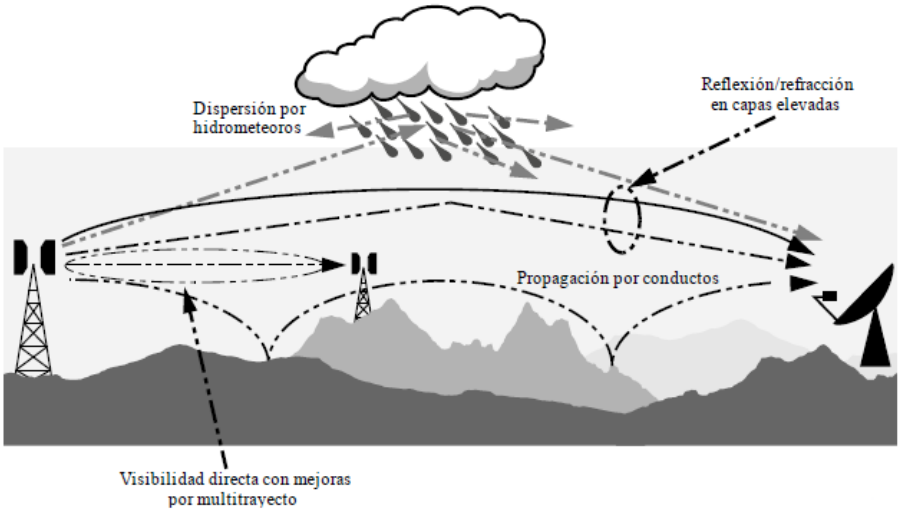


Figura 2.5. Mecanismos de propagación de la interferencia (II) [1].

2.3.4. Propagación por conductos de superficie

Este mecanismo puede provocar interferencia sobre el agua y zonas de costa plana, dando como resultado elevados niveles de señal en trayectos largos, por ejemplo, una distancia de 450 km sobre el mar. Este mecanismo es el más importante de corta duración.

2.3.5. Reflexión y refracción en capas elevadas

La repercusión de este mecanismo de interferencia puede ser bastante significativo en distancias muy largas como 200-300 km. Estas reflexiones y/o refracciones en capas altas, de cientos de metros, pueden suponer que las pérdidas de difracción del terreno sean superadas en situaciones donde la geometría del trayecto sea favorable.

2.3.6. Variación altura-ganancia de la ocupación del suelo

Esta atenuación, más que un mecanismo de propagación de la interferencia, es un procedimiento de corrección que nos permite predecir una pérdida adicional debida a la ocupación del suelo en uno o ambos extremos del trayecto, si se dispone de las características de dicho suelo (edificios, vegetación, etc.). La pérdida máxima adicional que tendremos con este procedimiento será de unos 20 dB por encima de los 0.9

Ghz e ira disminuyendo progresivamente hasta los 5 dB a los 0.1 Ghz. En zonas urbanas de edificios altos y separados por espacios abiertos, estas pérdidas serán menores que, por ejemplo, en lugares donde haya bloques de edificios más bajos y juntos.

2.3.7. Dispersión por hidrometeoros

Aunque este tipo de atenuación no se toma en cuenta en este proyecto, ya que lleva un procedimiento de calculo distinto a los mencionados anteriormente, no está de mal mencionar sus efectos ya que es una fuente potencial de interferencia entre transmisores y receptores de estaciones terrenales.

Estas pérdidas están referidas a las ocasionadas por una determinada célula de lluvia, que puede actuar de forma omnidireccional. No obstante, esta atenuación no suele ocasionar problemas significativos.

2.4. Recomendación UIT-R P.452-16. Datos necesarios para la formulación de una predicción

En este apartado vamos a comentar los datos necesarios con los que abordaremos los cálculos para una predicción usando modelos de propagación en cielo despejado (capítulos 3 y 4 de nuestra recomendación).

2.4.1. Datos de partida

Los datos básicos a partir de los cuales podremos sacar toda la información necesaria para el procedimiento del cálculo de la interferencia son:

- Frecuencia (GHz).
- Porcentaje de tiempo requerido durante el cual no se rebasa la pérdida básica de transmisión, *p*.
- Latitud de las estaciónes (grados).
- Longitud de las estaciones (grados).
- Altura del centro de la antena sobre el nivel del suelo (m).
- Altura del centro de la antena sobre el nivel medio del mar (m).
- Ganancia de la antena en la dirección del círculo máximo de interferencia (dBi).
- Polarización, vertical u horizontal.

2.4.2. Predicción de mes mas desfavorable

La elección de este parámetro esta ligada a los objetivos de calidad. Debemos determinar la transmisión mas desfavorable en la que deberán basarse las prdidas básicas de transmisión admisibles mínimas.

Existen modelos de predicción de la propagación que predicen la distribución anual de las perdidas básicas de transmisión. En ocasiones se utilizan directamente estas prediciones calculadas para cada mes. Sin embargo, si se quiere calcular las prediciones según el mes mas desfavrable, necesitaremos calcular el porcentaje de tiempo anual equivalente, *p*, en función del mes mas desfavorable p_w.

$$p = 10^{\left(\frac{\log(pw)+\log(\mathrm{GL})-0.186\omega-0.444}{0.816+0.078\omega}\right)} \quad \% \tag{2.1}$$

siendo:

ω: fracción del trayecto sobre el agua.

$$G_L = \begin{cases} \sqrt{1{,}1 + |\cos 2\varphi|^{0{,}7}} & \text{para } |\varphi| \le 45° \\ \sqrt{1{,}1 - |\cos 2\varphi|^{0{,}7}} & \text{para } |\varphi| > 45° \end{cases} \quad (2.2)$$

Ψ: latitud del centro del trayecto.

Si fuese necesario, p se limitaría de forma que $12\ p \geq p_w$.

2.4.3. Datos radiometeorológicos

Este procedimiento utiliza tres parámetros radiometeorológicos para describir la variación de las condiciones depropagación.

- ΔN, necesario para el cálculo efectivo de tierra. Es la proporción de variación del índice medio de refracción radioeléctrica a lo largo del primer kilómetro de la atmosfera. Este parámetro será positivo en nuestro proyecto.
- β_0, es el porcentaje de tiempo en el que pueden esperarse sobretasas de la variación de la refracción en la atmosfera inferior. Se determina con la expresión:

$$\beta_0 = \begin{cases} 10^{-0{,}015|\varphi| + 1{,}67}\,\mu_1\,\mu_4 & \% \quad \text{para } |\varphi| \le 70° \\ 4{,}17\,\mu_1\,\mu_4 & \% \quad \text{para } |\varphi| > 70° \end{cases} \quad (2.3)$$

donde:

$$\mu_1 = \left[10^{\frac{-d_{tm}}{16 - 6{,}6\tau}} + \left[10^{-(0{,}496 + 0{,}354\tau)} \right]^5 \right]^{0{,}2} \quad (2.4)$$

donde μ_1 está limitado a $\mu_1 \leq 1$, y:

$$\tau = \left[1 - e^{-\left(4{,}12 \times 10^{-4} \times d_{lm}^{2{,}41}\right)} \right] \quad (2.5)$$

donde:

d_{tm}: sección continua mas larga sobre la tierra (interior + costera) del trayecto en el circulo máximo.

d_{lm}: sección continua mas larga sobre la tierra (interior) del trayecto en el circulo máximo.

con:

$$\mu_4 = \begin{cases} 10^{(-0{,}935 + 0{,}0176|\varphi|)\log \mu_1} & \text{para } |\varphi| \le 70° \\ 10^{0{,}3 \log \mu_1} & \text{para } |\varphi| > 70° \end{cases} \quad (2.6)$$

- N_0, refractividad de la superficie a nivel del mar.

2.4.4. Radio efectivo de la Tierra

Teniendo en cuenta que vamos a suponer un radio real de la Tierra de 6371 km, el valor mediano del radio efectivo de la tierra queda determinado por:

$$a_e = 6371 \cdot k_{50} \tag{2.7}$$

donde:

$$k_{50} = \frac{157}{157 - \Delta N} \tag{2.8}$$

Para el radio efectivo de la Tierra rebasado durante el $\beta_{0\%}$ del tiempo queda determinado por:

$$a_\beta = 6371 \cdot k_\beta \tag{2.9}$$

donde $k_\beta = 3.0$.

2.4.5. Parámetros del análisis del trayecto

Por último, necesitaremos una serie de parámetros resultantes del análisis del trayecto que describieremos acontinuación:

- d, distancia del trayecto a lo largo del circulo máximo.
- d_{lt} *y* d_{lr}, si el trayecto es transhorizonte, serán la distancia desde las antenas a sus respectivos horizontes. En el caso de que el trayecto fuese de visibilidad directa, ambos parámetros se fijaran a la distancia desde los terminales a los puntos de perfil indentificados como bordes principales.
- θ_t *y* θ_r, ángulos de elevación de las antenas.
- θ, distancia angular del trayecto.
- h_{ts} *y* h_{rs}, altura del centro de las antenas sobre el nivel del mar.
- h_{te} *y* h_{re}, alturas efectivas de las antenas sobre el terreno.
- d_b, longitud combinada de las secciones del trayecto sobre el agua.
- ω, fracción del trayecto sobre el agua.
- d_{ct} *y* d_{cr}, distancia sobre tierra desde las antenas hasta la costa.

Con todos estos datos disponibles ya podríamos calcular la interferencia producida entre dos terminales siguiendo el procedimiento descrito en la recomendación.

3 HERRAMIENTAS DE DESARROLLO

El simulador está desarrollado en leguaje Python, dando uso a una de sus librerías para el entorno gráfico, TkInter, para dar al usuario mayor simplicidad y poder sacarle el mayor rendimiento posible. Este software es libre y no requiere licencia. Además, el usuario no tendrá que preocuparse por instalar nada, ya que todo lo necesario para que funcione correctamente se le proporcionará en una carpeta comprimida.

3.1 Python 3.8.2

Python es el lenguaje elegido para desarrollar el software del simulador por la legibilidad de su código. Es un lenguaje multiplataforma, dinámico e interpretado capaz de soportar una programación orientada a objetos, programación imperativa y programación funcional.

La version utilizda ha sido la 3.8.2. También se han instalado paquetes y añadido librerías para dessarollar nuestro trabajo. Aquí debemos mencionar el instalador de paquetes pip que ha sido el usado para la instalacion de dichos paquetes y librerías sin tener que descargarlas por internet.

Código 3.1 Ejemplo de uso del instalador de paquetes pip.

```
pip install [package]
```

3.1.1 Librerías usadas

Es muy común en los proyectos de programación el incluir nuevas librerías para el uso de funciones que no estén entre las predeterminadas. Para el desarrollo de nuestro proyecto hemos tenido que incorporar las siguientes:

- *Math*, para funciones matemáticas que no incorpora Python.
- *NumPy*, para otras funciones matemáticas y soporte de arrays.
- *Matplotlib*, para la generación de gráficas.
- *Astropy*, esta librería incluye muchos paquetes que son muy útiles en el ámbito de la astronomía, en nuestro caso la usasmos por su funcionalidad con las unidades.
- *PIL*, para el uso de Widget con imágenes.

- *Itu676*, este paquete nos proporciona diversas funciones para el cálculo de la atenuación debida a los gases.

3.2 TkInter

TkInter es la biblioteca gráfica de Python usada para el desarrollo de la interfaz gráfica del proyecto. Esta facilita la construcción y configuración de ventanas con *widgets*. Estos *widgets* muestran información al usuario y les permiten interconectar con la aplicación.

Ventana principal:

- La creación de la ventana principal se hace con la instancia *Tk()*. Luego se hace la llamada a la ventana, como si fuera una función.

Algunos de los elementos (*widgets*) más usados son:

- *Label*, para las etiquetas de texto.
- *Entry*, casilla para la introducción de texto o algún parámetro.
- *Button*, para la utilización de botones a los que aplicar eventos.
- *OptionMenu*, para seleccionar una opción entre una lista.
- *Frame*, contenedor para otros *widgets*.
- *Checkbuttom*, casillas de verificación.
- *Messagebox*, para ventanas emergentes de información.

Código 3.2 Ejemplo de ventana con *widgets*

```
from tkinter import *
ventana = Tk()
ventana.title("Welcome to LikeGeeks app")
ventana.geometry('350x200')
lbl = Label(ventana, text="Hello")
lbl.grid(column=0, row=0)
def clicked():
lbl.configure(text="Button was clicked !!")
btn = Button(ventana, text="Click Me", command=clicked)
btn.grid(column=1, row=0)
window.mainloop()
```

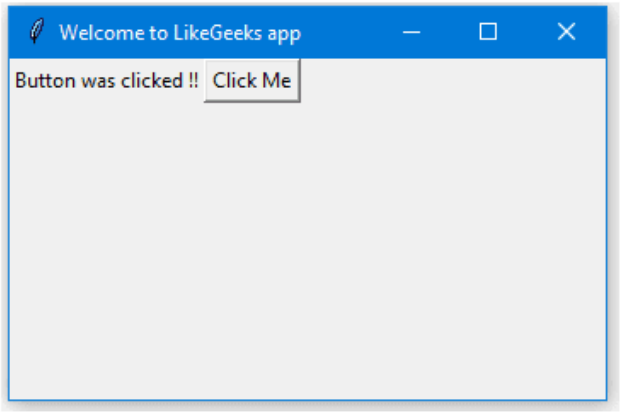


Figura 3.1. Ejemplo ventana TkInter.

3.3 Pycraf

Pycraft es un paquete de Python que nos proporciona funciones y procedimientos para el cálculo de diversas tareas, entre ellas, el cálculo de la atenuación entre una fuente interferente y la estación servicio. Y es por esto último por lo que lo hemos incluido en nuestro proyecto a modo de verificación de nuestro trabajo.

3.3.1 Instalación

Para el uso de este paquete hemos tenido que instalar varias bibliotecas y softwares adicionales que se complementan entre sí:

- *Setuptools*, usada para la instalación correcta del paquete.
- *Cython*, compilador estático.
- *SciPy*, conjunto de librerías de código abierto para funciones matemáticas, ciencia e ingeniería.
- *Pytest*, herramienta de testeo de la instalación del paquete Pycraf.
- *NumPy*, *Astropy* y *Matplotlib*, ya mencionadas anteriormente.

3.3.2 Funcionalidad

El uso de Pycraf es muy sencillo, una vez instalados todos los paquetes, inicializamos los datos de partida necesarios, calculamos el perfil del trayecto con la función *pathprof.PathProp()*, y ya podremos hacer uso de las distintas funciones para el cálculo de las diferentes atenuaciones.

Los datos iniciales serán:

- Latitud y longitud de ambas estaciones.
- Frecuencia.
- Una resolución de la altura del perfil que llamaremos, *hprof_step*.
- Omega, *w*, que será la fracción del trayecto sobre el agua.
- Temperatura y presión.
- Porcentaje de tiempo requerido durante el cual no se rebasa la pérdida básica de transmisión, *p*.
- Altura de las antenas sobre el nivel del suelo.

- Ganancia de las antenas.
- Categoría de la ocupación del suelo, *clutter zones*.

La función *pathprof.PathProp()* calcula el perfil del trayecto entre las dos estaciones proporcionadas, esto nos supone conocer todos los parámetros y características al detalle del trayecto, muy útil para comparar y validar datos a la hora de identificar las características de nuestro trayecto.

Para el uso de las diferentes funciones de cálculo de atenuaciones será necesario el resultado del análisis proporcionado por la función *pathprof.PathProp()*. En la siguiente imagen tenemos un ejemplo de código de todo lo explicado hasta el momento del uso de Pycraf.

```
1  from astropy import units as u
2  from pycraf import pathprof, conversions as cnv
3  pathprof.SrtmConf.set(download='missing')
4  freq = 1. * u.GHz         # Fracuencia
5  lon_tx, lat_tx = 6.8836 * u.deg, 50.525 * u.deg     #longitud y latitud del transmisor
6  lon_rx, lat_rx = 6.89 * u.deg, 50.530 * u.deg       #longitud y latitud del receptor
7  hprof_step = 100 * u.m  # resolucion de la altura del perfil
8  omega = 0. * u.percent  # fracion del trayecto sobre el mar
9  temperature = 290. * u.K   # temperatura
10 pressure = 1013. * u.hPa    # presion
11 time_percent = 0.01 * u.percent  # porcentaje de tiempo no excedido (p)
12 h_tg, h_rg = 5 * u.m, 50 * u.m  # alturas de las antenas sobre el nivel del suelo
13 G_t, G_r = 0 * cnv.dBi, 15 * cnv.dBi    # ganancia de antenas
14 # clutter zones
15 zone_t, zone_r = pathprof.CLUTTER.URBAN, pathprof.CLUTTER.SUBURBAN
16 pprop = pathprof.PathProp(
17      freq,
18      temperature, pressure,
19      lon_tx, lat_tx,
20      lon_rx, lat_rx,
21      h_tg, h_rg,
22      hprof_step,
23      time_percent,
24      zone_t=zone_t, zone_r=zone_r,
25      )
26 L_bfsg = pathprof.loss_freespace(pprop)[0]
27 L_bs = pathprof.loss_troposcatter(pprop, G_t, G_r)
28 L_complete = pathprof.loss_complete(pprop)
29 print("L_bfsg = ", "{0:.4f}".format(L_bfsg))  #perdida de transmision basica
30 print("L_bs = ","{0:.4f}".format(L_bs))    # pérdida por dispersión troposférica
31 print("L = ","{0:.4f}".format(pathprof.loss_complete(pprop, G_t, G_r)[6])) # perdidas totales
```

Figura 3.2. Ejemplo de uso de Pycraf.

```
L_bfsg =  89.6244 dB
L_bs =  113.0950 dB
L =  94.5585 dB
```

Figura 3.3. Resultados ejemplo Pycraf.

3.3.3 Datos SRTM

Llegados a este punto debemos destacar la línea 3 del código de la figura 3.2. Esta es la causante del uso de los datos SRTM, pero ¿qué son estos datos y para qué sirven?

En la complejidad del cálculo de la interferecia encontramos un problema, necesitamos datos

cartográficos para el análisis del trayecto y la generación del perfil de altura, estos son los datos SRTM. El programa descarga los datos necesarios automáticamente con dicha línea de código (línea 3). Estos archivos tienen extensión .htg y se descargan en el mismo directorio donde se encuentra el código fuente del programa. Están nombrados de una manera muy simple, el primer carácter es una "N" (norte) o "S" (sur) que indica si el siguiente número es una latitud positiva o negativa. De la misma forma, el cuarto carácter, es una "W" (oeste) o "E" (este) e indica si la longitud es positiva o negativa. Un ejemplo del formato de estos datos podría ser: *N50W006.htg*.

4 FUNCIONAMIENO DEL PROGRAMA

En este capítulo vamos a describir el funcionamiento de nuestro *software* así como el código usado para el desarrollo de este. El simulador se basa en una sencilla interfaz gráfica para interactuar con el usuario.

4.1 Pestaña principal

Para iniciar la aplicación, será necesario ejecutar el siguiente código por línea de comando una vez estemos en el directorio donde tengamos nuestro código fuente, usando el comando "*cd"* para cambiar de carpeta:

Código 4.1 Comando para iniciar la aplicación

```
cd programa python
programa.py
```

Una vez iniciado el programa nos aparecerá una ventana emergente con dos pestañas, la primera la llamaremos pestaña principal, en la que podremos meter los datos iniciales necesarios y elegir algunas opciones para calcular correctamente la interferencia. Esta ventana tiene un tamaño de 950x450 px ajustable a gusto del usuario. Esta programado con este tamaño ya que es el mínimo para tener una buena visión de todos los campos existentes en la ventana.

La pestaña principal podríamos decir que está dividida en dos secciones por el símbolo "‖". La primera, la parte izquierda, a su vez dividida en dos columnas, está dedicada a todos los parámetros que debemos introducir por cuadro de texto (*widget* "*Entry"*).

Cada parámetro tiene su nombre, la unidad en la que trabajaremos con dicho parámetro, entre paréntesis, y un botón señalizado con una "?" que hace referencia a una pequeña explicacion del parámetro, es lo que llamaremos un botón de ayuda.

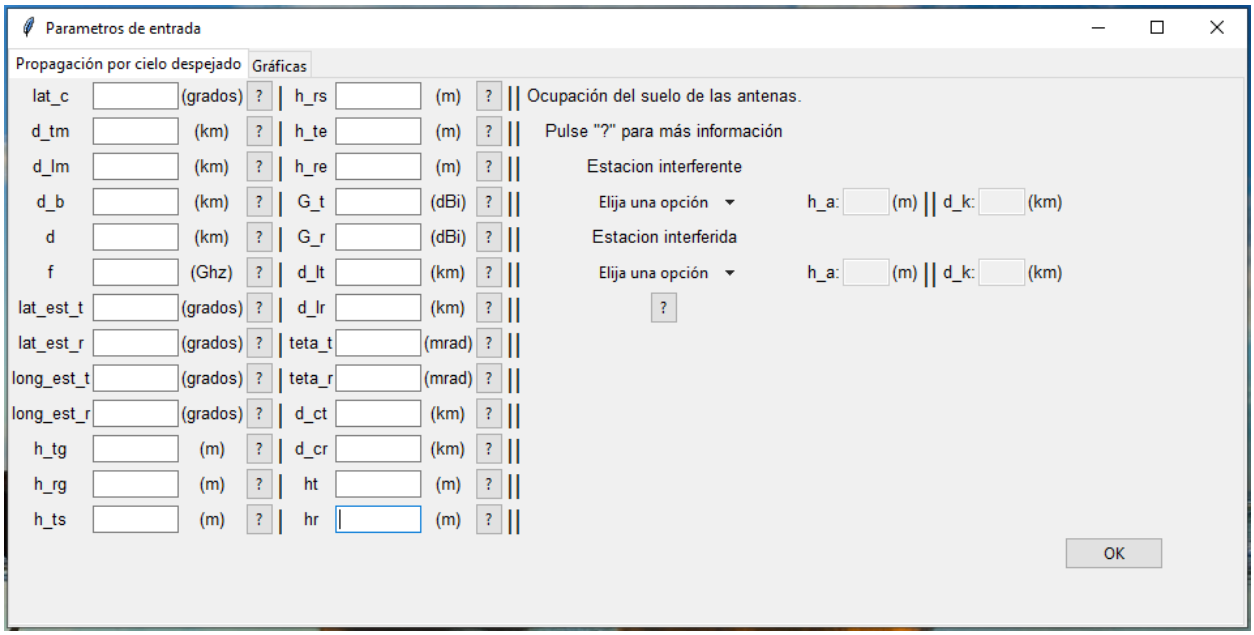


Figura 4.1. Ventana principal.

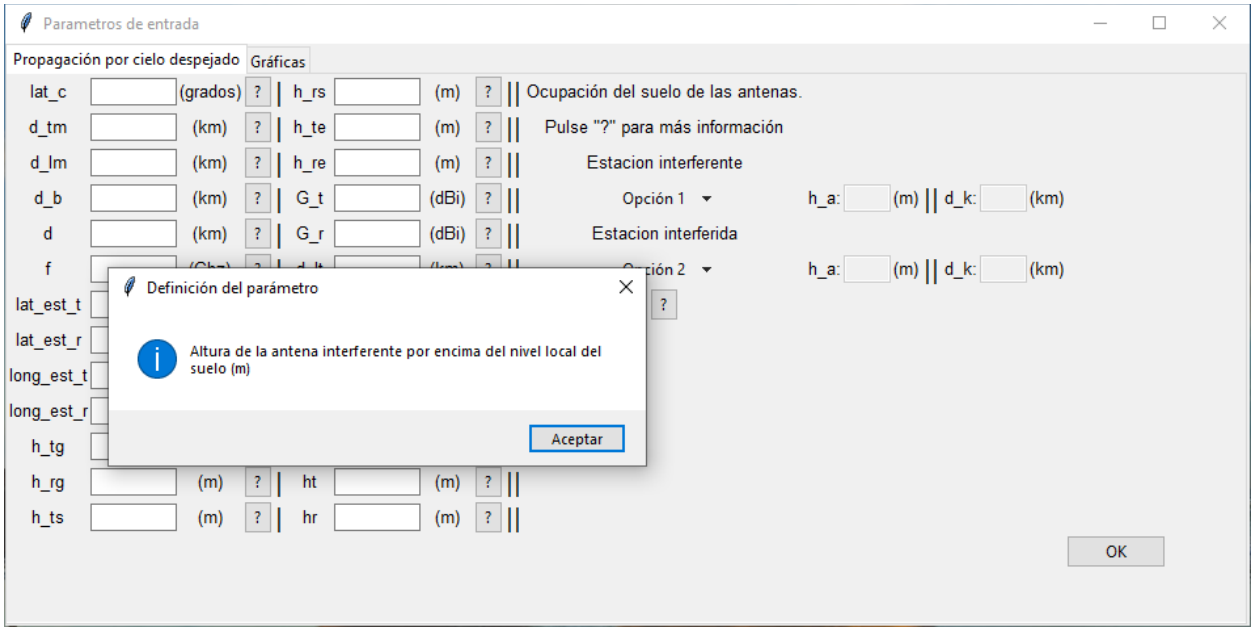


Figura 4.2. Ayuda para el parámetro "ht".

En cuanto a la parte derecha, está dedicada a la ocupación del suelo. En este caso hemos usado dos desplegables para que el usuario elija entre las diferentes opciones, dentro de las cuales está la de introducir el parámetro por cuadro de texto. En este caso el botón de ayuda proporcionado no ejecutará un cuadro de diálogo del tipo *menssagebox()*, sino más bien una foto explicativa, ya que es mucho más gráfico y fácil de entender. La foto emergente contiene la siguiente tabla:

Tabla 4.1 Alturas y distancias nominales de la ocupación del suelo

Opción	Categoría de ocupación del suelo (cobertura del terreno)	Altura nominal, h_a (m)	Distancia nominal, d_k (km)
1	Campos de cultivo alto Parques Árboles dispersos de forma irregular Cultivos (separación uniforme) Casas dispersas	4	0,1
2	Centro de población rural	5	0,07
3	Árboles de hoja caduca (separación irregular) Árboles de hoja caduca (separación uniforme) Bosque de árboles diversos	15	0,05
4	Coníferas (separación irregular) Coníferas (separación uniforme)	20	0,05
5	Selva tropical húmeda	20	0,03
6	Entorno suburbano	9	0,025
7	Entorno suburbano denso	12	0,02
8	Entorno urbano	20	0,02
9	Entorno urbano denso	25	0,02
10	Entorno urbano de edificios muy altos	35	0,02
11	Zona industrial	20	0,05

El cuadro de texto para introducir los parámetros *h_a* (altura nominal) y *d_k* (distancia nominal), a priori, estarán desactivados, ya que lo normal es elegir una de las opciones dadas. Pero en el caso de que necesitemos usarlo, bastará con elegir la opción "Introducir parámetros" y esta activará los cuadros de texto según la estación en la que estemos. En la siguiente imagen podemos observar con claridad como se activa el cuadro.

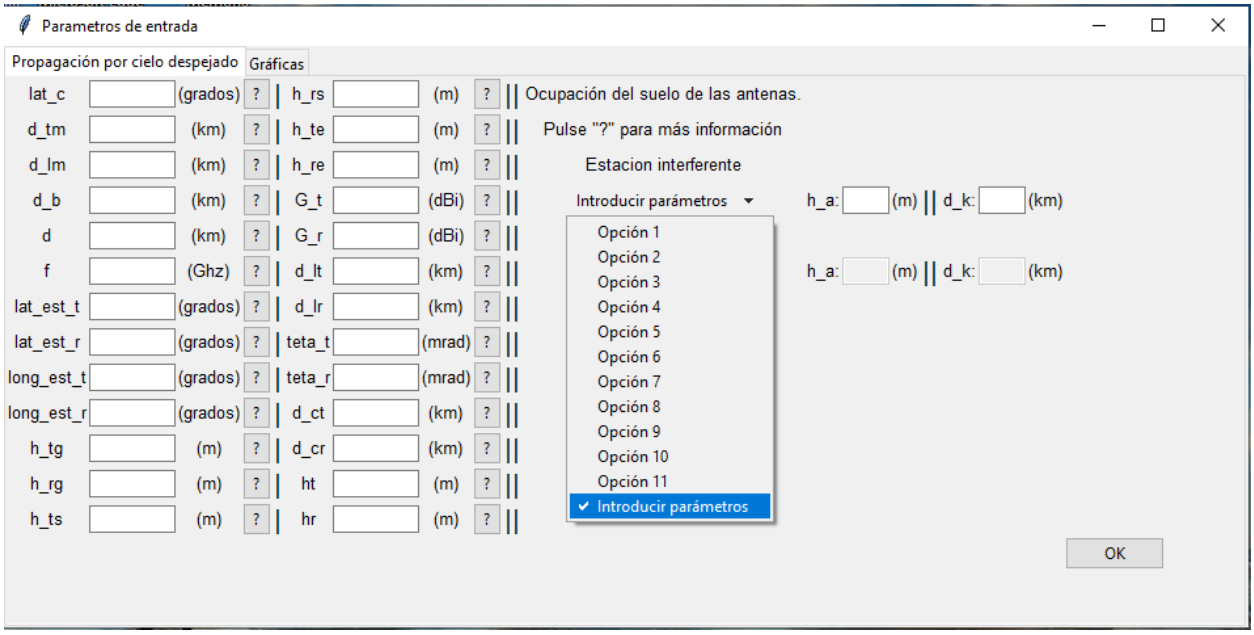


Figura 4.3. Desplegable para activar el cuadro de texto.

Una vez que tengamos todos los datos introducidos, presionaremos el botón "OK" y el programa

calculará la interferencia sufrida en el trayecto.

4.2 Pestaña secundaria o de gráficas

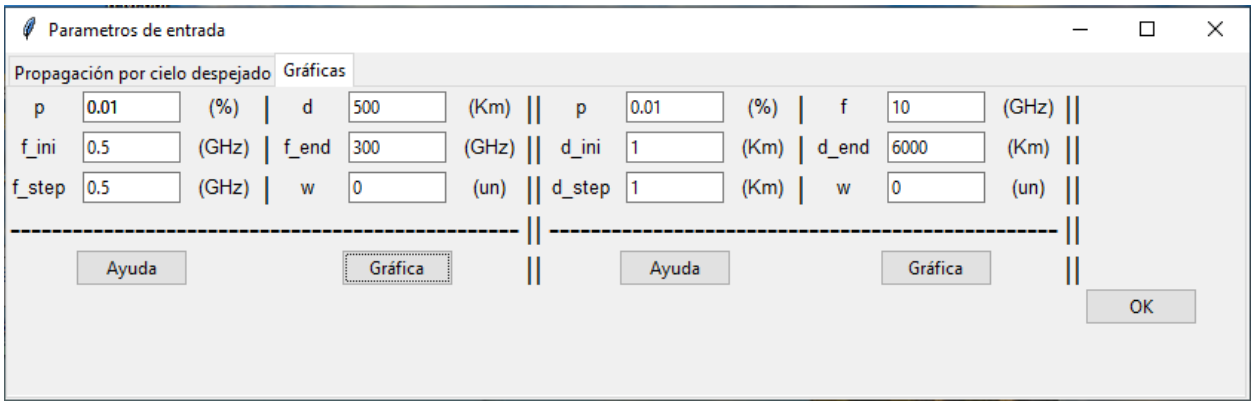


Figura 4.4. Pestaña de gráficas.

Pestaña secundaria, al igual que la primera, creada con el *widget* contenedor *"frame"*. Esta pestaña está dedicada a la representación de gráficas. En concreto dos. La primera representa la pérdida de transmisión básica debida a la propagación en espacio libre en función de la frecuencia y la segunda nos figurará la misma pérdida de transmisión, pero esta vez en función de la distancia. Ambos parámetros son dos de los más importantes en el cálculo de la interferencia entre dos estaciones.

El método de introducción de parámetros es exactamente igual que el de la pesataña principal. Además, como podemos ver en la Figura 4.4, cuenta con dos botones de ayuda (uno para cada gráfica) para disipar las posibles dudas generadas por los usuarios en relación con el significado de las posibles gráficas a representar.

Ambas gráficas se representan por separado. Sin embargo, conseguiremos una gran funcionalidad a nivel comparativo. Cada gráfica del mismo tipo se superpondrá, cambiando el color de cada una, en la misma figura de la anterior, siempre y cuando no se cierre la ventana de representación de la gráfica. Si cerramos esta ventana reiniciaremos el proceso.

Es decir, si quisiéramos comparar tres gráficas del primer tipo, bastaría con poner los datos necesarios en sus cuadros y seguidamente usar el botón "*Gráfica*". Este proceso se repetiría tres veces y ya tendríamos en la misma gráfica las tres curvas. Esto mismo ocurrirá si comparamos gráficas del tipo dos. En la Figura 4.5 podemos ver un ejemplo del resultado de dibujar varias gráficas con distintos parámetros inciales.

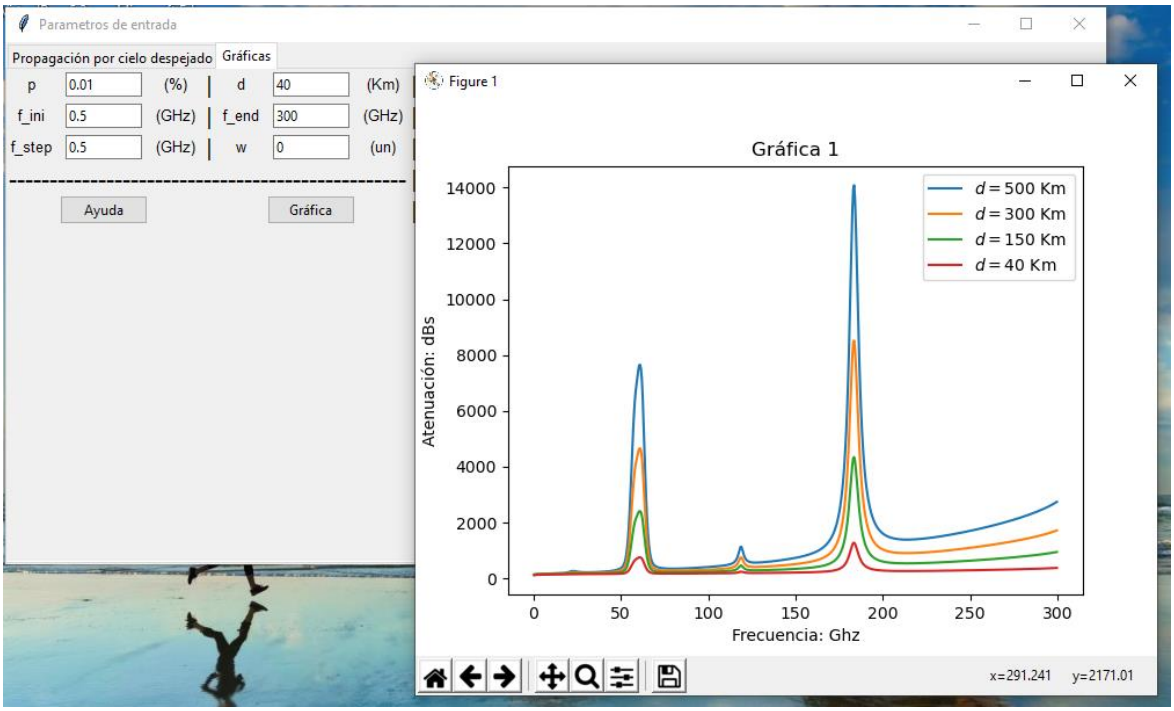


Figura 4.5. Ejemplo de gráficas.

Si nos es de utilidad las gráficas, pueden ser guardadas en distintos formatos según nos interese.

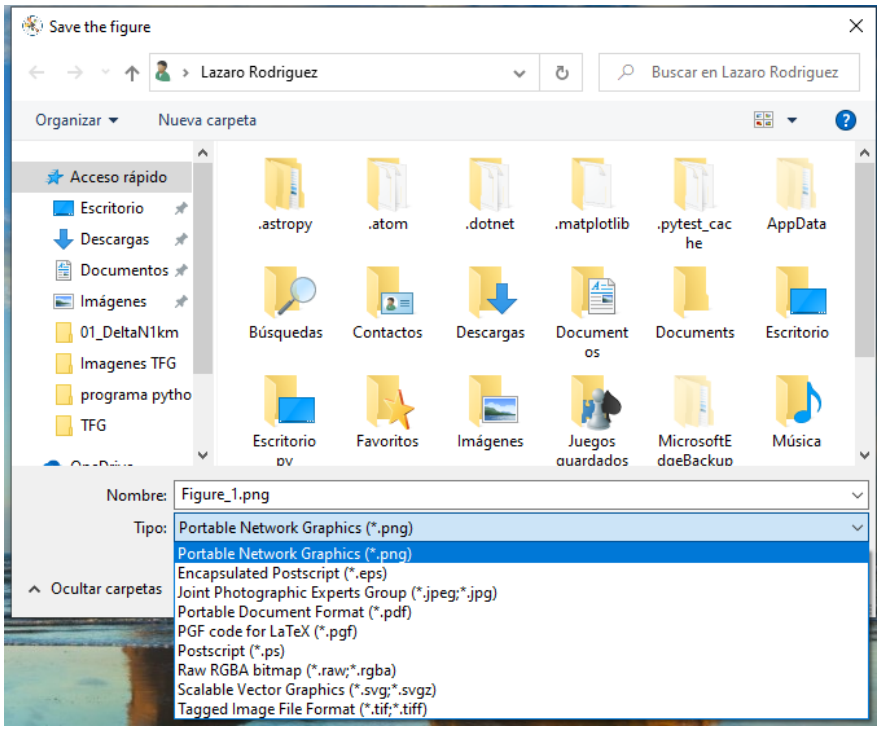


Figura 4.6. Ventana de selección de formato de las gráficas.

4.3 Ventanas emergentes de *Warnings*

Como puede ser normal, el usuario que incie por primera vez el *software* puede que no tenga el conocimiento de todas las especificaciones del programa, como está regulado o que simplemente cometa un error en la introducción de alguno de sus parámetros. Para evitar el cuelgue del programa en estas situaciones, se han previsto una serie de ventanas emergentes de modo que informe al usuario sobre el fallo cometido y su correspondiente solución. En este momento el programa se detendrá y esperará a la correcta introducción de estos parámetros para continuar con la ejecución de los cálculos. Estas ventanas han sido creadas con el *widget messagebox*.

Lo primero que se comprueba es que ningún elemento sea un campo vacio, es decir, que hayamos introducido valores a todos los parámetros de entrada. Si esto no fuese así un *warning* se encargará de avisarnos sobre nuestro error y no conseguiremos avanzar en el sistema hasta que el error no sea corregido.

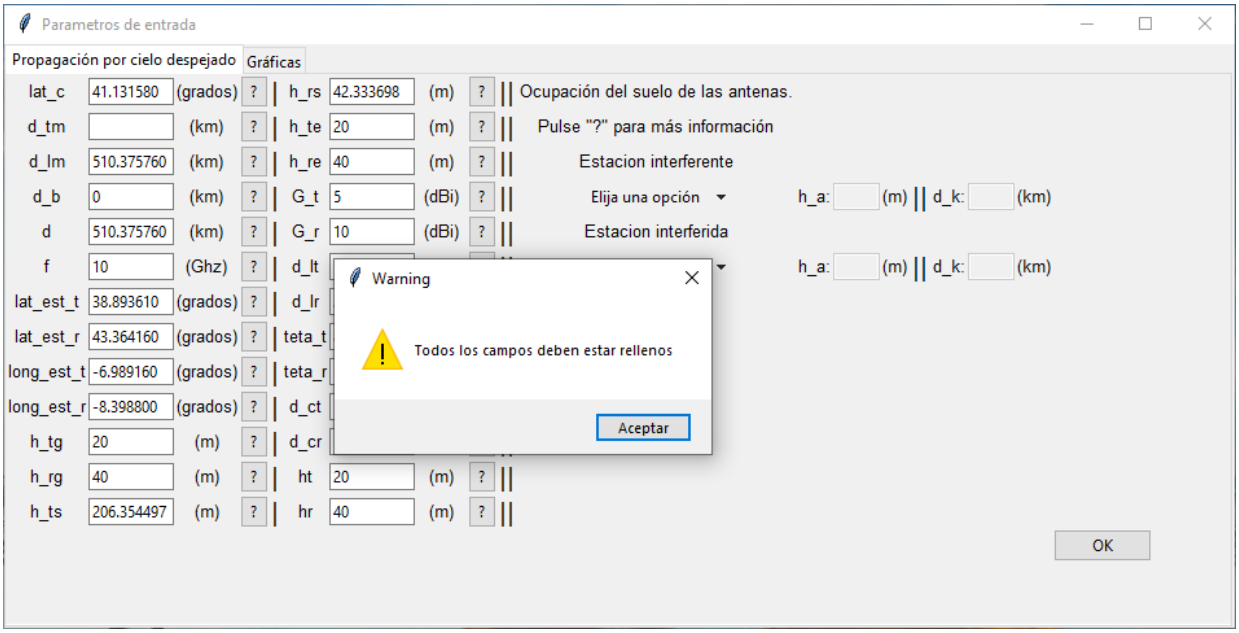


Figura 4.7. *Warning* debido a campos sin rellenar.

Una vez comprobado que todos los campos están rellenos se dispondrá a evaluar los diferentes parámetros. Y es en este punto donde se nos pueden generar varias ventanas de aviso.

Los primeros elementos a evaluar son los parámetros de caracter geográfico *lat_est_t, lat_est_r, long_est_t, long_est_r* y *lat_c*. Si alguno de estos parámetros no está bien introducido se mostrará una ventana indicando el rango que debe cumplir cada uno de ellos. La figura 4.7 nos muestra la respuesta del programa al introducir mal uno de estos campos.

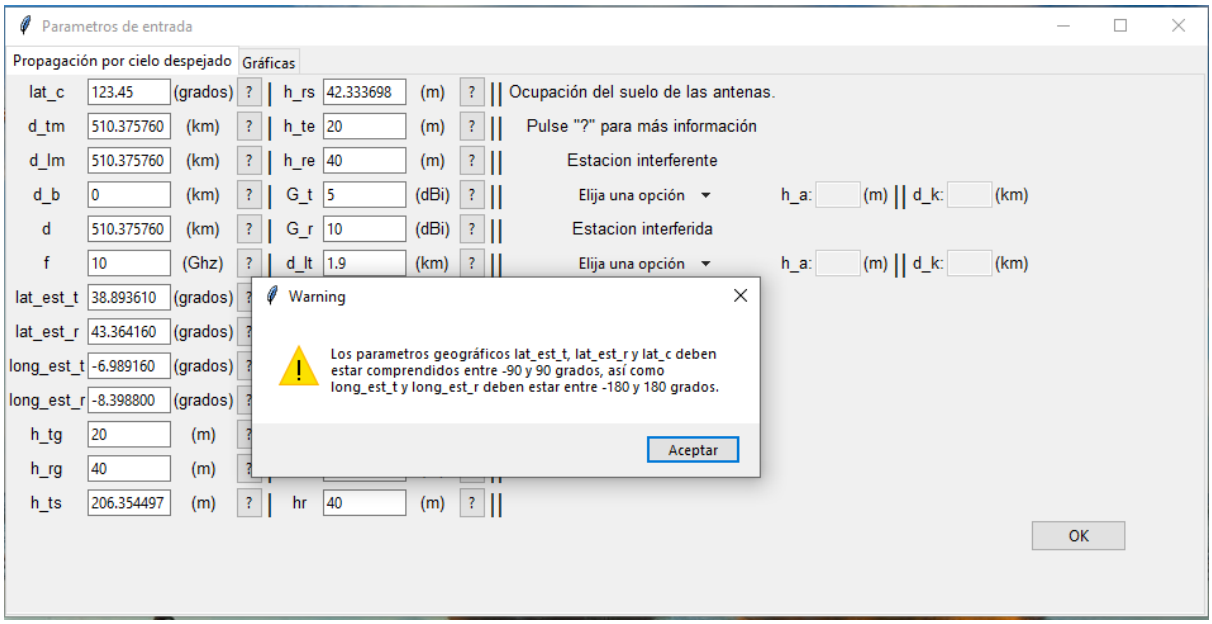


Figura 4.8. *Warning* debido a los parámetros geográficos.

No solo se evalúan los parámetros geográficos indicados anteriormente. Los parámetros *d* (distancia) y *f* (frecuencia), deben estar comprendidos en un determinado rango como lo indica la recomendación a la que hacemos referencia [1]. En el caso de la distancia, no debe superar los 10.000 Km y para la frecuencia el rango debe estar comprendido entre 0.1 y 50 GHz.

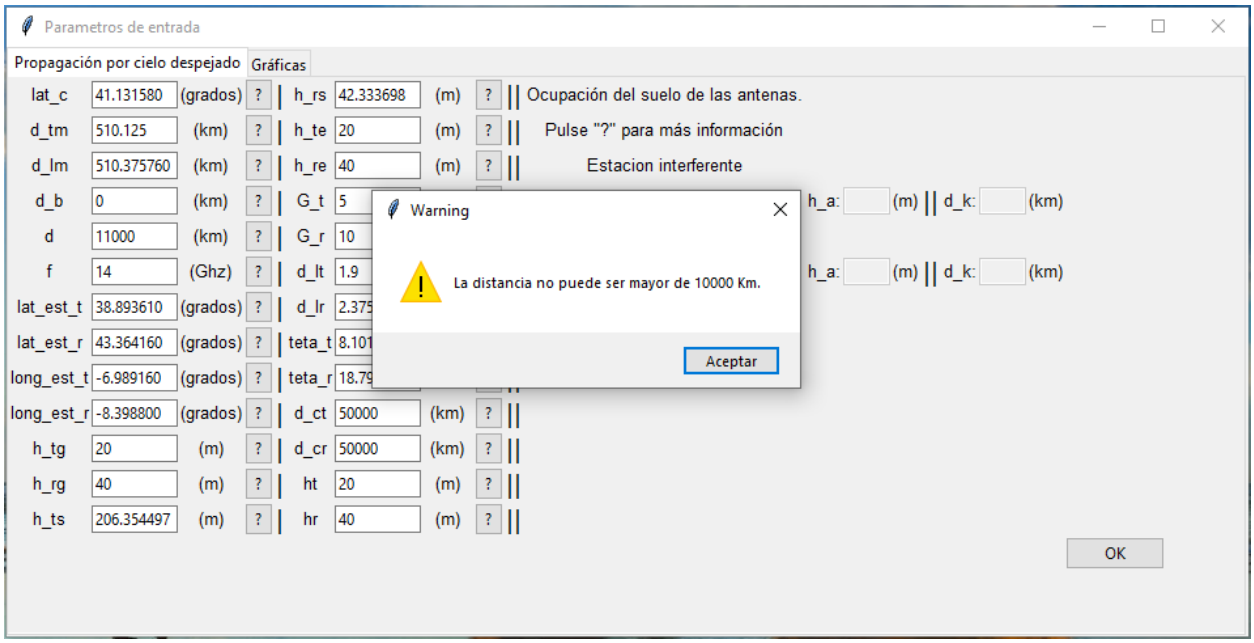


Figura 4.9. *Warning* debido al parámetro "*d*" fuera de rango.

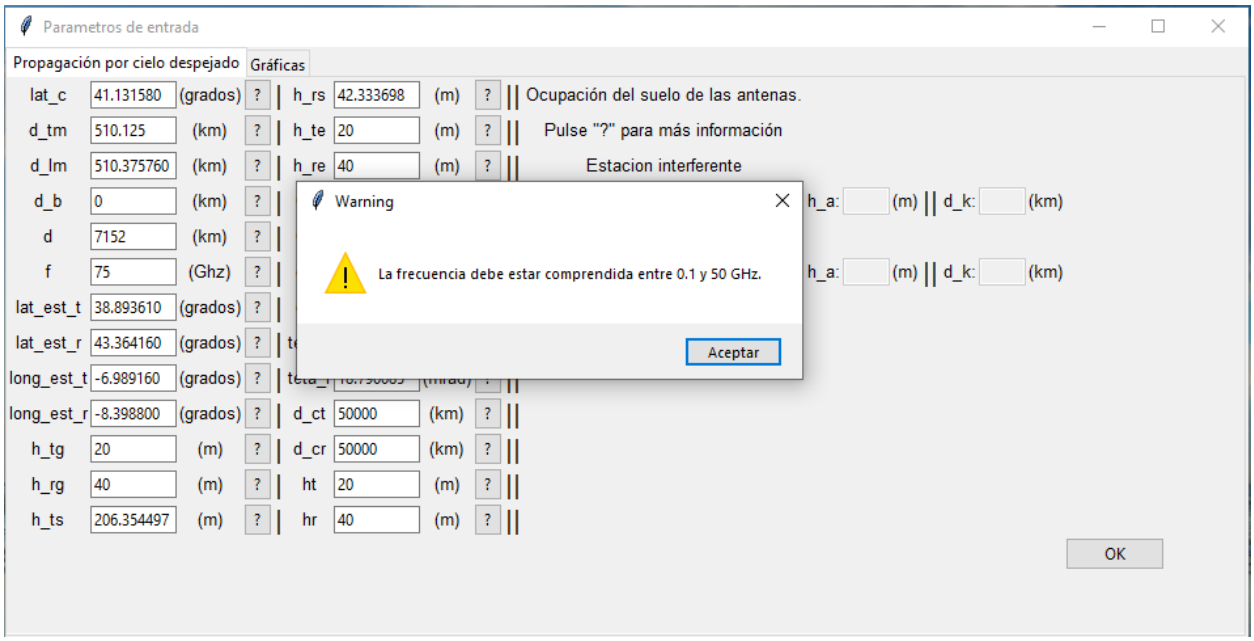


Figura 4.10. *Warning* debido al parámetro "*f*" fuera de rango.

También hemos añadido una ventana de aviso, como las anteriores, en la comprobación de la ocupación del suelo de las antenas. Si no seleccionamos ninguna opción de las sugeridas en el desplegable nos mostrará el aviso pertinente. De nuevo el programa se detiene hasta que elijamos una opción.

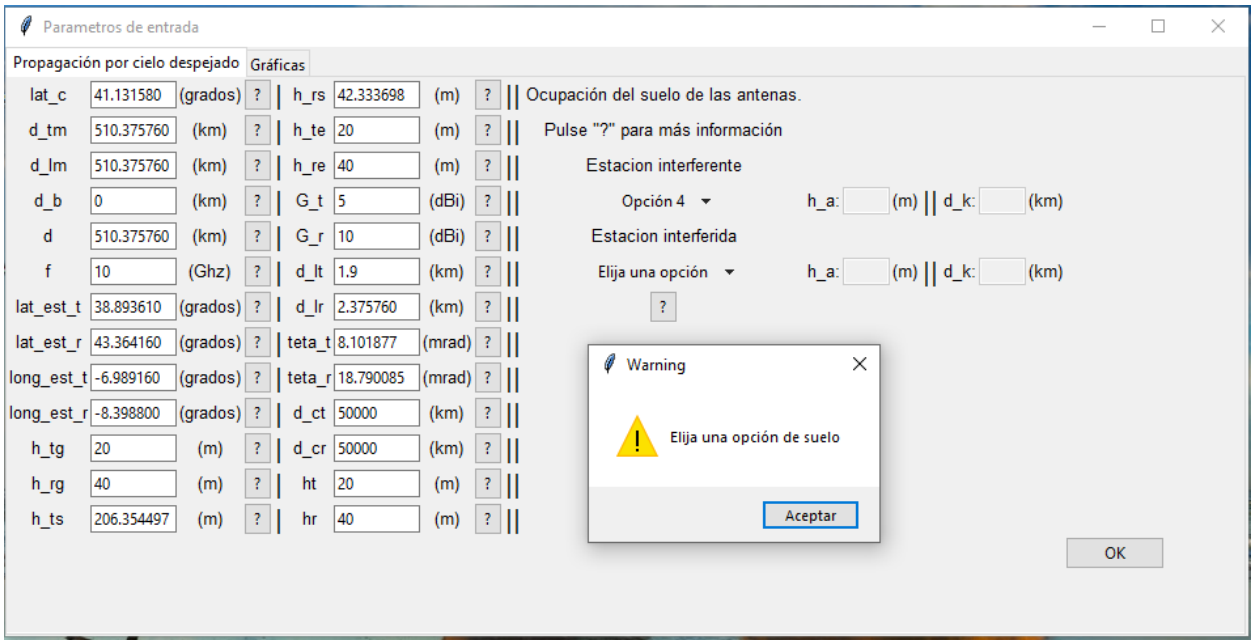


Figura 4.11. *Warning* debido a no elegir una opción en la ocupación del suelo de la estación interferida.

Evidentemente y como no podía ser menos, todas estas comprobaciones que se indican anteriormente y que están presentes en la pestaña principal también se mantienen en la pesataña de *Gráficas*. Por lo que de igual modo tendremos unos avisos por ventana emergente si no rellenamos algún campo, superamos en 10.000 Km la distancia o no elegimos la frencuancia dentro de su rango indicado.

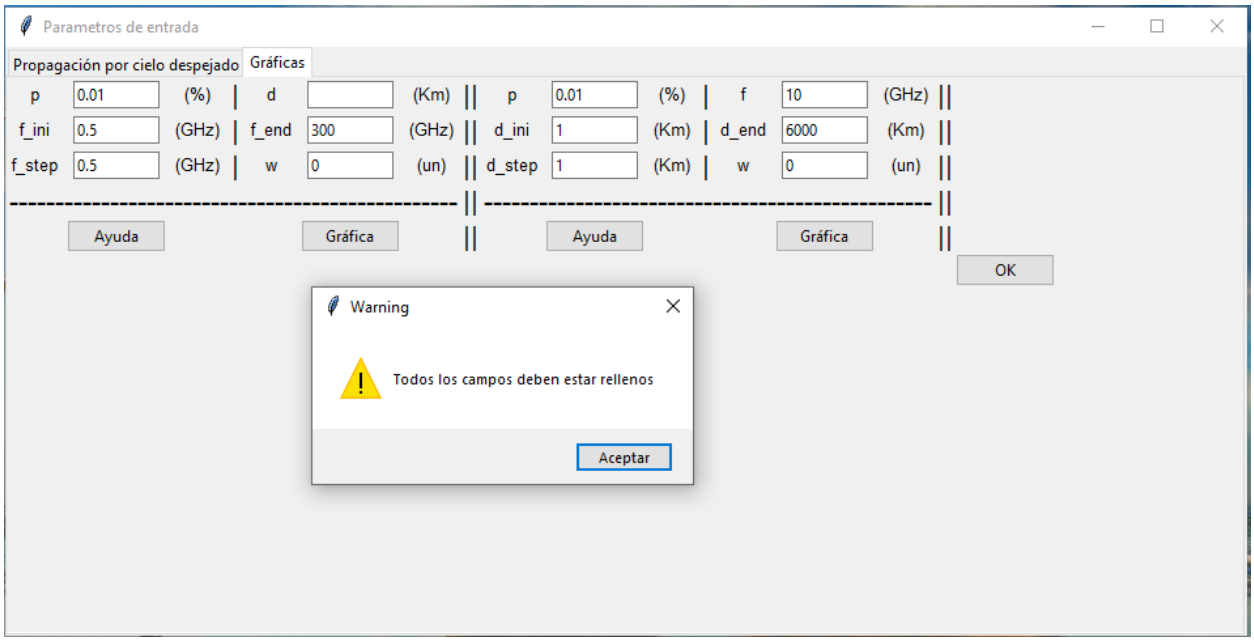


Figura 4.12. *Warning* en la pestaña de graficas (I).

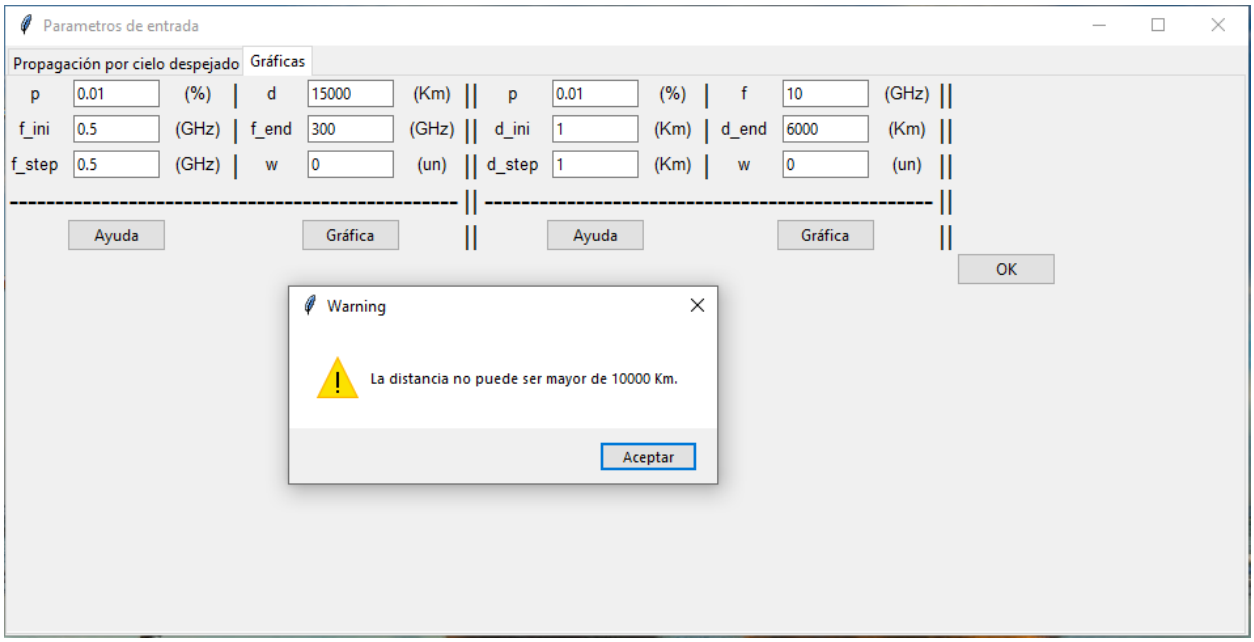


Figura 4.13. *Warning* en la pestaña de graficas (II).

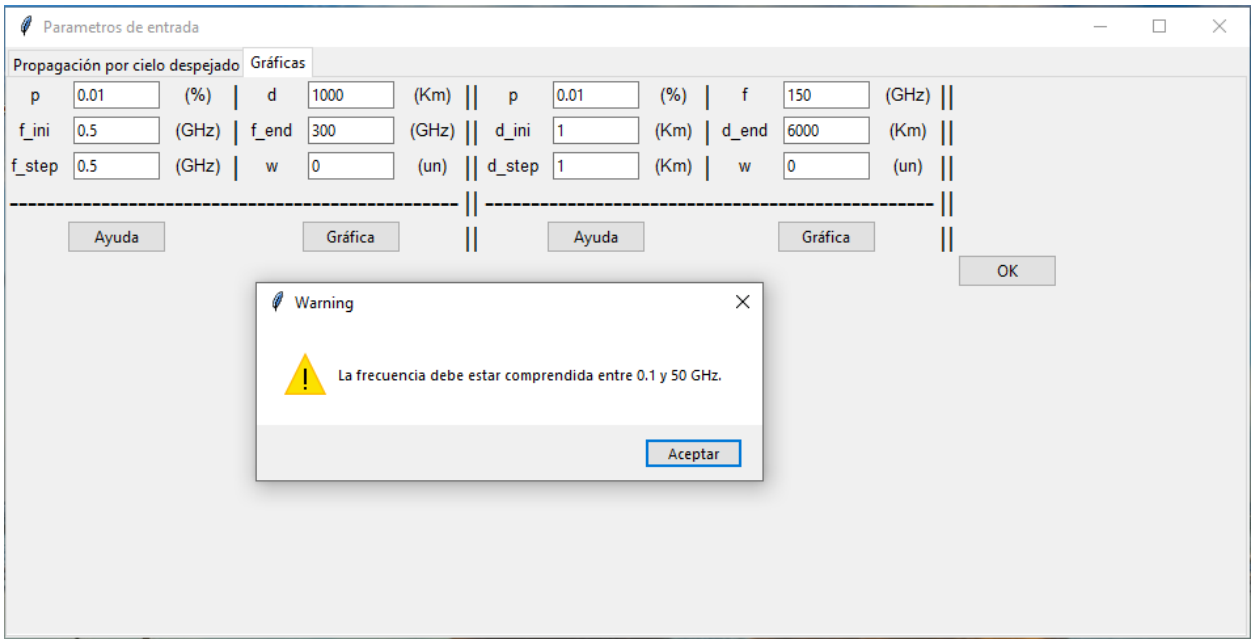


Figura 4.14. *Warning* en la pestaña de graficas (III).

4.4 Programa Python

Python es un lenguaje de programación muy potente y usado en la actualidad, con un gran crecimiento a nivel mundial, caracterizado por una gran flexibilidad y claridad que favorece a un código legible. Cuenta con una gran cantidad de librerías muy completas, ideales para implementar fórmulas y algoritmos matemáticos que nos facilitan la elaboración de programas.

4.4.1 Fichero programa.py

Este fichero marca el inicio de nuestro programa. Únicamente contiene la creación de la interfaz gráfica, de la cual haremos uso para introducir todos los valores iniciales, así como elegir las diferentes opciones de simulación y la llamada a una función "principal" (*abrirVentana*) en la que desarrollaremos el grueso de nuestro trabajo. Además, cuenta con la definición de todos los parámetros que usaremos a modo de comentario para una mayor aclaración de cada uno de ellos.

4.4.2 Fichero menu.py

Contiene nuestra función principal (*abrirVentana)* que usaremos para desarrollar nuestra interfaz gráfica y contendrá las llamadas a las distintas funciones para la simulación seleccionada.

Comenzamos importando las distintas librerías necesarias. A continuación, tenemos una línea de código muy importante, es la encargada de descargar los datos cartográficos, los anteriormente mencionados como datos SRTM.

Nuestra función principal *(abrirVentana()*) está compuesta por varias funciones a su vez. Primero definimos unos parámetros necesarios para la interfaz gráfica, como son la geometría, el titulo o las distintas pestañas que podremos seleccionar.

Funciones para usar en *abrirVentana()*:

1. Funciones ayuda (*clicked_num()*, siendo "num" un número entero del 1 al 28, *ayuda1()* y *ayuda2()*): estas funciones, las cuales llamaremos al pulsar sobre un botón de ayuda, nos permitirán conocer la definición de cada parámetro que debamos introducir en los valores iniciales. Son funciones

puramente informativas. En el caso de *ayuda1()* y *ayuda2()* están destinadas a la pestaña de gráficas como información de ayuda de uso.

2. Función *seleccionar1()* y *selccionar2()*: funciones cuyo uso es desbloquear o bloquear la entrada de un parámetro según la opción que hayamos elegido. Si deseamos que el valor de dicho parámetro sea uno de los valores estándar proporcionados, bloqueará la introducción del parámetro por el entorno gráfico. Si nuestra opción es introducirlo manualmente, nos desbloqueará el cuadro de entrada del parámetro.

3. Función *cierraPrincipal()*: función principal de la parte simulada de la propagación en cielo despejado. Lo primero que haremos será una pequeña comprobación de parámetros a modo de evitar un cuelgue del programa. Se comprobará que todos los campos de introducción de parámetros no estén vacios. Seguido, se transformarán dichos parámetros en variables para su uso en la ejecución del programa. Una vez que tenemos todos los parámetros en variables y los campos no son nulos, comprobaremos que aquellos parámetros que necesariamente tengan que estar acotados estén correctamente dentro de su rango de uso.

   Puesto que los últimos parámetros que comprobamos son los referentes a la ocupación del suelo y con motivo de que el código sea lo mas claro y legible posible, el primer cálculo que haremos será el de las pérdidas adicionales por la ocupación del suelo, así todo el código referente a esto estará seguido y en la misma sección.

   La siguiente sección de código viene referida a la difracción, donde le daremos uso al paquete Pycraf ya descrito anteriormente. Inicializamos párametros, llamamos a la función *pathprof.PathProp()* para analizar el trayecto y de esta forma podremos calcular la difracción con la función implementada en la librería *pathprof.loss_diffraction()*.

   Una vez llegados a este punto, calcularemos algunos parámetros necesarios para continuar con nuestra simulación como son el radio efectivo de la tierra, los parámetros *mu_x* (siendo x un valor del 1-4), *beta_0* o la corrección por la rugosidad del terreno.

   Con todos los parámetros ya calculados, hacemos las llamadas a las distintas funciones que nos calcularán el resto de las atenuaciones sufridas (atenuación por visibilidad directa, difracción, dispersión troposférica y por conductos y reflexión en las capas). Por último y, en cuanto a cálculos se refiere, calcularemos la interfrencia total según el procedimiento de la recomendación.

4. Función *cierraSecundaria()*: esta función tiene como único objetivo cerrar el programa.

5. Función *grafica1()*: esta función genera una gráfica que representa la pérdida de transmisión básica debida a la propagación en espacio libre en función de la frecuencia. Primero se comprueba que los campos no estén vacíos y que los parámetros de entrada no estén fuera de rango. Una vez hecha las comprobaciones pertinentes se ejecuta el código necesario para a representación de la gráfica.

6. Función *grafica2()*: esta función genera una gráfica que representa la pérdida de transmisión básica debida a la propagación en espacio libre en función de la distancia. Al igual que en la función anterior, se comprueba que los campos no estén vacíos, que los parámetros de entrada no estén fuera de rango y se ejecuta el código necesario para a representación gráfica.

7. Todo el código restante pertenece a la creación, configuración y uso de los distintos *widgets* para la representación de nuestra interfaz gráfica.

   Para la creación de las etiquetas de texto hemos usado el *widget Label()* que se complementa junto con el *widget Entry()* y formaremos cada linea de parámetros de entrada. El formato sería:

   - *Label – Entry – Label*. Lo que equivale en el entorno gráfico al nombre del parámetro, cuadro de introducción de texto y unidad del parámetro.

Además de estos *widgets*, hay que añadirles el *Buttom()*, con "?" como texto de botón, al que le adjudicaremos cada función de ayuda.

Para la parte derecha de la pestaña principal, aparte de los *widgets Label()* y *Entry()*, ya mencionados anteriormente, hemos hecho uso del elemento *OptionMenu()* para la creación de las dos listas de opciones que proporcionamos.

En cuanto a la pestaña de Gráficas, debemos comentar que los *widgets* usados son los mismos descritos anteriormente, *Label()* para las etiquetas de texto, *Entry()* para los campos con cuadro de texto y *Button()* para la asignación de eventos.

4.4.3 Fichero funciones.py

Este fichero contiene todas aquellas funciones que debemos ir usando para el desarrollo de nuestro programa. Las agrupamos en otro fichero para una mayor organización y comprensión del código. Como en todos los ficheros anteriores, lo primero es añadir las distintas librerías y paquetes que usemos.

Funciones disponibles:

1. *calcula_beta (lat_c, mu_1, mu_4)*: función que calcula la "beta_0" a partir de la latitud del centro de trayecto y de dos parámetros calculados al inicio de *cierraPrincipal()*. Este parámetro es muy importante y tendrá una gran influencia en la simulación.

2. *calcula_absorcion_gaseosa(f, d, rho)*: función que nos permite calcular la absorción gaseosa dada una frecuencia, la distancia del trayecto y la densidad de vapor de agua. Esta función a su vez, usa una función contenida en una de las librerías de la ITU-R, importadas para el cálculo total de la absorción gaseosa (*itur.models.itu676.gamma_exact(f, P, rho, T),* siendo "P" la presión atmosférica y "T" la temperatura).

3. *calcula_propagacion_visibilidad_directa(p_o_beta, f, w, d, d_lt, d_lr)*: función que calcula la interferencia producida por una antena con la que comparte visibilidad directa. Esta función depende del primer argumento que le pasemos, "*p"* (% de tiempo requerido durante el cual no se rebasa la perdida básica de transmisión) o "beta". Los parámetros siguientes requeridos respectivamente son la frecuencia, la fracción del trayecto total sobre el agua, la distancia del trayecto a lo largo del círculo máximo y la distancia desde las antenas de transmisión y recepción a sus respectivos horizontes.

4. *calcula_dispersion_troposferica(d, f, G_t, G_r, teta, p, N0)*: función que nos permite calcular las pérdidas básicas de transmisión debida a la dispersión troposférica. Dicha función usa como argumentos de entrada la distancia del trayecto a lo largo del círculo máximo, la frecuencia, la ganancia de ambas antenas, la distancia angular del trayecto, *p* (parámetro comentado anteriormente) y la refractividad de la superficie a nivel del mar.

5. *calcula_propagacion_conductos_reflexion(d_lt, d_lr, f, teta_t, teta_r, w, d_ct, d_cr, h_ts, h_rs, a_e, d, p, beta_0, mu_2, mu_3)*: esta función calcula las pérdidas básicas de transmisión debidas a la propagación por conductos y reflexión en las capas. Entre sus argumentos de entrada destacamos aquellos que no hemos mencionado con anterioridad como son: los ángulos de elevación de las antenas de transmisión y recepción respecto del horizonte (*teta_t* y *teta_r,*), distancia sobre tierra desde las antenas hasta la costa a lo largo del trayecto de círculo máximo de la interferencia (*d_ct* y *d_cr*), altura del centro de la antena interferente e interferida sobre el nivel del mar (*h_ts* y *h_rs*), el valor mediano del radio efectivo de la Tierra (*a_e*) y dos parámetros calculados en la función *cierraPrincipal()* como son *mu_2* y *mu_3.*

5 CASOS PRÁCTICOS Y COMPARACIÓN DE RESULTADOS

Para terminar, vamos a presentar algunos ejemplos prácticos de estaciones terrenas existentes a las que les calcularemos la interferencia sufrida y comprobaremos los resultados con el fin de asegurarnos el correcto funcionamiento de nuestro programa. Para la comprobación de resultados usaremos el paquete Pycraf.

5.1 Caso 1

En este primer caso vamos a usar una estación situada en Sevilla y vamos a suponer que queremos instalar otra estación cerca de la ciudad de Córdoba. Veremos que interferencia puden sufrir.

- Sevilla: latitud = 37.345555, longitud = -5.971388
- Córdoba: latitud = 37.877222, longitud = -4.787777

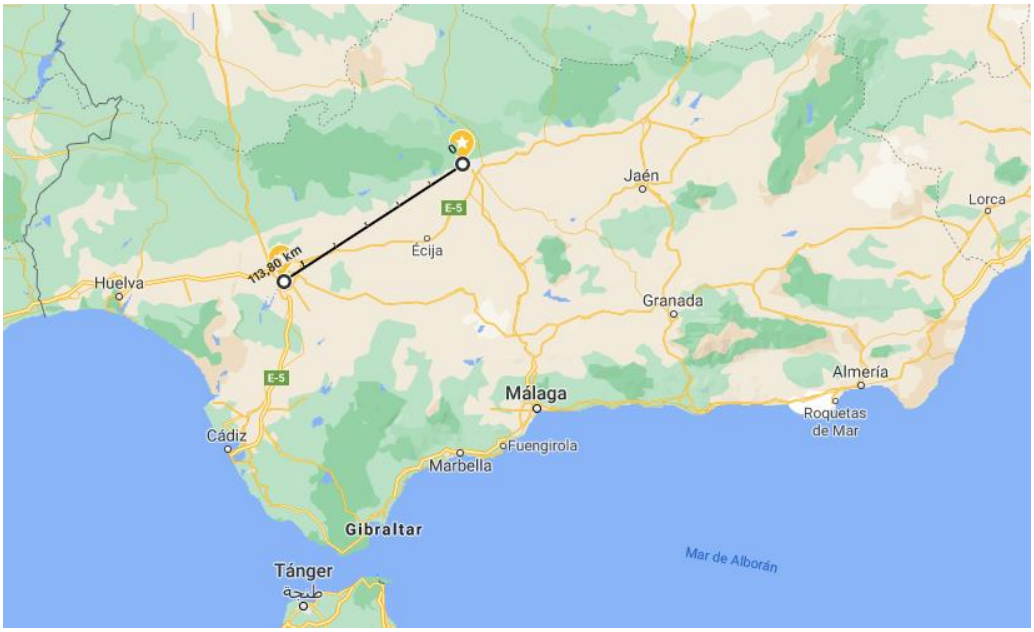


Figura 5.1. Enlace caso 1.

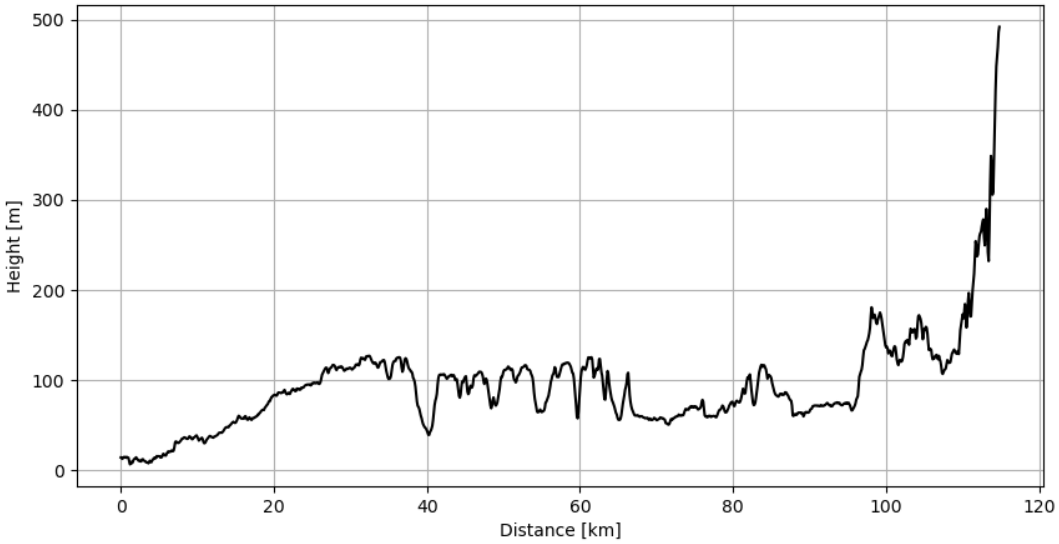


Figura 5.2. Perfil del trayecto caso 1.

Para este caso vamos a utilizar los datos mostrados en la Figura 5.2.

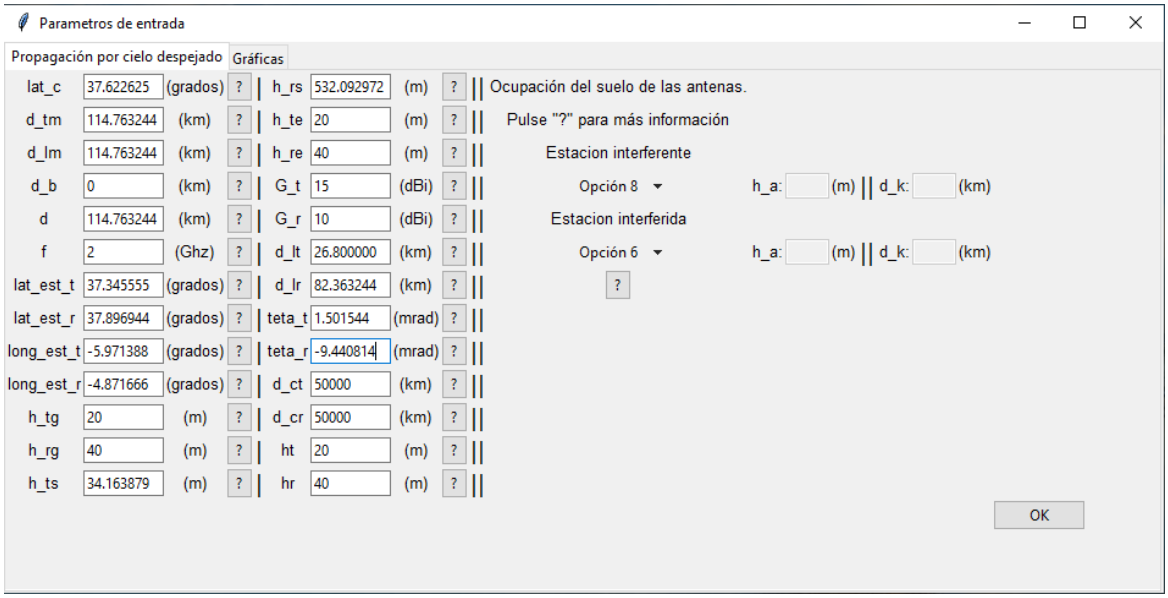


Figura 5.3. Datos de entrada para el caso 1.

Con estos datos de partida nuestro programa nos da un resultado de la interferencia de:

```
C:\Users\lazar\Escritorio\programa python>programa.py
L =  112.0412  dBs
```

Figura 5.4. Resultado caso 1.

Para comprobar que nuestros resultados son correctos, vamos a usar el módulo Pycraf con los mismos datos de entrada y así verificar que nuestro resultado se asemeja a lo deseable. En la Figura 5.5 podemos

observar el código fuente para el uso de Pycraf que equivale al uso de nuestro *software*.

```python
from astropy import units as u
from pycraf import pathprof, conversions as cnv

pathprof.SrtmConf.set(download='missing')
freq = 2 * u.GHz
lon_tx, lat_tx = -5.971388 * u.deg, 37.345555 * u.deg
lon_rx, lat_rx = -4.871666 * u.deg, 37.896944 * u.deg
hprof_step = 100 * u.m  # resolution of height profile
temperature = 290. * u.K
pressure = 1013. * u.hPa
time_percent = 0.01 * u.percent  # see P.452 for explanation
h_tg, h_rg = 20 * u.m, 40 * u.m
G_t, G_r = 15 * cnv.dBi, 10 * cnv.dBi
# clutter zones
zone_t, zone_r = pathprof.CLUTTER.URBAN, pathprof.CLUTTER.SUBURBAN
pprop = pathprof.PathProp(
     freq,
     temperature, pressure,
     lon_tx, lat_tx,
     lon_rx, lat_rx,
     h_tg, h_rg,
     hprof_step,
     time_percent,
     zone_t=zone_t, zone_r=zone_r,
     )
print("L = ","{0:.4f}".format(pathprof.loss_complete(pprop, G_t, G_r)[6]))
```

Figura 5.5. Módulo Pycraf para el caso 1.

Como resultado de este código tenemos:

```
C:\Users\lazar\Escritorio\programa python>pruebas2.py
L =  112.0382 dB
```

Figura 5.6. Resultado de la simulación con el módulo Pycraf para el caso 1.

Como podemos observar, el resultado es muy similar al obtenido en nuestros cálculos. Los decimales en los que varían ambas simulaciones son prácticamente despreciables, por lo que podemos eventuar que nuestro programa simula bastante bien la interferencia sufrida en las estaciones.

Como contenido adicional y usando los datos de este mismo ejemplo, hemos elaborado una gráfica con la intención de comparar los resultados al variar la frecuencia de trabajo.

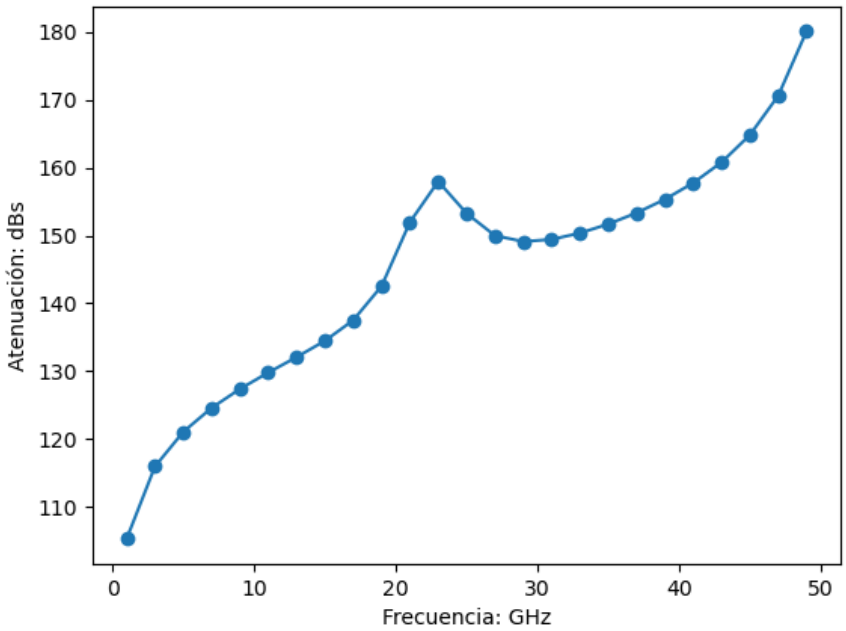


Figura 5.7. Gráfica caso 1.

Como podemos observar, la influencia de la frecuencia es muy grande, a medida que aunmentamos la frecuencia, la interferencia aumenta. Cabe destacar el pico en torno a los 25 GHz en los que la interfrencia se dis para.

5.2 Caso 2

Para nuestro segundo caso vamos a usar de nuevo nuestra estación terrena de Sevilla, utilizada en el caso anterior y como emplazamiento de la estación receptora, una localización cerca de la ciudad de Jerez de la Frontera. En este caso las latitudes y longitudes de estas estaciones se corresponden con:

- Sevilla: latitud = 37.345555, longitud = -5.971388
- Jerez de la Frontera: latitud = 36.739112, longitud = -6.131243

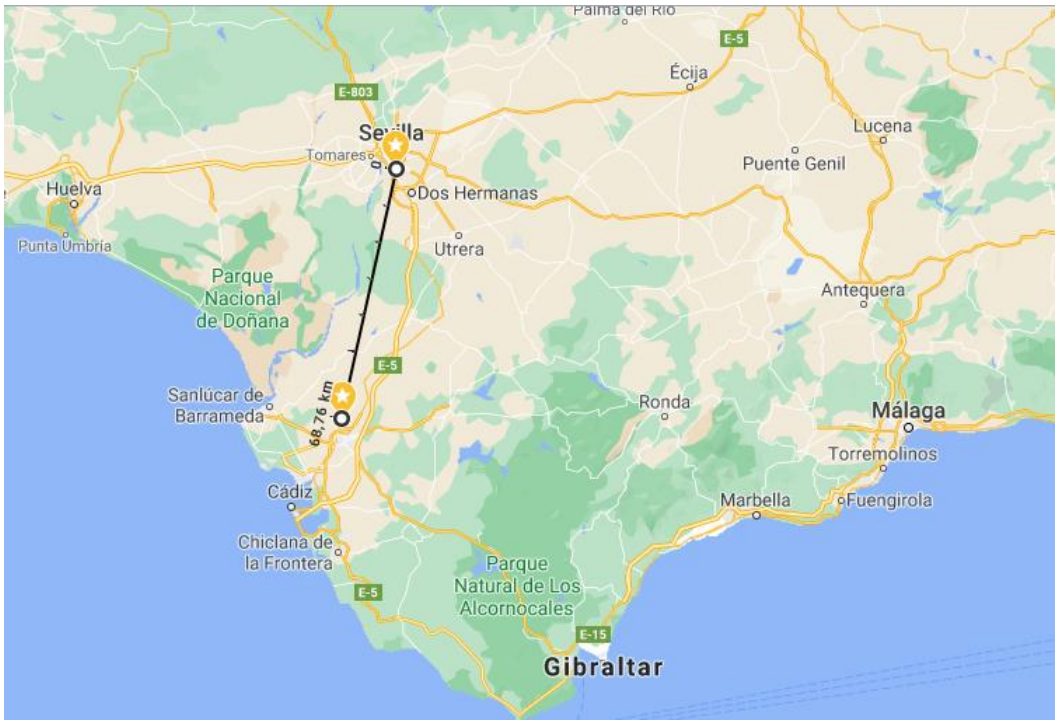


Figura 5.8. Enlace caso 2.

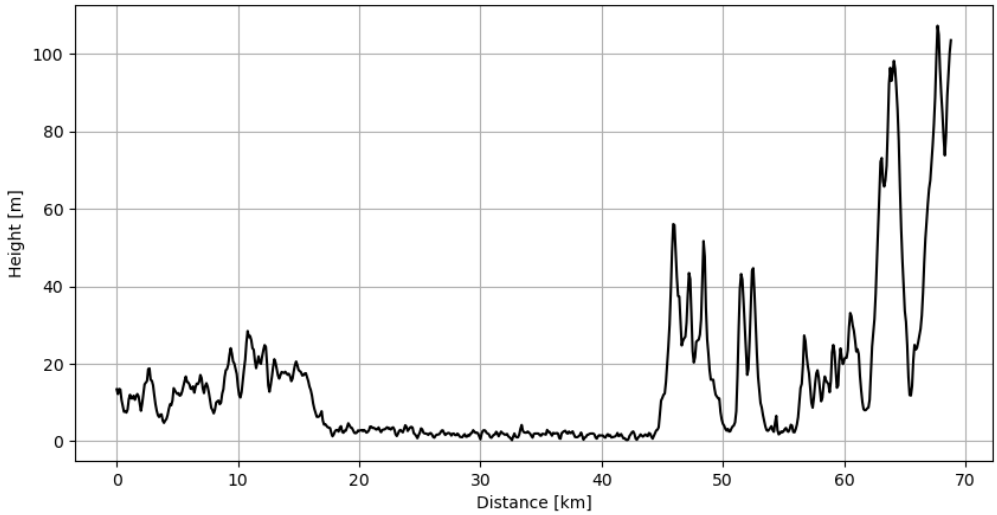


Figura 5.9. Perfil del trayecto caso 2.

Para nuestro caso número dos, los datos inicales introducidos son los ilustrados en la Figura 5.10.

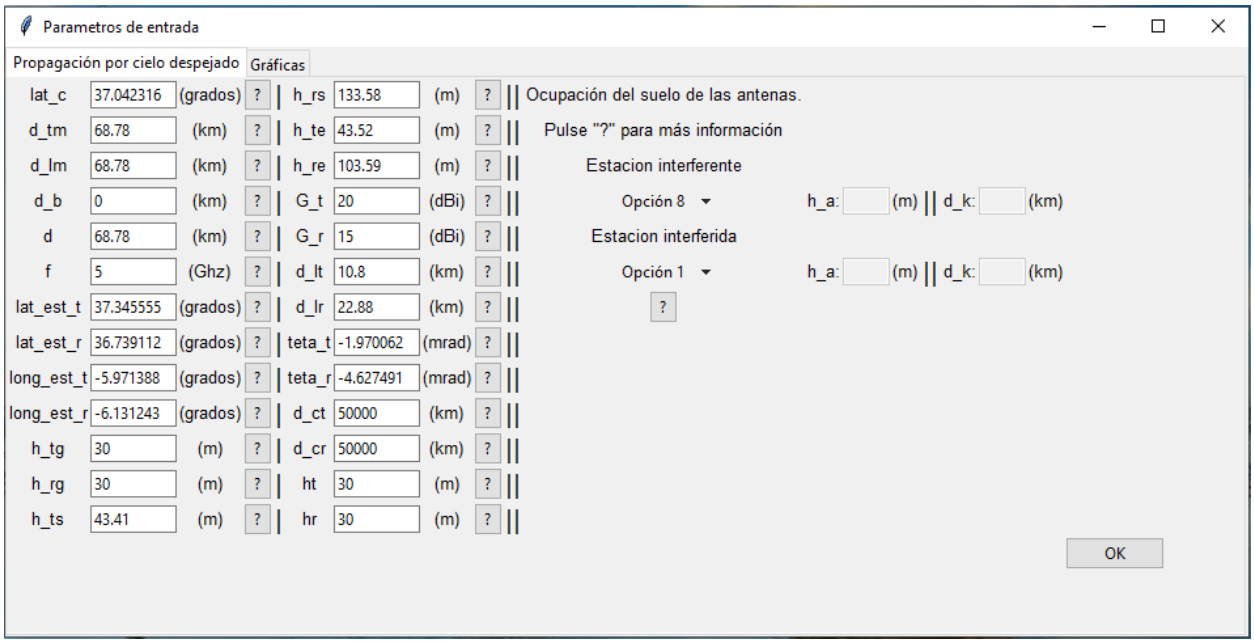


Figura 5.10. Datos de entrada para el caso 2.

Estos datos nos proporcionan un nivel de interferencia de:

```
C:\Users\lazar\Escritorio\programa python>programa.py
L =  99.0707  dBs
```

Figura 5.11. Resultado caso 2.

De nuevo para contrastar los resultados obtenidos en nuestro trabajo vamos a usar el módulo Pycraf. En la Figura 5.12 tenemos el código usado para este objetivo.

```python
from astropy import units as u
from pycraf import pathprof, conversions as cnv

pathprof.SrtmConf.set(download='missing')
freq = 5 * u.GHz
lon_tx, lat_tx = -5.971388 * u.deg, 37.345555 * u.deg
lon_rx, lat_rx = -6.131243 * u.deg, 36.739112 * u.deg
hprof_step = 100 * u.m  # resolution of height profile
temperature = 290. * u.K
pressure = 1013. * u.hPa
time_percent = 0.01 * u.percent  # see P.452 for explanation
h_tg, h_rg = 30 * u.m, 30 * u.m
G_t, G_r = 20 * cnv.dBi, 15 * cnv.dBi
# clutter zones
zone_t, zone_r = pathprof.CLUTTER.URBAN, pathprof.CLUTTER.SPARSE
pprop = pathprof.PathProp(
    freq,
    temperature, pressure,
    lon_tx, lat_tx,
    lon_rx, lat_rx,
    h_tg, h_rg,
    hprof_step,
    time_percent,
    zone_t=zone_t, zone_r=zone_r,
    )
print(pprop)
print("L = ","{0:.4f}".format(pathprof.loss_complete(pprop, G_t, G_r)[6]))
```

Figura 5.12. Módulo Pycraf para el caso 2.

El código anterior da como resultado una interferencia de:

```
C:\Users\lazar\Escritorio\programa python>pruebas2.py
L =  99.0704 dB
```

Figura 5.13. Resultado de la simulación con el módulo Pycraf para el caso 2.

Como podemos observar, en nuestro segundo ejemplo, el resultado obtenido con el uso del módulo de Pycraf es muy parecido al obtenido por nuestro programa. De nuevo podemos afirmar que nuestro *software* cumple perfectamente los objetivos tratados.

Vamos a volver a comparar nuestros resultados en función de la frecuencia. La gráfica de la Figura 5.14 nos muestra justamente esto.

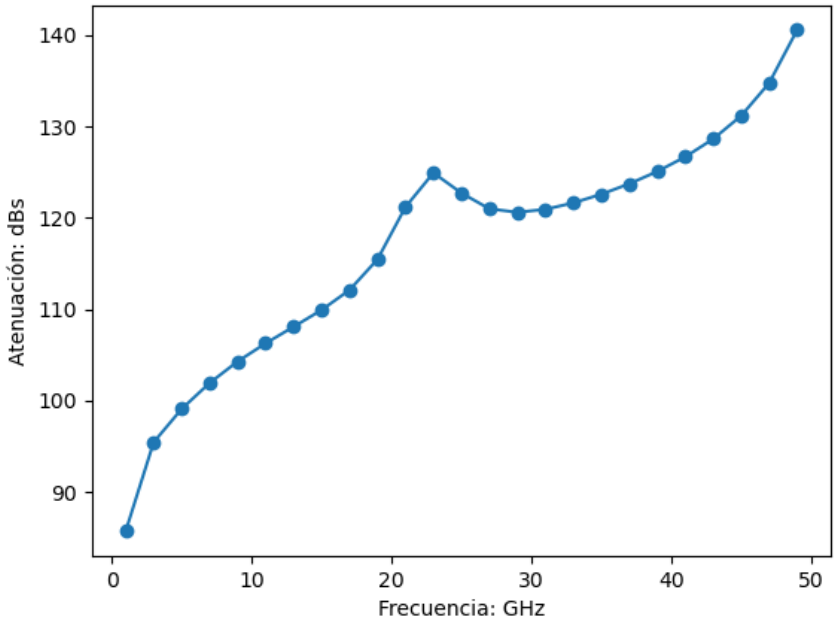


Figura 5.14. Gráfica caso 2.

6 CONCLUSIONES Y FUTURAS LINEAS DE TRABAJO

Durante el duro y largo camino recorrido para la realización de este Proyecto Fin de Grado he conseguido varios objetivos. El trabajo ha resultado de la unión de las intensificaciones de Sistemas de Telecomunicación y Telemática. Esto me ha ayudado a expandir mis conocimientos en ambas áreas, para superar los objetivos propuestos en el trabajo.

Lo primero a lo que tuve que hacer frente es a la perfecta comprensión de la recomendación en la que está basado el proyecto. En algunos apartados no estaban claros los procedimientos que debíamos seguir para los cálculos. Estas dudas fueron resueltas sin problemas con la ayuda de mi tutora, Susana, y la búsqueda de información adicional.

Una vez comprendido por completo todos los procedimientos que se llevaban a cabo, la dificultad se centraba en los conocimientos en el lenguaje de programación Python, los cuales he afianzado y, además, comprobado que he mejorado hasta el punto de conseguir un código robusto, bien estructurado, claro y lo más importante, que cumple con los objetivos. La dificultad se basaba en el entendimiento y funcionamiento de las distintas librerías necesarias usadas y que requerían de un pequeño trabajo de investigación. Tengo que destacar que, en lo que a la programación se refiere, suponía todo un reto para mí ya que era el ámbito en el que me sentía menos seguro dentro de la carrera. Creo que lo supe afrontar y con gran resultado.

Otro de los grandes problemas fue la intregracion de parámetros esenciales de tipo cartográficos, como podían ser las alturas de los perfiles, distancias o parámetros dependientes de estos mismos. Tras varios días de trabajo y búsqueda, dimos con la respuesta. El módulo Pycraf, el cual hemos descrito brevemente en capítulos anteriores, nos permitía integrar dichos parámetros y, de una forma fácil, acceder a ellos para poder manipularlos debidamente.

Una vez resueltos estos problemas la realización del proyecto fue cuestión de dedicarle tiempo y trabajo.

Como líneas de trabajo futuras, podemos destacar una serie de áreas que podrían complementar perfectamente con mi trabajo:

- Este trabajo parte de la Recomendación P.452-16, pero se centra solo en la parte de la predicción de la interferencia en cielo despejado. Esto es básicamente la mitad de la recomendación. Como trabajo complementario, se podría mencionar la implementación en el *software* el cálculo para de la predicción de la interferencia en presencia de hidrometeoros.
- Tanto este trabajo como la posible implementación comentada anteriormente, podríamos tratarlos como módulos independientes y trabajar sobre un simulador completo general que implemente todos estos módulos basados en las recomendaciondes de la ITU-R.
- Podríamos trabajar también con la implementación de un *sofware* que pueda ser capaz de

sacar parámetros cartográficos sin necesidad de usar fuentes externas.

- Mejorar la interfáz gráfica con paquetes externos más potentes.

Por último, me gustaría comentar la satisfacción que me produce terminar el proyecto debido a la trascendencia que supondrá el fin de este. Será el incio de mi futuro.

APÉNDICE A
CÓDIGO FUENTE

programa.py

```
#Parametros de entrada


# f --> frecuencia (Ghz)

# p --> % de tiempo requerido durante el cual no se rebasa la perdida basica de transmision (usaremos 0.01)

# lat_est_t --> latitud estacion interferente (grados)

# lat_est_r --> latitud estacion interferida (grados)

# long_est_t --> longitud estacion interferente (grados)

# long_est_r --> longitud estacion interferida (grados)

# h_tg --> altura del centro de la antena interferente sobre el nivel del suelo (m)

# h_rg --> altura del centro de la antena interferida sobre el nivel del suelo (m)

# h_ts --> altura del centro de la antena interferente sobre el nivel del mar (m)

# h_rs --> altura del centro de la antena interferida sobre el nivel del mar (m)

# h_te, h_re --> alturas efectivas de las antenas sobre el terreno (m)

# G_t --> ganancia de la antena interferente en la direccion del horizonte a lo largo del trayecto ortodromico o del circulo maximo de interferencia (dBi)

# G_r --> ganancia de la antena interferida en la direccion del horizonte a lo largo del trayecto ortodromico o del circulo maximo de interferencia (dBi)

# Polarizacion

# lat_c --> latitud del centro del trayecto (grados)

# d_tm --> seccion continua mas larga sobre la tierra (interior + costera) del trayecto en el circulo maximo (km)

# d_lm --> seccion continua mas larga sobre la tierra (interior) del trayecto en el circulo maximo (km)

# d --> distancia del trayecto a lo largo del circulo maximo (km)

# d_lt, d_lr --> (km) Para un trayecto transhorizonte, la distancia desde las antenas de transmision y recepcion a sus respectivos horizontes

#                         Para un trayecto de visibilidad directa, cada uno se fija a la distancia desde la terminal hasta el punto de perfil identificado

#                      como el borde principal en el metodo de difraccion para el 50%

# teta_t, teta_r --> Para un trayecto transhorizonte, los angulos de elevacion de las antenas de transmision y recepcion respecto del horizonte (mrad)

#                    Para un trayecto de visibilidad directa, cada uno se fija al angulo de elevacion del otro terminal

# teta --> distancia angular del trayecto (mrad)

# d_b --> longitud combinada de las secciones del trayecto sobre el agua (km)

# w --> fraccion del trayecto total sobre el agua. Para trayectos realizados totalmente sobre tierra w = 0

# d_ct, d_cr --> distancia sobre tierra desde las antenas hasta la costa a lo largo del trayecto de circulo maximo de la interferencia (km)
```

```
#               Se fija en 0 en el caso de un terminal ubicado en un barco o en una plataforma maritima d_ct

#########################################################################################################################
####

import math

from tkinter import *

from menu import *

######################  Menu  ##########################

win = Tk()

abrirVentana(win)
```

funciones.py

```
import math
import itur
import itur.models.itu676 as itu676
from tkinter import *
from tkinter import scrolledtext
from tkinter import messagebox
from tkinter.ttk import *

#############################################################
####    Función que calcula beta_0   ####################
#############################################################
def calcula_beta (lat_c,mu_1,mu_4):
    if (math.fabs(lat_c) <= 70) :
        beta_0 = 10**(-0.015*(math.fabs(lat_c))+1.67)*mu_1*mu_4
    else :
        beta_0 = 4.17*mu_1*mu_4

    return beta_0

#########################################################################
#####    Función que calcula la absorción gaseosa (Ag)   #############
#########################################################################
def calcula_absorcion_gaseosa(f, d, rho):

    T = 290               # Temperatura
    P = 1013            # Presion
    Ag = (itur.models.itu676.gamma_exact(f, P, rho, T)) * d         # Absorcion gaseosa total
    return Ag

#################################################################################################
#####    Función que calcula la propagación con visibilidad directa (L_b0)   ########      (varia en 1 dB
con el de pycraft)
#################################################################################################
def calcula_propagacion_visibilidad_directa(p_o_beta, f, w, d, d_lt, d_lr): #

    rho = 7.5 + 2.5*w                       # densidad del vapor de angua
    Ag = calcula_absorcion_gaseosa(f, d, rho)
    E_sp = 2.6 * (1 - math.e**(-0.1*(d_lt+d_lr))) * math.log10(p_o_beta/50)   # Correccion para p o beta_0

    L_bfsg =  92.5 + 20*(math.log10(f)) + 20*(math.log10(d)) + Ag.value    # Perdida de transmision basica
debida a la pripagacion en el espacio libre y atenuacion por gases

    L_b0 = L_bfsg + E_sp    #L_b0p o L_b0b --> perdida de transmision basica no rebasada durante el
porcentaje de tiempo p/beta_0, debido a la propagacion con visibilidad directa

    return L_b0

#########################################################################
#####    Función para calcular la dispersión troposférica  (L_bs)   ######
#########################################################################
def calcula_dispersion_troposferica(d, f, G_t, G_r, teta, p, N0):
    rho = 3
    Ag = calcula_absorcion_gaseosa(f, d, rho)

    Lc = 0.051*math.e**(0.055*(G_t+G_r))
```

```
    Lf = 25*math.log10(f) - 2.5*((math.log10(f/2))**2)

    L_bs = 190 + Lf + 20*math.log10(d) + 0.573*teta - 0.15*N0 + Lc + Ag.value - 10.1*((-
1*math.log10(p/50))**0.7)

    return L_bs

#############################################################################################################
#############
#####    Función para calcular la propagación por conductos y por reflexión en las capas (L_ba = Af + Ad +
Ag)   #####
#############################################################################################################
#############
def calcula_propagacion_conductos_reflexion(d_lt, d_lr, f, teta_r, teta_t, w, d_ct, d_cr, h_ts, h_rs, a_e,
d, p, beta_0, mu_2, mu_3):

    # Af perdidas totales fijas de acoplamiento

    #####  Alf: correccion empirica para tener en cuenta el aumento de la atenuacion con la longitud de
onda en la propagacion por conductos

    if (f < 0.5):
        Alf = 45.375 - 137.0 * f + 92.5 * f**2
    else :
        Alf = 0

    teta_2_t = teta_t - 0.1 * d_lt
    teta_2_r = teta_r - 0.1 * d_lr

    ##### Ast, Asr: perdidas de difraccion debidas al apantallamiento del emplazamiento

    if (teta_2_t > 0):
        Ast = 20*math.log10(1+0.361*teta_2_t*(f*d_lt)**0.5) + 0.264*teta_2_t * f**(1/3)
    else :
        Ast = 0

    if (teta_2_r > 0):
        Asr = 20*math.log10(1+0.361*teta_2_r*(f*d_lr)**0.5) + 0.264*teta_2_r * f**(1/3)
    else :
        Asr = 0

    ##### Act, Acr : correciones del acoplamiento por conductos en la superficie sobre el mar

    if (w >= 0.75 and d_ct >= d_lt and d_ct <= 5):
        Act = -3*math.e**(-0.25*d_ct**2)*(1 + math.tanh(0.07*(50-h_ts/1000)))
    else :
        Act = 0

    if (w >= 0.75 and d_cr >= d_lr and d_cr <= 5):
        Acr = -3*math.e**(-0.25*d_cr**2)*(1 + math.tanh(0.07*(50-h_rs/1000)))
    else :
        Acr = 0

    Af = 102.45 + 20*math.log10(f) + 20*math.log10(d_lt+d_lr) + Alf + Ast + Asr + Act + Acr

    ########################################################################

    # Ad perdidas dependientes del % de tiempo y de la distancia angular

    ganma_d = 5*10**(-5)*a_e*f**(1/3)   # atenuacion especifica

    if (teta_t <= 0.1*d_lt):
        teta_1_t = teta_t
    else:
        teta_1_t = 0.1*d_lt

    if (teta_r <= 0.1*d_lr):
        teta_1_r = teta_r
    else:
        teta_1_r = 0.1*d_lr

    teta_1 = ((d*10**3)/a_e) + teta_1_t + teta_1_r          # distancia angular

    beta = beta_0*mu_2*mu_3
```

```
    r = 1.076/((2.0058-math.log10(beta))**1.012) * math.e**(-(9.51-
4.8*math.log10(beta)+0.198*(math.log10(beta))**2)*10**(-6)*d**1.13)

    Ap = -12 + (1.2+3.7*10**(-3)*d)*math.log10(p/beta) + 12*(p/beta)**r      # Ap variabilidad del % de
tiempo

    Ad = ganma_d*teta_1 + Ap

    ##########################################################################

    rho = 7.5 + 2.5*w
    Ag = calcula_absorcion_gaseosa(f, d, rho)

    ##########################################################################

    L_ba = Af + Ad + Ag.value

    return L_ba
```

menu.py

```
import math
import itur
import itur.models.itu676 as itu676
import itur.models.itu835 as itu835
from pylab import *
import matplotlib.pyplot as plt
from tkinter import *
from tkinter import scrolledtext
from tkinter import messagebox
from funciones import *
from PIL import ImageTk, Image
from astropy import units as u
from pycraf import pathprof, conversions as cnv
import numpy as np
##########################################################################

pathprof.SrtmConf.set(download='missing')

optionlist = [
'Elija una opción',
'Opción 1',
'Opción 2',
'Opción 3',
'Opción 4',
'Opción 5',
'Opción 6',
'Opción 7',
'Opción 8',
'Opción 9',
'Opción 10',
'Opción 11',
'Introducir parámetros'
]
def abrirVentana(win):

    win.title("Parametros de entrada")
    win.geometry('950x450')

    nb = Notebook(win)
    nb.pack(fill = "both", expand = "yes")

    p1 = Frame(nb)
    p2 = Frame(nb)

    opcion1 = StringVar()
    opcion2 = StringVar()

    ######## Funciones de ayuda del parametro ############################

    def clicked_1():
```

```
            messagebox.showinfo('Definición del parámetro', 'Latitud del centro del trayecto (grados)')

    def clicked_2():

            messagebox.showinfo('Definición del parámetro', 'Sección continua mas larga sobre la tierra
(interior + costera) del trayecto en el circulo máximo (km)')

    def clicked_3():

            messagebox.showinfo('Definición del parámetro', 'Sección continua mas larga sobre la tierra
(interior) del trayecto en el círculo máximo (km)')

    def clicked_4():

            messagebox.showinfo('Definición del parámetro', 'Longitud combinada de las secciones del
trayecto sobre el agua (km)')

    def clicked_5():

            messagebox.showinfo('Definición del parámetro', 'Distancia del trayecto a lo largo del círculo
máximo (km)')

    def clicked_6():

            messagebox.showinfo('Definición del parámetro', 'Frecuencia (Ghz)')

    def clicked_7():

            messagebox.showinfo('Definición del parámetro', 'Latitud de la estación interferente
(grados)')

    def clicked_8():

            messagebox.showinfo('Definición del parámetro', 'Latitud de la estación interferida (grados)')

    def clicked_9():

            messagebox.showinfo('Definición del parámetro', 'Longitud de la estación interferente
(grados)')

    def clicked_10():

            messagebox.showinfo('Definición del parámetro', 'Longitud de la estación interferida
(grados)')

    def clicked_11():

            messagebox.showinfo('Definición del parámetro', 'Altura del centro de la antena interferente
sobre el nivel del suelo (m)')

    def clicked_12():

            messagebox.showinfo('Definición del parámetro', 'Altura del centro de la antena interferida
sobre el nivel del suelo (m)')

    def clicked_13():

            messagebox.showinfo('Definición del parámetro', 'Altura del centro de la antena interferente
sobre el nivel del mar (m)')

    def clicked_14():

            messagebox.showinfo('Definición del parámetro', 'Altura del centro de la antena interferida
sobre el nivel del mar (m)')

    def clicked_15():

            messagebox.showinfo('Definición del parámetro', 'Alturas efectiva de la antena interferente
sobre el terreno (m)')

    def clicked_16():

            messagebox.showinfo('Definición del parámetro', 'Altura efectiva de la antena interferida
sobre el terreno (m)')

    def clicked_17():
```

```
            messagebox.showinfo('Definición del parámetro', 'Ganancia de la antena interferente en la
dirección del horizonte a lo largo del trayecto ortodrómico o del círculo máximo de interferencia (dBi)')

    def clicked_18():

            messagebox.showinfo('Definición del parámetro', 'Ganancia de la antena interferida en la
dirección del horizonte a lo largo del trayecto ortodrómico o del círculo máximo de interferencia (dBi)')

    def clicked_19():

            messagebox.showinfo('Definición del parámetro', 'Para un trayecto transhorizonte, la distancia
desde la antena de transmision a su horizonte.\nPara un trayecto de visibilidad directa, cada uno se fija
a la distancia desde la terminal hasta el punto de perfil identificado como el borde principal en el
método de difracción para el 50% (km).')

    def clicked_20():

            messagebox.showinfo('Definición del parámetro', 'Para un trayecto transhorizonte, la distancia
desde la antena de recepcion a su horizonte.\nPara un trayecto de visibilidad directa, cada uno se fija a
la distancia desde la terminal hasta el punto de perfil identificado como el borde principal en el método
de difracción para el 50% (km).')

    def clicked_21():

            messagebox.showinfo('Definición del parámetro', 'Para un trayecto transhorizonte, el ángulo de
elevación de la antena de transmisión respecto del horizonte.\nPara un trayecto de visibilidad directa,
cada uno se fija al ángulo de elevación del otro terminal (mrad).')

    def clicked_22():

            messagebox.showinfo('Definición del parámetro', 'Para un trayecto transhorizonte, el ángulo de
elevación de la antena de recepción respecto del horizonte.\nPara un trayecto de visibilidad directa, cada
uno se fija al ángulo de elevación del otro terminal (mrad).')


    def clicked_24():

            messagebox.showinfo('Definición del parámetro', 'Distancia sobre tierra desde la antena
transmisora hasta la costa a lo largo del trayecto de círculo máximo de la interferencia (km)\nSe fija en
0 en el caso de un terminal ubicado en un barco o en una plataforma marítima')

    def clicked_25():

            messagebox.showinfo('Definición del parámetro', 'Distancia sobre tierra desde la antena
receptora hasta la costa a lo largo del trayecto de círculo máximo de la interferencia (km)\nSe fija en 0
en el caso de un terminal ubicado en un barco o en una plataforma marítima')

    def clicked_26():

            messagebox.showinfo('Definición del parámetro', 'Altura de la antena interferente por encima
del nivel local del suelo (m)')

    def clicked_27():

            messagebox.showinfo('Definición del parámetro', 'Altura de la antena interferida por encima
del nivel local del suelo (m)')

    def clicked_28():
        root = Toplevel(win)
        root.geometry('800x550')

        load = Image.open("Cuadro4.png")
        render = ImageTk.PhotoImage(load)

        img = Label(root, image=render)
        img.pack(fill=BOTH, expand = YES)
        img.image = render
        img.place(x=0, y=0)

    def seleccionar1(p1):
        if (opcion1.get() == 'Introducir parámetros') :
            txt_ha_1.config(state='enabled')
            txt_dk_1.config(state='enabled')
        else :
            txt_ha_1.config(state='disabled')
```

```
            txt_dk_1.config(state='disabled')

    def seleccionar2(p1):
        if (opcion2.get() == 'Introducir parámetros') :
            txt_ha_2.config(state='enabled')
            txt_dk_2.config(state='enabled')
        else :
            txt_ha_2.config(state='disabled')
            txt_dk_2.config(state='disabled')

    #######  Funcion principal del modelo de cielo despejado
    def cierraPrincipal():



        if (txt_1.get()=='' or txt_2.get()=='' or txt_3.get()=='' or txt_4.get()=='' or txt_5.get()=='' or
txt_6.get()=='' or txt_7.get()=='' or txt_8.get()=='' or txt_9.get()=='' or
            txt_10.get()=='' or txt_11.get()=='' or txt_12.get()=='' or txt_13.get()=='' or
txt_14.get()=='' or txt_15.get()=='' or txt_16.get()=='' or txt_17.get()=='' or txt_18.get()=='' or
            txt_19.get()=='' or txt_20.get()=='' or txt_21.get()=='' or txt_22.get()=='' or
txt_24.get()=='' or txt_25.get()=='' or txt_26.get()=='' or txt_27.get()==''):

            messagebox.showwarning('Warning', 'Todos los campos deben estar rellenos')
        else:

            lat_c = float(txt_1.get())
            d_tm = float(txt_2.get())
            d_lm = float(txt_3.get())
            d_b = float(txt_4.get())
            d = float(txt_5.get())
            f = float(txt_6.get())
            lat_est_t = float(txt_7.get())
            lat_est_r = float(txt_8.get())
            long_est_t = float(txt_9.get())
            long_est_r = float(txt_10.get())
            h_tg = float(txt_11.get())
            h_rg = float(txt_12.get())
            h_ts = float(txt_13.get())
            h_rs = float(txt_14.get())
            h_te = float(txt_15.get())
            h_re = float(txt_16.get())
            G_t = float(txt_17.get())
            G_r = float(txt_18.get())
            d_lt = float(txt_19.get())
            d_lr = float(txt_20.get())
            teta_t = float(txt_21.get())
            teta_r = float(txt_22.get())
            # teta = float(txt_23.get())
            d_ct = float(txt_24.get())
            d_cr = float(txt_25.get())
            ht = float(txt_26.get())
            hr = float(txt_27.get())

            if (lat_c > 90 or lat_c <-90 or lat_est_t > 90 or lat_est_t <-90 or lat_est_r > 90 or
lat_est_r <-90 or long_est_t > 180 or long_est_t <-180 or long_est_r > 180 or long_est_r <-180):
                messagebox.showwarning('Warning', 'Los parametros geográficos lat_est_t, lat_est_r y lat_c
deben estar comprendidos entre -90 y 90 grados, así como long_est_t y long_est_r deben estar entre -180 y
180 grados.')
            else:

                if (d > 10000):
                    messagebox.showwarning('Warning', 'La distancia no puede ser mayor de 10000 Km.')
                else:

                    if (f < 0.1 or f > 50):
                        messagebox.showwarning('Warning', 'La frecuencia debe estar comprendida entre 0.1
y 50 GHz.')
                    else:


##########################################################################################
Perdidas adicionales por la ocupacion del suelo
##########################################################################################

                        #Estacion interferente
```

```
    if (opcion1.get() == 'Elija una opción' or opcion2.get() == 'Elija una opción'):

        messagebox.showwarning('Warning', 'Elija una opción de suelo')
    else:

        if (opcion1.get() == 'Opción 1'):
            ha_t = 4
            dk_t = 0.1
            zone_t = pathprof.CLUTTER.SPARSE
        elif (opcion1.get() == 'Opción 2'):
            ha_t = 5
            dk_t = 0.07
            zone_t = pathprof.CLUTTER.VILLAGE
        elif (opcion1.get() == 'Opción 3') :
            ha_t = 15
            dk_t = 0.05
            zone_t = pathprof.CLUTTER.DECIDIOUS_TREES
        elif (opcion1.get() == 'Opción 4') :
            ha_t = 20
            dk_t = 0.05
            zone_t = pathprof.CLUTTER.CONIFEROUS_TREES
        elif (opcion1.get() == 'Opción 5') :
            ha_t = 20
            dk_t = 0.03
            zone_t = pathprof.CLUTTER.TROPICAL_FOREST
        elif (opcion1.get() == 'Opción 6') :
            ha_t = 9
            dk_t = 0.025
            zone_t = pathprof.CLUTTER.SUBURBAN
        elif (opcion1.get() == 'Opción 7') :
            ha_t = 12
            dk_t = 0.02
            zone_t = pathprof.CLUTTER.DENSE_SUBURBAN
        elif (opcion1.get() == 'Opción 8') :
            ha_t = 20
            dk_t = 0.02
            zone_t = pathprof.CLUTTER.URBAN
        elif (opcion1.get() == 'Opción 9') :
            ha_t = 25
            dk_t = 0.02
            zone_t = pathprof.CLUTTER.DENSE_URBAN
        elif (opcion1.get() == 'Opción 10') :
            ha_t = 35
            dk_t = 0.02
            zone_t = pathprof.CLUTTER.HIGH_URBAN
        elif (opcion1.get() == 'Opción 11') :
            ha_t = 20
            dk_t = 0.05
            zone_t = pathprof.CLUTTER.INDUSTRIAL_ZONE
        else :
            ha_t = float(txt_ha_1.get())
            dk_t = float(txt_dk_1.get())
            zone_t = pathprof.CLUTTER.UNKNOWN

            #Estacion Interferida

        if (opcion2.get() == 'Opción 1'):
            ha_r = 4
            dk_r = 0.1
            zone_r = pathprof.CLUTTER.SPARSE
        elif (opcion2.get() == 'Opción 2'):
            ha_r = 5
            dk_r = 0.07
            zone_r = pathprof.CLUTTER.VILLAGE
        elif (opcion2.get() == 'Opción 3') :
            ha_r = 15
            dk_r = 0.05
            zone_r = pathprof.CLUTTER.DECIDIOUS_TREES
        elif (opcion2.get() == 'Opción 4') :
            ha_r = 20
            dk_r = 0.05
            zone_r = pathprof.CLUTTER.CONIFEROUS_TREES
        elif (opcion2.get() == 'Opción 5') :
            ha_r = 20
            dk_r = 0.03
```

```
                            zone_r = pathprof.CLUTTER.TROPICAL_FOREST
                        elif (opcion2.get() == 'Opción 6') :
                            ha_r = 9
                            dk_r = 0.025
                            zone_r = pathprof.CLUTTER.SUBURBAN
                        elif (opcion2.get() == 'Opción 7') :
                            ha_r = 12
                            dk_r = 0.02
                            zone_r = pathprof.CLUTTER.DENSE_SUBURBAN
                        elif (opcion2.get() == 'Opción 8') :
                            ha_r = 20
                            dk_r = 0.02
                            zone_r = pathprof.CLUTTER.URBAN
                        elif (opcion2.get() == 'Opción 9') :
                            ha_r = 25
                            dk_r = 0.02
                            zone_r = pathprof.CLUTTER.DENSE_URBAN
                        elif (opcion2.get() == 'Opción 10') :
                            ha_r = 35
                            dk_r = 0.02
                            zone_r = pathprof.CLUTTER.HIGH_URBAN
                        elif (opcion2.get() == 'Opción 11') :
                            ha_r = 20
                            dk_r = 0.05
                            zone_r = pathprof.CLUTTER.INDUSTRIAL_ZONE
                        else :
                            ha_r = float(txt_ha_2.get())
                            dk_r = float(txt_dk_2.get())
                            zone_r = pathprof.CLUTTER.UNKNOWN


                        F_fc = 0.25 + 0.375*(1+math.tanh(7.5*(f-0.5)))

                        A_ht = 10.25 * F_fc * math.e**(-dk_t) * (1-math.tanh(6*(ht/ha_t - 0.625))) -
0.33
                        A_hr = 10.25*F_fc*math.e**(-dk_r)*(1-math.tanh(6*(hr/ha_r - 0.625))) - 0.33
############################################################################
#########   Calculo de la difracción
############################################################################

                        Temp = 290 * u.K                # Temperatura
                        Pres = 1013.25 * u.hPa          # Presion
                        p_p = 0.01 * u.percent          #

                        hprof_step = 100 * u.m   # resolution of height profile
                        freq = f * u.GHz
                        lat_tx = lat_est_t * u.deg
                        lat_rx = lat_est_r * u.deg
                        lon_tx = long_est_t * u.deg
                        lon_rx = long_est_r * u.deg
                        h_tg_1 = h_tg * u.m
                        h_rg_1 = h_rg * u.m

                        pprop = pathprof.PathProp(
                                freq,
                                Temp, Pres,
                                lon_tx, lat_tx,
                                lon_rx, lat_rx,
                                h_tg_1, h_rg_1,
                                hprof_step,
                                p_p,
                                zone_t=zone_t, zone_r=zone_r,
                                )

                        L_bd = pathprof.loss_diffraction(pprop)[3].value


############################################################################
#################### Radio efectivo de la tierra ###########################

                        hm = pprop.h_m.value      # rugosidad del terreno

                        incremento_N = pprop.delta_N.value
                        N0 = pprop.N0.value  ## Refractividad de la superficie a nivel del mar
                        k_beta = 3.0   #estimacion del factor del radio efectivo de la Tierra excecido
duerante el beta_0 % del tiempo
```

```
                            k_50 = 157 /(157-incremento_N)

                            a_e = 6371*k_50   #valor mediano del radio efectivo de la Tierra (Km)
                            a_beta = 6371*k_beta   #radio efectivo de la Tierra rebasado durante el beta_0
% (Km)
                            p = 0.01

                            #### calculo teta

                            teta = 10**3*d/a_e + teta_t + teta_r


###############################################################################
###########     calculo mu      ###############################################
###############################################################################

                            E = 3.5
                            tau = 1-(math.e**-(0.000412*(d_lm**2.41)))

                            mu_1 = (10**(-d_tm/(16-6.6*tau))+(10**-(0.496+0.354*tau))**5)**0.2

                            alfa = -0.6 - E*(10**-9) * d**3.1 * tau

                            mu_2 = ((500 * d**2) / (a_e * (math.sqrt(h_te) + math.sqrt(h_re))**2))**alfa
# correccion por la geometria del trayecto

                            dI = min(d-d_lt-d_lr, 40)
                            # mu_3 correccion por la rugosidad del terreno
                            if (hm <= 10):
                                mu_3 = 1
                            else:
                                mu_3 = math.e**(-4.6*(10**-5)*(hm-10)*(43+6*dI))

                            # mu_4

                            if (math.fabs(lat_c) <= 70) :
                                mu_4 = 10**((-0.935+0.0176*math.fabs(lat_c))*math.log10(mu_1))
                            else :
                                mu_4 = 10**(0.3*math.log10(mu_1)
                            ###################################################

                            beta_0 = calcula_beta (lat_c,mu_1,mu_4)

                            if (d_b == 0):
                                w = 0
                            else :
                                w = d_b/d   #fraccion del trayecto total sobre el agua

                            ###############################################################################

                            L_b0p = calcula_propagacion_visibilidad_directa(p, f, w, d, d_lt, d_lr)
                            L_b0b = calcula_propagacion_visibilidad_directa(beta_0, f, w, d, d_lt, d_lr)
                            L_bs = calcula_dispersion_troposferica(d, f, G_t, G_r, teta, p, N0)
                            L_ba = calcula_propagacion_conductos_reflexion(d_lt, d_lr, f, teta_r, teta_t,
w, d_ct, d_cr, h_ts, h_rs, a_e, d, p, beta_0, mu_2, mu_3)
                            ###############################################################################

                            L_dp = pathprof.loss_diffraction(pprop)[1].value
                            L_bd_50 = pathprof.loss_diffraction(pprop)[2].value

                            S_tim = pprop.S_tim_50.value
                            S_tr = pprop.S_tr_50.value
                            Fj = 1 - 0.5 * (1 + math.tanh(3 * 0.8 * (S_tim - S_tr)/0.3))        ### factor
de interpolacion para tener en cuenta la distancia angular del trayecto

                            Fk = 1 - 0.5 * (1 + math.tanh(3 * 0.5 * (d - 20)/20))               ### factor
de interpolacion para tener en cuenta la distancia del circulo maximo

                            if (50 > p and p > beta_0):
                                Fi = func_I(p/100) / func_I(beta_0/100)
                            else:
                                Fi = 1

                            if (p < beta_0):
                                L_min_b0p = L_b0p + (1 - w)*L_dp
                            else:
```

```
                                L_min_b0p = L_bd_50 + (L_b0b + (1-w)*L_dp - L_bd_50)*Fi

                            L_min_bap = 2.5*math.log(math.e**(L_ba/2.5) + math.e**(L_b0p/2.5))

                            if (L_min_bap > L_bd):
                                L_bda = L_bd
                            else:
                                L_bda = L_min_bap + (L_bd - L_min_bap)*Fk

                            L_bam = L_bda + (L_min_b0p - L_bda)*Fj

                            L_b = -5 * math.log10(10**(-0.2*L_bs) + 10**(-0.2*L_bam)) + A_ht + A_hr

                            L = L_b - G_t - G_r
                            print("L = ", "{0:.4f}".format(L), " dBs")

                            win.quit()

##############    Fin función principal 1    ##############################
################################################################################
#############   Widgets ventana 1 ##########################################
################################################################################

    lbl_1 = Label(p1, text="lat_c", font=("Arial Bold", 10))
    lbl_1.grid(column=0, row=0)
    txt_1 = Entry(p1,width=10)
    txt_1.insert(0, "41.131580")
    txt_1.grid(column=1, row=0)
    lbl_1_1 = Label(p1, text="(grados)", font=("Arial Bold", 10))
    lbl_1_1.grid(column=2, row=0)

    lbl_2 = Label(p1, text="d_tm", font=("Arial Bold", 10))
    lbl_2.grid(column=0, row=1)
    txt_2 = Entry(p1,width=10)
    txt_2.insert(0, "510.375760")
    txt_2.grid(column=1, row=1)
    lbl_2_1 = Label(p1, text="(km)", font=("Arial Bold", 10))
    lbl_2_1.grid(column=2, row=1)

    lbl_3 = Label(p1, text="d_lm", font=("Arial Bold", 10))
    lbl_3.grid(column=0, row=2)
    txt_3 = Entry(p1,width=10)
    txt_3.insert(0, "510.375760")
    txt_3.grid(column=1, row=2)
    lbl_3_1 = Label(p1, text="(km)", font=("Arial Bold", 10))
    lbl_3_1.grid(column=2, row=2)

    lbl_4 = Label(p1, text="d_b", font=("Arial Bold", 10))
    lbl_4.grid(column=0, row=3)
    txt_4 = Entry(p1,width=10)
    txt_4.insert(0, "0")
    txt_4.grid(column=1, row=3)
    lbl_4_1 = Label(p1, text="(km)", font=("Arial Bold", 10))
    lbl_4_1.grid(column=2, row=3)

    lbl_5 = Label(p1, text="d", font=("Arial Bold", 10))
    lbl_5.grid(column=0, row=4)
    txt_5 = Entry(p1,width=10)
    txt_5.insert(0, "510.375760")
    txt_5.grid(column=1, row=4)
    lbl_5_1 = Label(p1, text="(km)", font=("Arial Bold", 10))
    lbl_5_1.grid(column=2, row=4)

    lbl_6 = Label(p1, text="f", font=("Arial Bold", 10))
    lbl_6.grid(column=0, row=5)
    txt_6 = Entry(p1,width=10)
    txt_6.insert(0, "10")
    txt_6.grid(column=1, row=5)
    lbl_6_1 = Label(p1, text="(Ghz)", font=("Arial Bold", 10))
    lbl_6_1.grid(column=2, row=5)

    lbl_7 = Label(p1, text="lat_est_t", font=("Arial Bold", 10))
    lbl_7.grid(column=0, row=6)
    txt_7 = Entry(p1,width=10)
    txt_7.insert(0, "38.893610")
    txt_7.grid(column=1, row=6)
```

```
lbl_7_1 = Label(p1, text="(grados)", font=("Arial Bold", 10))
lbl_7_1.grid(column=2, row=6)

lbl_8 = Label(p1, text="lat_est_r", font=("Arial Bold", 10))
lbl_8.grid(column=0, row=7)
txt_8 = Entry(p1,width=10)
txt_8.insert(0, "43.364160")
txt_8.grid(column=1, row=7)
lbl_8_1 = Label(p1, text="(grados)", font=("Arial Bold", 10))
lbl_8_1.grid(column=2, row=7)

lbl_9 = Label(p1, text="long_est_t", font=("Arial Bold", 10))
lbl_9.grid(column=0, row=8)
txt_9 = Entry(p1,width=10)
txt_9.insert(0, "-6.989160")
txt_9.grid(column=1, row=8)
lbl_9_1 = Label(p1, text="(grados)", font=("Arial Bold", 10))
lbl_9_1.grid(column=2, row=8)

lbl_10 = Label(p1, text="long_est_r", font=("Arial Bold", 10))
lbl_10.grid(column=0, row=9)
txt_10 = Entry(p1,width=10)
txt_10.insert(0, "-8.398800")
txt_10.grid(column=1, row=9)
lbl_10_1 = Label(p1, text="(grados)", font=("Arial Bold", 10))
lbl_10_1.grid(column=2, row=9)

lbl_11 = Label(p1, text="h_tg", font=("Arial Bold", 10))
lbl_11.grid(column=0, row=10)
txt_11 = Entry(p1,width=10)
txt_11.insert(0, "20")
txt_11.grid(column=1, row=10)
lbl_11_1 = Label(p1, text="(m)", font=("Arial Bold", 10))
lbl_11_1.grid(column=2, row=10)

lbl_12 = Label(p1, text="h_rg", font=("Arial Bold", 10))
lbl_12.grid(column=0, row=11)
txt_12 = Entry(p1,width=10)
txt_12.insert(0, "40")
txt_12.grid(column=1, row=11)
lbl_12_1 = Label(p1, text="(m)", font=("Arial Bold", 10))
lbl_12_1.grid(column=2, row=11)

lbl_13 = Label(p1, text="h_ts", font=("Arial Bold", 10))
lbl_13.grid(column=0, row=12)
txt_13 = Entry(p1,width=10)
txt_13.insert(0, "206.354497")
txt_13.grid(column=1, row=12)
lbl_13_1 = Label(p1, text="(m)", font=("Arial Bold", 10))
lbl_13_1.grid(column=2, row=12)

lbl_14 = Label(p1, text="h_rs", font=("Arial Bold", 10))
lbl_14.grid(column=5, row=0)
txt_14 = Entry(p1,width=10)
txt_14.insert(0, "42.333698")
txt_14.grid(column=6, row=0)
lbl_14_1 = Label(p1, text="(m)", font=("Arial Bold", 10))
lbl_14_1.grid(column=7, row=0)

lbl_15 = Label(p1, text="h_te", font=("Arial Bold", 10))
lbl_15.grid(column=5, row=1)
txt_15 = Entry(p1,width=10)
txt_15.insert(0, "20")
txt_15.grid(column=6, row=1)
lbl_15_1 = Label(p1, text="(m)", font=("Arial Bold", 10))
lbl_15_1.grid(column=7, row=1)

lbl_16 = Label(p1, text="h_re", font=("Arial Bold", 10))
lbl_16.grid(column=5, row=2)
txt_16 = Entry(p1,width=10)
txt_16.insert(0, "40")
txt_16.grid(column=6, row=2)
lbl_16_1 = Label(p1, text="(m)", font=("Arial Bold", 10))
lbl_16_1.grid(column=7, row=2)

lbl_17 = Label(p1, text="G_t", font=("Arial Bold", 10))
```

```
lbl_17.grid(column=5, row=3)
txt_17 = Entry(p1,width=10)
txt_17.insert(0, "5")
txt_17.grid(column=6, row=3)
lbl_17_1 = Label(p1, text="(dBi)", font=("Arial Bold", 10))
lbl_17_1.grid(column=7, row=3)

lbl_18 = Label(p1, text="G_r", font=("Arial Bold", 10))
lbl_18.grid(column=5, row=4)
txt_18 = Entry(p1,width=10)
txt_18.insert(0, "10")
txt_18.grid(column=6, row=4)
lbl_18_1 = Label(p1, text="(dBi)", font=("Arial Bold", 10))
lbl_18_1.grid(column=7, row=4)

lbl_19 = Label(p1, text="d_lt", font=("Arial Bold", 10))
lbl_19.grid(column=5, row=5)
txt_19 = Entry(p1,width=10)
txt_19.insert(0, "1.9")
txt_19.grid(column=6, row=5)
lbl_19_1 = Label(p1, text="(km)", font=("Arial Bold", 10))
lbl_19_1.grid(column=7, row=5)

lbl_20 = Label(p1, text="d_lr", font=("Arial Bold", 10))
lbl_20.grid(column=5, row=6)
txt_20 = Entry(p1,width=10)
txt_20.insert(0, "2.375760")
txt_20.grid(column=6, row=6)
lbl_20_1 = Label(p1, text="(km)", font=("Arial Bold", 10))
lbl_20_1.grid(column=7, row=6)

lbl_21 = Label(p1, text="teta_t", font=("Arial Bold", 10))
lbl_21.grid(column=5, row=7)
txt_21 = Entry(p1,width=10)
txt_21.insert(0, "8.101877")
txt_21.grid(column=6, row=7)
lbl_21_1 = Label(p1, text="(mrad)", font=("Arial Bold", 10))
lbl_21_1.grid(column=7, row=7)

lbl_22 = Label(p1, text="teta_r", font=("Arial Bold", 10))
lbl_22.grid(column=5, row=8)
txt_22 = Entry(p1,width=10)
txt_22.insert(0, "18.790085")
txt_22.grid(column=6, row=8)
lbl_22_1 = Label(p1, text="(mrad)", font=("Arial Bold", 10))
lbl_22_1.grid(column=7, row=8)

    lbl_24 = Label(p1, text="d_ct", font=("Arial Bold", 10))
lbl_24.grid(column=5, row=9)
txt_24 = Entry(p1,width=10)
txt_24.insert(0, "50000")
txt_24.grid(column=6, row=9)
lbl_24_1 = Label(p1, text="(km)", font=("Arial Bold", 10))
lbl_24_1.grid(column=7, row=9)

lbl_25 = Label(p1, text="d_cr", font=("Arial Bold", 10))
lbl_25.grid(column=5, row=10)
txt_25 = Entry(p1,width=10)
txt_25.insert(0, "50000")
txt_25.grid(column=6, row=10)
lbl_25_1 = Label(p1, text="(km)", font=("Arial Bold", 10))
lbl_25_1.grid(column=7, row=10)

lbl_26 = Label(p1, text="ht", font=("Arial Bold", 10))
lbl_26.grid(column=5, row=11)
txt_26 = Entry(p1,width=10)
txt_26.insert(0, "20")
txt_26.grid(column=6, row=11)
lbl_26_1 = Label(p1, text="(m)", font=("Arial Bold", 10))
lbl_26_1.grid(column=7, row=11)

lbl_27 = Label(p1, text="hr", font=("Arial Bold", 10))
lbl_27.grid(column=5, row=12)
txt_27 = Entry(p1,width=10)
txt_27.insert(0, "40")
txt_27.grid(column=6, row=12)
```

```
lbl_27_1 = Label(p1, text="(m)", font=("Arial Bold", 10))
lbl_27_1.grid(column=7, row=12)

btn_1 = Button(p1,text='?', width = 2, command=clicked_1)
btn_1.grid(column=3,row=0)

btn_2 = Button(p1,text='?', width = 2, command=clicked_2)
btn_2.grid(column=3,row=1)

btn_3 = Button(p1,text='?', width = 2, command=clicked_3)
btn_3.grid(column=3,row=2)

btn_4 = Button(p1,text='?', width = 2, command=clicked_4)
btn_4.grid(column=3,row=3)

btn_5 = Button(p1,text='?', width = 2, command=clicked_5)
btn_5.grid(column=3,row=4)

btn_6 = Button(p1,text='?', width = 2, command=clicked_6)
btn_6.grid(column=3,row=5)

btn_7 = Button(p1,text='?', width = 2, command=clicked_7)
btn_7.grid(column=3,row=6)

btn_8 = Button(p1,text='?', width = 2, command=clicked_8)
btn_8.grid(column=3,row=7)

btn_9 = Button(p1,text='?', width = 2, command=clicked_9)
btn_9.grid(column=3,row=8)

btn_10 = Button(p1,text='?', width = 2, command=clicked_10)
btn_10.grid(column=3,row=9)

btn_11 = Button(p1,text='?', width = 2, command=clicked_11)
btn_11.grid(column=3,row=10)

btn_12 = Button(p1,text='?', width = 2, command=clicked_12)
btn_12.grid(column=3,row=11)

btn_13 = Button(p1,text='?', width = 2, command=clicked_13)
btn_13.grid(column=3,row=12)

btn_14 = Button(p1,text='?', width = 2, command=clicked_14)
btn_14.grid(column=8,row=0)

btn_15 = Button(p1,text='?', width = 2, command=clicked_15)
btn_15.grid(column=8,row=1)

btn_16 = Button(p1,text='?', width = 2, command=clicked_16)
btn_16.grid(column=8,row=2)

btn_17 = Button(p1,text='?', width = 2, command=clicked_17)
btn_17.grid(column=8,row=3)

btn_18 = Button(p1,text='?', width = 2, command=clicked_18)
btn_18.grid(column=8,row=4)

btn_19 = Button(p1,text='?', width = 2, command=clicked_19)
btn_19.grid(column=8,row=5)

btn_20 = Button(p1,text='?', width = 2, command=clicked_20)
btn_20.grid(column=8,row=6)

btn_21 = Button(p1,text='?', width = 2, command=clicked_21)
btn_21.grid(column=8,row=7)

btn_22 = Button(p1,text='?', width = 2, command=clicked_22)
btn_22.grid(column=8,row=8)

btn_24 = Button(p1,text='?', width = 2, command=clicked_24)
btn_24.grid(column=8,row=9)

btn_25 = Button(p1,text='?', width = 2, command=clicked_25)
btn_25.grid(column=8,row=10)

btn_26 = Button(p1,text='?', width = 2, command=clicked_26)
```

```
    btn_26.grid(column=8,row=11)

    btn_27 = Button(p1,text='?', width = 2, command=clicked_27)
    btn_27.grid(column=8,row=12)

    btn_28 = Button(p1,text='?', width = 2, command=clicked_28)
    btn_28.grid(column=10,row=6)

    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=0)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=1)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=2)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=3)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=4)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=5)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=6)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=7)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=8)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=9)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=10)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=11)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=9, row=12)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=0)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=1)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=2)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=3)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=4)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=5)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=6)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=7)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=8)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=9)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=10)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=11)
    Label(p1, text="|", font=("Arial Bold", 15)).grid(column=4, row=12)

    Label(p1, text="Ocupación del suelo de las antenas.", font=("Arial Bold", 10)).grid(column=10, row=0)
    Label(p1, text="Pulse "'"?"'" para más información", font=("Arial Bold", 10)).grid(column=10, row=1)

    etiq = Label(p1, text="Estacion interferente", font=("Arial Bold", 10)).grid(column=10, row=2)
    etiq = Label(p1, text="Estacion interferida", font=("Arial Bold", 10)).grid(column=10, row=4)

    li = OptionMenu(p1, opcion1, *optionlist, command=seleccionar1)
    li.grid(column=10, row=3)
    Label(p1, text="h_a:", font=("Arial Bold", 10)).grid(column=11, row=3)
    txt_ha_1 = Entry(p1,width=5, state='disabled')
    txt_ha_1.grid(column=12, row=3)
    Label(p1, text="(m)", font=("Arial Bold", 10)).grid(column=13, row=3)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=14, row=3)
    Label(p1, text="d_k:", font=("Arial Bold", 10)).grid(column=15, row=3)
    txt_dk_1 = Entry(p1,width=5, state='disabled')
    txt_dk_1.grid(column=16, row=3)
    Label(p1, text="(km)", font=("Arial Bold", 10)).grid(column=17, row=3)

    lo = OptionMenu(p1, opcion2, *optionlist, command=seleccionar2)
    lo.grid(column=10, row=5)
    Label(p1, text="h_a:", font=("Arial Bold", 10)).grid(column=11, row=5)
    txt_ha_2 = Entry(p1,width=5, state='disabled')
    txt_ha_2.grid(column=12, row=5)
    Label(p1, text="(m)", font=("Arial Bold", 10)).grid(column=13, row=5)
    Label(p1, text="||", font=("Arial Bold", 15)).grid(column=14, row=5)
    Label(p1, text="d_k:", font=("Arial Bold", 10)).grid(column=15, row=5)
    txt_dk_2 = Entry(p1,width=5, state='disabled')
    txt_dk_2.grid(column=16, row=5)
    Label(p1, text="(km)", font=("Arial Bold", 10)).grid(column=17, row=5)

    button = Button(p1, text="OK", command=cierraPrincipal)
    button.grid(column=18, row=20)

#############   Fin Widgets ventana 1   ####################################
##############################################################################
##############################################################################
####### Funcion secundaria    ################################################

    def cierraSecundaria():
```

```
        win.quit()

#######  Fin funcion secundaria   ##########################################
###############################################################################
#######     Graficas    ######################################################
    def grafica1():

        if (txt_30.get()=='' or txt_31.get()=='' or txt_32.get()=='' or txt_33.get()=='' or
txt_34.get()=='' or txt_35.get()==''):

            messagebox.showwarning('Warning', 'Todos los campos deben estar rellenos')

        p = float(txt_30.get())
        f_ini = float(txt_31.get())
        f_end = float(txt_32.get())
        f_step = float(txt_33.get())
        w = float(txt_34.get())
        d = float(txt_35.get())

        if (d > 10000):
            messagebox.showwarning('Warning', 'La distancia no puede ser mayor de 10000 Km.')
        else:

            def f1(x):

                rho = 7.5 + 2.5*w                # densidad del vapor de angua
                Ag = calcula_absorcion_gaseosa(x, d, rho)
                L_bfsg =  92.5 + 20*(np.log10(x)) + 20*(math.log10(d)) + Ag.value  # Perdida de
transmision basica debida a la pripagacion en el espacio libre y atenuacion por gases
                return L_bfsg

            x = arange(f_ini, f_end, f_step)          # array de floats, de f_ini a f_end cada f_step

            p1 = plot(x, f1(x), '-', label='$d=%d$ Km ' % d)    # generar el gráfico de la función y=x
            grid()
            title('Gráfica 1')
            ylabel('Atenuación: dBs')
            xlabel('Frecuancia: Ghz')
            legend(loc=1)
            show()

    def grafica2():

        if (txt_36.get()=='' or txt_38.get()=='' or txt_39.get()=='' or txt_40.get()=='' or
txt_41.get()=='' or txt_37.get()==''):

            messagebox.showwarning('Warning', 'Todos los campos deben estar rellenos')

        p = float(txt_36.get())
        d_ini = float(txt_38.get())
        d_end = float(txt_39.get())
        d_step = float(txt_40.get())
        w = float(txt_41.get())
        f = float(txt_37.get())

        if (f < 0.1 or f > 50):
            messagebox.showwarning('Warning', 'La frecuencia debe estar comprendida entre 0.1 y 50 GHz.')
        else:

            def f1(x):

                rho = 7.5 + 2.5*w                      # densidad del vapor de angua
                Ag = calcula_absorcion_gaseosa(f, x, rho)
                L_bfsg =  92.5 + 20*(np.log10(f)) + 20*(np.log10(x)) + Ag.value    # Perdida de
transmision basica debida a la pripagacion en el espacio libre y atenuacion por gases
                return L_bfsg

            x = arange(d_ini, d_end, d_step)          # array de floats, de d_ini a d_end cada d_step

            p1 = plot(x, f1(x), '-', label='$f=%f$ GHz' % f)    # generar el gráfico de la función
            grid()
            title('Gráfica 2')
            ylabel('Atenuación: dBs')
            xlabel('Distancia: Km')
            legend(loc=1)
            show()
```

```
#############################################################################

    def ayuda1():

        messagebox.showinfo('Grafica 1', 'Esta gráfica nos representa la pérdida de transmisión básica
debida a la propagación en espacio libre en funcion de la frecuencia.')

    def ayuda2():

        messagebox.showinfo('Grafica 2', 'Esta gráfica nos representa la pérdida de transmisión básica
debida a la propagación en espacio libre en funcion de la distancia.')

#############   Widgets ventana 2 ###################################
#############################################################################
######### grafica 1 ###########################################################

    lbl_30 = Label(p2, text="p", font=("Arial Bold", 10))
    lbl_30.grid(column=2, row=2)
    txt_30 = Entry(p2,width=10)
    txt_30.insert(0, "0.01")
    txt_30.grid(column=3, row=2)
    lbl_30_1 = Label(p2, text="(%)", font=("Arial Bold", 10))
    lbl_30_1.grid(column=4, row=2)

    lbl_35 = Label(p2, text="d", font=("Arial Bold", 10))
    lbl_35.grid(column=6, row=2)
    txt_35 = Entry(p2,width=10)
    txt_35.insert(0, "500")
    txt_35.grid(column=7, row=2)
    lbl_35_1 = Label(p2, text="(Km)", font=("Arial Bold", 10))
    lbl_35_1.grid(column=8, row=2)

    Label(p2, text="|", font=("Arial Bold", 15)).grid(column=5, row=2)
    Label(p2, text="|", font=("Arial Bold", 15)).grid(column=5, row=3)
    Label(p2, text="|", font=("Arial Bold", 15)).grid(column=5, row=4)

    Label(p2, text="||", font=("Arial Bold", 15)).grid(column=9, row=2)
    Label(p2, text="||", font=("Arial Bold", 15)).grid(column=9, row=3)
    Label(p2, text="||", font=("Arial Bold", 15)).grid(column=9, row=4)
    Label(p2, text="||", font=("Arial Bold", 15)).grid(column=9, row=5)

    Label(p2, text="-------------------------------------------------", font=("Arial Bold",
15)).grid(column=2, row=5, columnspan = 7)

    lbl_31 = Label(p2, text="f_ini", font=("Arial Bold", 10))
    lbl_31.grid(column=2, row=3)
    txt_31 = Entry(p2,width=10)
    txt_31.insert(0, "0.5")
    txt_31.grid(column=3, row=3)
    lbl_31_1 = Label(p2, text="(GHz)", font=("Arial Bold", 10))
    lbl_31_1.grid(column=4, row=3)

    lbl_32 = Label(p2, text="f_end", font=("Arial Bold", 10))
    lbl_32.grid(column=6, row=3)
    txt_32 = Entry(p2,width=10)
    txt_32.insert(0, "300")
    txt_32.grid(column=7, row=3)
    lbl_32_1 = Label(p2, text="(GHz)", font=("Arial Bold", 10))
    lbl_32_1.grid(column=8, row=3)

    lbl_33 = Label(p2, text="f_step", font=("Arial Bold", 10))
    lbl_33.grid(column=2, row=4)
    txt_33 = Entry(p2,width=10)
    txt_33.insert(0, "0.5")
    txt_33.grid(column=3, row=4)
    lbl_33_1 = Label(p2, text="(GHz)", font=("Arial Bold", 10))
    lbl_33_1.grid(column=4, row=4)

    lbl_34 = Label(p2, text="w", font=("Arial Bold", 10))
    lbl_34.grid(column=6, row=4)
    txt_34 = Entry(p2,width=10)
    txt_34.insert(0, "0")
    txt_34.grid(column=7, row=4)
    lbl_34_1 = Label(p2, text="(un)", font=("Arial Bold", 10))
    lbl_34_1.grid(column=8, row=4)
```

```
    button = Button(p2, text="Ayuda", command=ayuda1)
    button.grid(column=3, row=6)

    Label(p2, text="||", font=("Arial Bold", 15)).grid(column=9, row=6)

    button = Button(p2, text="Gráfica", command=grafica1)
    button.grid(column=7, row=6)

    ######### grafica 2 ############################################################3

    lbl_36 = Label(p2, text="p", font=("Arial Bold", 10))
    lbl_36.grid(column=10, row=2)
    txt_36 = Entry(p2,width=10)
    txt_36.insert(0, "0.01")
    txt_36.grid(column=11, row=2)
    lbl_36_1 = Label(p2, text="(%)", font=("Arial Bold", 10))
    lbl_36_1.grid(column=12, row=2)

    lbl_37 = Label(p2, text="f", font=("Arial Bold", 10))
    lbl_37.grid(column=14, row=2)
    txt_37 = Entry(p2,width=10)
    txt_37.insert(0, "10")
    txt_37.grid(column=15, row=2)
    lbl_37_1 = Label(p2, text="(GHz)", font=("Arial Bold", 10))
    lbl_37_1.grid(column=16, row=2)

    Label(p2, text="|", font=("Arial Bold", 15)).grid(column=13, row=2)
    Label(p2, text="|", font=("Arial Bold", 15)).grid(column=13, row=3)
    Label(p2, text="|", font=("Arial Bold", 15)).grid(column=13, row=4)

    Label(p2, text="||", font=("Arial Bold", 15)).grid(column=17, row=2)
    Label(p2, text="||", font=("Arial Bold", 15)).grid(column=17, row=3)
    Label(p2, text="||", font=("Arial Bold", 15)).grid(column=17, row=4)
    Label(p2, text="||", font=("Arial Bold", 15)).grid(column=17, row=5)

    Label(p2, text="-------------------------------------------------", font=("Arial Bold",
15)).grid(column=10, row=5, columnspan = 7)

    lbl_38 = Label(p2, text="d_ini", font=("Arial Bold", 10))
    lbl_38.grid(column=10, row=3)
    txt_38 = Entry(p2,width=10)
    txt_38.insert(0, "1")
    txt_38.grid(column=11, row=3)
    lbl_38_1 = Label(p2, text="(Km)", font=("Arial Bold", 10))
    lbl_38_1.grid(column=12, row=3)

    lbl_39 = Label(p2, text="d_end", font=("Arial Bold", 10))
    lbl_39.grid(column=14, row=3)
    txt_39 = Entry(p2,width=10)
    txt_39.insert(0, "6000")
    txt_39.grid(column=15, row=3)
    lbl_39_1 = Label(p2, text="(Km)", font=("Arial Bold", 10))
    lbl_39_1.grid(column=16, row=3)

    lbl_40 = Label(p2, text="d_step", font=("Arial Bold", 10))
    lbl_40.grid(column=10, row=4)
    txt_40 = Entry(p2,width=10)
    txt_40.insert(0, "1")
    txt_40.grid(column=11, row=4)
    lbl_40_1 = Label(p2, text="(Km)", font=("Arial Bold", 10))
    lbl_40_1.grid(column=12, row=4)

    lbl_41 = Label(p2, text="w", font=("Arial Bold", 10))
    lbl_41.grid(column=14, row=4)
    txt_41 = Entry(p2,width=10)
    txt_41.insert(0, "0")
    txt_41.grid(column=15, row=4)
    lbl_41_1 = Label(p2, text="(un)", font=("Arial Bold", 10))
    lbl_41_1.grid(column=16, row=4)

    button = Button(p2, text="Ayuda", command=ayuda2)
    button.grid(column=11, row=6)

    Label(p2, text="||", font=("Arial Bold", 15)).grid(column=17, row=6)
```

```
    button = Button(p2, text="Gráfica", command=grafica2)
    button.grid(column=15, row=6)


    button = Button(p2, text="OK", command=cierraSecundaria)
    button.grid(column=18, row=20)


#############   Fin Widgets ventana 2   ###################################
###############################################################################

    nb.add(p1, text ="Propagación por cielo despejado")
    nb.add(p2, text ="Gráficas")

    win.mainloop()
```

REFERENCIAS

[1] *Procedimiento de predicción para evaluar la interferencia entre estaciones situadas en la superficie de la Tierra a frecuencias superiores a unos 0,1 GHz.* Rec. UIT-R P.452-16

[2] John A. Richards, *Radio Waves Propagation.* Springer.

[3] Abdollah Ghasemi, Ali Abedi, Farshid Ghasemi, *Propagation Engineering in Radio Links Design*. Springer.

[4] Charles R. Severance, *Python para todos.*

[5] *Atenuación debida a los gases atmosféricos*. Rec. UIT-R P.676-6

Índice de Códigos

3.1 Ejemplo del uso del instalador de paquetes pip 13
3.2 Ejemplo de ventana con Widgets 14
4.1 Comando para inicar la aplicación 19

GLOSARIO

ITU: International Telecommunications Union 3

SRTM: Shuttle Radar Topography Mission 14