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

Simulador en Python para calcular las pérdidas por difracción debidas a obstáculos según la Rec. ITU-R P.526

Autor: Juan Antonio Castro Fernández
Tutor: Susana Hornillo Mellado

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

Sevilla, 2013

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

Simulador en Python para calcular las pérdidas por difracción debidas a obstáculos según la Rec. ITU-R P.526

Autor:
Juan Antonio Castro Fernández

Tutor:
Susana Hornillo Mellado
Profesora Contratada Doctora

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

Trabajo de Fin de Grado: Simulador en Python para calcular las pérdidas por difracción debidas a obstáculos según la Rec. ITU-R P.526

Autor: Juan Antonio Castro Fernández
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:

El Secretario del Tribunal

Sevilla, 2020

Agradecimientos

Quiero agradecer haber llegado aquí a aquellos profesores que me hicieron ver lo apasionante que era estudiar mi carrera y hacer lo que hacían, a todos los compañeros que alguna vez han compartido horas de estudio, dolores de cabeza y risas; a mí por no haberme abandonado nunca y haber confiado siempre en que podría y, sobre todo y con mención especial, a aquellos incondicionales que siempre me han apoyado independientemente de cuáles fueran las circunstancias.

Juan Antonio Castro Fernández
Sevilla, 2020

Resumen

El estudio de los fundamentos de la radiocomunicación y el comportamiento de las ondas electromagnéticas en su propagación por cualquier medio son una parte indispensable en nuestra formación como graduados en la ingeniería de telecomunicación. De igual forma lo es disponer de medios y recursos para que este estudio pueda ser eficiente y accesible a todos los estudiantes.

En este trabajo se abordan ambos aspectos, ya que se expondrán los conocimientos estudiados sobre propagación por difracción y se planteará una solución que permita en un futuro disponer de una herramienta gratuita que ofrezca a los estudiantes de la escuela la posibilidad de poner en práctica los conocimientos que se vayan adquiriendo sobre la materia de manera accesible y cómoda.

Índice

ÍNDICE DE FIGURAS

Notación

TX	Transmisor
RX	Receptor
sen	Función seno
tan	Función tangente
$\cos^{-1}$	Funcion arcocoseno
cos	Funcion coseno
:	Tal que
<	Menor o igual
>	Mayor o igual

1 INTRODUCCIÓN

La elaboración de un software que continúe el camino emprendido por algunos compañeros y siente las bases necesarias para que otros nuevos sigan explorándolo ha sido una idea ilusionante desde el día en que se planteó. Implementar una solución que pudiera ser parte de otra mayor, que fuera gratuita y pudiera en un futuro prestar servicio a alumnos de la escuela era una idea emocionante para cualquiera que disfrute un reto y desee aportar su grano de arena a la comunidad de la escuela. Sin embargo, basta con dar un paso para descubrir que cualquier planificación, ideal o esquema puede venirse abajo ante el más mínimo cambio. Y esto es justamente lo que ocurrió cuando, una vez planteados los objetivos, se inició el estudio de las soluciones a los problemas planteados.

En este capítulo se definirán el punto de partida y los objetivos originales para posteriormente exponer los escollos encontrados y cómo hubo que sortearlos para poder aproximarse lo máximo posible a la meta establecida.

1.1 Objetivo original

El punto de partida de este trabajo de fin de grado es el que se elaborase en 2018 bajo el nombre Simulación en Python del efecto de múltiples obstáculos sobre un radioenlace según la Rec. ITU-R P.526[1]. En él, se estimaban las pérdidas de un radioenlace en el que se detectasen hasta dos obstáculos, nutriéndose para tal fin de la aplicación Elevation API[2] provista gratuitamente por Google. Esta aplicación ofrecía al usuario, dados dos pares de coordenadas, una imagen del perfil geográfico entre ambos puntos y una estructura de datos en la que podía consultarse a demanda tanto la cota del punto elegido como las coordenadas del mismo. Así, tomando debidamente el número de muestras que fueran necesarias podía obtenerse perfil y cotas con la precisión suficiente como para estimar las pérdidas que hubiera entre unos supuestos transmisor y receptor.

El objetivo que se planteó en un principio fue el de terminar de implementar los métodos recogidos en la recomendación[3] y, dado que estos eran escasos y a priori de fácil implementación, valorar la opción de proveer al usuario de una interfaz gráfica.

Sin embargo, apenas un año después de la finalización del trabajo original[1] la aplicación dejó de ser gratuita, por lo que la fuente que proporcionaba los datos ya no estaba disponible y no era posible continuar desarrollando los métodos y algoritmos que quedaban pendientes en la recomendación[3]. Encontrar un camino para obtener estos datos de forma fiable se convirtió por tanto en la primera tarea que cumplir de forma obligada.

La solución que se encontró no era en absoluto sencilla, y supuso tal carga de trabajo que trajo consigo un cambio forzado en las metas que se hubieran propuesto originalmente.

1.2 Nuevos objetivos

Ya que la implementación de una solución supuso un trabajo extra inesperado, hubo que descartar algunas de las metas fijadas, y la primera en ser eliminada fue la de crear una interfaz gráfica. Con el tiempo se vio que tampoco se podrían implementar al completo todos los algoritmos de la norma, limitándose estos a aquellos que afectaban directamente a los obstáculos de un radioenlace y obviando otros como los que estudian específicamente el efecto de pantallas[4].

Esto no significa en absoluto que este trabajo haya estado carente de material, porque como se verá en los próximos capítulos, el desarrollo y corrección de una herramienta que aporte perfil, datos y precisión con garantías ha ocupado la mayor parte del tiempo de trabajo empleado en él.

Algunas de las razones que han supuesto una inversión adicional de tiempo es el cambio en el modelo de datos con que se trabaja. Elevation API[2] ofrecía una estructura matricial en la que jugando con filas y columnas se podía obtener latitud, longitud o cota para un punto cualquiera, mientras que la solución que finalmente se implementó no aportaba tanta facilidad, por lo que hubo que trabajar con distintas cadenas que contenían cada uno de estos distintos datos por separado. Esto obligó a reestructurar todos los métodos de cálculo y procesamiento de datos ya que los originales no eran compatibles, lo que al final supuso rehacer el trabajo de partida[1] prácticamente de cero.

Aunque era necesario rediseñar el software y la versión de Python[5] usada en el programa de partida (2.7) quedaba atrás, se decidió mantener el mismo lenguaje por su potencia y la gran variedad de recursos gratuitos que ofrece, subiendo en cambio la versión a la 3.7.

En adelante se hará rereferncia a nombres de métodos y estructuras de datos con los que se han trabajado para ir resolviendo los distintos problemas que se han ido encontrando. Se invita al lector a acudir tanto al anexo donde se encuentra el código fuente como al capítulo 6 en el que se detallan cada uno de los métodos empleados.

2 OBTENCIÓN DE DATOS DE ELEVACIÓN

Vista la necesidad de disponer de una herramienta que proporcione el perfil terrestre entre dos puntos seleccionados, así como los datos de elevación en el perfil entre ellos, se centró la primera etapa de trabajo en estudiar distintas tecnologías y recursos que permitieran alcanzar tal fin.

En este capítulo se expondrán todas ellas, detallando tanto los procedimientos que se siguieron como los motivos que llevaron o no a su aceptación.

2.1 Soluciones investigadas

Una vez se hubo descartado el uso de Elevation API[2] (aplicación usada en el trabajo original), se centró la primera etapa del desarrollo en buscar una alternativa a la misma para así disponer de una herramienta que proporcionara unas funcionales similares a la anterior. A continuación, se presentarán las diferentes soluciones que se investigaron en esta fase, pero antes sería preciso hablar del modelo SRTM, en cuya base de datos se basan todas las soluciones estudiadas.

2.1.1 Modelo SRTM

El modelo SRTM (*Shuttle Radar Topography Mission)* es un proyeto de la NASA[6] que recoge datos de elevación de toda la superficie terrestre. Fue anunciado en el año 2000, y en aquel entonces estaba previsto que estuviera terminado a finales del año 2015. Estos datos están recogidos en diferentes grados de precisión, según los arcosegundos usados en el modelo. Actualmente, las bases de datos disponibles tienen una precisión de 3 arcosegundos (90 metros) para toda la superficie terrestre, a excepción de Estados Unidos, que dispone de datos con una precisión de 1 arcosegundo (10 metros).

El acceso a estos datos es totalmente gratuito, y disfrutan de un amplio abanico de opciones de uso. En los siguientes puntos se abordarán las herramientas basadas en estos modelos que se contemplaron durante el desarrollo de esta primera fase.

2.1.2 Herramientas basadas en SRTM

2.1.2.1 Elevation 1.0.6

Elevation[7] es una API proporcionada por Python, que permite hacer una llamada a la API, con dos coordenadas como parámetros. Esta devuelve el resultado en un archivo en formato TIFF. El formato TIFF se usa para compresión de imágenes sin pérdida, y tiene sus ventajas frente a otros como JPEG, cuya ventaja radica en el poco peso de sus imágenes y la rápida apertura de las mismas, con la consecuente pérdida de calidad. El hecho de que estos resultados vengan en formato TIFF dificulta el tratamiento de los datos a la hora de llevarlos a una gráfica, o usarlos para cálculos posteriores.

Teniendo en cuenta estas desventajas, se descartó su uso.

2.1.2.2 Archivos en formato *hgt*

Habiendo rechazado el uso de la API de Python, se intentó trabajar directamente con los archivos SRTM. La base de datos completa consta de aproximadamente 55000 archivos para los datos de toda la superficie terrestre, en sus diferentes precisiones, como ya se comentaba en el apartado 1.1.1. Estos archivos están disponibles en formato *hgt*. Cada archivo consta de una matriz, cuyas entradas son las diferentes cotas para la coordenada correspondiente, de manera que cada zona de superficie está mapeada a través de matrices divididas en distintos archivos. En el siguiente apartado se detallará el formato *hgt*.

2.1.2.2.1 Estructura de los archivos *hgt*

Los archivos *hgt* (abreviación de *height*, del inglés, altura) definen en su nombre la cobertura que tienen, de forma que sea el nombre de un archivo *N54E117.hgt*, se tiene este abarcará todas las elevaciones desde 54º N 117º E hasta 55ºN 118º E. Estos archivos están disponibles, como se menciona en el apartado 1.1.1, en dos resoluciones distintas:

- $SRTM_1$ (muestras de 1 arcosegundo)
- $SRTM_3$ (muestras de 3 arcosegundos)

Si esto se lleva a unidades angulares se tienen los siguientes cálculos:

$$1\ segundo = \frac{1}{60}\ minutos = \frac{1}{3600}\ grados$$

$$3\ segundos = \frac{1}{20} minutos = \frac{1}{1200} grados$$

Así, partiendo del fichero de ejemplo anterior, y suponiendo que este fuera de 3 arcosegundos, se tiene que este abarcaría las siguientes latitudes:

$$55º, 55º + \frac{1}{1200}º, 55º + \frac{1}{1200}º, \dots, 55º + \frac{1200}{1200} = 56º$$

Si esto mismo aplica a las longitudes se tiene una matriz final de 1201x1201 elevaciones. Para el caso de un fichero de 1 arcosegundo, esta matriz sería de 3601x3601 elevaciones. En la siguiente figura se representa un ejemplo de estructura para un fichero $SRTM_3$ cuyo nombre es N27E086.hgt, de donde se extrae que abarca desde 27º N 86 º E hasta 28 º N 87º E.

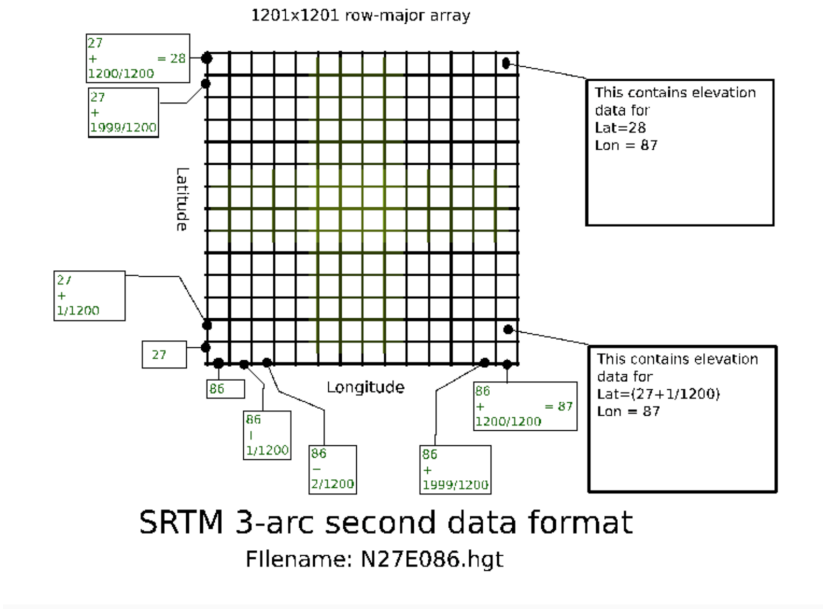


Figura 1 - Estructura de los archivos *hgt*[8]

Si se observa la entrada (1,1), esta corresponde a 28 $\left(27 + \frac{1200}{1200}\right)$, y se ha llegado a ese valor aumentando gradualmente desde la entrada (1201,1), que corresponde a 27º N 86º E. Se concluye por tanto observando la imagen que:

- Fijando una fila se mantiene constante la latitud y se varía la longitud.
- Fijando una columna se mantiene constante la longitud y se varía la latitud.

Para facilitar el trabajo con este formato de archivo se dispone de métodos[9] para su tratamiento, que truncan las coordenadas proporcionadas en las unidades, y usan los decimales restantes para seleccionar la entrada de la matriz adecuada. Aunque la fuente anteriormente citada proporciona documentación sobre los métodos necesarios, su funcionamiento no era el esperado. Para comprobar que los datos obtenidos son correctos, se comparan con los radioenlaces usados en el trabajo de partida[1], con el fin de tener una referencia fiable sobre las medidas. La forma de proceder es realizar una interpolación entre las coordenadas de transmisor y receptor y obtener, para cada una de las coordenadas, la elevación correspondiente. Para conseguir esto, se hace uso de un método que obtiene todas las coordenadas necesarias entre dos puntos dados así como la elevación de cada uno de ellos.

Una vez calculados todos los puntos a lo largo del radioenlace, se procede a leer la elevación correspondiente de los archivos *hgt*, Esto se hacía invocando a unos métodos que obtenían el nombre del archivo *hgt* según las coordenadas, truncando como se ha explicado en los párrafos anteriores. De este archivo se obtenía la cota exacta usando la parte de cimal de las coordenadas, accediendo a la celda exacta de la matriz.

Pero como se hubo comentado anteriormente, estos resultados no solo no arrojan la precisión necesaria, sino que distan mucho de ser una aproximación al perfil del radioenlace.

Tras probar a obtener la base de datos de varias fuentes distintas e intentar afinar la precisión de los métodos sin éxito, se descartó su uso.

2.1.2.3 Open-elevation Service

Open-elevation es un servicio gratuito proporcionado por Amazon[10] y que permite obtener la elevación del terreno haciendo una llamada a su API a través de una consulta en el navegador. La base de datos de la que se dispone está facilitada por Mapzen[11], y este servicio permite trabajar sin tener la base de datos descargada en el entorno local accediendo a ella a través de consultas, o descargándola mediente unos scripts. El peso total de la base de datos es de aproximadamente 200Gb, por lo que por falta de recursos se optó por trabajar con consultas a la base de datos.

Ya sea de un modo u otro, la aplicación viene empaquetada en una imagen de Docker[12], y para su uso es necesariapor tanto la instalación de Docker y el montado del contenedor. Antes de entrar en detalles, se explicará qué es Docker, cómo funciona, y en qué consiste el concepto de contenedor.

2.1.2.3.1 Conceptos sobre Docker

Docker es un proyecto de código abierto que facilita el despliegue de aplicaciones, automatizándolo y ofreciendo una capa de abstracción, de manera que facilita que estas aplicaciones se desplieguen en distintos sistemas operativos sin necesidad de cambios en ella. Este despliegue se hace mediante contenedores[13], que son encapsulaciones autocontenidas donde se incluyen todos los programas y dependencias necesarias para que una aplicación funcione.

Un esquema que represente cómo funciona Docker sería el siguiente:

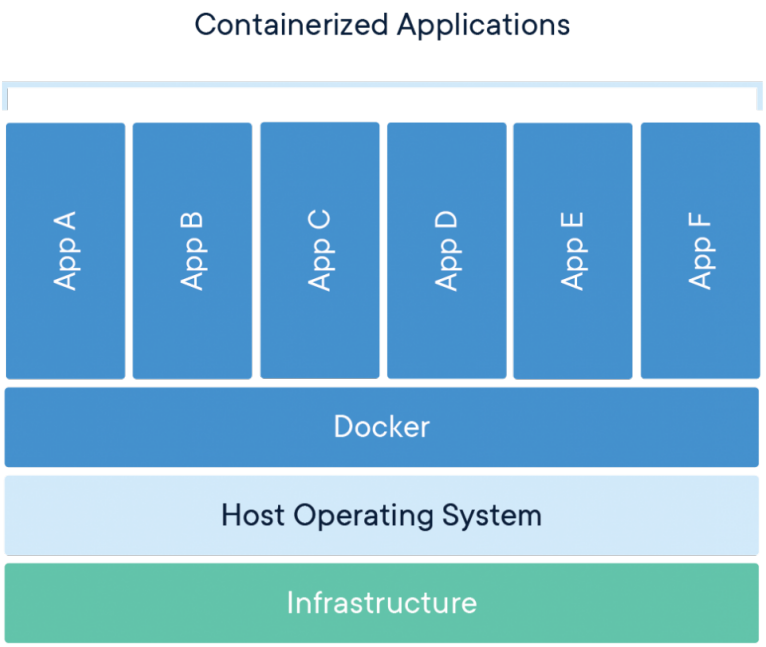


Figura 2 - Funcionamiento de Docker[12]

Es de esta forma que se consigue levantar la aplicación que accederá a la base de datos de la elevación terrestre. El uso de esta aplicación en un contenedor tiene la ventaja de que una futura ampliación de este trabajo o integración con otros módulos no requerirá una inversión de tiempo y recursos para tener disponible la base de datos.

2.1.2.3.1 Open-elevation Service en Docker

Una vez comprendido Docker y cómo usarlo, se procede a obtener la imagen de la aplicación de los repositorios de Docker. Siguiendo los pasos de la propia documentación se levanta el contenedor, que se mapea al puerto 3000 de manera que haciendo una llamada a localhost:3000 se accede a la aplicación.

Para poder acceder a los datos, es necesario pasar como parámetros las coordenadas deseadas (con la precisón necesaria), de forma que una llamada exitosa a la aplicación sería:

http://localhost:3000?lat=12.3&lng=45.6

Esto devuelve el dato de la elevación en forma de objeto JSON, por lo que basta con leerlo y convertirlo al tipo necesario para su tratamiento y representación. En la siguiente imagen se muestra el resultado que se obtiene al hacer una llamada a la API desde el navegador:

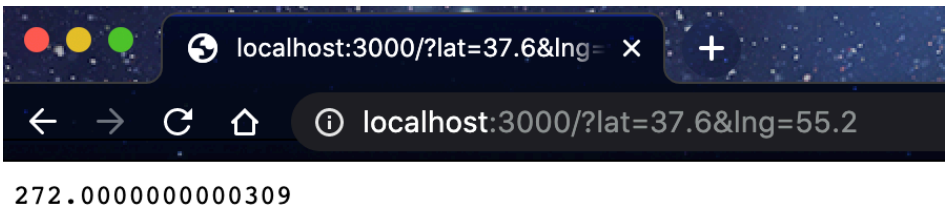


Figura 3 - Resultado devuelto por la API en el navegador.

Esta vez, los datos obtenidos son válidos y se obtienen en un formato que favorece trabajar con los fines propuestos, de forma que en el siguiente capítulo se abordará cómo se consigue una representación fiable del perfil geográfico.

3 REPRESENTACIÓN GRÁFICA

Una vez conseguidos los datos, el siguiente paso es representarlos en una gráfica junto con una simulación del radioenlace, lo que incluye representación gráfica de las antenas, rayo directo y primer elipsoide de Fresnel. En este capítulo se abordarán los métodos y la lógica seguidos para alcanzar tal objetivo. Para ello, antes de la representación es necesario procesar los datos y adecuarlos a los métodos y medios de los que se dispone en Python.

3.1 Procesamiento y representación a partir de los datos de elevación

En el capítulo anterior se había conseguido obtener la elevación de puntos equidistantes entre las coordenadas de transmisor y receptor. En esta sección se verá el procesamiento de estos datos para conseguir un conjunto de variables y métodos que permitan obtener una representación fiable y que facilite el estudio que este trabajo se propone.

3.1.1 Procesamiento del perfil geográfico

Los primeros datos que se prepararon para su representación fueron los del propio perfil. Ya se disponía de una variable que contuviera los valores de la elevación, y de una variable que proporcionara las diferentes distancias en kilómetros a las que se había muestreado, por tanto era cuestión de representar una frente a otra para obtener el perfil calculado.

Antes de pasar a su representación, se comprueba que se pueda aplicar el modelo de Tierra plana. Para esto, se usa el siguiente método, que calcula la protuberancia en el punto medio del radioenlace, así como en cada uno de los puntos en los que se ha calculado la elevación previamente:

De esta forma, si la protuberancia terrestre en el punto medio es mayor que 5 metros, se aplica una corrección calculando la protuberancia terrestre en los mismos puntos en los que se habían obtenido las elevaciones, de acuerdo con la siguiente fórmula:

$$f(x) = \frac{x \cdot (d-x)}{2kRo} \qquad (1)$$

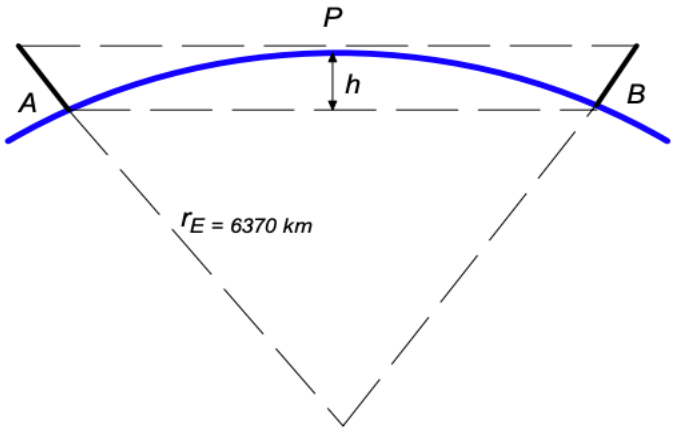


Figura 4 - Protuberancia terrestre[14]

Donde *x* es el punto donde se quiere calcular la protuberancia, *d* es la distancia del radioenlace, *k* es el factor de radio efectivo, y *Ro* el radio de la Tierra. Así, representando los datos obtenidos (corregidos o no) frente a los puntos muestreados en el radioenlace se obtiene el perfil geográfico. Para esta representación se ha simulado un radioenlace entre Monesterio y Pedroso. A continuación se muestra un ejemplo ilustrativo con ambas localizaciones para clarificar el proceso:

Monesterio:
- Latitud: 38.0784
- Longitud: -6.2800

Pedroso:
- Latitud: 37.8570
- Longitud: -5.7699

En la siguiente Figura 3 puede verse su ubicación geográfica con Google Earth[15]:

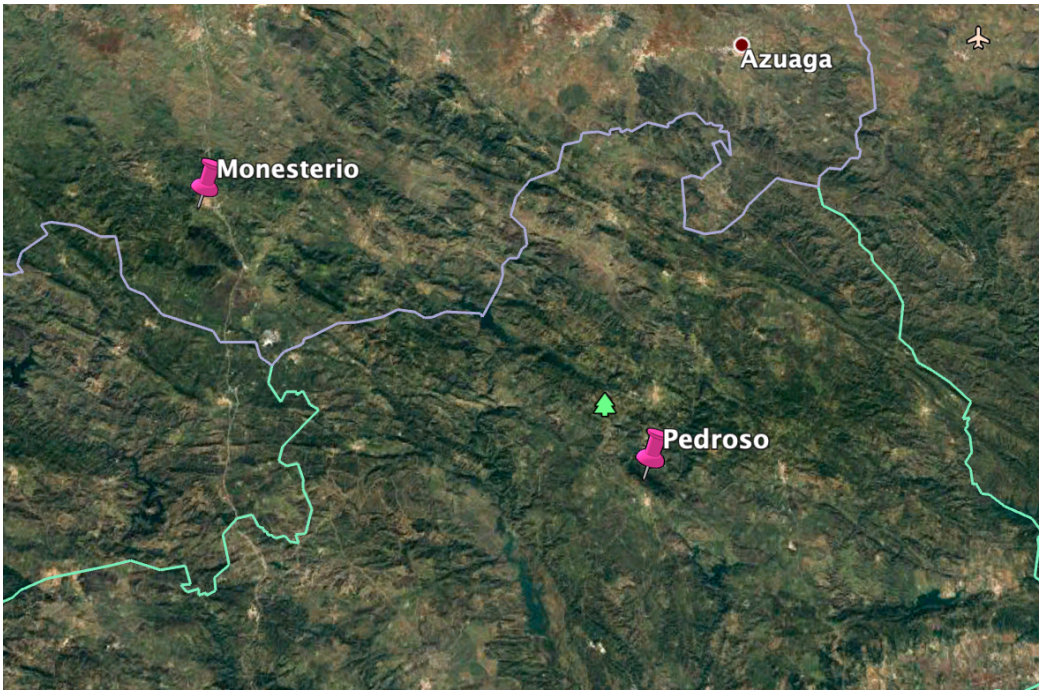
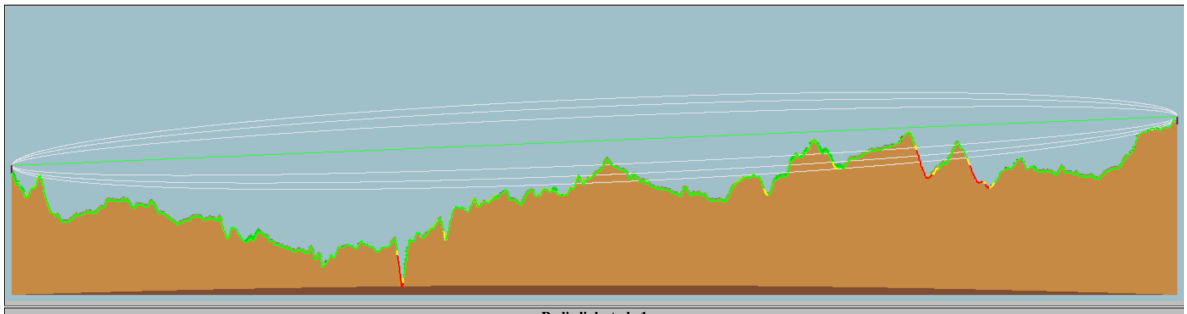


Figura 5 - Enlace entre Monesterio y Pedroso

Para obtener el perfil de elevación se ha usado la herramienta de software libre Radio Mobile[16]. Esta herramienta facilita un simulador de radioenlaces que proporciona tanto el perfil de elevación como los cálculos para pérdidas de propagación, difracción y otros modelos. Permite configurar parámetros como ganancias, potencia de transmisión, sensiblidad etc, que hacen de esta herramienta un gran apoyo para los aficionados a estos estudios. Gracias a su condición de software libre, así como a las distintas funcionalidades que ofrece se ha eligido como herramienta con la que tener una referencia fiable de datos y comprarlos con los obtenidos en este trabajo.



Radio link study 1			
pedroso (1)		(2) monesterio	
Latitude	37.857000 °	Latitude	38.078400 °
Longitude	-5.769900 °	Longitude	-6.280000 °
Ground elevation	654.7 m	Ground elevation	847.4 m
Antenna height	30.0 m	Antenna height	30.0 m
Azimuth	298.99 TN | 300.60 MG °	Azimuth	118.68 TN | 120.45 MG °
Tilt	-0.01 °	Tilt	-0.45 °

Figura 6 - Perfil de elevación para el radioenlace Pedroso-Monesterio

Finalmente, se comprueba que el resultado obtenido con la representación de los datos extaídos de la base de datos frente a la distancia es igual que el que arroja Radio Mobile, salvando la relación de aspecto de las imágenes representadas.

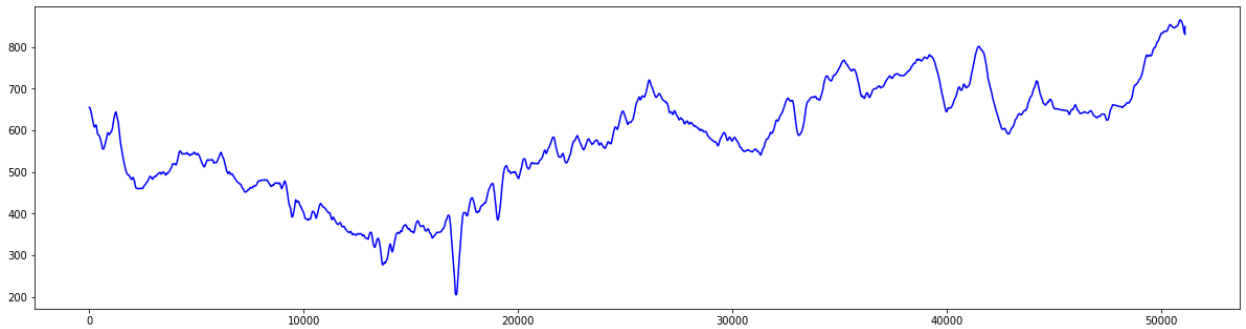


Figura 7 - Perfil geográfico obtenido con el programa diseñado.

El siguiente paso es dibujar las antenas y el rayo directo entre estas. Para dibujar las antenas, se parte de la elevación origen y destino, y se les suma la longitud del mátil de las antenas que previamente haya introducido el usuario. Esta lógica se implementa en el método ***dibuja_antena*** que, con los datos anteriores, dibujará una recta entre dicha elevación (base de la antena) y la elevación final (elevación base más la longitud del mástil) tanto para transmisor como receptor.

Para el rayo, se interpolan los puntos entre origen y destino, de manera que se obtenga una recta que dibujar. La recta se obtiene con el método ***dibuja_rayo***, y luego se representa sobre la misma gráfica que las antenas y el perfil. El resultado final se muestra en la Figura 6:

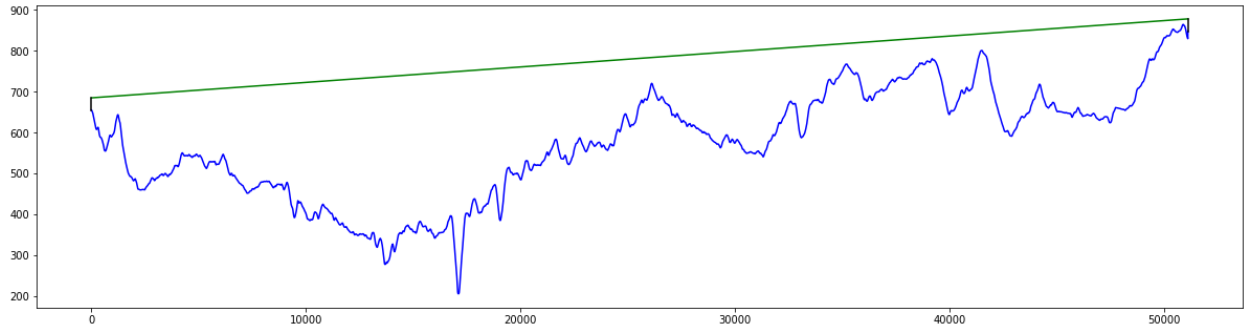


Figura 8 - Perfil con antenas y rayo de visión directa

Con esto, solo queda representar el primer elipsoide de Fresnel. Su cálculo es sencillo ya que se calcula el radio de Fresnel para cada punto de acuerdo con la ecuación disponible en la recomendación en la que se basa este trabajo.

Sin embargo, su representación supone un problema mayor. Con la variable obtenida del apartado anterior son necesarias algunas correcciones antes de pasar a la representación. En primer lugar, hay que sumar la altura final de la antena (cota terrestre más la altura del mástil) que esté más baja. En este caso sería la antena del transmisor, para que así el origen del elipsoide esté situado justo en el extremo de dicha antena.

El siguiente problema viene cuando hay que situar el otro extremo del elipsoide en la antena restante. No basta con calcular el ángulo entre ambas para así girar la figura, ya que esto describiría una circunferencia y no haría coincidir los extremos. En este caso, hay que usar el método ***rotate_around_point_lowperf***, que hace un *reshape* de la variable para hacer coincidir los extremos.

Tras este ajuste, ya se puede representar el elipsoide. Hay que tener en cuenta que los cálculos expuestos solo calculan la mitad superior del elipsoide. Para poder representarlo completamente, sólo hay que restar la altura final anterior, en lugar de sumarla, y rotar con el método comentado en el párrafo anterior. El resultado final es el que se muestra en la Figura 7:

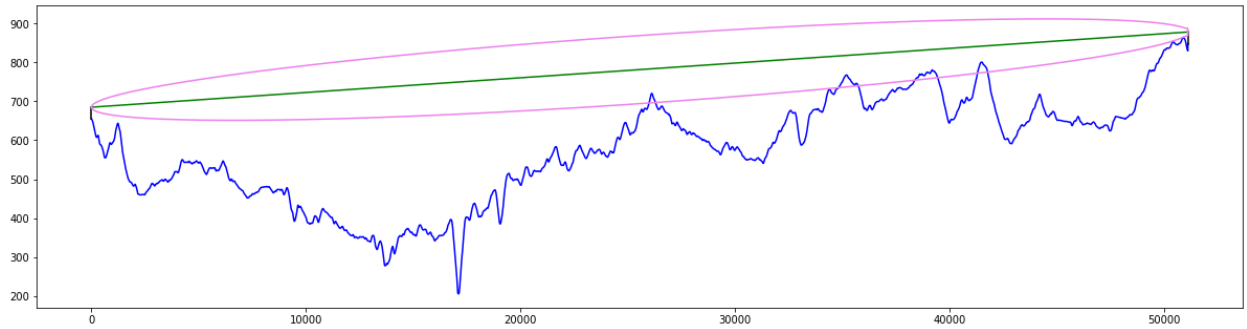


Figura 9 - Representación completa del perfil con antenas, rayo y elipsoide de Fresnel.

Aunque se ha representado el elipsoide situando los extremos en las antenas, hay que considerar que esto es una aproximación de la representación gráfica para facilitar la comprensión del proceso al usuario. La realidad es que las antenas quedan en los focos del elipsoide, y no en los extremos, de manera que una representación fiel a la realidad sería de esta forma:

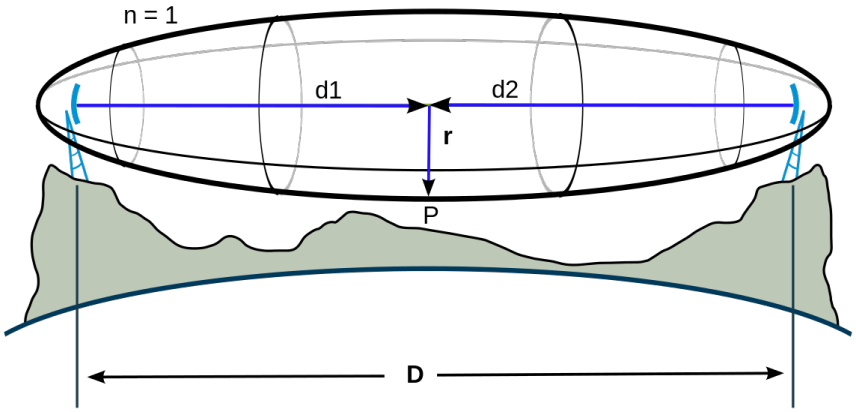


Figura 10 - Representación real del elipsoide de Fresnel[17]

Como apunte final, hay que recordar que esto no es más que una representación gráfica y que es posible que para algunas simulaciones la imagen mostrada presente algún error o desajuste entre los elementos representados, pero a la hora de los cálculos se han prevenido estos posibles errores para que los cálculos realizados con los datos no se vean afectados por estas variaciones.

3.1.2 Procesamiento y representación de los obstáculos

Tras disponer de una representación gráfica tanto del perfil geográfico como del rayo de visión directa y el primer elipsoide de Fresnel, el siguiente paso para poder estimar las pérdidas por difracción es determinar qué accidentes del perfil suponen una interferencia, sea esta constructiva o destructiva, durante la propagación.

Para ello, se elabora un algoritmo que encuentra todos los máximos relativos que haya disponible en la cadena de elevaciones del perfil. De esta forma se obtiene, con una primera batida, una lista de posibles obstáculos que puedan interferir en la propagación. En el siguiente diagrama se muestra dicho algoritmo, en el que cuando un punto del perfil sea mayor que el anterior y menor que el siguiente se almacenará el índice de la cadena. La razón por la que se almacena el índice y no el valor de la elevación del terreno en ese punto es que en futuras operaciones en el programa será necesario acceder, por ejemplo, a la distancia a la que se encuentra el obstáculo del transmisor o al valor del elipsoide de Fresnel en ese mismo punto; es por eso que conociendo los índices de la cadena que son conflictivos se puede acceder a cualquier información necesaria en el punto en cuestión sin más que acceder a los valores salvados por este algoritmo:

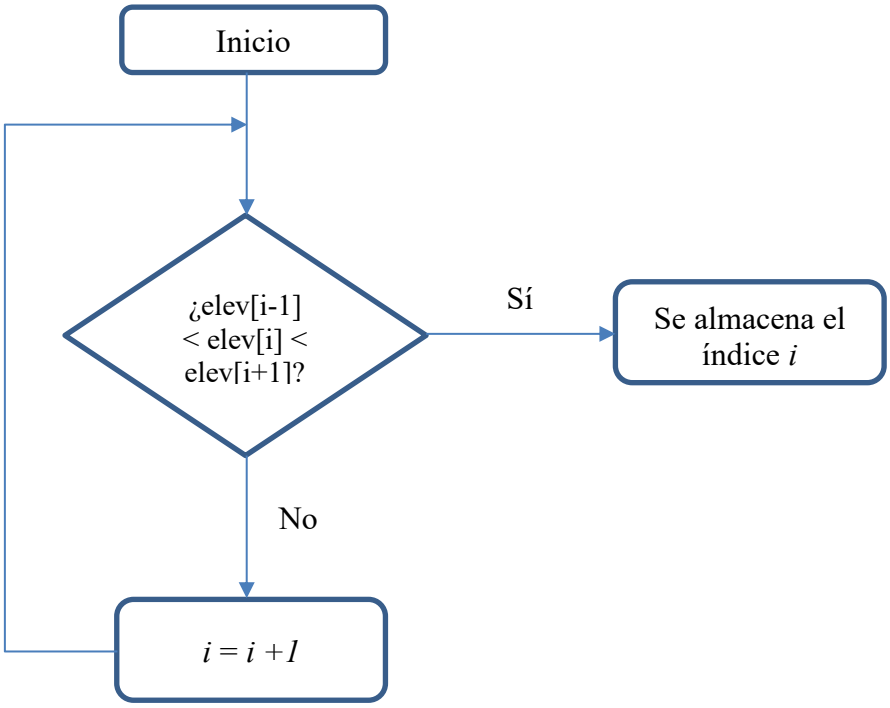


Figura 11 - Diagrama de flujo para selección de máximos relativos

A continuación, hay que comprobar si estos máximos interfieren o no en el primer elipsoide de Fresnel. Para ello se comprueba que el despejamiento en el punto procesado se encuentre dentro de los límites aceptables. El despejamiento se define como la distancia desde el rayo de visión directa hasta la cima del obstáculo. En la siguiente figura se muestra el despejamiento para un obstáculo en arista filo de cuchillo, tanto en valor positvo como negativo:

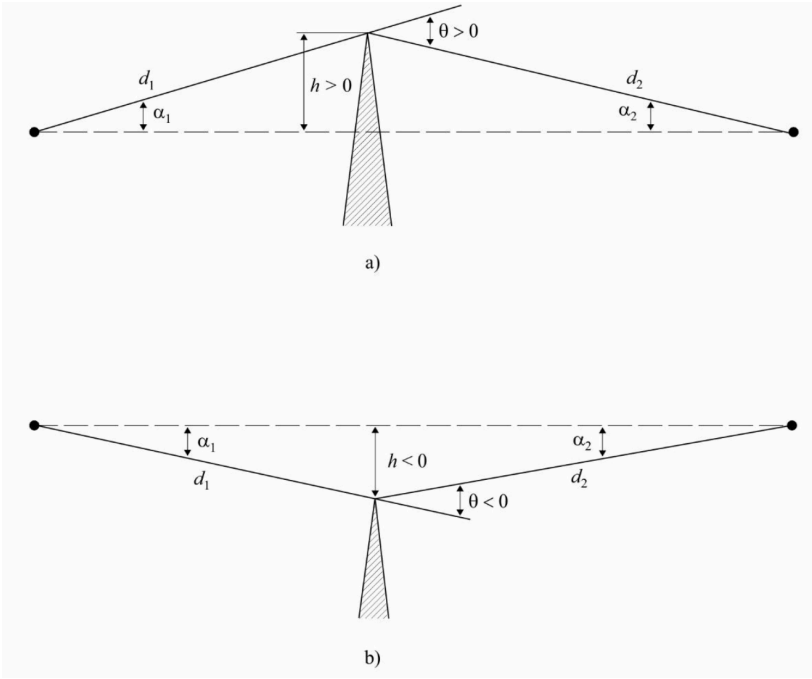


Figura 12 – Despejamiento de un obstáculo[18]

Para determinar la afectación de estos obstáculos en la propagación se evalúa el despejamiento normalizado, de manera que éste se encuentre dentro de intervalo de valores admisibles. El despejamiento normalizado se define como el despejamiento dividido por el valor del radio de la primera zona de Fresnel en el punto analizado:

$$\frac{hi}{R1_i} \tag{2}$$

El intervalo de trabajo se define tal que: $-0.6 < \frac{hi}{R1_i} < 0.5$, lo que se traduce en que:

- Se debe dejar libre al menos el 60% de la primera zona de Fresnel, en su mitad inferior.
- Se debe dejar libre al menos el 50% de la primera zona de Fresnel, en su mitad superior.

Para valores del despejamiento normalizado por debajo de -0.6 el punto será descartado como obstáculo, y para valores superiores a 0.5 se considerará un obstáculo que hace inviable el radioenlace. Para cualquier otro caso, se considera que es un obstáculo que afecta a la propagación y por tanto habrá que estudiar de qué forma afecta a la propagación por difracción.

En el siguiente algoritmo se recoge la lógica implementada que no sólo determinará aquellos valores que están dentro o no de los límites, sino que además proporcionará una salida que al ser procesada determinará si el programa calculará las pérdidas por difracción o no, dependiendo de si hay o no obstáculos y en caso de haberlos, si el radioenlace es viable o no:

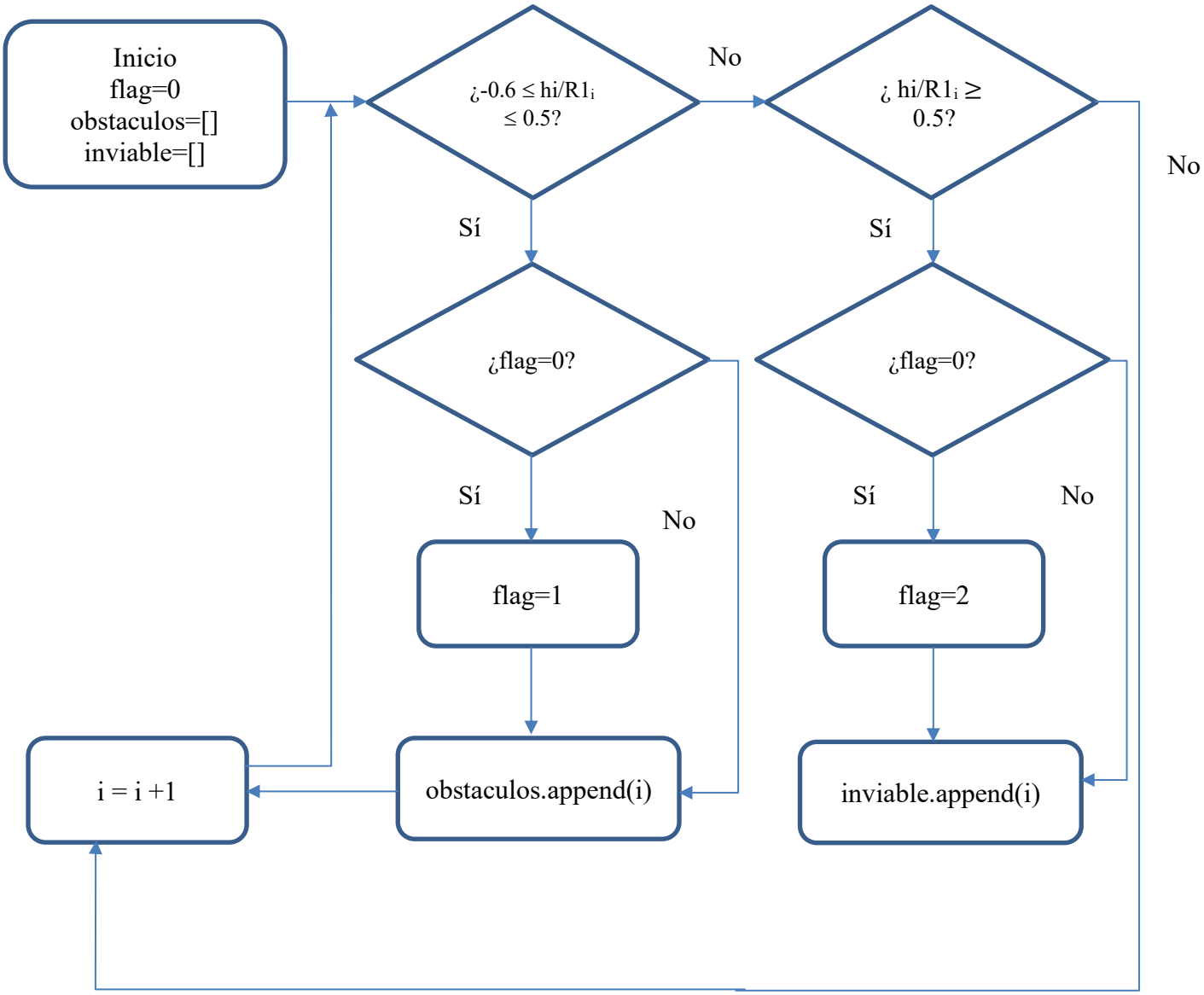


Figura 13 - Diagrama de flujo para clasificación de obstáculos

Con la variable *flag* se consigue determinar en el bloque principal del programa el comportamiento que se seguirá según se explicaba en el párrafo anterior. Finalmente, una vez procesados los datos y se ha obtenido la lista de obstáculos de un tipo u otro sólo queda su representación en la gráfica que se ha ido completando a lo largo del capítulo:

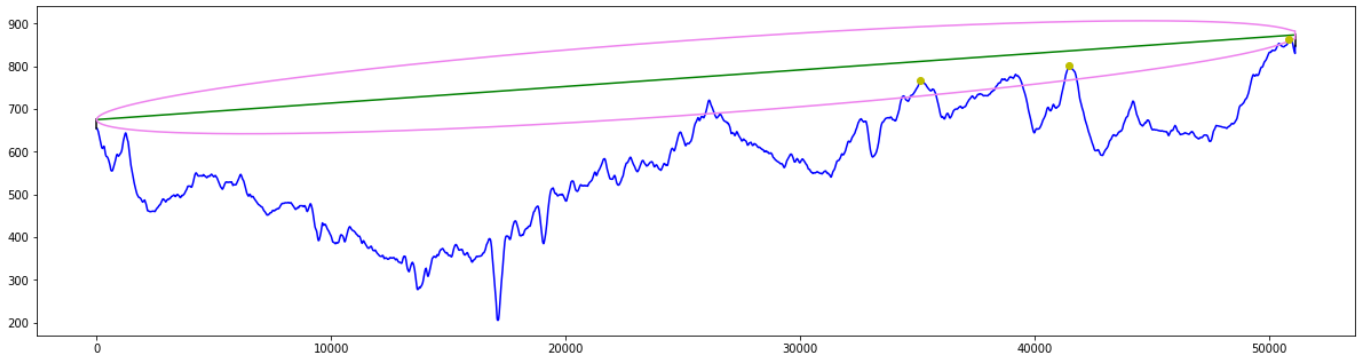


Figura 14 - Perfil geográfico con obstáculos sobre la gráfica

Para la representación se ha elegido el color amarillo para representar aquellos obstáculos con despejamiento normalizado dentro del intervalo de valores admisibles mientras que, en rojo, aunque no se muestre en este ejemplo, se representarán aquellos que hacen inviable el radioenlace. De esta forma, se proporciona al usuario con un simple vistazo la situación del radioenlace y de aquellos obstáculos que se hayan encontrado y se vayan a analizar en la simulación.

4 REFINAMIENTO DE LA HERRAMIENTA GRÁFICA

Los medios de los que se disponían para implementar una herramienta gráfica que sustituyera a la original no eran los más óptimos, ya que tanto el motor gráfico del entorno de desarrollo como la base de datos con que se trabaja no tienen toda la precisión que podría esperarse, por lo que fue necesario realizar un trabajo extra que subsanara todos los posibles errores que surgieran en las distintas casuísticas del programa.

En este capítulo se abordan los distintos errores, tanto gráficos como de cálculo, que se fueron encontrando durante las distintas simulaciones, así como aquellos que se anticiparon de cara a futuras ejecuciones.

Durante el mismo, se hará referencia a nombres de variables que se usan para corregir los errores y tratar los puntos. Se invita al lector a consultar la sección 6.2 en cuya introducción se exponen los nombres de estas variables para facilitar el entendimiento del programa.

4.1 Errores en la resolución

El primer error, y quizás el más simple de los resueltos, que se encontró afectaba a la resolución del perfil geográfico. El programa está diseñado para tomar una muestra cada 5 metros de manera que con una simple división se puede calcular el número de muestras con el que se trabajará en la simulación correspondiente. El problema que genera esta configuración es que para distancias muy cortas una muestra de 5 metros no es suficiente para obtener una precisión aceptable.

En el siguiente ejemplo, tomado del trabajo original[1] se muestra el perfil obtenido para un radioenlace de aproximadamente 350 metros:

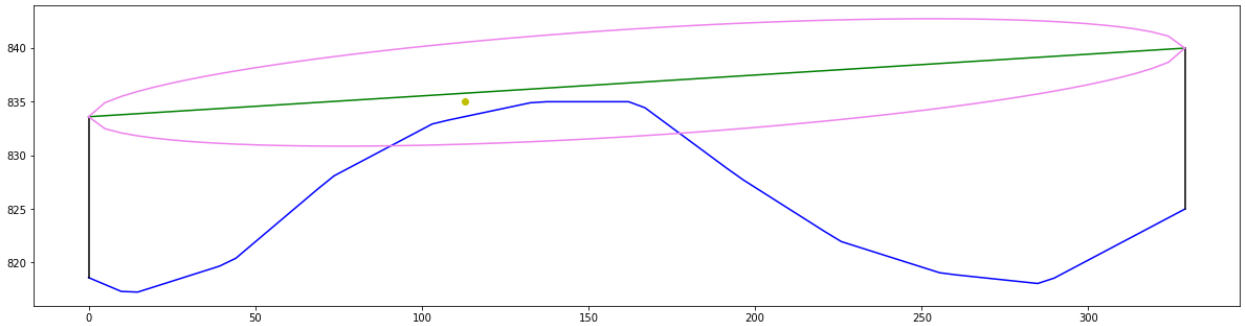


Figura 15 - Perfil con errores en la resolución

Como se puede observar, la herramienta ha detectado y dibujado el obstáculo en un punto que no está en el perfil calculado. Esto no solo tiene implicaciones gráficas, como es obvio, sino que también arroja errores a la hora de calcular las pérdidas por difracción como se podrá comprobar más adelante.

Haciendo un simple cálculo, si se aproxima la distancia entre transmisor y receptor a unos 350 metros y considerando la distancia de muestreo de 5 metros se obtienen un total de 70 muestras, que resultan no ser suficientes para alcanzar una precisión que no arroje errores. La solución pasa por tanto por aumentar el número de muestras en el radioenlace, y esto se puede abordar desde dos perspectivas distintas:

4.1.1 Cálculo de muestras según la longitud del radioenlace

La primera solución que se planteó fue modificar la distancia de muestreo en función de la longitud del radioenlace, de forma que esta se adaptara automáticamente para garantizar una representación gráfica correcta y unos cálculos sin errores. Para ello, había que partir del número mínimo de muestras que se quería en un radioenlace cualquiera, de forma que tras varias simulaciones y estimaciones se planteó la siguiente correspondencia:

- Para longitudes superiores a 1 km: 200 muestras mínimas, con una muestra cada 5 metros.
    - Este valor resultó ser óptimo para radioenlaces en torno a 1km, llegando a alcanzar una precision más que suficiente y sin apenas variación en los cálculos ante modificaciones de la distancia de muestreo, pero no lo fue para distancias muy superiores a ésta.
- Para longitudes entre 0.5 y 1 km: 100-150 muestras, con una muestra cada 5 ó 3.33 metros repectivamente.
    - Las variaciones entre ambas distancias de muestreo apenas suponían un cambio tanto en la representación gráfica como en los cálculos.
- Para longitudes inferiores a 500 metros, 200 muestras.
    - Para distancias inferiores a 500 metros, como es el caso del ejemplo, había que garantizar un mínimo de 200 muestras para que no se produjeran errores significantes en la representación y por tanto en los cálculos.

El inconveniente que plantea esta solución es que no se encuentra una cantidad mínima con la que acotar aquellas distancias superiores a 1km, ya que a medida que aumenta la distancia, también debe hacerlo el número de muestras para que no se acumulen errores en la precisión que a la larga suponga una variación notable de las pérdidas por difracción. Además, aunque se encontrase, aumentar de forma indiscriminada el número de muestras estaría haciendo que el programa no fuera eficiente ya que, para distancias entre 1 y 5 km por ejemplo, se habría sobremuestreado el perfil con el consiguiente aumento de la carga computacional sin apenas notar diferencia en los resultados.

Teniendo en cuenta este inconveniente, además de atender a la correspondencia establecida, y considerando también que la carga computacional no se ve afectada prácticamente nada al variar entre 100 y 200 el número de muestras, se decidió descartar esta solución y optar por la siguiente.

4.1.2 Garantizar un mínimo de muestras para todos los radioenlaces

Ya que, como se comenta en la sección 4.1.1, la carga computacional no se ve afectada por las variaciones entre el número de muestras y que además resulta complejo determinar el número de muestras/km para una distancia cualquiera, se decidió fijar todas las simulaciones a un mínimo

de 200 muestras, de forma que cualquier radionlace con una distancia inferior a 1km tuviera 200 muestras, y para distancias superiores se calculase a razón de una muestra cada 5 metros.

Hay que tener en cuenta que la decisión de fijarlas a 200 ya que no hay afectación en la carga computacional viene dada porque en la sección 4.1.1 las muestras variaban entre 100 y 200. Obviamente, para longitudes muy grandes el número de muestras será lo suficientemente elevado como para que se requiera un tiempo de procesamiento. Por ejemplo, para una distancia de 30 km se tendría un total de 6000 muestras, que sí supondrían una carga considerable. No hay que olvidar que estas muestras se obtienen de hacer peticiones a la aplicación disponible en el contenedor de Docker y que por tanto este tiempo de espera dependerá, además de velocidad del procesador del equipo que aloje la aplicación, de la conexión a internet que haya disponible.

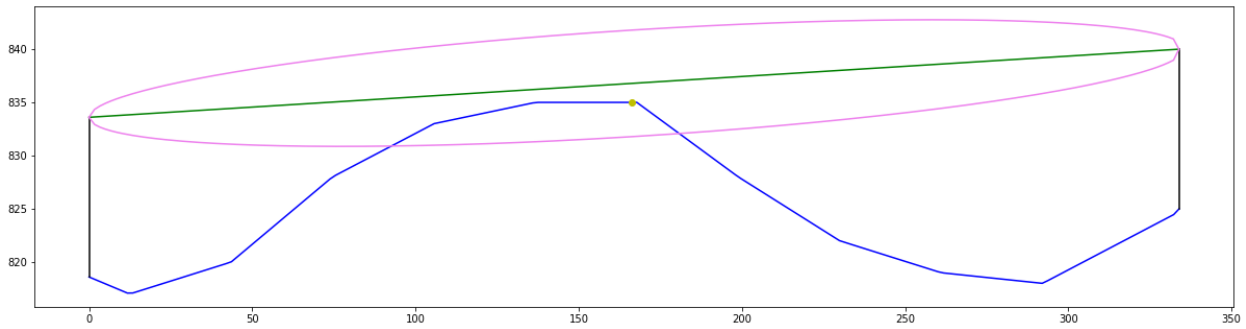


Figura 16 - Perfil tras fijar el mínimo de muestras a 200

Tras esto, se simula de nuevo el mismo perfil y se obtiene el siguiente resultado:

Ahora, el punto amarillo que determina el obstáculo sí se corresponde con la cima de la colina dibujada en la gráfica, por lo que se elimina el error gráfico.

Además, como se comentaba antes, estas variaciones suponían también un error en el cálculo de las pérdidas por difracción. En la siguiente imagen se puede ver el valor obtenido tras la simulación en el primer caso, cuando no se había fijado ningún número mínimo de muestras:

```
Las pérdidas totales por difracción para este radioenlace son 4.013496485518214 dB

Análisis finalizado
```

Figura 17 - Cálculo de pérdidas sin corrección de muestras

Mientras que, tras fijar el número de muestras a 200, se obtiene una corrección de aproximadamente 0.5 dB para el radioenlace.

```
Las pérdidas totales por difracción para este radioenlace son 3.570243041876463 dB

Análisis finalizado
```

Figura 18 - Cálculo de pérdidas con corrección de muestras

Aunque la variación parece insignificante, hay que considerar que estos errores son totalmente impredecibles, y que para distintas elevaciones y distancias la proporción puede ser lo suficientemente grande como para que se obtenga un error mayor. Además, cuanto menor sea la distancia, más propensa a tener errores es la simulación.

Aún así, ya sea con las 70 muestras originales o con las 200 que se tienen tras la corrección, la precisión de esta aplicación es mayor que la que se ofrecía en la herramienta original de Google[2]. Esto se puede comprobar, por ejemplo, para el mismo perfil representado en el ejemplo de esta sección. En la siguiente imagen se comparan el perfil obtenido por la herramienta original frente a representado con los datos de la API elegida.

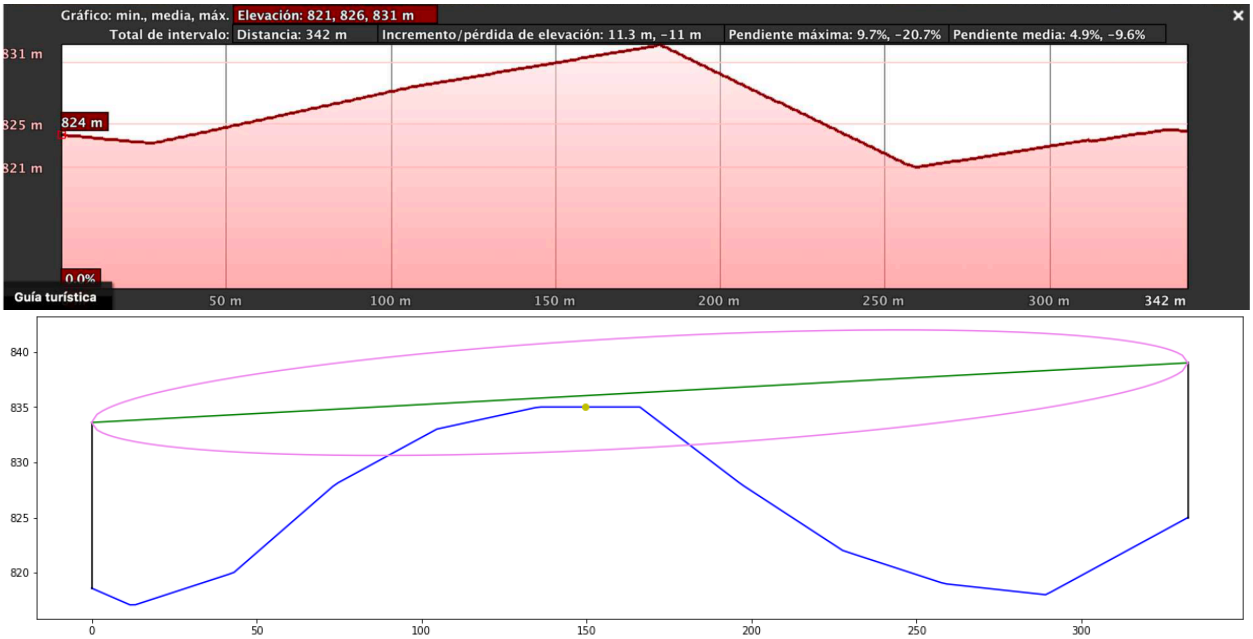


Figura 19 - Comparación de perfiles obtenidos por ambas herramientas.

La mejoría es visible a simple vista. Esto es debido a que la aplicación utiliza distintas fuentes[19] para distintas partes del globo terráqueo, de manera que al tener varios modelos del geoide la precisión aumenta. Por su parte, la aplicación original de Google sólo toma los datos[20] del SRTM, al menos en su versión gratuita, para todo el globo exceptuando Estados Unidos donde dispone de modelos con más precisión.

4.2 Errores en la representación, detección y procesamiento de los obstáculos

La representación de los obstáculos en la gráfica también supone una fuente de errores debidos a las imprecisiones entre los distintos cálculos realizados. La herramienta diseñada dista mucho de ser perfecta, ya que no se disponen de los recursos ni la tecnología necesaria, por lo que es frecuente encontrar errores de cálculo y procesamiento. En esta sección se detallan a qué afectan ambos errores y cuál es la solución propuesta para ellos.

4.2.1 Error en las cercanías del transmisor y receptor

En algunas simulaciones es posible que en las distancias más próximas a transmisor y receptor se produzcan errores aleatorios que vienen generados porque se está trabajado con valores muy

cercanos entre sí y la acumulación de errores en las operaciones y la falta de precisión pueden originar resultados inesperados e incongruentes.

Por ejemplo, en el siguiente perfil se puede observar cómo estando la cota del transmisor por encima del receptor y para una altura de 15 metros en ambos mástiles se obtienen dos obstáculos en la simulación:

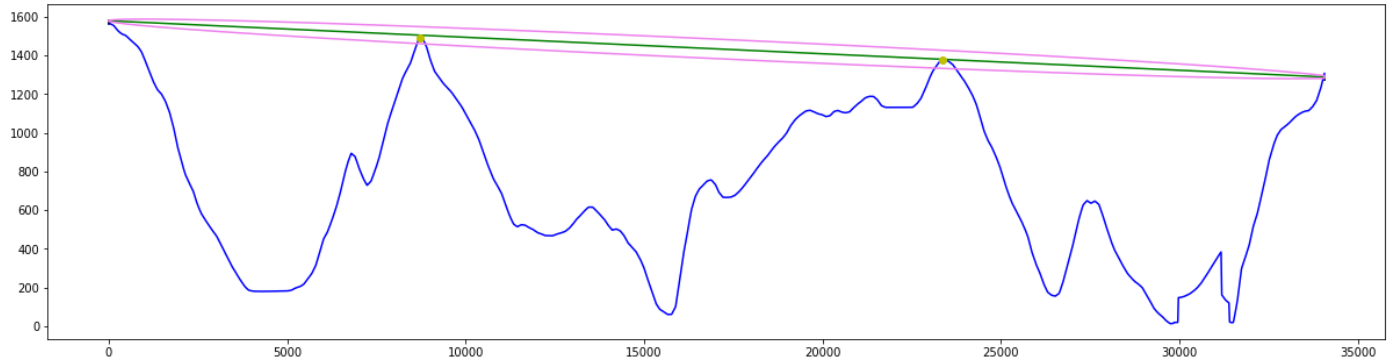


Figura 20 - Perfil con dos obstáculos y antenas con mástiles de 15 metros.

En primer lugar, hay que observar que de por sí, la representación gráfica presenta un error ya que el elipsoide de Fresnel no coincide en su extremo derecho con la antena receptora. Como ya se comentó en secciones anteriores, esta herramienta gráfica no es todo lo precisa que debiera por falta de recursos y tecnología. Aún así, los cálculos realizados con los valores procesados coinciden con los obtenidos analíticamente. Atendiendo luego a los dos obstáculos, cabe esperar que aumentando la altura de ambos mástiles se reduzca el número de obstáculos en el trayecto o, al menos, las pérdidas por difracción.

Sin embargo, este comportamiento esperado no se corresponde para nada con el obtenido en la simulación. Para esta nueva ejecución se aumenta el mástil de las antenas de 15 a 40 metros, y el resultado obtenido es el siguiente:

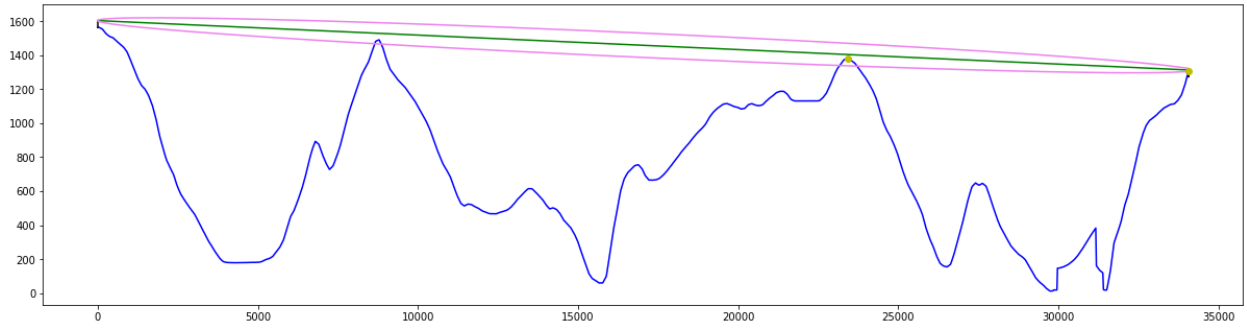


Figura 21 - Error de obstáculo en el extremo receptor.

Aunque el obstáculo del primer pico ha desaparecido, en el extremo receptor ha aparecido un nuevo obstáculo. Este resultado no tiene sentido ya que, siendo las cotas de elevación las mismas en ambas ejecuciones, los mástiles han aumentado y por tanto debieran salvar cualquier obstáculo que pudiera haber en la zona. No hay que olvidar cómo se han obtenido los obstáculos, aplicando el algoritmo descrito en la sección 3.1.2, y es que la más mínima variación en el valor del radio del elipsoide de Fresnel o el despejamiento normalizado puede detectar un obstáculo en

un punto donde antes no había ninguno. También es necesario tener en cuenta que el elipsoide se ha representado haciendo un reajuste de las cadenas como se explica en la sección 3.1.1, lo que favorece que se den errores como éste.

Ya que en las múltiples simulaciones y pruebas en búsqueda de errores sólo se detectó este comportamiento en las proximidades de transmisor y receptor se decidió abordar la siguiente solución: eliminar las muestras más cercanas a ambos extremos para eliminar cualquier posible falso positivo. Concretamente se decidió obviar las 10 primeras y las 10 últimas muestras de la cadena de cotas. Esto es así porque, como se vio en la sección 4.1.2, las muestras se toman cada 5 metros para un trayecto general y se fijan a 200 para aquellos que tengan una distancia menor de 1km, por lo que en cualquier caso se estarían suprimiendo como mucho 50 metros desde cada extremo.

Tras aplicar esta corrección, el resultado es el siguiente:

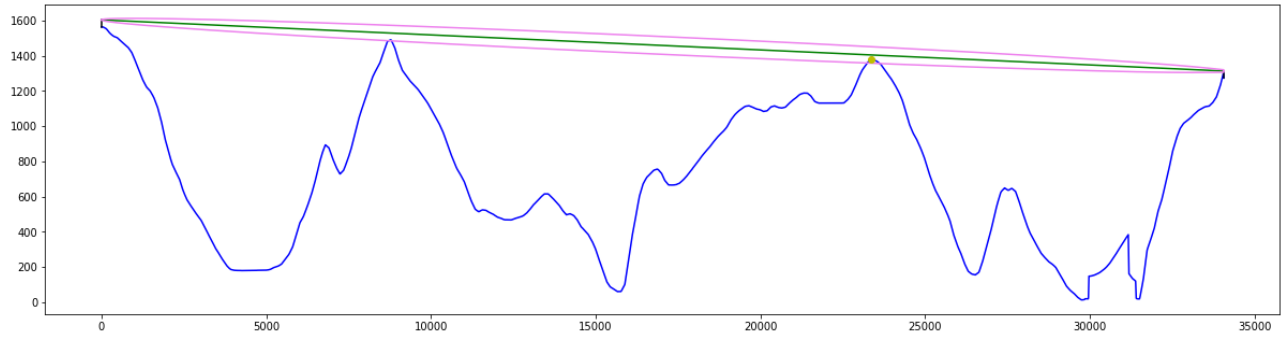


Figura 22 - Perfil con corrección en los extremos.

En este caso, ya sólo se tiene un obstáculo y los extremos están limpios. Tras ver que el funcionamiento era óptimo, se ajustó el número de muestras eliminadas en función de la distancia para evitar que en trayectos cortos se suprimieran posibles obsáculos reales.

4.2.2 Error en la detección de obstáculos múltiples

Los extremos del radioenlace no han sido los únicos tramos conflictivos a la hora de procesar y detectar obstáculos. De hecho, el mayor problema viene a la hora de determinar si un enlace tiene múltiples obstáculos o no debido a que, para una misma cima, la herramienta puede detectar más de un punto positivo. Este problema se ilustra en la siguiente imagen:

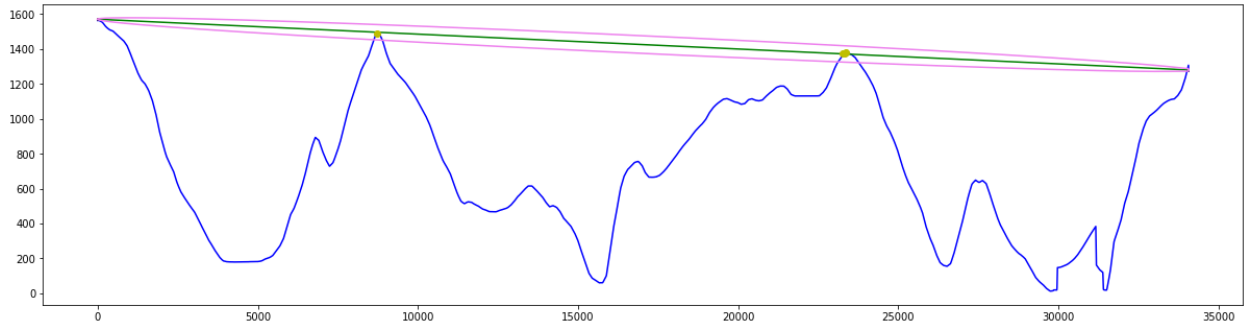


Figura 23 - Detección errónea de múltiples obstáculos

Donde, si se amplía la imagen sobre el segundo obstáculo, se puede observar cómo la herramienta ha determinado que hay más de un punto conflictivo:

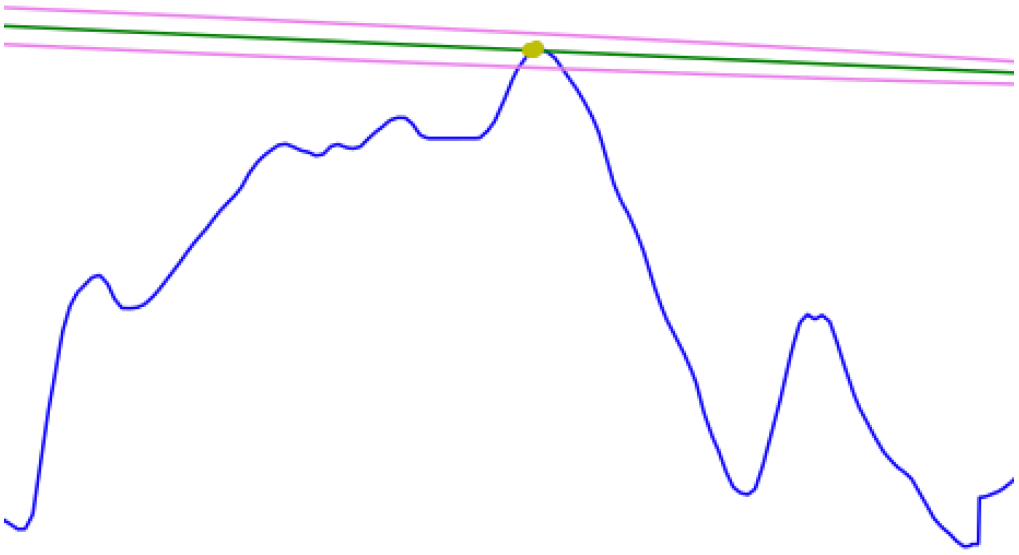


Figura 24 - Múltiples obstáculos en una misma cima

Esto ocurre porque como se detalla en la sección 3.1.2 los obstáculos se determinan calculando los máximos relativos del perfil, y para los puntos próximos entre sí una mínima variación entre las cotas puede dar lugar a un máximo relativo. Este comportamiento supone un problema considerable en el cálculo de las pérdidas por difracción ya que un trayecto que a simple vista se evalúa como trayecto de dos obstáculos se tratará como uno con múltiples, por lo que diseñar un procedimiento que determine cuántos obstáculos hay realmente en un perfil se convierte en una necesidad.

Este procedimiento se plantea a priori tan sencillo como tomar los valores máximos de cada cima y elimiar el resto, de forma que se estaría garantizando que hay un único punto por cima. El primer escollo que se encuentra es determinar qué puntos pertenecen a una cima y cuáles a otra, ya que, de tomarse equívocamente un punto cualquiera como perteneciente a una cima que no es la suya, se podría falsear el valor máximo de la cota en las cimas implicadas. La solución pasa por agrupar los puntos según lo cercanos que estén entre sí, y para ello se introduce un margen en función de la longitud del trayecto que permita crear intervalos entre los puntos y poder agruparlos según estén o no incluidos en este intervalo. Así, siendo m el número de muestras del perfil procesado, se plantean los siguientes márgenes:

- Si $m < 500$:
  - Margen de 10 muestras, es decir un intervalo de aproximadamente 50 metros a izquierda y derecha del punto estudiado.
- Si $500 < m < 1000$:
  - Margen de 50 muestras, es decir un intervalo de aproximdamente 250 metros a izquierda y derecha del punto estudiado.
- Si $m > 1000$:
  - Margen de 150 muestras, es decir un intervalo de aproximadamente 750 metros a izquierda y derecha del punto estudiado.

Para estos cálculos se recuerda que en la mayoría de los casos se toma una muestra cada 5 metros. Una vez establecidos los márgenes, se procesan los puntos de acuerdo al siguiente algoritmo, en el que dado un punto (índice) cualquiera se evalúa si:

- *obstaculos[i]* > *obstaculos[i+1]* – margen, para los puntos a la derecha del punto actual.
- *obstaculos[i]* < *obstaculos[i-1]* + margen, para los puntos a la izquierda del punto actual.

Es importante recordar que la cadena *obstaculos* contiene los índices sobre el total de muestras del trayecto de aquellas en las que hay obstáculos (para más detalle, consultar la sección 6.2). Teniendo en cuenta esto, se considerarán adyacentes dos puntos que cumplan una o ambas condiciones. De esta forma se agrupan los puntos que estén lo suficientemente próximos entre sí, aunque habrá considerar algunas excepciones y aspectos adicionales:

- Si un punto es adyacente a su anterior pero no al siguiente se considera el fin de una agrupación de puntos de una misma cima, y los nuevos puntos procesados pasarían a ser parte de una nueva agrupación. Con esto se consigue determinar qué puntos pertenecen a una cima u otra.
- La cadena que contiene los despejamientos de estos obstáculos (*desp*) también debe ser reajustado, ya que se habrán eliminado obstáculos para los que previamente se había calculado y almacenado su despejamiento.
- Para los puntos primero y último se tendrá en cuenta que sólo se evalúa su proximidad al siguiente y anterior punto, respectivamente.

Todas estas condiciones y salvedades se detallan en el siguiente diagrama y sus pasos complementarios, donde se recoge toda la lógica implementada para el procesamiento y eliminación de obstáculos redundantes:

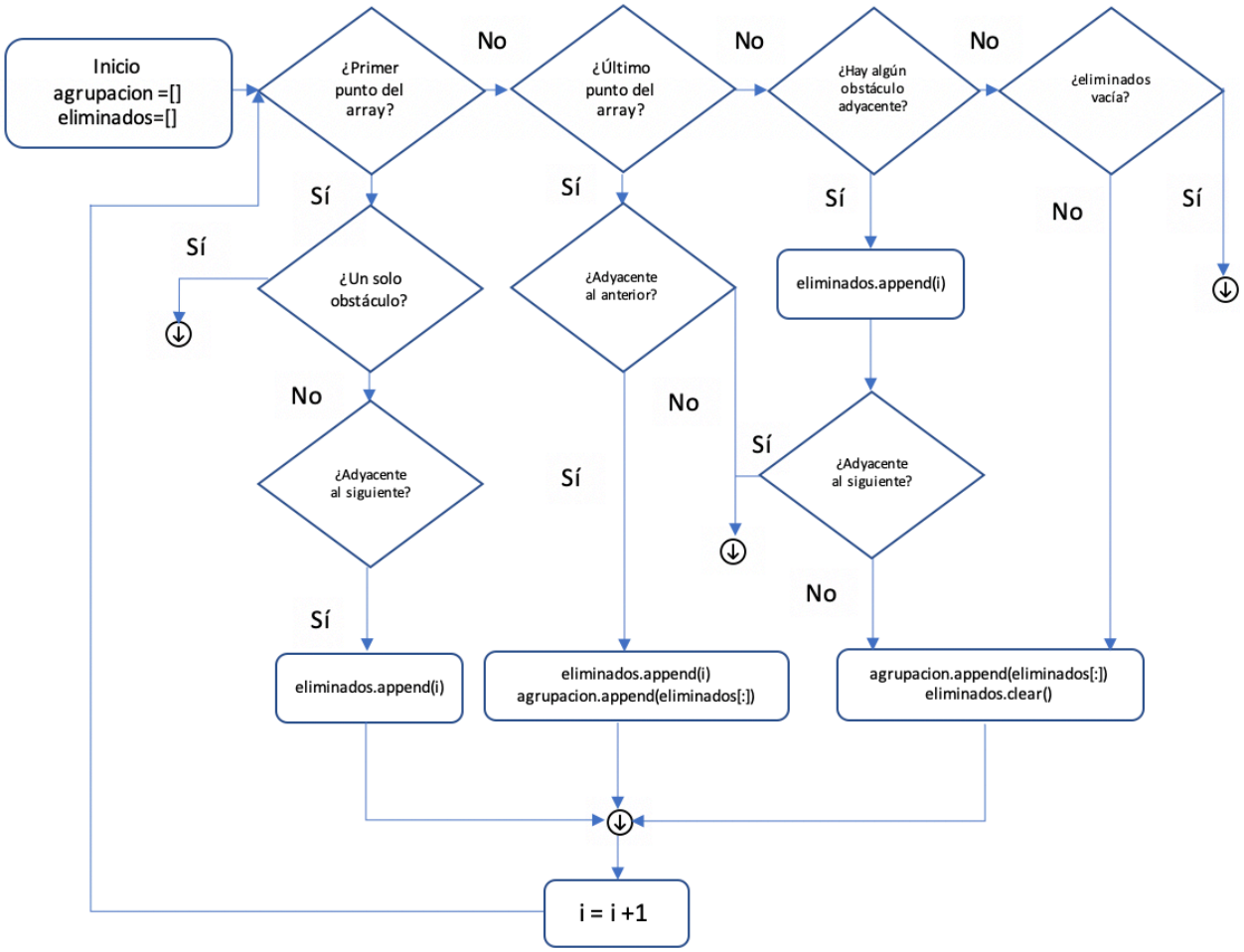


Figura 25 - Diagrama de flujo para eliminación de obstáculos redundantes

Este primer algoritmo, donde *eliminados* es una cadena auxiliar para los puntos pertenecientes a una misma cima, obtiene un conjunto de cadenas, *agrupacion*, que contiene todos los subgrupos de puntos para cada una de las cimas conflictivas. Hay que tener en cuenta que si hubiera una cima para la que sólo se hubiese detectado un punto (es decir, el comportamiento ideal) éste no estaría incluido en la cadena *agrupacion*. Por ejemplo, para el perfil de la Figura X, la cadena *agrupacion* sólo contendría los índices de los puntos de la segunda cima.

El siguiente paso es más sencillo, ya que sólo consiste en evaluar cada subconjunto individual de puntos sobre la cadena *elev*, que contiene como ya se ha explicado con anterioridad todas las cotas del perfil, y quedarnos con aquellos valores que proporcionen la cota máxima. Mismamente, es necesario eliminar del vector *desp* los valores del despejamiento para estos mismos puntos de forma que todos los cálculos y operaciones con cadenas sigan cuadrando. Por ejemplo, supóngase que se parte de:

- Una cadena $obstaculos[1450,1460,1470]$, con tres puntos adyacentes de cual el máximo se obtiene para *elev*[1460], o lo que es lo mismo, *elev*[*obstaculos*[1]]
- Una cadena *desp* con tres valores cualesquiera, sean $[A, B, C]$, que corresponden al despejamiento en los puntos 1450, 1460 y 1470, o lo que es lo mismo, despejamiento a $1450\ x\ 5\ metros = 7250\ metros$ del transmisor, etc.

A partir de aquí, se procede tal y como se indica en el apartado anterior:

- Se eliminan los valores redundantes 1450 y 1470, que están en las posiciones 0 y 2.
- Se elimina de la cadena *desp* los valores en estos dos puntos, 0 y 2, independientemente de cuál fuera el valor almacenado.

El resultado final es una cadena *obstaculos* con un solo valor, que es el índice donde se encuentra el obstáculo, y una cadena *desp* que contiene el valor del despejamiento para ese obstáculo, en este caso el valor B.

Finalmente se representan los puntos calculados sobre la gráfica ahora sí con una correspondencia de un solo punto por cima, evitando cualquier confusión y asegurando que se aplique el método adecuado al perfil en cuestión. En la siguiente imagen se muestra el resultado final:

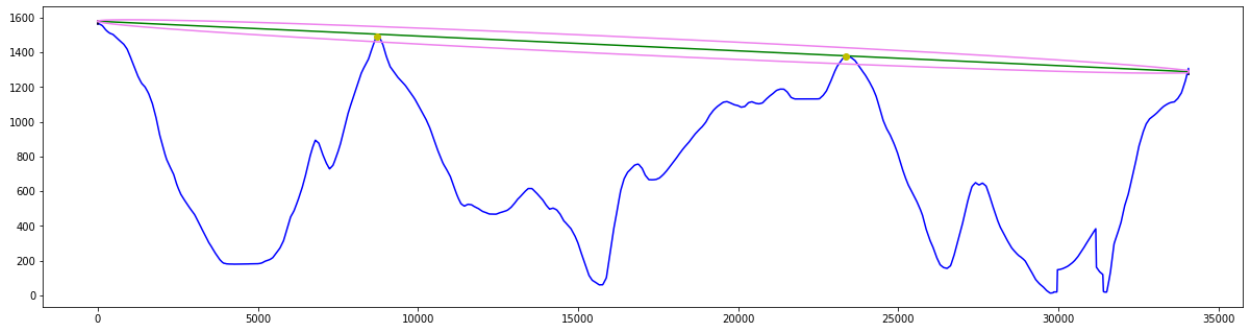


Figura 26 - Perfil geográficos con detección única de punto por cima.

Bajo estas condiciones ya se puede aplicar, para este caso por ejemplo, el método para dos obstáculos ya que se tiene la certeza de que no hay error en la detección de los obstáculos. Otro ejemplo de los resultados que arroja este algoritmo puede verse si se simula un radioenlace entre

Monesterio y Pedroso como en la sección 3.1.1 usando una frecuencia de 300MHz, con el fin de hacer mayor el radio de la primera zona de Fresnel y provocar que aparezcan más obstáculos:

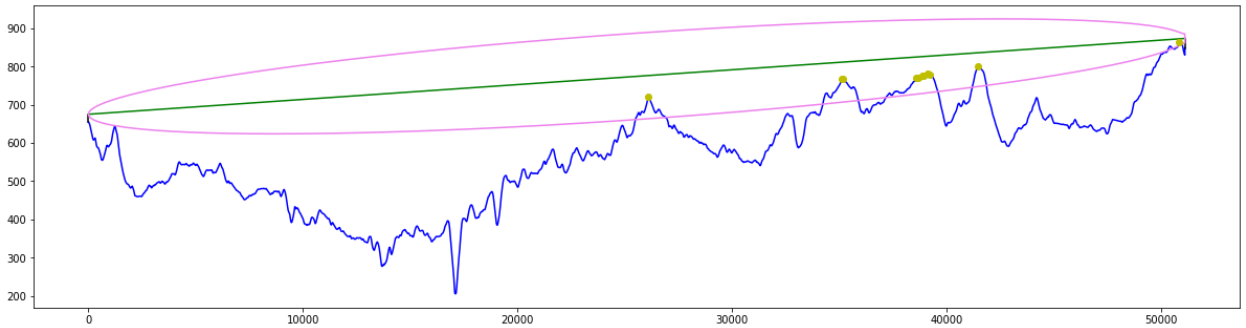


Figura 27 - Enlace Monesterio-Pedroso sin procesamiento de puntos adyacentes.

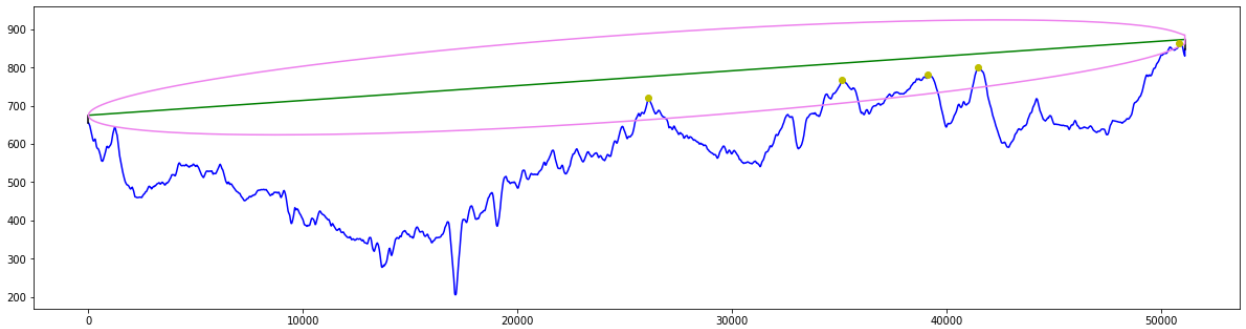


Figura 28 - Enlace Monesterio-Pedroso con corrección en la detección de obstáculos

Es cierto que el reajuste del vector *desp* podría haberse ahorrado filtrando antes los obstáculos obtenidos en el primer barrido, pero esta solución se antojaba más compleja por la forma en que está estructurado el programa, las llamadas a funciones etc, por lo que con unas simples operaciones que no tienen impacto sobre los tiempos de ejecución se solventa el problema.

5 RESOLUCIÓN ANALÍTICA

Una vez solventados los problemas con la herramienta gráfica el programa ya estaba en condiciones de realizar cálculos con mayor precisión, por lo que el siguiente paso es simular distintos radioenlaces para calcular sus pérdidas por difracción.

Sin embargo, previo a este paso es necesario realizar los mismos cálculos que hará el programa pero de forma analítica, de esta forma se pueden comparar ambos resultados para evaluar el nivel de precisión del software. Se analizarán por tanto radioenlaces de distinta casuística de manera que se puedan aplicar los distintos métodos implementados. Habrá que tener en cuenta que todos los cálculos realizados analíticamente están hechos con aproximaciones de los valores obtenidos, por lo que cabe esperar que haya diferencias con los datos calculados automáticamente por la acumulación de errores.

Para todos los ejemplos y por simplicidad las fórmulas usadas se presentarán con los parámetros sustituidos por su valor corrspondiente. Para consultar cuál es la fórmula general se puede acudir a la Rec. P-526[3] en la que se basa este trabajo.

5.1 Medio llano

Para este radioenlace se evaluará un perfil en el que no se encuentra ningún obstáculo que obstruya la primera zona del elipsoide de Fresnel. Para este ejemplo, los datos son:

- Frecuencia: 1 GHz
- a_e : 6370 km
- Distancia entre ambas antenas: 601.11 m
- Altura del mástil de la antena TX: 15 m
- Altura del mástil de la antena RX: 20 m
- Altura del transmisor (cota + mástil): 507.96 m
- Altura del receptor (cota + mástil): 442.00 m

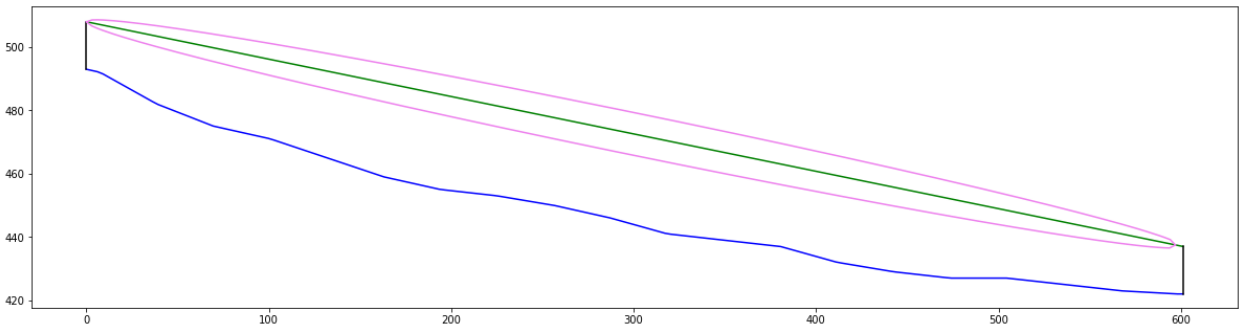


Figura 29 – Radioenlace en medio llano

El primer paso es calcular la distancia de visión directa, que viene dada por la siguiente fórmula:

$$d_{los} = \sqrt{2 \times 6370 \cdot 10^3} \times \left(\sqrt{507.96} + \sqrt{442.00}\right) = 154288.93\ m$$

Dado que $d < d_{los}$, se continúa siguiendo el procedimiento descrito[21] en la recomendación:

$$m = \frac{601^2}{4\times6370\cdot10^3\times(507.96+442)} = 1.49 \cdot 10^{-5}$$

$$c = \frac{507.96 - 442}{507.96 + 442} = 0.069$$

$$b = 2\sqrt{\frac{1.49 \cdot 10^{-5} + 1}{3 \times 1.49 \cdot 10^{-5}}} \times \cos\left(\frac{\pi}{3} + \frac{1}{3}\cos^{-1}\left(\frac{3 \times 0.069}{2}\sqrt{\frac{3 \times 1.49 \cdot 10^{-5}}{(1.49 \cdot 10^{-5} + 1)^3}}\right)\right) = 0.068$$

$$d_1 = \frac{601}{2} \times (1 + 0.068) = 320.93\ metros$$

$$d_2 = 601 - 320.93 = 280.07\ metros$$

$$h = \frac{\left(507.96 - \frac{320.93^2}{2\times6370\cdot10^3}\right)\times280.07 + \left(442.00 - \frac{280.07^2}{2\times6370\cdot10^3}\right)\times320.93}{601} = 472.73 \text{ metros}$$

Donde h resulta ser la altura máxima libre de obstáculos. Por último, se calcula h_{req}:

$$h_{req} = 0.552 \times \sqrt{\frac{320.93 \times 280.07 \times 0.3}{601}} = 3.69\ metros$$

Como $h > h_{req}$, se concluye que las pérdidas totales por difracción para este radioenlace son L = 0 dB.

Además de este caso, hay que considerar aquellos trayectos que son transhorizonte, sin embargo no se han podido realizar los cálculos de forma teórica por falta de datos reales con los que hacerlos. Aún así, toda la lógica que corresponde a este procedimiento está disponible en el software y fue debidamente probada (forzando la lógica) para comprobar que las magnitudes que se obtenían correspondían al análisis teórico.

5.2 Medio con obstáculo único

En este apartado se considerarán dos casos: obstáculo único en arista filo de cuchillo y obstáculo único redondeado, ambos sobre el mismo perfil, de manera que pueda evaluarse el efecto de la curvatura del obstáculo a la hora de añadir una atenuación extra.

Se empiezan los cálculos con el obstáculo en arista filo de cuchillo, sin evaluar la curvatura del obstáculo.

5.2.1 Obstáculo único en arista filo de cuchillo

Para este ejemplo, los datos son:

- Frecuencia: 1 GHz
- a_e : 6370 km
- Distancia entre ambas antenas: 333.95 m
- Altura del mástil de la antena TX: 15 m
- Altura del mástil de la antena RX: 15 m
- Altura del transmisor (cota + mástil): 833.60 m
- Altura del receptor (cota + mástil): 834.00 m
- Distancia del TX al obstáculo: 166.13 m

Y presenta el siguiente perfil:

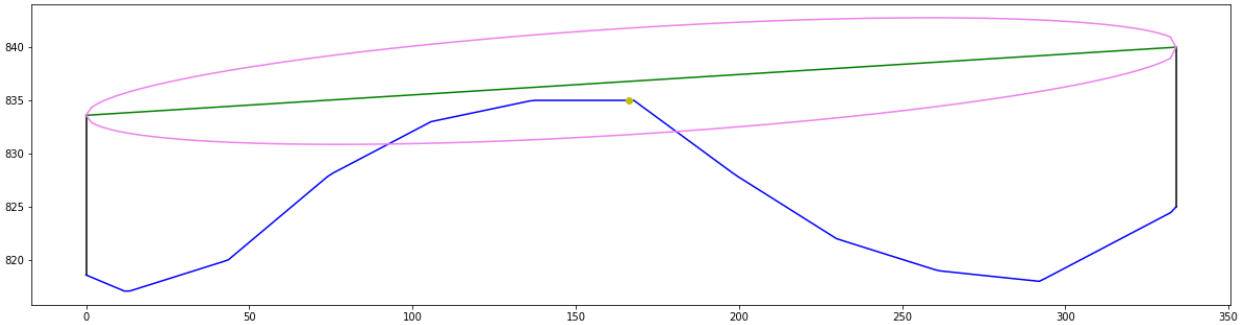


Figura 30 - Radioenlace con obstáculo en arista filo de cuchillo.

Este perfil ya ha sido presentado antes y aunque a simple vista se aprecie que es un obstáculo redondeado se le tratará como un obstáculo en arista filo de cuchillo.

El primer paso es calcular el despejamiento, y para ello se necesita saber tanto la altura del obstáculo como la altura del rayo de visión directa a la distancia donde se encuentra el obstáculo. La altura del obstáculo son 835 metros, y para calcular la altura del rayo de visión directa se usa la ecuación de la recta que pasa por dos puntos, siendo estos dos puntos el extremo transmisor y el extremo receptor:

$$y(166.13) = \frac{166.13 \times (840 - 833.6)}{333.95} + 833.6 = 836.78\, metros$$

Y por tanto el despejamiento:

$$h = 835 - 836.78 = -1.78\, metros$$

Se puede ahora entonces calcular el radio de Fresnel en la primera zona para el punto en cuestión, y así obtener luego el despejamiento normalizado:

$$R = \sqrt{\frac{1 \times 0.3 \times 166.13 \times (333.95 - 166.13)}{333.95}} = 5.00\ metros$$

Y así, el despejamiento normalizado queda:

$$h/_R = -1.78/_{5.00} = -0.356$$

Este valor se encuentra dentro de los límites aceptables, es decir: $-0.6 \leq h/_R \leq 0.5$, por lo que se pueden calcular las pérdidas por difracción. Para ello, se empieza calculando el parámetro v como indica el procedimiento[22]:

$$v = -1.78\sqrt{\frac{2}{0.3} \times \left(\frac{1}{166.13} + \frac{1}{333.95 - 166.13}\right)} = -0.50$$

Como $v > -0.78$:

$$J(v) = 6.9 + 20\log_{10}\left(\sqrt{(-0.5 - 0.1)^2 + 1} - 0.5 - 0.1\right) = 1.95\ dB$$

Por tanto, se concluye que L = 1.95 dB para este radioenlace con un solo obstáculo.

Hay que tener en cuenta que en la fórmula original v se define como:

$$v = h\sqrt{\frac{2}{\lambda}\left(\frac{1}{d_1} + \frac{1}{d_2}\right)} \qquad (3)$$

Donde d_1 y d_2 son las distancias desde los extremos del trayecto a la cima del obstáculo, y aunque en el software se tomen estos valores exactos, en la resolución analítica se ha usado en este ejemplo y se usará en los posteriores la distancia sobre la superficie terrestre, ya que el error es mínimo y se facilitan los cálculos.

5.2.2 Obstáculo único redondeado

Para el estudio de un obstáculo único con radio conocido se reutilizará, como se ha dicho, el caso de estudio del apartado anterior. Por tanto, a los datos anteriores sólo hay que añadir el radio de curvatura del obstáculo:

- Frecuencia: 1 GHz
- a_e : 6370 km
- Distancia entre ambas antenas: 333.95 m
- Altura del mástil de la antena TX: 15 m
- Altura del mástil de la antena RX: 15 m

- Altura del transmisor (cota + mástil): 833.60 m
- Altura del receptor (cota + mástil): 834.00 m
- Distancia del TX al obstáculo: 166.13 m
- Radio de curvatura del obstáculo: 950 m

Este radio de curvatura se ha obtenido de aplicar la siguiente fórmula disponible en la sección correspondiente de la recomendación[23], iterando únicamente dos veces:

$$r = \frac{1}{N}\sum_{1}^{N}\frac{{x_i}^2}{{y_i}^2} \tag{4}$$

Tomándose x_i e y_i tal y como se representan en la siguiente figura:

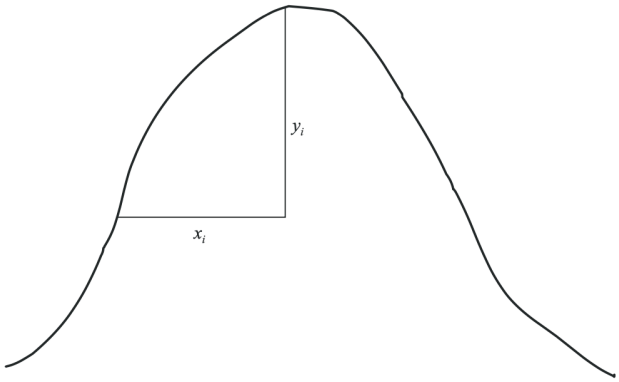


Figura 31 - Perfil vertical de un obstáculo redondeado

Al tratarse del mismo perfil no es necesario rehacer los cálculos, sino que pueden tomarse las mismas pérdidas base del perfil con el obstáculo en arista filo de cuchillo ya que la diferencia es que hay que considerar un término extra que será el que aporte la atenuación adicional por la curvatura. Así, las pérdidas por difracción se definen como:

$$A = J(v) + T(m,n) \tag{5}$$

Donde $J(v)$ son las pérdidas de Fresnel-Kirchoff[22] que se reutilizan del apartado anterior y $T(m,n)$ el término de atenuación por la curvatura.

Para determinar la atenuación adicional es necesario calcular previamente dos parámetros:

$$m = 950 \times \left[\frac{333.95}{166.13 \times (333.95 - 166.13)}\right] \Big/ \left[\frac{950 \cdot \pi}{0.3}\right]^{1/3} = 0.52$$

$$n = -1.78 \times \left[\frac{950 \cdot \pi}{0.3}\right]^{2/3} / 950 = -0.86$$

Ya que $m \cdot n = -0.45 \leq 4$, el término $T(m,n)$ se calcula como:

$T(m,n) = 7.2 \times 0.52^{1/2} - \left(2 - 12.5 \times (-0.86)\right) \times 0.52 + 3.6 \times (0.52)^{3/2} - 0.8 \times 0.52^2 = -0.304$ dB

Curiosamente, en este radioenlace la curvatura del obstáculo está provocando una interferencia constructiva de manera que el término $T(m,n)$ en lugar de actuar como pérdida lo hace como ganancia, disminuyendo las pérdidas totales por difracción, por lo que el resultado final es:

$$A = 1.95\ dB + (-0.304) = 1.64\ dB$$

Para comprobar que efectivamente el término $T(m,n)$ supone un aumento de la atenuación se rehacen los cálculos disminuyendo la frecuencia de forma que la primera zona de Fresnel se haga más amplia y el obstáculo ocupe un porcentaje mayor sobre la misma, aumentando así las pérdidas.

Por tanto, y teniendo en cuenta que ahora $\lambda = 0.6\ m$, queda:

$$R = 7.07\ m$$

$$v = -0.36$$

$$J(v) = 3.03\ dB$$

El término $J(v)$ ha sufrido un incremento de poco más de 1 dB. Por su parte, el término $T(m,n)$ queda como:

$$m = 0.67$$

$$n = -0.55$$

De nuevo $m \cdot n \leq 4$, por lo que:

$$T(m,n) = 1.56\ dB$$

Finalmente:

$$A = J(v) + T(m,n) = 4.59\ dB$$

Con un aumento de las pérdidas como se esperaba en primera instancia.

5.3 Medio con dos obstáculos aislados

En el estudio de un radioenlace con dos obstáculos aislados se usará la misma estructura que se emplease para el cálculo de las pérdidas en un medio con un obstáculo aislado: cálculo de las pérdidas por difracción para dos obstáculos en arista filo de cuchillo y un cálculo posterior para las pérdidas de los dos mismos obstáculos considerando su curvatura.

5.3.1 Dos obstáculos aislados en arista filo de cuchillo

Los datos de este análisis son:

- Frecuencia: 500 MHz
- a_e : 6370 km
- Distancia entre ambas antenas: 34067.88 m
- Altura del mástil de la antena TX: 7 m
- Altura del mástil de la antena RX: 10 m
- Altura del transmisor (cota + mástil): 1570.22 m
- Altura del receptor (cota + mástil): 1282.85 m
- Distancia del TX al primer obstáculo: 8780.74 m
- Distancia del TX al segundo obstáculo: 23417.00 m

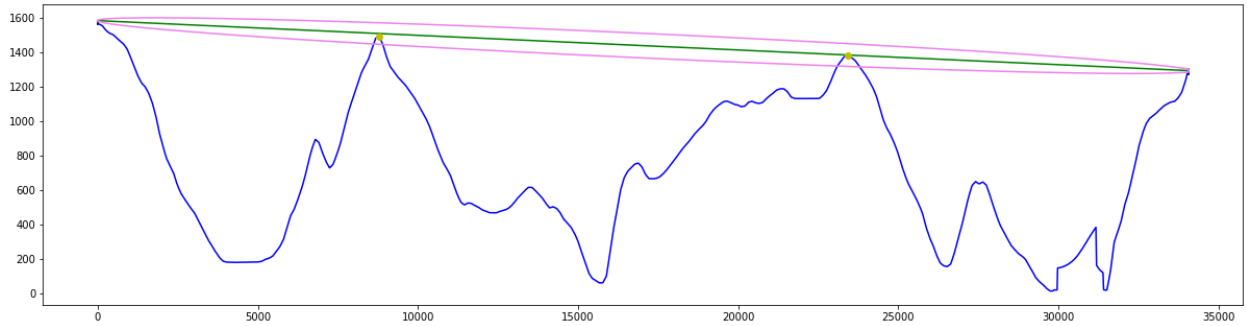


Figura 32- Radioenlace con dos obstáculos aislados en arista filo de cuchillo

Para calcular las pérdidas por difracción cuando se tienen dos obstáculos es necesario aplicar los cálculos de difracción para cada obstáculo por separado, teniendo en cuentas algunas particularidades.

En primer lugar, es necesario determinar qué obstáculo es el que predomina de los dos, y para ello se evaluará cuál de los dos tiene mayor relación h/R, siendo h la altura del obstáculo medida desde el rayo de visión directa y R el radio de Fresnel el los puntos evaluados:

- Primer obstáculo:

Con una altura de unos 1490 metros, se calcula el radio del elipsoide de Fresnel en el punto donde se encuentra el obstáculo y la altura del rayo con la ecuación de la recta, ya que se conocen las coordenadas a ambos extremos.:

$$y_{rayo}(8780.74) = \frac{8780.74 \times (1282.85 - 1570.22)}{34067.88} + 1570.22 = 1498.72\ metros$$

Por lo que con:

$$h_1 = 1490 - 1498.72 = -8.72\ metros$$

$$R_1 = \sqrt{\frac{0.6 \times \left(8780.74 \times (34067.88 - 8780.74)\right)}{34067.88}} = 62.53 \text{ metros}$$

Se tiene que $h_1/R_1 = -0.13$

- Segundo obstáculo:

El segundo obstáculo tiene una altura de 1378.03 metros. Se realizan los mismos cálculos con las distancias y los valores del radio de Fresnel correspondiente:

$$y_{rayo}(23417.00\ ) = \frac{23417.00\ \times (1282.85 - 1570.22)}{34067.88} + 1570.22 = 1370.18\ metros$$

$$h = 1378.03 - 1370.18 = 7.85\ metros$$

$$R_1 = \sqrt{\frac{0.6 \times \left(23417.00\ \times (34067.88 - 23417.00\ )\right)}{34067.88}} = 66.27\ metros$$

Y finalmente $h_2/R_2 = 0.12$. Como $h_1/R_1 < h_2/R_2$ se reajustan los despejamientos como se indica en la siguiente figura:

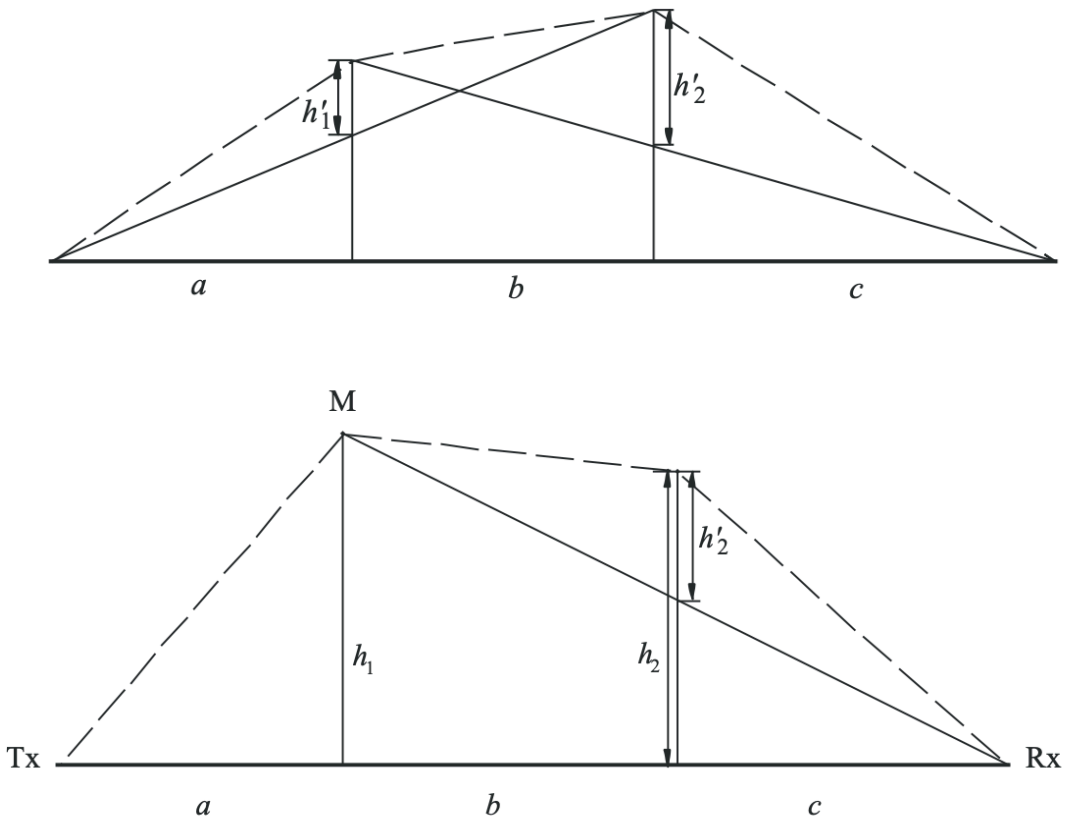


Figura 33 - Ajuste del despejamiento para dos obstáculos aislados

Ya que el caso de estudio se corresponde con el primer esquema, es necesario calcular h'_1 y h'_2, y para ello se vuelve a usar la ecuación de la recta para calcular la altura a la que se encuentran los rayos TX-obstáculo 2 y obstáculo 1-RX en sus puntos correspondientes y así hacer la diferencia con la altura de las cimas:

- Primer obstáculo:

$$y_{rayo-TX-obs2}(8780.74) = \frac{8780.74 \times (1378.00 - 1570.22)}{23417.00} + 1570.22 = 1498.14\, metros$$

$$h'_1 = 1490 - 1498.14 = -8.14\, metros$$

- Segundo obstáculo:

$$y_{rayo-obs1-RX}(23417.00) = \frac{(23417.00 - 8780.74) \times (1282.85 - 1490)}{34067.88 - 8780.74} + 1490 = 1370.01\, metros$$

$$h'_2 = 1378.03 - 1370.01 = 8.02\, metros$$

Ahora ya se pueden calcular las pérdidas por separado de cada obstáculo usando el procedimiento seguido en el apartado 5.2.1, pero teniendo en cuenta que se particularizará cada cálculo considerando que:

- El obstáculo 1 afecta al subtrayecto $a + b$, según la figura.
- El obstáculo 2 afecta al subtrayecto $b + c$.

Esto implica que los valores d_1 y d_2 se tomarán según corresponda como las distancias $a,\ b$ ó c. Así, para el obstáculo 1 se tiene:

$$v = -8.14 \times \sqrt{\frac{2}{0.6} \times \left(\frac{1}{8780.14} + \frac{1}{14636.23}\right)} = -0.20$$

$$J(v) = 6.9 + 20 \log_{10}(\sqrt{(-0.2 - 0.1)^2 + 1} - 0.2 - 0.1 = 4.33\, dB = L_1$$

Y para el segundo:

$$v = 8.02 \times \sqrt{\frac{2}{0.6} \times \left(\frac{1}{14636.23} + \frac{1}{10650.90}\right)} = 0.18$$

$$J(v) = 6.9 + 20\log_{10}(\sqrt{(0.18-0.1)^2+1} + 0.2 - 0.1 = 7.75\ dB = L_2$$

Queda entonces que las pérdidas totales por difracción son:

$$L = L_1 + L_2 - T_c \quad (6)$$

Donde T_c es un término de corrección que se aplica para considerar la separación entre las antenas:

$$T_c = \left[12 - 20\log_{10}\left(\frac{2}{1+\alpha}\right)\right]\left(\frac{q}{p}\right)^{2p} \quad (7)$$

Donde:

$$p = \left(\frac{2}{\lambda}\frac{(a+b+c)}{(b+c)a}\right)^{1/2} h_1 \quad (8)$$

$$q = \left(\frac{2}{\lambda}\frac{(a+b+c)}{(a+b)c}\right)^{1/2} h_2 \quad (9)$$

$$\tan\alpha = \left(\frac{b(a+b+c)}{ac}\right)^{1/2} \quad (10)$$

Sustituyendo los valores en cada uno de los parámetros se tiene que T_c es aproximadamente 1.85 dB, por lo que las pérdidas finales son:

$$L = 4.33 + 7.75 - 1.85 = 10.23\ dB$$

5.3.2 Dos obstáculos aislados redondeados

Al igual que se hiciera con un único obstáculo redondeado, en el estudio de dos obstáculos aislados redondeados se reutilizarán los cálculos de los obstáculos en arista filo de cuchillo y se calcularán los términos de atenuación adicional por la curvatura del obstáculo. Los datos para este estudio son:

- Frecuencia: 500 MHz
- a_e : 6370 km
- Distancia entre ambas antenas: 34067.88 m
- Altura del mástil de la antena TX: 7 m
- Altura del mástil de la antena RX: 10 m
- Altura del transmisor (cota + mástil): 1570.22 m
- Altura del receptor (cota + mástil): 1282.85 m
- Distancia del TX al primer obstáculo: 8780.74 m
- Distancia del TX al segundo obstáculo: 23417.00 m

- Radio de curvatura del primer obstáculo: 1100 m
- Radio de curvatura del segundo obstáculo: 1000 m

Como ya se ha visto, las pérdidas vienen dadas por la ecuación (5):

$$A = J(v) + T(m, n)$$

Se sabe del apartado anterior que para el primer obstáculo $L_1 = J_1(v) = 4.33\ dB$. El término asociado a la cuvatura para este obstáculo se calcula como:

$$m_1 = 1100 \times \left[\frac{8780.14 + 14636.23}{8780.14 \times 14636.23}\right] \Big/ \left[\frac{1100 \cdot \pi}{0.6}\right]^{1/3} = 0.01$$

$$n_1 = -8.14 \times \left[\frac{1100 \cdot \pi}{0.6}\right]^{2/3} / 1100 = -2.37$$

$$T_1(m, n) = 7.2 \times (0.01)^{1/2} - \big(2 - 12.5 \times (-2.37)\big) \times 0.01 + 3.6 \times (0.01)^{3/2} - 0.8 \times (0.01)^2 = 0.42\ dB$$

$$A_1 = 4.33\ dB + 0.42\ dB = 4.75\ dB$$

Análogamente se tiene $L_2 = J_2(v) = 7.75\ dB$ para el segundo obstáculo y el término de atenuación adicional tal que:

$$m_2 = 1000 \times \left[\frac{14636.23 + 10650.90}{14636.23 \times 10650.90}\right] \Big/ \left[\frac{1000 \cdot \pi}{0.6}\right]^{1/3} = 0.009$$

$$n_2 = 8.02 \times \left[\frac{1000 \cdot \pi}{0.6}\right]^{2/3} / 1000 = 2.41$$

$$T_2(m, n) = 7.2 \times (0.009)^{1/2} - \big(2 - 12.5 \times (2.41)\big) \times 0.009 + 3.6 \times (0.009)^{3/2} - 0.8 \times (0.009)^2 = 0.94\ dB$$

$$A_2 = 7.75\ dB + 0.94\ dB = 8.69\ dB$$

Ya que el término T_c no depende del radio de curvatura se mantiene constante y las pérdidas totales quedan como:

$$L_{total} = A_1 + A_2 - T_c = 4.75\ dB + 8.69\ dB - 1.85\ dB = 11.59\ dB$$

5.4 Medio con múltiples obstáculos

A diferencia de los métodos anteriores, el procedimiento descrito en la recomendación[24] para medios con múltiples obstáculos no puede ser resuelto analíticamente, ya que está pensado para ser implementado y automatizado por software y son de una complejidad elevada.

La intención de esta sección no es entrar en el detalle estos cálculos sino describir superficialmente los dos métodos propuestos par poder adquirir una idea a alto nivel de cómo funcionan.

En el siguiente capítulo podrá verse cómo se han calculado las pérdidas de forma automática.

5.4.1 Método de Bulington

Este método utiliza la pendiente de la línea que une transmisor y receptor (suponiendo un trayecto sin obstáculos) como referencia para determinar si el trayecto es de visión directa o es transhorizonte. Para ello, calcula cuál es el punto de todos los disponibles que tiene la mayor pendiente para la línea entre el transmisor y el punto procesado. Este punto se conocerá como punto de Bullington[25].

Así, siendo S_{tim} la pendiente para el punto de Bullington y S_{tr} la pendiente entre transmisor y receptor, se distinguen dos casos:

- $S_{tim} < S_{tr}$: trayecto con visión directa.
- $S_{tim} \geq S_{tr}$: trayecto transhorizonte.

Cada uno de estas casuísticas contempla distintas formas de calcular el parámetro v de difracción, además de reajustarse las pérdidas si el trayecto fuera transhorizonte; pero en cualquier caso estas pérdidas se calculan usando la misma fórmula $J(v)$ que se usara en el apartado 5.2.1, considerando algunas salvedades[25] en función del tipo de trayecto.

5.4.2 Método completo

El método completo para la estimación automática de pérdidas por difracción combina el método de Bullington con técnicas para reajustar las antenas a perfiles ficticios y de esta forma refinar el cálculo de las pérdidas.

Estos reajustes consisten en una serie de cálculos[26] que permiten hayar las alturas ficticias de transmisor y receptor relativas a una superficie lisa. De esta forma, usando las nuevas alturas de las antenas modificadas, el método completo estima las pérdidas tal que:

$$L = L_{ba} + \max\{L_{sph} - L_{bs}, 0\} \tag{11}$$

Donde:

- L_{ba} son las pérdidas de Bullington calculadas tal y como se describe en el apartado 5.4.1 para el trayecto real original, es decir, sin modificar las alturas de las antenas y con las cotas de elevación reales.

- L_{sph} son las pérdidas para el trayecto, tomando la longitud real en una tierra esférica y con las alturas modificadas.
- L_{ba} son las pérdidas suponiendo una tierra lisa y con las alturas modificadas.

Con estas nuevas pérdidas se modifican las pérdidas base añadiendo un término de corrección que corresponde al máximo entre 0 y la diferencia de las pérdidas en una supuesta tierra eférica y una tierra lisa.

6 PROGRAMA EN PYTHON

Antes de ver los resultados que se obtienen en las simulaciones y compararlos con los que se han calculado analíticamente es necesario ver el funcionamiento del programa, para poder comprender tanto su lógica a alto nivel como el detalle de los métodos que utiliza.

Este capítulo pretende ser una guía con la que comprender qué camino se sigue en cada simulación además de servir de fuente de información para consultar el funcionamiento de todos los métodos que se hayan mencionado a lo largo del documento.

6.1 Lógica del programa

En la siguiente figura, extraída de la recomendación[3] , se muestra la lógica que se debe seguir a la hora de estimar las pérdidas por difracción en un radioenlace. Hay que considerar que no es del todo exacto, por supuesto, y es que el procedimiento 4.4 (cilindros múltiples aislados) de la recomendación no ha sido implementado debido a su complejidad y a que se cuenta con el procedimiento 4.3 (obstáculos de forma redondeada múltiple) para estimar las pérdidas.

Teniendo en cuenta esta salvedad, el programa empezará al usuario solicitando los parámetros necesarios para los cálculos, siendo estos: frecuencia, altura de los mástiles de ambas antenas, radio efectivo de la tierra y coordenadas de transmisor y receptor. Una vez obtenidos se calculan las coordenadas en la recta que une ambos puntos y se obtiene y representa el perfil como se explica en el capítulo 3. De esta forma el usuario puede ver el perfil antes de realizar los cálculos, para poder indicar si quiere considerar los radios de curvatura de los obstáculos, además de introducir si fuera necesario la permitividad relativa efectiva y la conductividad efectiva[27]. Al margen de esta interacción, todo el procedimiento interno sigue la lógica que se muestra a continuación de forma totalmente transparente al usuario, a excepción de una simulación con múltiples obstáculos en la que, antes de presentar el resultado final, se preguntará al usuario qué método de los dos vistos en el apartado 5.4 quiere aplicar:

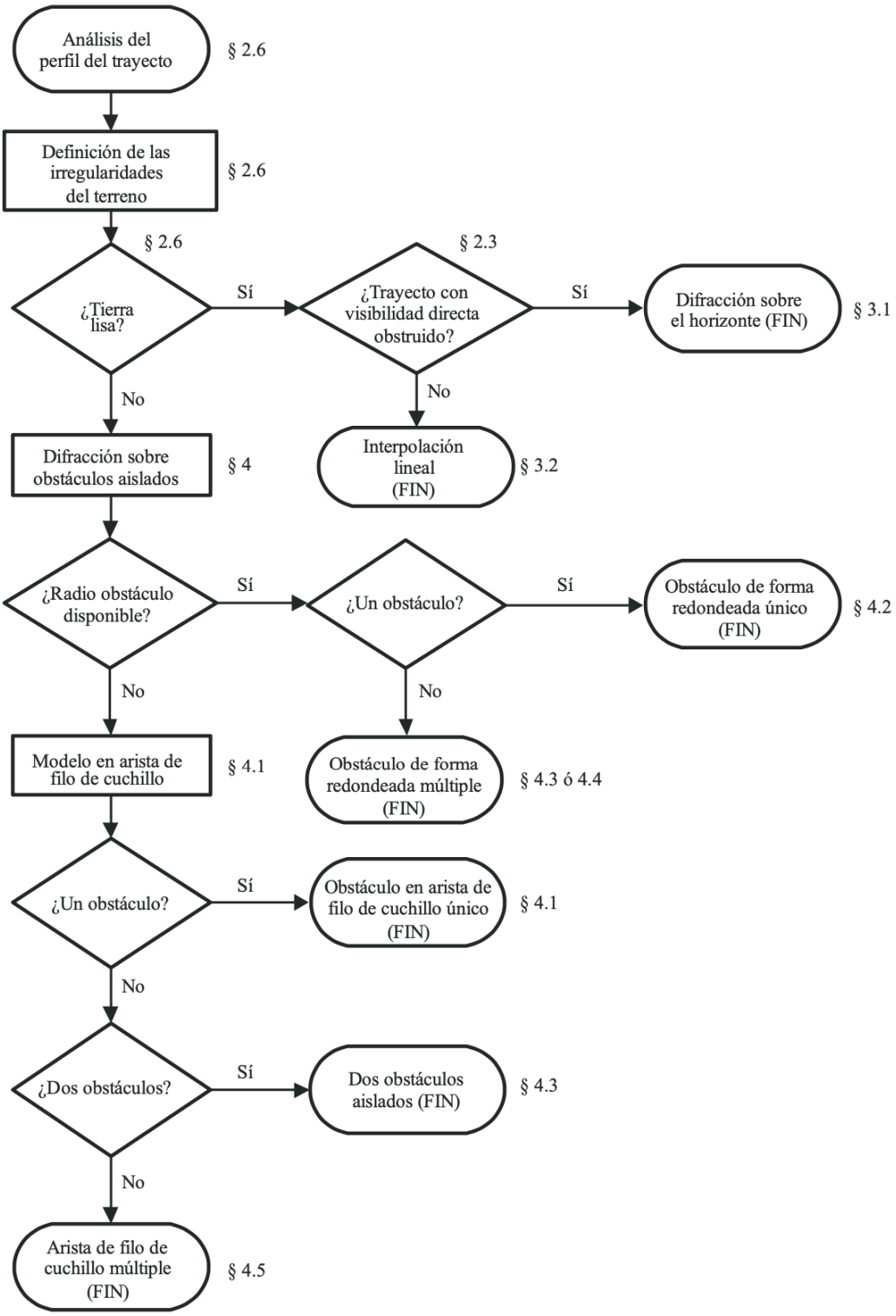


Figura 34 - Lógica de la recomendación implementada en Python.[28]

6.2 Métodos usados en el programa

Esta sección es la que recoge el funcionamiento de cada método así como su definición para que tanto el lector, los usuarios y los desarrolladores que trabajen con el software puedan conocer el detalle de los mismos.

Todos los métodos se han agrupado en distintos archivos según pertenezcan a un tipo de terreno u otro, separando también aquellos que hacen cálculos globales a todo el programa. Las variables que se utilizan pueden consultarse en el Anexo I del código donde se explican aquellas que puedan dar lugar a confusión. Para todos los métodos se ha procurado mantener la nomenclatura que se utiliza en la recomendación[3] para facilitar el entendimiento al lector, por lo que alguas de estas variables no estarán explicadas con detenimiento.

Aún así, se detallan aquellas más importantes en las que se fundamenta todo el programa:

- Cadena *coord*: es una cadena con dos dimensiones, $(2, m)$, donde m es el total de muestras, de forma que accediendo a cualquier punto entre 0 y $m - 1$ se puede obtener el par de coordenadas latitud-longitud.
- Cadena *elev*: contiene la cota del perfil para cada muestra que se haya tomado. El primer punto corresponde a la base de la antena transmisora y el último a la base de la antena receptora.
- Cadena *d_x*: contiene las distancias desde el transmisor (que es el punto de origen) hasta el receptor para todas las muestras que hay, y con un equiespaciado que viene de dividir la distancia total (desde el transmisor al receptor) entre el número de muestras. Así, si se representase *elev* frente a *d_x* se obtendría el perfil tal y como se muestra en las figuras de los capítulos anteriores.
- Cadena *desp*: contiene el despejamiento para cada obstáculo que se haya encontrado en el perfil.
- Cadena *radio_f*: esta cadena tiene el radio de la primera zona de Fresnel para todos los puntos muestreados.
- Cadena *obstaculos*: es la cadena en la que se van a guardar todos los índices en los que hay un obstáculo. La idea de estructurarlo de esta manera es que accediendo a cualquier posición de esta cadena se puede acceder a cualquier información sobre ese punto. Por ejemplo, si *obstaculos*[0] = 10, tomando ese valor en una variable podría saberse a qué distancia se encuentra el obstáculo del transmisor o su cota simplemente haciendo *elev*[10] y *d_x*[10], respectivamente.

6.2.1 Funciones.py

Este archivo contiene funciones genéricas para el procesamiento de las entradas del usuario así como algunos cálculos necesarios en el bloque principal, sin entrar en cálculos específicos de pérdidas.

6.2.1.1 distance(lat1,lng1,lat2,lng2)

Este método calcula la distancia entre dos puntos, dados por dos pares de coordenadas, usando la fórmula de Haversine[29]

6.2.1.2 calculateBearing(lat1,lon1,lat2,lon2)

Calcula el azimut en grados entre los puntos origen y destino, es decir transmisor y receptor. Este método será llamado por el método *coordinates* que se presenta a continuación para poder interpolar las coordenadas entre ambos puntos.

6.2.1.3 coordinates(d,samp,lat1,lon1,lat2,lon2)

Devuelve tantos pares de coordenadas entre transmisor y receptor como muestras haya disponibles en el radioenlace.

6.2.1.4 fresnel(dist,freq,cord,h_coor,h_TX,h_RX,w_length)

Devuelve el radio de la primera zona de Fresnel en cada uno de los puntos disponibles en el radioenlace en una sola cadena, a razón de un punto por muestra. Recibe para ello todos los parámetros necesarios, como la distancia, frecuencia, alturas de transmisor y receptor, coordenadas etc para poder obtener las distancias desde los extremos al punto en cuestión y poder devolver el radio en ese punto.

6.2.1.5 CalculaMuestras(d)

Devuelve el número de muestras para el radioenlace en función de la distancia. Como se ha comentado en capítulos anteriores, cuando el número de muestras es inferior a 200 se fijará a este mismo valor para garantizar una resolución mínima.

6.2.1.6 open_elevation(cord)

Es el método que hace las llamadas a la aplicación de elevación que está disponible en el contenedor y las devuelve en el formato adecuado para los cálculos. Hará una petición por cada par de coordenadas que haya en la cadena *cord*.

6.2.1.7 dibuja_rayo(x,y,h_TXm,h_RXm,d_x)

Dibuja el rayo de visión directa entre transmisor y receptor a partir de las coordenadas de estos y las alturas de los mástiles.

6.2.1.8 dibuja_antena(h_b,h_t,base)

Recibe el punto base de una antena (su posición sobre el eje x) y las cotas de la los extremos de la antena para representarla. Ha de llamarse dos veces, una para cada antena.

6.2.1.9 rotate_around_point_lowperf(point, radians, origin=(0, 0))

En el capítulo 3 se expone el problema que supone representar el elipsoide de Fresnel entre dos antenas que no están a la misma cota. Este método es el que se encarga de hacer el reajuste de los puntos para que la representación sea adecuada para el usuario.

6.2.1.10 protuberancia(r_e, d,k,d_x)

Calcula la protuberancia terrestre para la distancia del trayecto, de manera que se pueda determinar si se usa el modelo de tierra plana o se corrige el perfil según esta protuberancia.

6.2.1.11 dibuja_obstaculo(obs_rep,elev,d_x,type)

Este método dibujará los obstáculos sobre el perfil en función del nivel de afectación, siendo amarillos aquellos que provocan una pérdida por difracción y rojos los que hacen inviable el radioenlace.

6.2.1.12 procesa_obstaculos(obstaculos,m,elev,desp)

Como ya se comentase en la sección 4.2.2, la detección de obstáculos requiere un algoritmo que pueda discernir si todos los puntos detectados son realmente obstáculos únicos o no. Este método se encarga de aplicar dicho algoritmo diseñado y devuelve dos cadenas, una con los puntos en los que hay obstáculos y otra con el despejamiento de los mismos, pero con la certeza de que cada punto corresponde a un único obstáculo.

6.2.1.13 tipo_terreno(elev,radio_f_aux,rayo,d_x,m)

Este método es de los más importantes de este archivo, ya que utiliza todos los datos que se han ido obteniendo para determinar qué tipo de terreno es el que se ha proporcionado. Así, trabajará con los datos de elevación, rayo de visión directa y radio de Fresnel y calculará qué puntos son potenciales obstáculos en una primera batida, para después usar el método *procesa_obstaculos* y quedarse con los puntos que realmente suponen un obstáculo. Con este resultado, devolverá una bandera que determinará el flujo del programa, llamando a uno u otro método de estimación de pérdidas en función del número de obstáculos que haya.

6.2.1.14 calcula_distancia_obstaculos(h1,h2,x1,x2,d_x,obstaculos,elev)

Por último, este es un método genérico que será llamado en algunas ocasiones para calcular las distancias entre las cimas de los obstáculos y/o transmisor y receptor para poder realizar algunos cálculos específicos como por ejemplo los que se hacen en la sección 5.3.

6.2.2 funciones_terreno_liso.py

Todos aquellos métodos relativos a la estimación de pérdidas para un terreno liso se encuentran en este archivo.

6.2.2.1 medio_llano(dis,a_e,h_TXm,h_RXm,h1,h2,wave_len,freq,rafio_f)

Es el método principal de este archivo y que se encargará de determinar qué tipo de terreno liso se tiene (transhorizonte o visión directa), para así ir llamando a los demás métodos según sea un caso u otro. Es el que devuelve las pérdidas para este radioenlace al bloque principal.

6.2.2.2 polarizacion()

Este método no recibe argumentos ya que su único fin es solicitar la polarización (vertical u horizontal) al usuario para así determinar la admitancia de superficie[27].

6.2.2.3 perdidas_difraccion(d,a_e,wave_len,K,pol,h_TX,h_RX)

Calcula las pérdidas por difracción a partir de los datos de admitancia de superficie, polarización, distancia, etc. Los cálculos que este método realiza no se han presentado de forma teórica ya que no había datos reales disponibles para su simulación, pero pueden consultarse en la sección correspondiente de la recomendación.

6.2.2.4 calcula_G(Y,beta,K)

Obtiene los términos de ganancia de altura para ambas antenas a partir de las fórmulas presentadas en la sección del terreno liso de la recomendación.

6.2.2.5 calcula_beta(pol,K,f)

Calcula el parámetro β relativo a la naturaleza del suelo y la polarización y que se usará para determinar la ganancia de altura de las antenas.

6.2.2.6 admitancia(a_e,wave_len,pol)

Ofrece una entrada más al usuario en la que se pide la permitividad relativa efectiva y la conductividad efectiva y devuelve la admitancia en función de estos dos valores y la polarización.

6.2.2.7 calc_h_libre(dis,h1,h2,wave_len,a_e)

Calcula y devuelve la altura mínima libre de osbtáculos que permitirá determinar si las pérdidas se fija automáticamente a cero o si necesitan algún cálculo adicional, según sea el valor de ésta respecto a la altura libre de obstáculos necesaria para unas pérdidas por difracción cero.

6.2.3 funciones_obstaculos.py

El último archivo con definición de métodos es el que reúne todos aquellos que sirven para tratar todos los perfiles en los que haya cualquier obstáculo, independientemente de la cantidad de éstos que haya.

6.2.3.1 medio_obstaculos(r_e,h_TXm,h_RXm,h1,h2,d,w_length,freq,obstaculos,radio_f,desp,distancias,elev,d_x)

Método principal para el tratamiento de perfiles con obstáculos. Es el que, en función del número de obstáculos, llama a una u otra función para estimar las pérdidas. Recibe todos los parámetros necesarios (todas las cotas, las alturas de antenas, frecuencia, radios de la tierra y de Fresnel, despejamiento, etc) para poder pasarlo a cada método individual aquellos que necesite. Es la encargada de devolver al bloque principal las pérdidas calculadas.

6.2.3.2 selecciona_metodo(obstaculos)

Este método recibe el número de obstáculos y devuelve el método que se debe aplicar para estimar las pérdidas. Si hay más de dos obstáculos no se preguntará el radio de curvatura. Si hay dos o menos, y el usuario ha indicado que sí quiere introducir el radio de curvatura, se dará al usuario una entrada por cada obstáculo para poder salvar el radio de cada uno de ellos. A partir de la respuesta del usuario y el número de obstáculos se devolverá al *medio_obstaculos* una

variable con el nombre del método (de la recomendación) que se usará, para que pueda ser llamado y se calculen las pérdidas.

6.2.3.3 obstaculo_unico(w_length,desp,distancias,R,v)

Aquí está toda la lógica necesaria para calcular las pérdidas por difracción cuando sólo se tiene un obstáculo. Entre otras variables se recibe el radio de curvatura para poder calcular el efecto de la atenuación por curvatura del obstáculo, si se ha proporcionado. Este método también será llamado por el método *dos_obstaculos_aislados* para calcular las pérdidas de cada uno de los obstáculos por separado, particularizados para sus subtrayectos como se explica en el análisis teórico. Devuelve las pérdidas del obstáculo independientemente de la parte del programa en la que haya sido invocado.

6.2.3.4 dos_obstaculos_aislados(w_length,desp,fresnel,h_TXm,h_RXm,h1,h2,obstaculos,elev,radio,d_x)

Cuando se tienen dos obstáculos se llama a este método, que determina cuál de los dos es el predominante y, en función del resultado, reajusta los despejamientos y hace uso del método anterior para calcular las pérdidas de cada obstáculo en su subtrayecto con el despejamiento debidamenta ajustado y así poder obtener las pérdidas totales.

6.2.3.5 obstaculos_multiples(obstaculos, elev, d_x, r_e, d, h1, h2, w_length, desp, distancias, radio, h_TXm, h_RXm)

Este método no hace cálculos de pérdidas, sino que sirve para determinar cuál de los dos procedimientos disponibles para múltiples obstáculos (Bullington o método completo) se ha de usar. Para ello, llegado el momento proporciona una entrada al usuario pidiendo que se introduzca el método deseado y se encargará de invocar al que corresponda.

6.2.3.6 metodo_bullington(dkm,d_xkm,Ce,obstaculos,elev,h1,h2,w_length,desp,distancias,radio,h_TXm, h_RXm)

Obtiene las pérdidas siguiendo todos los pasos que se detallan en la sección de Bullington de la recomendación[25] a partir de las variables que se le proporcionan. Como diferencia con los métodos anteriores en los que se calculaban pérdidas, este método trabaja con las distancias en kilómetros, por lo que previa invocación es necesario reajustar las dos primeras variables que recibe (*dkm* y *d_xkm*) para que equivalgan a las originales en kilómetros.

6.2.3.7 metodo_completo(dkm,d_xkm,Ce,obstaculos,elev,h1,h2,w_length,desp,distancias,radio,h_TXm, h_RXm,Lba)

Al igual que *metodo_bullington*, éste trabaja con las distancias en kilómetros y además, recibe las pérdidas de Bullington para el radioenlace que se esté tratando. Esto es así porque como se explicó en la sección 5.4.1, el método completo parte de las pérdidas de Bullington y aplica un factor de corrección particularizando el trayecto para tierra lisa y esférica, por lo que durante su transcurso deberá hacer uso de *metodo_bullington* para calcular las pérdidas en estos dos supuestos.

7 RESOLUCIÓN AUTOMÁTICA

En este capítulo se pretende mostar el cálculo automático de las pérdidas por difracción mediante el software diseñado y comparar los resultados con los obtenidos en el capítulo 5.

Es importante recordar que estos resultados pueden diferir con los obtenidos analíticamente ya que el software trabaja internamente con todos los decimales disponibles mientras que en el cálculo analítico se ha trabajado con slos decimales redondeados a las decenas. Hay algunas casuísticas que no han podido ser probadas con ejemplos reales, como por ejemplo, aquellas en la que la distancia real del trayecto es mayor a la distancia de visión directa. Aún así, en las simulaciones que se realizaron durante la fase de pruebas se forzó al software a evaluar el radioenlace como si cumpliera dicha condición, y los cálculos eran correctos.

Para la comparación se usarán perfiles e imágenes obtenidas de Google Earth[15].

7.1 Medio llano

El medio llano que se va a simular responde a las siguientes coordenadas de transmisor y receptor:

- Transmisor:
  - Latitud: 39.2416,
  - Longitud: -6.41
- Receptor:
  - Latitud: 39.2470
  - Longitud: -6.41

El perfil que se obtiene como referencia se muestra en la siguiente figura:

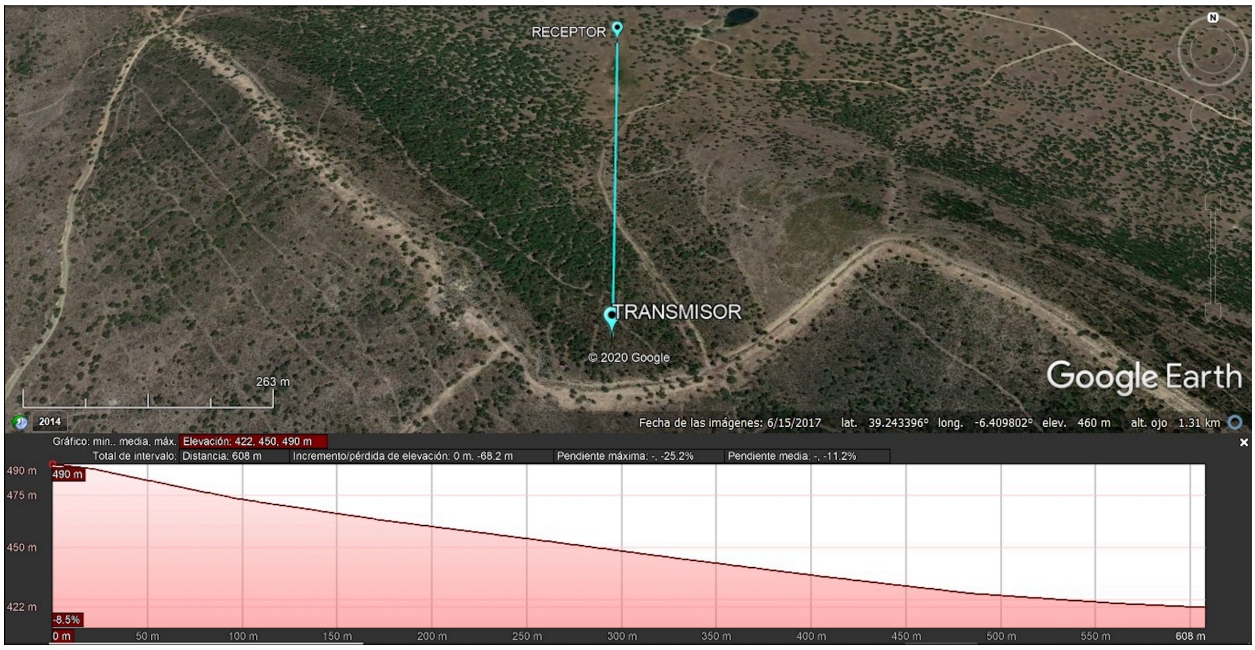


Figura 35 - Perfil llano obtenido con Google Earth

Hay un detalle que es necesario relativo a la entrada que se solicita al usuario para la polarización. Aunque este dato sólo es necesario si se cumplen ciertas condiciones en el procedimiento, se ha decidido pedirlo al inicio del método ya que de esta forma si hay un número elevado de muestras que requiera un tiempo de computación mayor, se puede procesar todo de forma completamente desatendida sin tener que pausar el programa hasta que el usuario introduzca los datos:

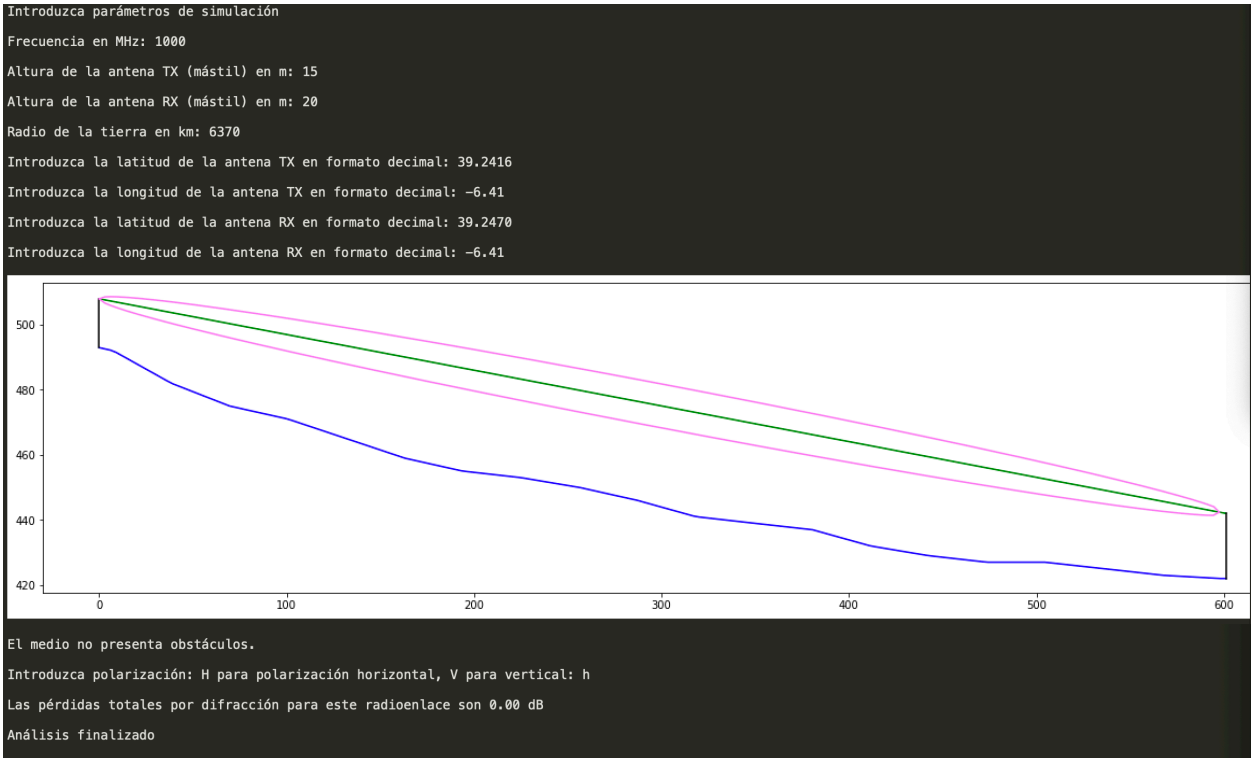


Figura 36 - Simulación por software para un medio llano

El resultado coincide con el encontrado en el apartado 5.1, y se ha obtenido de manera inmediata al cumplirse las condiciones ya estudiadas[21] con anterioridad.

7.2 Obstáculo único

Como ya se ha visto en el capítulo anterior, se utilizará el mismo enlace para los casos en los que hay un obstáculo único redondeado y un obstáculo único en arista filo de cuchillo.

7.2.1 Obstáculo único en arista filo de cuchillo

Las coordenadas para este radioenlace son :

- Transmisor:
  - Latitud: 40.7890
  - Longitud: -3.0300

- Receptor:
  - Latitud: 40.7860
  - Longitud: -3.0300

El radioenlace presenta el siguiente perfil:

Figura 37 - Perfil geográfico con un obstáculo uno obtenido con Google Earth

Y la ejecución en el programa muestra el siguiente resultado:

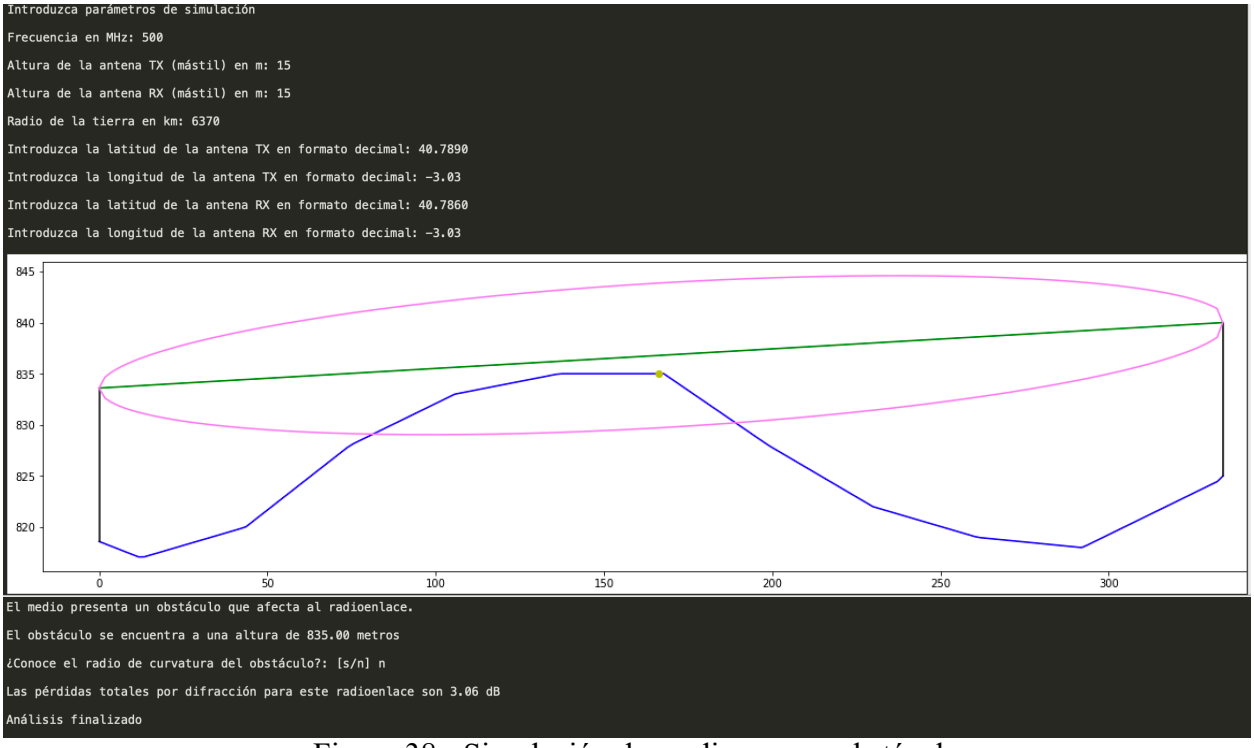


Figura 38 - Simulación de medio con un obstáculo

Se comprueba, al igual que en el capítulo 5, que el perfil se asemeja al obtenido desde Google Earth y que las pérdidas, salvando los errores acumulados por el redondeo, son similares a las

que se calcularon en el apartado 5.2.1. Aunque hay una diferencia de casi 1 dB, hay que tener en cuenta que la distancia entre tansmisor y receptor para este radioenlace es muy pequeña, por lo que tanto los cálculos realizados con el software como los cálculos realizados a mano (los datos de elevación, distancias, etc. se han obtenido por inspección en la gráfica y/o del programa) pueden acumular una cantidad mayor de errores, de ahí que la diferencia sea mayor.

7.2.2 Obstáculo único redondeado

En esta ocasión no es necesario definir la longitud ni latitud ni de transmisor ni receptor ya que se reutiliza el enlace. Se presenta entonces las pérdidas obtenidas desde el software para un obstáculo con un radio de curvatura aproximado de 950 metros:

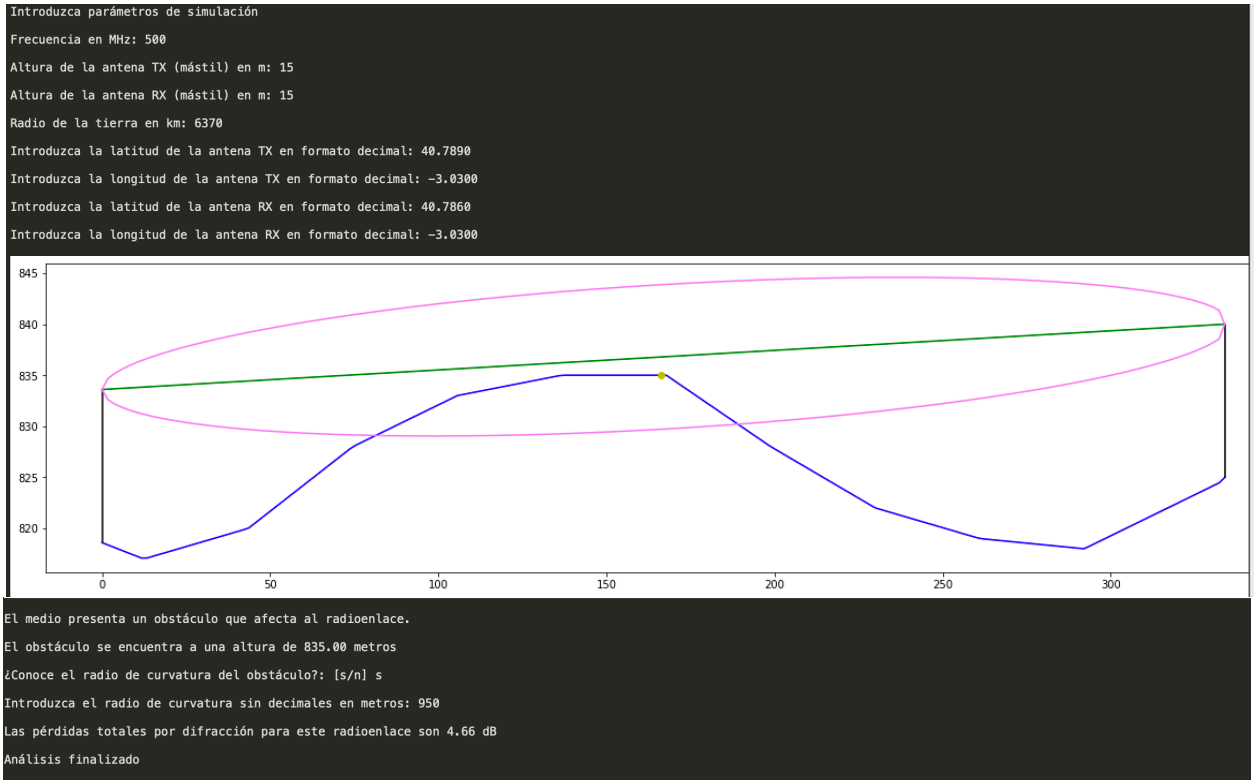


Figura 39 - Simulación en software de radioenlace con obstáculo único redondeado

Si se atiende a la Figura X, se puede observar cómo el software en esta ocasión, al haber respondido el usuario ‘s’, pregunta por el radio de curvatura del radioenlace, y cómo el término asociado a la atenuación extra por la curvatura ha supuesto un aumento de las pérdidas totales. El resultado apenas difiere del que se calculó en el apartado 5.2.2.

7.3 Dos obstáculos aislados

Una vez más se plantea un problema en el que se van a reutilizar los cálculos realizados para un medio con dos obstáculos en arista filo de cuchillo para estimar las pérdidas por difracción para dos obstáculos redondeados.

7.3.1 Dos bstáculos aislados en arista filo de cuchillo

Aunque este perfil ya es conocido, pues se ha visto numerosas veces a lo largo del documento, se presentan sus coordenadas:

- Transmisor
  - Latitud: 61.4633
  - Longitud: 6.9627
- Receptor:
  - Latitud: 61.4578
  - Longitud: 7.6031

El perfil obtenido con Google Earth es:

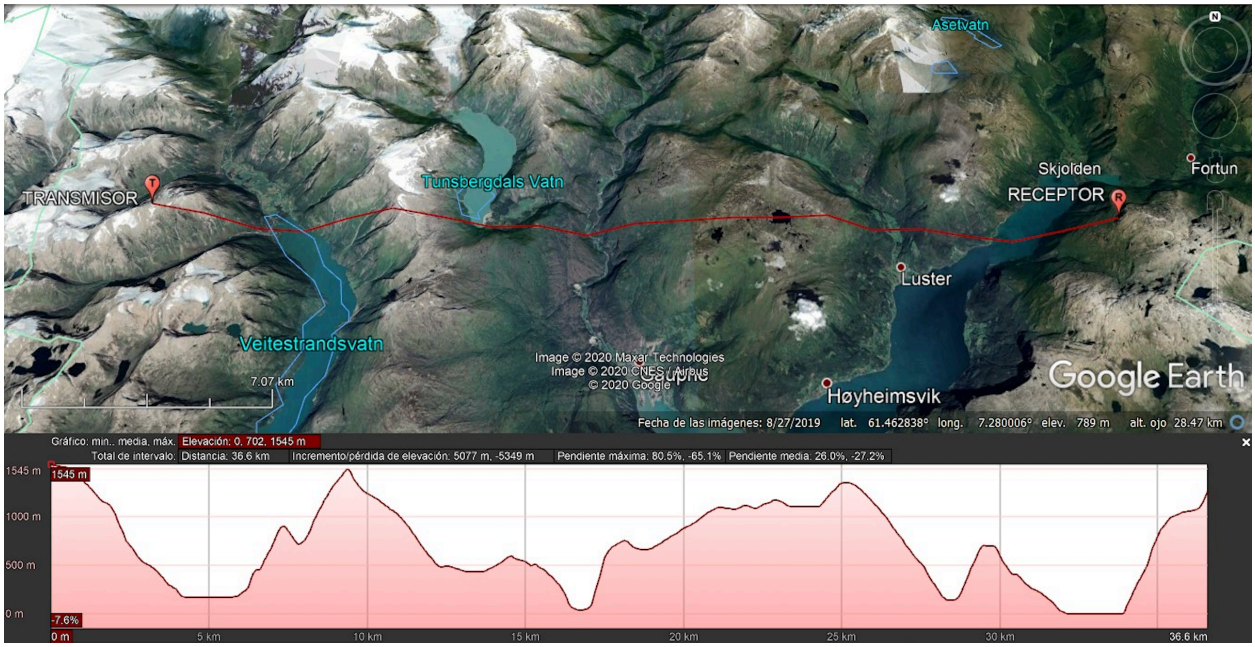


Figura 40 - Perfil con dos obstáculos obtenidos con Google Earth

La salida del programa presenta el siguiente resultado:

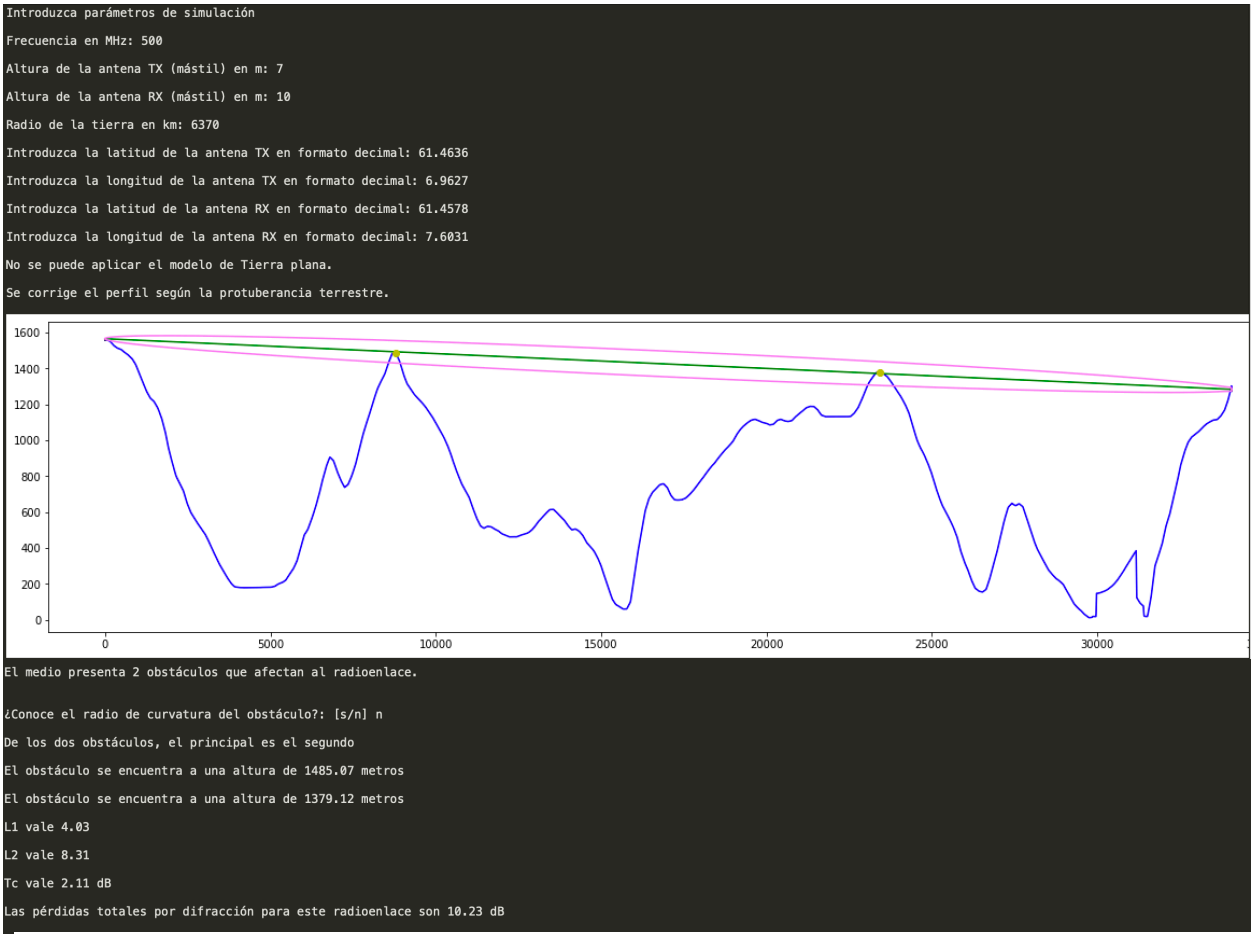


Figura 41 - Medio simulado con dos obstáculos filo de cuchillo.

Donde se ve que las pérdidas, en esta ocasión, son iguales a las calculadas analíticamente.

7.3.2 Dos obstáculos aislados redondeados

Como ya se ha comentado, se va a simular el mismo radioenlace pero considerando los radios de curvatura de los obstáculos. Estos se han estimado en:

- Radio curvatura primer obstáculo: 1100 metros.
- Radio curvatura segundo obstáculo: 1000 metros.

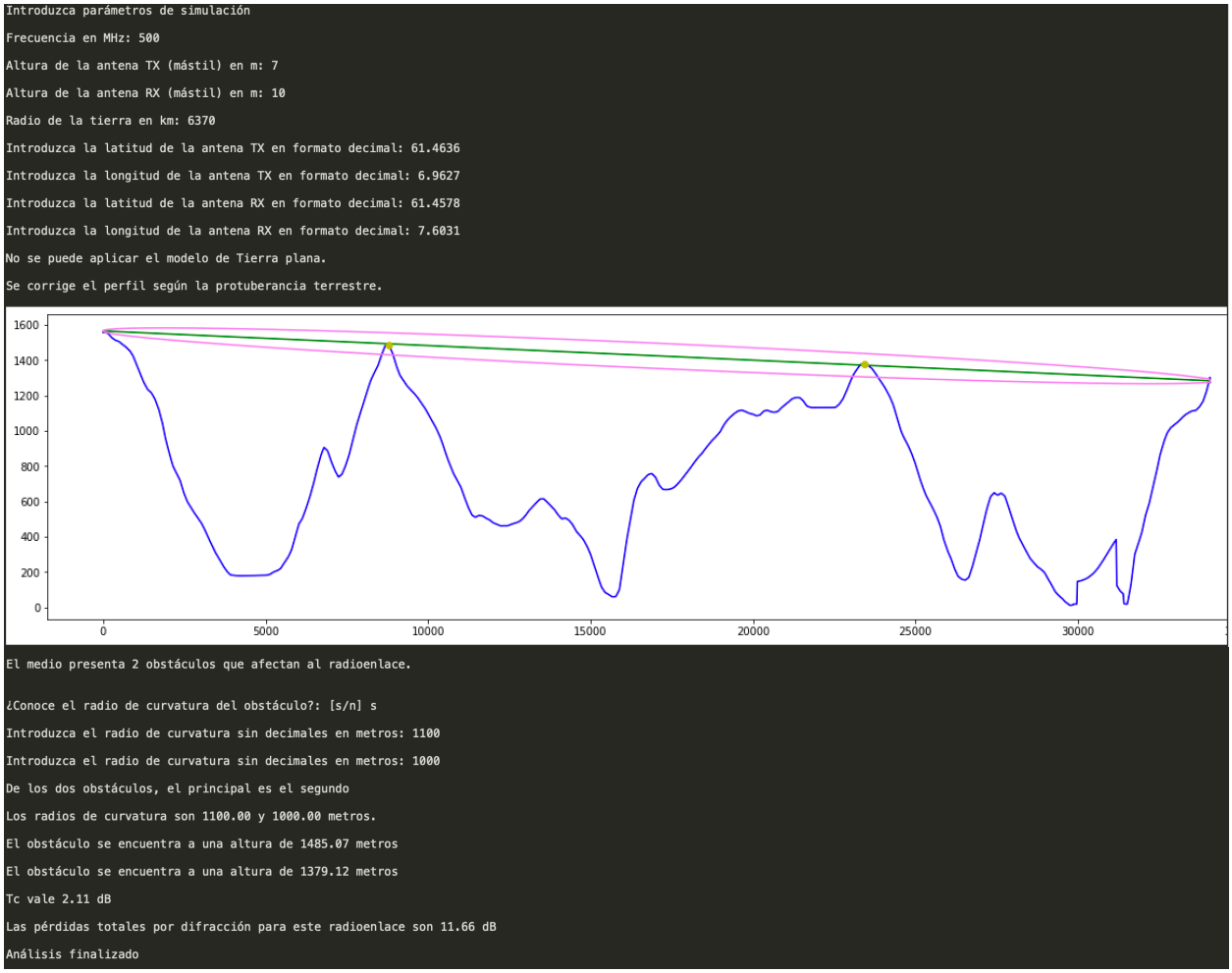


Figura 42 - Simulación de radioenlace con dos obstáculos redondeados

Al haber seleccionado el usuario la opción ‘s’ se vuelve a preguntar los radios de curvatura, esta vez en dos ocasiones, de manera que la primera entrada se asigne al radio de curvatura del primer obstáculo y la segunda al segundo. Esta vez las pérdidas no son las mismas que las que se calcularon en el apartado 6.3.2, pero la diferencia es despreciable.

7.4 Obstáculos múltiples

Como ya se comentó en el apartado 5.4, las pérdidas para un radioenlace con múltiples obstáculos serían únicamente calculadas mediante el software, debido en primer lugar a que este algoritmo está pensado para resolución automática, y en segundo lugar a la complejidad que supondría resolverlo analíticamente.

Las coordenadas son:

- Transmisor:
  - Latitud: 37.8570
  - Longitud: -5.7699

- Receptor:
  - Latitud: 38.0784
  - Longitud: -6.2800

Para esta simulación, al detectarse varios obstáculos, se preguntará al usuario qué método quiere aplicar para el cálculo de las pérdidas, siendo el primer método el método de Bullington, y el segundo método el método completo para un radioenlace de múltiples obstáculos. En ambos casos, el perfil es el que sigue:

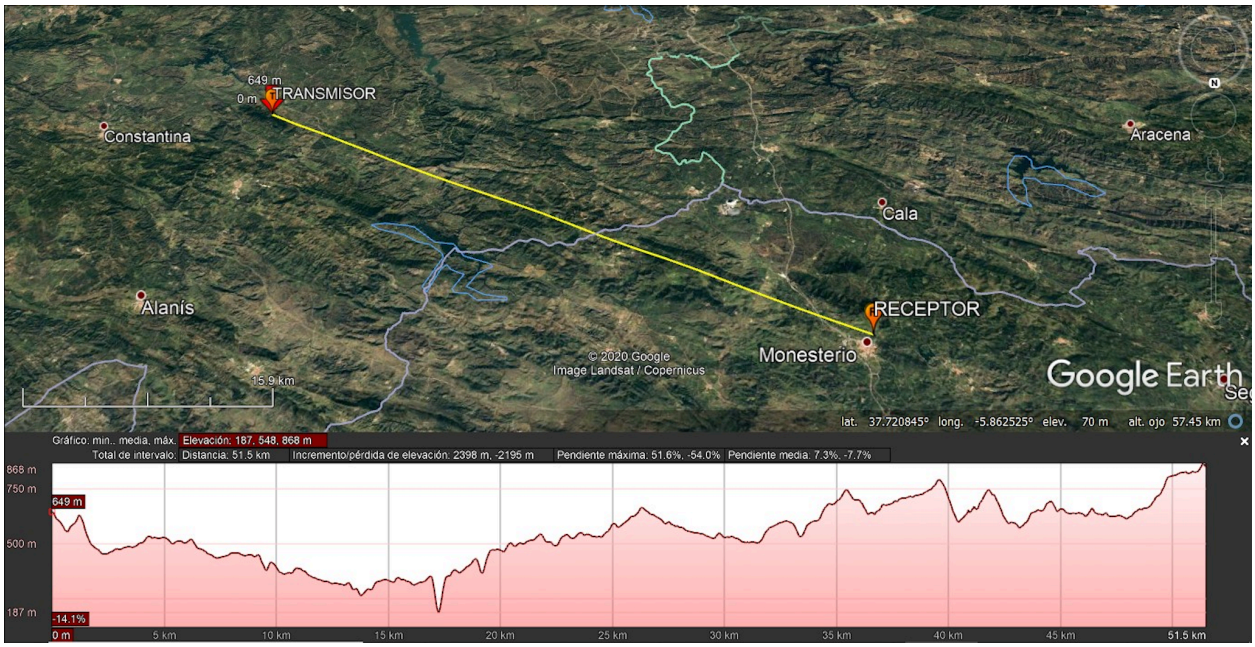


Figura 43 - Perfil en Google Earth para enlace con múltiples obstáculos

7.4.1 Obstáculos múltiples: método de Bullington

Como se ha adelantado en la introducción de la sección, se usará la entrada del software solicitada al usuario para indicar que el cálculo de pérdidas se haga por el método de Bullington.

Figura 44 - Simulación de medio con múltiples obstáculos

Aunque no se indique, de manera interna el software ha aplicado el método de Bullington para estimar las pérdidas por difracción, abstrayendo al usuario de cualquier operación y procesamiento de valores que se haya realizado.

7.4.2 Obstáculos múltiples: método de Bullington

Se repite ahora la simulación para estimar las pérdidas por el método completo:

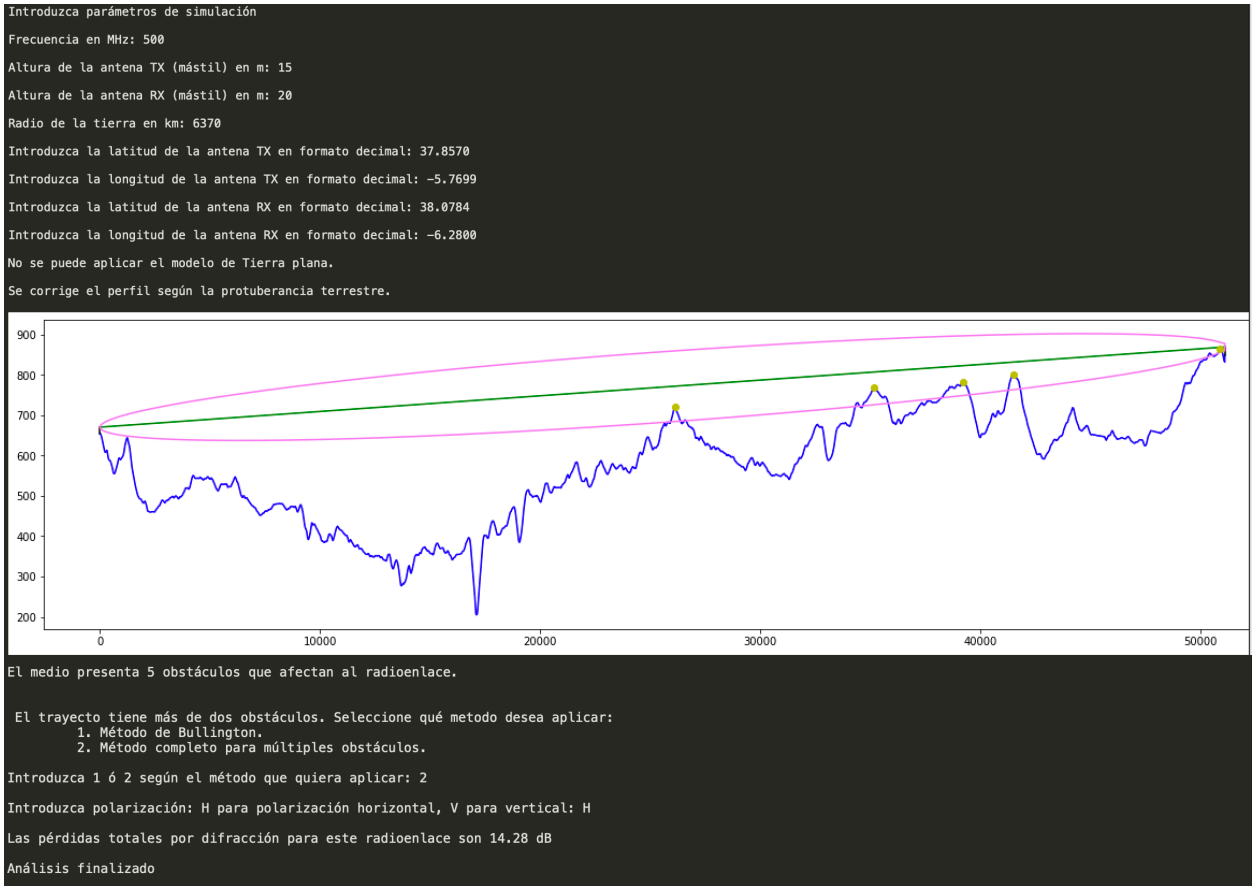


Figura 45 - Cálculo de pérdidas por software para múltiples obstáculos con método compelto.

Para esta simulación, aunque se pueda esperar que el cálculo de las pérdidas sea más preciso, el resultado es exactamente el mismo que el que se obtuvo con el método Bullington. Esto puede deberse a que tanto los métodos aplicados en 6.4.1 como en 6.4.2 están pensados para ser exclusivamente utilizados para cálculos automáticos, y como tal pueden requirir una precisión y un juego de datos que con la tecnología y los recursos disponibles no es posible garantizar. Aún así, se puede apreciar cómo el método completo, a diferencia del método de Bullington, hace uso de los algoritmos para cálculos de medio llano con el fin de aumentar la precisión de las pérdidas calculadas.

8 CONCLUSIONES

Sintetizar lo aprendido durante el desarrollo de este trabajo se antoja complicado ya que han sido muchas y muy distintas las áreas en las que se ha podido profundizar, ampliar y reforzar conocimientos. Lo más obvio y evidente es todo lo que atañe al conocimiento de la radiocomunicación y la programación. Estudiar las distintas tecnologías ha supuesto un enriquecimiento notable de los conocimientos en desarrollo software, lógica y pensamiento lateral, además de los conocimientos en tecnologías concretas como Docker o Python, de los cuáles ha quedado patente su potencia y por qué están tan demandados en esta área de conocimiento. De igual forma, estudiar la teoría en la que se basa este trabajo ha permitido recordar y reforzar los conocimientos de radiopropagación (y aunque en menor medida, de otras materias) que se adquirieran a lo largo de estos años atrás. Ver distintas formas de aplicarlos además de analizarlos con perspectiva y experiencia sin duda hace que todo este camino recorrido hasta aquí haya merecido la pena.

La idea de que este trabajo pueda formar parte de un módulo mayor y que en un futuro pueda ser usado por otros estudiantes ha sido un aliciente extra desde el primer día, tanto en motivación como a la hora de afrontar un reto que ante esta circunstancia se crece, ya que las soluciones que se plantean no pueden ser excluyentes y han de dejar puertas abiertas para una futura integración con el resto de módulos ya desarrollados o por desarrollar. Por último y no por ello menos importante, compartir una meta común con otros compañeros hace que recorrer este camino haya sido más llevadero y satisfactorio, ya que recibir ayuda de aquellos que trabajan en paralelo y ser capaz de devolverla es algo sin duda gratificante.

En cuanto a lo que queda por recorrer, hay muchos puntos en los que podría ponerse el foco, pero la gran mayoría destaca por tener en común el mismo punto de partida: la herramienta de obtención y representación de datos de elevación que se ha implementado. Como ya se dijera en capítulos anteriores, esta herramienta no es perfecta y aunque se hayan reforzado muchas de sus debilidades y solventados algunos de sus problemas, es cierto que aún queda mucho por mejorar y refinar: calidad de la representación gráfica, interfaz de usuario, etc. Además, esta herramienta no está separada del módulo de estimación de pérdidas por difracción por lo que sería de utilidad que en un futuro fuera una entidad independiente, ya que cualquiera que quiera hacer uso de ella deberá extraer del código aquellos métodos que apliquen a tal fin.

No sólo es necesario refinar la herramienta gráfica, sino que también debería considerarse el estudio y mejora del método 4.5 de la recomendación[3], mencionado en el apartado 5.4, ya que como se ha comentado puede no disponer de toda la precisión necesaria, por lo que una mejora en el procesamiento de los datos podría beneficiar la estimación de las pérdidas. De igual modo, el método 4.4 podría también ser implementado para disponer de otro recurso de cálculo de pérdidas con el que aumentar la precisión o comparar resultados, así como los algoritmos para evaluar las pérdidas por pantallas, etc.

Otro desarrollo futuro que merece la pena mencionar es el de una herramienta que, haciendo uso de la que aquí se proporciona, represente un mapa en tres dimensiones. Este programa obtiene las coordenadas en una línea recta entre dos puntos, pero si en su lugar se se pudieran proporcionar

cuatro (como las cuatro esquinas de una sección de mapa) y hacer una matriz con el tamaño adecuado podría obtenerse una representación en tres dimensiones de la zona elegida.

Por último, hay un aspecto muy importante que tener en cuenta y que en el futuro deberá ser estudiado con detenimiento, y es la infraestructura que se proporcione para esta herramienta. Ya se comentó que la aplicación estaba hospedada en un contenedor, pero son el procesador del equipo y la conexión a internet los que determinan el tiempo que se tardará en obtener, representar y procesar los datos en función del número de muestras. Para las pruebas que se han hecho en este trabajo ha bastado con un equipo personal (aunque algunas simulaciones requerían mucho tiempo), pero si llega la ocasión en que un módulo mayor pueda utilizarse en una sesión de prácticas por varios alumnos se necesitarían muchos más recursos. Por ello, y dado el tamaño que supondría tener la base de datos disponible en un almacenamiento local, así como el número variable de usuarios y peticiones, se plantea estudiar el uso de orientar esta herramienta final a microservicios y usar una arquitectura en la nube en la que los recursos pudieran autoescalarse, evitándose así las limitaciones que pudiera ofrecer una máquina física.

ANEXO

bloque_principal.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Feb 28 12:58:17 2019

@author: Juan Antonio
"""
import sys
import os
import json
import numpy as np
from funciones import *
from funciones_terreno_liso import *
from funciones_obstaculos import *
import matplotlib.pyplot as plt
from matplotlib import transforms as trf
from pylab import *
import pycurl
import requests
import scipy


'''Velocidad de la luz'''
c=float(3e8)

print("Introduzca parámetros de simulación")

freq = float(input('Frecuencia en MHz: '))
h_TXm = int(input("Altura de la antena TX (mástil) en m: "))
h_RXm = int(input("Altura de la antena RX (mástil) en m: "))
r_e = float(input("Radio de la tierra en km: "))

k=4/3
#Longitud de onda
w_length=c/(freq*1e6)



LatTX = input('Introduzca la latitud de la antena TX en formato decimal: ')
LonTX = input('Introduzca la longitud de la antena TX en formato decimal: ')
LatRX = input('Introduzca la latitud de la antena RX en formato decimal: ')
LonRX = input('Introduzca la longitud de la antena RX en formato decimal: ')

#Se calcula la distancia entre TX y RX
d=distance(LatTX,LonTX,LatRX,LonRX)
#Se calcula el número de muestras
m=CalculaMuestras(d)
#Se calculan las coordenadas entre ambos puntos
cord=coordinates(d,m,LatTX,LonTX,LatRX, LonRX)
#Se calculan las cotas para todos los puntos
elev = open_elevation(cord)
#Se calculan el radio de Fresnel para todos los puntos
radio_f=fresnel(d,freq,cord,elev,h_TXm,h_RXm,w_length)
#Radio de fresnel positivo y negativo desde el rayo para representarlo
radio_f_pos=[x + float(elev[0]) + h_TXm for x in radio_f]
radio_f_neg=[float(elev[0]) - x + h_TXm for x in radio_f]
#Se forma la cadena d_x con las distancias a lo largo de todo el eje x
d_x=[]
d_x=np.linspace(0,d,m)
```

```
#Se calcula la protuberancia terrestre y se corrige el perfil si es > 5.
prot,correccion = protuberancia(r_e,d,k,d_x)
if prot >= 5:
    print("\r\nNo se puede aplicar el modelo de Tierra plana.\r")
    print("\r\nSe corrige el perfil según la protuberancia terrestre.\r")
    for i in range(0,len(correccion)):
        elev[i]=float(elev[i])+float(correccion[i])

else:
    for i in range(0,len(correccion)):
        elev[i]=float(elev[i])+float(correccion[i])
    '''Se convierten las cordenadas en formato string a float para su
    representación'''
    for i in range(0,len(elev)):
        elev[i]=float(elev[i])
    '''Se calcula la dimensión del eje x'''

#Se definen la figura y los ejes y se representa el perfil
fig = plt.figure(figsize=(20,5))
axes = fig.add_axes([0.1,0.1,0.8,0.8])
axes.plot(d_x,elev, 'b-')

#Se definen todos las variables para representar antenas,
#y se pintan sobre la misma gráfica
hy=[elev[0],elev[len(elev)-1]]
hx=[d_x[0],d_x[len(d_x)-1]]
altura_rayo = dibuja_rayo(hx,hy,h_TXm,h_RXm,d_x)
htx_b = elev[0]
htx_t = htx_b+h_TXm
hrx_b = elev[len(elev)-1]
hrx_t = hrx_b+h_RXm
dibuja_antena(htx_b,htx_t,d_x[0])
dibuja_antena(hrx_b,hrx_t,d_x[len(d_x)-1])

#Se definen la parte superior e inferior de la elipse
rotacion = math.atan((elev[-1]+h_RXm-elev[0]-h_TXm)/d)
rotacion2 = math.atan(((elev[-1]+h_RXm)-(elev[0]+h_TXm))/(d_x[-1]-d_x[0]))
xs = 0
ys = elev[0]+h_TXm

points_pos = [d_x,radio_f_pos]
points_neg = [d_x,radio_f_neg]
xfinal = []
yfinal = []

#Se procesa la elipse para rotarla y hacer reajuste de la dimensión.
for i in range(0,len(points_pos[1]),1):
    qx,qy=rotate_around_point_lowperf((points_pos[0][i],points_pos[1][i]), 0-
        rotacion2, origin=(xs, ys))
    xfinal.append(qx)
    yfinal.append(qy)

#Se representa la parte superior
plt.plot(xfinal,yfinal,'violet')

xfinal2 = []
yfinal2 = []
```

```
#Se procesa la elipse para rotarla y hacer reajuste de la dimensión.
for i in range(0,len(points_pos[1]),1):
    qx,qy=rotate_around_point_lowperf((points_pos[0][i],points_pos[1][i]), 0-
        rotacion2, origin=(xs, ys))
    xfinal.append(qx)
    yfinal.append(qy)

#Se representa la parte superior
plt.plot(xfinal,yfinal,'violet')

xfinal2 = []
yfinal2 = []

for i in range(0,len(points_neg[1]),1):
    qx,qy=rotate_around_point_lowperf((points_neg[0][i],points_neg[1][i]), 0-
        rotacion2, origin=(xs, ys))
    xfinal2.append(qx)
    yfinal2.append(qy)

#Se representa la parte inferior
plt.plot(xfinal2,yfinal2,'violet')


radio_f_aux = []

for t in range(0,len(altura_rayo)-1):
    diff = yfinal[t]-altura_rayo[t]
    radio_f_aux.append(diff)

#Se llama a la función tipo_terreno para determinar el tipo de medio,
#los obstáculos y el despejamiento.
terreno,obstaculos,inviable,desp = tipo_terreno(elev, radio_f_aux,altura_rayo,
    d_x,m)


#Se calcula las cota de los extremos transmisor y receptor
h1 = elev[0]+h_TXm
h2 = elev[len(elev)-1]+h_RXm

#Se presenta toda la gráfica
plt.show()

#En función del terreno, se llama a un método u otro para calcular las
pérdidas.
if terreno == 0:
    print('\r\nEl medio no presenta obstáculos. ')
    Ldif = medio_llano(d,r_e,h_TXm,h_RXm,h1,h2,w_length,freq,radio_f)

if terreno == 1:
    if len(obstaculos) == 1:
        print('\r\nEl medio presenta un obstáculo que afecta al radioenlace.')
    else:
        print('\r\nEl medio presenta '+repr(len(obstaculos))+' obstáculos que
            afectan al radioenlace. \r\n')
    distancias = calcula_distancia_obstaculos(h1,h2,d_x[0],d_x[-1],d_x,
        obstaculos,elev)
    Ldif = medio_obstaculos(r_e,h_TXm,h_RXm,h1,h2,d,w_length,freq,obstaculos,
        radio_f,desp,distancias,elev,d_x)

if terreno == 2:
    print('El radioenlace es inviable. \r\n')

print(f'\r\nLas pérdidas totales por difracción para este radioenlace son {Ldif
    :.2f} dB')

print('\r\nAnálisis finalizado')
```

funciones.py

```
# -*- coding: utf-8 -*-
"""
Created on Thu Feb 28 17:17:33 2019

@author: Juan Antonio
"""

import os
import json
import numpy as np
import math
import pycurl
import requests
from pylab import *



#Función que dará la distancia del radioenalce usando la fórmula de Haversine:
def distance(lat1,lng1,lat2,lng2):
    '''calculates the distance between two lat, long coordinate pairs'''
    R = 6378000 # radius of earth in m
    lat1f = float(lat1)
    lng1f = float(lng1)
    lat2f = float(lat2)
    lng2f = float(lng2)
    lat1rads = math.radians(lat1f)
    lat2rads = math.radians(lat2f)
    deltaLat = math.radians((lat2f-lat1f))
    deltaLng = math.radians((lng2f-lng1f))
    a = math.sin(deltaLat/2) * math.sin(deltaLat/2) + math.cos(lat1rads) * math
        .cos(lat2rads) * math.sin(deltaLng/2) * math.sin(deltaLng/2)
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
    d = R * c
    return d

#Función que calcula el azimut
def calculateBearing(lat1,lng1,lat2,lng2):
    '''Calcula el azimut en grados entre un punto origen y un punto destino'''
    lat1f = float(lat1)
    lng1f = float(lng1)
    lat2f = float(lat2)
    lng2f = float(lng2)
    startLat = math.radians(lat1f)
    startLong = math.radians(lng1f)
    endLat = math.radians(lat2f)
    endLong = math.radians(lng2f)
    dLong = endLong - startLong
    dPhi = math.log(math.tan(endLat/2.0+math.pi/4.0)/math.tan(startLat/2.0+math
        .pi/4.0))
    if abs(dLong) > math.pi:
        if dLong > 0.0:
            dLong = -(2.0 * math.pi - dLong)
        else:
            dLong = (2.0 * math.pi + dLong)
    bearing = (math.degrees(math.atan2(dLong, dPhi)) + 360.0) % 360.0;
    return bearing
```

```
def getDestinationLatLong(lat,lng,azimuth,distance):
    '''devuelve la latitud y longitud de un punto destino
    dados las coordenadas origen, azimut y distancia al punto'''
    lat1f = float(lat)
    lng1f = float(lng)
    R = 6378.1 #Radio de la Tierra en km
    brng = math.radians(azimuth) #Azimut en grados se convierte a radianes
    d = distance/1000

    lat1 = math.radians(lat1f) #lat punto actual
    lon1 = math.radians(lng1f) #Lon punto actual

    lat2 = math.asin( math.sin(lat1) * math.cos(d/R) + math.cos(lat1)* math.sin
        (d/R)* math.cos(brng))

    lon2 = lon1 + math.atan2(math.sin(brng) * math.sin(d/R)* math.cos(lat1),
           math.cos(d/R)- math.sin(lat1)* math.sin(lat2))

    lat2 = math.degrees(lat2)
    lon2 = math.degrees(lon2)

    return[lat2, lon2]

def coordinates(d,samp,lat1,lon1,lat2,lon2):
    '''Devuelve cada par de coordenadas entre dos puntos dados.
    Habrá tantas muestras como haya.'''
    azimuth=calculateBearing(lat1,lon1,lat2,lon2)
    coords = []
    interval=d/samp
    remainder, nsamp = math.modf((interval))
    counter = float(interval)
    coords.append([lat1,lon1])
    for i in  range(1,int(samp)-1):
        c = getDestinationLatLong(lat1,lon1,azimuth,counter)
        counter += float(interval)
        coords.append(c)
    #counter +=1
    coords.append([lat2,lon2])
    return coords



def fresnel(dist,freq,cord,h_coor,h_TX,h_RX,w_length):
    r_fresnel = []
    for i in range(0,len(h_coor),1):
        dist_TX=distance(cord[0][0],cord[0][1],cord[i][0],cord[i][1])
        dist_RX=distance(cord[i][0],cord[i][1],cord[len(cord)-1][0],cord[len(
            cord)-1][1])
        r_aux=math.sqrt((w_length*dist_RX*dist_TX)/(dist))
        r_fresnel.append(r_aux)
    return r_fresnel
```

```python
def CalculaMuestras(d):
    '''En base a la longitud del enlace, se toma un número de muestras para
    obtener
    las coordenadas y alturas en el mismo, de manera que se tenga una buena
    estimación
    del perfil geográfico del radioenlace. Para tal fin, se va a calcular una
    altura cada
    dos metros. Se ha decidido hacer una función para que en caso de que se
    decida modificar
    la distancia de muestreo, dicha modificación sea imediata y no requiera
    tocar el
    bloque principal'''
    muestras=round(d/5)
    if muestras < 200:
        muestras = 200
    return muestras



def open_elevation(cord):
    elev = []
    for i in range(0,len(cord)):
        peticion = 'http://localhost:3000?lat='+str(cord[i][0])+'&lng='+str(
            cord[i][1])
        r = requests.get(peticion)
        r.json()
        elev.append(r.text)

    return elev


def dibuja_rayo(x,y,h_TXm,h_RXm,d_x):
    x1, x2 = x[0], x[1]
    y1 = y[0] + h_TXm
    y2 = y[1] + h_RXm
    rayoY = np.linspace(y1,y2,len(d_x))
    plt.plot([x1,x2],[y1,y2],'g-')
    return rayoY

def dibuja_antena(h_b,h_t,base):
    plt.plot([base,base],[h_b,h_t],'k-')



def rotate_around_point_lowperf(point, radians, origin=(0, 0)):
    x, y = point
    ox, oy = origin

    qx = ox + math.cos(radians) * (x - ox) + math.sin(radians) * (y - oy)
    qy = oy + -math.sin(radians) * (x - ox) + math.cos(radians) * (y - oy)

    return qx, qy
```

```
'''Se va a procesar el array de obstáculos para eliminar aquellos que estén muy
próximos, ya que el programa puede introducir errores e interpretar un sólo
obstáculo como si fuerand dos'''
def procesa_obstaculos(obstaculos,m,elev,desp):

    eliminados = []
    agrupacion = []
    elimina_desp = []
    if m < 500:
        elimina = 10
    elif 500 < m < 1000:
        elimina = 50
    else:
        elimina = 150

    for i in range(0,len(obstaculos),1):
        if i == 0:
            if not len(obstaculos) == 1:
                if obstaculos[i] > obstaculos[i+1] - elimina:
                    eliminados.append(obstaculos[i])
        elif i == len(obstaculos)-1:
            if obstaculos[i] < obstaculos[i-1] + elimina:
                eliminados.append(obstaculos[i])
                agrupacion.append(eliminados[:])
        elif obstaculos[i] < obstaculos[i-1] + elimina or obstaculos[i] >
            obstaculos[i+1] - elimina:
            eliminados.append(obstaculos[i])
            if not obstaculos[i] > obstaculos[i+1] - elimina:
                agrupacion.append(eliminados[:])
                eliminados.clear()
        else:
            if len(eliminados) != 0:
                agrupacion.append(eliminados[:])
                eliminados.clear()

    for i in range(0,len(agrupacion)):
        elevaciones=[]
        for j in range(0,len(agrupacion[i])):
            elevaciones.append(elev[agrupacion[i][j]])
        maximo = max(elevaciones)
        for h in range(0,len(elevaciones)):
            if elevaciones[h] < maximo:
                elimina_desp.append(desp[obstaculos.index(elev.index(
                    elevaciones[h]))])
        for k in range(0,len(elevaciones)):
            if not elev.index(elevaciones[k]) == elev.index(maximo):
                obstaculos.remove(elev.index(elevaciones[k]))
        for i in elimina_desp:
            desp.remove(i)
        elimina_desp.clear()



    return obstaculos,desp
```

```
'''En este método se analizan todos los puntos disponibles del radioenlace
    y se seleccionan aquellos que representan obstáculos. En cada punto, se
    calcula el despejamiento normalizado a partir del valor del radio de
    Fresnel
    y la cota'''
def tipo_terreno(elev,radio_f_aux,rayo,d_x,m):
    flag = 0
    obstaculos = [] #Obstáculos toma los indíces del array donde hay
    obstaculos que afecten al radioenlace
    inviable = []
    aux=[]
    daux=[]
    indices=[]
    desp=[]

    '''Se van a eliminar aquellos índices que se encuentren entre las
            10 primeras y las 10 últimas muestras para refinar el cálculo de
            obstáculos
            y evitar errores en aquellos puntos cercanos al transmisor y
            receptor'''

    margin = 10
    '''Puntos que son mayores que el anterior y menores que el siguiente:
        potenciales obstáculos'''
    for j in range(margin,len(elev)-margin,1):
        if elev[j]>elev[j-1] and elev[j]>elev[j+1]:
            indices.append(j)


    if indices == []:
        indices.append(elev.index(max(elev)))

    for i in range(0,len(indices)):
        despejamiento=float(elev[indices[i]])-float(rayo[indices[i]])

        '''Si el obstáculo se encuentra dentro de los valores aceptables,
        se guarda la posición del obstáculo en una variable y se fija la
        bandera a 1 para identificar terreno con obstáculos'''
        if not indices[i] == 0 or indices[i] == len(elev)-1:
            if despejamiento/float(radio_f_aux[indices[i]]) >= -0.6 and
                despejamiento/float(radio_f_aux[indices[i]]) <=0.5:
                obstaculos.append(indices[i])
                desp.append(despejamiento)
                if flag == 0:
                    flag = 1

            if despejamiento/float(radio_f_aux[indices[i]]) >= 0.5:
                inviable.append(indices[i])
                if flag != 2:
                    flag = 2



    '''Se procesan los obstaculos y se dibujan'''
    if len(obstaculos) != 0:
        obstaculos,desp = procesa_obstaculos(obstaculos,m,elev,desp)

    if len(inviable) != 0:
        inviable,desp = procesa_obstaculos(inviable,m,elev,desp)
```

```
    for i in range(0,len(obstaculos)):
        plot(d_x[obstaculos[i]],elev[obstaculos[i]],'oy')

    for i in range(0,len(inviable)):
        plot(d_x[inviable[i]],elev[inviable[i]],'or')

    return flag,obstaculos,inviable,desp


'''Esta variable contendrá las distancias de TX a primer obstáculo, del segundo
al tercero, etc hasta llegar al receptor'''

def calcula_distancia_obstaculos(h1,h2,x1,x2,d_x,obstaculos,elev):
    distancias=[]
    if len(obstaculos) == 1:
        altura = elev[obstaculos[0]]
        print(f'\r\nEl obstáculo se encuentra a una altura de {elev[obstaculos[
            0]]:.2f} metros')
        d1 = math.sqrt((d_x[obstaculos[0]]-x1)**2-(elev[obstaculos[0]]-h1)**2)
        d2 = math.sqrt((d_x[obstaculos[0]]-x2)**2-(elev[obstaculos[0]]-h2)**2)
        distancias.append(d1)
        distancias.append(d2)

    return distancias

def dibuja_obstaculo(obs_rep,elev,d_x,type):
    for i in range(0,len(obs_rep)-1):
        raux=[]
        dxaux=[]
        for k in range (0,len(obs_rep[i])-1):
            indice = d_x.index(obs_rep[k])
            raux.append(elev[indice])
            dxaux.append(obs_rep[k])
            if type == 1:
                plt.plot(dxaux,raux,'b-')
            if type == 2:
                plt.plot(dxaux,raux,'r-')
```

funciones_terreno_liso.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Nov  2 17:56:23 2019

@author: Juan Antonio
"""

import math
import sys


def medio_llano(dis,a_e,h_TXm,h_RXm,h1,h2,wave_len,freq,rafio_f):
#Se calcula la distancia limite de visibilidad directa:
    d_los=(math.sqrt(2*a_e*1e3))*(math.sqrt(h1)+math.sqrt(h2))
   #d_los2=(math.sqrt(2*a_e*1e3))*(math.sqrt(h_TXm)+math.sqrt(h_RXm))
    pol=polarizacion()
    #Si la distancia del enlace es mayor a la d_los se calcula K y la fuerza
    de difraccion del campo en dBs
    '''Si dis>d_los estamos en un trayecto transhorizonte. Aplicamos el
    apartado
    3.1 de la recomendación'''
    if dis>=d_los:
        K=admitancia(a_e,wave_len,pol)
        Ldif=perdidas_difraccion(dis,a_e,wave_len,K,pol,h_TXm,h_RXm)
        #perdidas_total = 0
        #print('tus muertos son llanos y K vale ' + repr(K))
        #print('las pérdidas son '+repr(Ldif))
    else:
        h_libre,hreq=calc_h_libre(dis,h1,h2,wave_len,a_e)
        if h_libre > hreq:
            Ldif=0
        else:
            a_em=0.5*(dis/(math.sqrt(h1)+math.sqrt(h2)))**2
            K=admitancia(a_em,wave_len,pol)
            Ah=perdidas_difraccion(dis,a_em, wave_len,K,pol,h_TXm,h_RXm)
            if Ah < 0:
                Ldif=0
            else:
                Ldif=(1-h_libre/hreq)*Ah
            '''MÉTODO 3.1.1'''
    return Ldif

def polarizacion():
    flag = True
    while(flag):
        pol=input('Introduzca polarización: H para polarización horizontal, V
            para vertical: ')
        if pol == 'h' or pol == 'H':
            flag = False
        elif pol == 'v' or pol == 'V':
            flag = False
        else:
            print('Valor no válido.')
    return pol
```

```
def perdidas_difraccion(d,a_e,wave_len,K,pol,h_TX,h_RX):
    f=3e8/wave_len
    beta = 0

    if K < 0.001:
        loss = 0
        return loss
    elif K > 1:
        loss = "Error"
        return loss
    else:
        beta = calcula_beta(pol,K,f)
        '''Duda sobre las unidades de a_e. D debe ir en km, y la longitud
        de onda en m, por tanto no sé si esa conversión es válida o no'''
        X=beta*((math.pi/(wave_len*(a_e*1e3)**2))**(1/3))*d
        X2=beta*d*(math.pi/(wave_len*(a_e*1e3)**2))**(float(1)/float(3))
        Y_TX=2*beta*((math.pi**2/(wave_len**2*a_e*1e3))**(1/3))*(h_TX)
        Y_RX=2*beta*((math.pi**2/(wave_len**2*a_e*1e3))**(1/3))*(h_RX)

        if X >= 1.6:
            #Fx=11 + math.log10(X)-17.6*X
            F=11+10*math.log10(X)-17.6*X
        elif X < 1.6:
            Fx=-20*math.log10(X)-5.6488*(X**1.425)
            F=-20*math.log10(X)-5.6488*(X**float(1.425))
        print('F(X) vale ' + repr(Fx))
        G_TX=calcula_G(Y_TX,beta,K)
        G_RX=calcula_G(Y_RX,beta,K)

        loss = Fx + G_TX + G_RX
        return loss


def calcula_G(Y,beta,K):
    B=beta*Y
    if B > 2:
        ganancia = 17.6*((B-1.1)**(1/2))-5*math.log10(B-1.1)-8
    else:
        ganancia=20*math.log10(B+0.1*B**3)

    if ganancia < (2+20*math.log10(K)):
        ganancia = 2 + 20*math.log10(K)
    return ganancia

def calcula_beta(pol,K,f):
    superficie=input('Por favor, indique si se trata de un trayecto por mar (
        m) o por tierra (t): ')
    if pol == 'h' or pol == 'h':
        beta = 1
    elif ((superficie=='t' or superficie =='T') and f>20e6) or ((superficie ==
        'm' or superficie == 'M') and f> 300e6):
        beta=(1+1.6*K**2+0.67*K**4)/(1+4.5*K**2+1.53*K**4)
    return beta
```

```
def admitancia(a_e,wave_len,pol):
    flag = True
    while(flag):
        try:
            eps=float(input('Introduzca la permitividad relativa efectiva: '))
            if isinstance(eps,int) or isinstance(eps,float):
                flag = False
            else:
                print('Valor no válido')
        except:
            print('Formato no válido, introdúzcalo de nuevo')
    flag = True
    while(flag):
        try:
            sigma=float(input('Introduzca la conductividad efectiva (S/m): '))
            if isinstance(sigma,int) or isinstance(sigma,float):
                flag = False
            else:
                print('Valor no válido')
        except:
            print('Formato no válido, introdúzcalo de nuevo')
    Kh=((2*math.pi*a_e/wave_len)**(-1/3))*((eps-1)**2+(60*wave_len*sigma)**2)**
        (-1/4)
    #Kh1=0.36*(a_e*(3e8/wave_len))**(-1/3)*((eps-1)**2+(18000*sigma/(3e8/wave_l
    en))**2)**(-1/4)
    if pol == 'v' or pol == 'V':
        Kv=Kh*(eps**2+(60*wave_len**sigma)**2)**(1/2)
        return Kv
    else:
        return Kh
```

```
def calc_h_libre(dis,h1,h2,wave_len,a_e):
    #Se calculam c y m
    a_e=a_e*1000
    c=(h1-h2)/(h1+h2)
    m=(dis**2)/(4*a_e*(h1+h2))
    #Se calcula b
    b=2*math.sqrt(float((m+1))/float(3*m))*math.cos(math.pi/3+(1/3)*math.acos((
        3*c/2)*math.sqrt((3*m)/(m+1)**3)))
    d1=dis/2*(1+b)
    d2=dis-d1
    h=((h1-(d1**2/(2*a_e*1e3)))*d2+(h2-(d2**2/(2*a_e*1e3)))*d1)/dis
    h_libre=((h1-(d1**2/(2*a_e)))*d2+(h2-d2**2/(2*a_e))*d1)/dis
    hreq=0.552*math.sqrt(d1*d2*wave_len/dis)

    return h,hreq
```

funciones_obstaculos.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Feb 16 11:17:50 2020

@author: Juan Antonio
"""

import math
import sys
from funciones import *
from Cv_Sv import *
from funciones_terreno_liso import *

#Método principal que llamará al método de cálculi de pérdidas que corresponda
#según al tipo de terreno
def medio_obstaculos(r_e,h_TXm,h_RXm,h1,h2,d,w_length,freq,obstaculos,radio_f,
    desp,distancias,elev,d_x):

    metodo, radio = selecciona_metodo(obstaculos)

    if metodo == '4.2' or metodo == '4.1':
        if radio != []:
            R = radio[0]
        else:
            R = F'n'
        Ldb = obstaculo_unico(w_length,desp,distancias,R,False)
    elif metodo == '4.3':
        Ldb = dos_obstaculos_aislados(w_length,desp,radio_f,h_TXm,h_RXm,h1,h2,
            obstaculos,elev,radio,d_x)
    elif metodo == '4.5':
        Ldb = obstaculos_multiples(obstaculos, elev, d_x, r_e, d, h1, h2,
            w_length,desp,distancias,radio,h_TXm,h_RXm)
    return Ldb

#Método que determina qué método se deberá aplicar según el número de
obstáculos
#y de la disponibilidad del radio de curvatura
def selecciona_metodo(obstaculos):
    flag = True
    metodo = ''
    radio = []
    num_obstaculos = len(obstaculos)

    '''Si hay más de dos obstáculos no se pedirá el radio de curvatura, sino
    que se aplicará el método 4.5 para un trayecto general'''

    if len(obstaculos) > 2:
        flag = False
        ask_radio = 'n'
    while(flag):
        ask_radio = input("¿Conoce el radio de curvatura del obstáculo?: [s/n]
            ")
        #print('\r\n')
        if ask_radio == 's' or ask_radio == 'n':
            flag = False
        else:
            print('\r\nRespuesta no válida, por favor introduzca \'s\' en caso
                afirmativo y \'n\' en caso negativo')
```

```
    flag = True
    #Si está disponible el radio de curvatura se pregunta al usuario y se
    selecciona
    -3el método.
    if ask_radio == 's':
        for i in range(0,len(obstaculos)):
            #Se preguntará hasta que se introduzca un valor válido
            while(flag):
                try:
                    ans = int(input("Introduzca el radio de curvatura sin
                        decimales en metros: "))
                    if isinstance(ans,int):
                        radio.append(ans)
                        flag = False
                except:
                    print('\r\nValor no válido, introduzca un número entero')
            flag = True

        if num_obstaculos == 1:
            metodo = '4.2'
        else:
            metodo = '4.3'
        return metodo, radio
    else:
        if num_obstaculos == 1:
            metodo = '4.1'
        elif num_obstaculos == 2:
            metodo = '4.3'
        else:
            metodo = '4.5'
        return metodo, radio

#Método para calcular las pérdidas producidas por un único obstáculo
def obstaculo_unico(w_length,desp,distancias,R,v):
    d1 = distancias[0]
    d2 = distancias[1]

    '''En primer lugar se calculan v y Jv, independientemente del radio de
    curvatura del obstáculo
    '''
    if not v:
        v = desp[0]*math.sqrt((2/w_length)*(1/d1+1/d2))

    if v > -0.78:
        Jv= 6.9 + 20*math.log10(math.sqrt((v-0.1)**2+1) + v - 0.1)
    else:
        v_mod="{0:.2f}".format(round(v,2))
        Jv=-20*math.log10((math.sqrt((1-Cv_Sv[v_mod][0]-Cv_Sv[v_mod][1])**2)+(
            Cv_Sv[v_mod][0]-Cv_Sv[v_mod][1])**2)/2)

    A = Jv

    '''Después,si se ha proporcionado un valor para el radio de curvatura
    se calcula el factor T(m,n) que da la atenuación adicional por la curvatura
    '''
```

```
    if isinstance(R,int):
        m = R*((d1+d2)/(d1*d2))/((math.pi*R/w_length)**(1/3))
        n = desp[0]*((math.pi*R/w_length)**(2/3))/R
        if m*n <= 4:
            Tmn = 7.2*(m**(1/2))-(2-12.5*n)*m+3.6*(m**(3/2))-0.8*(m**2)
        else:
            Tmn = -6-20*math.log10(m*n) + 7.2*(m**(1/2))-(2-17*n)*m+3.6*(m**(3/
                2))-0.8*(m**2)
        A = Jv + Tmn

    return A


def dos_obstaculos_aislados(w_length,desp,fresnel,h_TXm,h_RXm,h1,h2,obstaculos,
    elev,radio,d_x):
    M = 0
    Ldb = 1000
    #Se calcula la relación h/R para ver qué obstáculo predomina.
    y_tray_obs1 = d_x[obstaculos[0]]*(h2-h1)/d_x[-1]+h1
    y_tray_obs2 = d_x[obstaculos[1]]*(h2-h1)/d_x[-1]+h1
    h_1 = elev[obstaculos[0]]-y_tray_obs1
    h_2 = elev[obstaculos[1]]-y_tray_obs2
    rel1 = h_1/fresnel[obstaculos[0]]
    rel2 = h_2/fresnel[obstaculos[1]]
    primero = []
    segundo = []
    l_h1 = []
    l_h2 = []
    v = False

    #Si es el primero, se reajusta el despejamiento del segundo obstáculo
    if rel1 > rel2:
        print('\r\nDe los dos obstáculos, el principal es el primero')
        y_MRx = (d_x[obstaculos[1]]-d_x[obstaculos[0]])*(h2-elev[obstaculos[0]]
            )/(d_x[-1]-d_x[obstaculos[0]])+elev[obstaculos[0]]
        h2prima=elev[obstaculos[1]]-y_MRx
        h1prima=h_1
    #Si es el segundo, se reajustan ambos despejamientos.
    else:
        print('\r\nDe los dos obstáculos, el principal es el segundo')

        y_obs1_RX = (d_x[obstaculos[1]]-d_x[obstaculos[0]])*(h2-elev[obstaculos
            [0]])/(d_x[-1]-d_x[obstaculos[0]])+elev[obstaculos[0]]
        h2prima=elev[obstaculos[1]]-y_obs1_RX

        y_TX_obs2 = (d_x[obstaculos[0]]-d_x[0])*(elev[obstaculos[1]]-h1)/(d_x[
            obstaculos[1]]-d_x[0])+h1
        h1prima=elev[obstaculos[0]]-y_TX_obs2


    #Para pasarlas al método obstaculo_unico los despejamientos deben estar en
    una cadena.
    l_h1.append(h1prima)
    l_h2.append(h2prima)
```

```
dis_a = d_x[obstaculos[0]]-d_x[0]
dis_b = d_x[obstaculos[1]]-d_x[obstaculos[0]]
dis_c = d_x[len(d_x)-1]-d_x[obstaculos[1]]
primero.append(obstaculos[0])
segundo.append(obstaculos[1])
#Se calculan las perdidas teniendo en cuenta si hay o no radio de
curvatura, haciendo uso de
#obstaculo_unico
if not radio:
    distancias = calcula_distancia_obstaculos(h1,elev[obstaculos[1]],d_x[0]
        ,d_x[obstaculos[1]],d_x,primero,elev)
    L1 = obstaculo_unico(w_length,l_h1,distancias,radio,v)
    distancias = calcula_distancia_obstaculos(elev[obstaculos[0]],h2,d_x[
        obstaculos[0]],d_x[-1],d_x,segundo,elev)
    L2 = obstaculo_unico(w_length,l_h2,distancias,radio,v)

    if L1 > 15 and L2 > 15:
        Lc = math.log10((dis_a+dis_b)*(dis_b+dis_c) / (dis_b*(dis_a+dis_b+
            dis_c)))
        L = L1+L2+Lc
    else:
        L = L1 + L2
    print(f'\r\nL1 vale {L1:.2f}')
    print(f'\r\nL2 vale {L2:.2f}')
else:
    print(f'\r\nLos radios de curvatura son {radio[0]:.2f} y {radio[1]:.2f}
         metros.')
    distancias = calcula_distancia_obstaculos(h1,elev[obstaculos[1]],d_x[0]
        ,d_x[obstaculos[1]],d_x,primero,elev)
    L1 = obstaculo_unico(w_length,l_h1,distancias,radio[0],v)
    distancias = calcula_distancia_obstaculos(elev[obstaculos[0]],h2,d_x[
        obstaculos[0]],d_x[-1],d_x,segundo,elev)
    L2 = obstaculo_unico(w_length,l_h2,distancias,radio[1],v)
    if L1 > 15 and L2 > 15:
        Lc = math.log10((dis_a+dis_b)*(dis_b+dis_c) / (dis_b*(dis_a+dis_b+
            dis_c)))
        L = L1+L2+Lc
    else:
        L = L1 + L2
#En cualquier caso, factor de corrección de Tc por la separación de las
antenas
alfa = math.atan(math.sqrt(dis_b*(dis_a+dis_b+dis_c)/(dis_a*dis_c)))
q = math.sqrt((2*(dis_a+dis_b+dis_c))/(w_length*dis_c*(dis_a+dis_b)))*abs(
    h_2)
p = math.sqrt((2*(dis_a+dis_b+dis_c))/(w_length*dis_a*(dis_b+dis_c)))*abs(
    h_1)
Tc=(12 - 20*math.log10(2/(1-alfa/math.pi)))*((q/p)**(2*p))
print(f'\r\nTc vale {Tc:.2f} dB')
L = L - Tc
return L
```

```
#Método para el cálculo de pérdidsa para múltiples obstáculos
def obstaculos_multiples(obstaculos, elev, d_x, r_e,d,h1,h2,w_length,desp,
    distancias,radio,h_TXm,h_RXm):
    flag = True
    Ce=1/r_e
    print('\r\n El trayecto tiene más de dos obstáculos. Seleccione qué metodo
        desea aplicar:')
    print('\t 1. Método de Bullington.')
    print('\t 2. Método completo para múltiples obstáculos.')
    '''Se convierte d_x a kilómetros'''
    d_xkm = [i/1000 for i in d_x]
    dkm=d/1000
    #Se pregunta al usuario por el método que quiere aplicar.
    while(flag):
        try:
            opcion = input("Introduzca 1 ó 2 según el método que quiera
                aplicar: ")
            if opcion == '1' or opcion == '2':
                flag = False
        except:
            print('\r\nRespuesta no válida, por favor introduzca 1 ó 2 según
                el método que quiera aplicar')

    #Si opción 1, se llama a bullington, si 2, se llama a bullington y se
    pasan sus pérdidas
    #al método completo.
    if opcion == '1':
        Lb = metodo_bullington(dkm,d_xkm,Ce,obstaculos,elev,h1,h2,w_length,desp
            ,distancias,radio,h_TXm,h_RXm)
    elif opcion == '2':
        Lba = metodo_bullington(dkm,d_xkm,Ce,obstaculos,elev,h1,h2,w_length,
            desp,distancias,radio,h_TXm,h_RXm)
        Lb = metodo_completo(dkm,d_xkm,Ce,obstaculos,elev,h1,h2,w_length,desp,
            distancias,radio,h_TXm,h_RXm,Lba)

    return Lb

#Método de Bullington
def metodo_bullington(dkm,d_xkm,Ce,obstaculos,elev,h1,h2,w_length,desp,
    distancias,radio,h_TXm,h_RXm):
    Saux = []
    for i in range(2,len(d_xkm)):
        aux = (elev[i]+500*Ce*d_xkm[i]*(dkm-d_xkm[i])-h1)/d_xkm[i]
        Saux.append(aux)
    Stim = max(Saux)
    Str = (h2-h1)/dkm
```

```
    #Se comparan las pendientes. En cada caso, se calculan las pérdidas como
    si hubiera
    #un obstáculo único, con el parámetro v definido según corresponda, y se
    reajustan
    #si es necesario.
    if Stim < Str:
        vaux = []
        for i in range(2,len(d_xkm)-1):
            aux = (elev[i]+500*Ce*d_xkm[i]*(dkm-d_xkm[i])-(h1*(dkm-d_xkm[i])+h2
                *d_xkm[i])/dkm)*sqrt(0.002*dkm/(w_length*d_xkm[i]*(dkm-d_xkm[i]
                )))
            vaux.append(aux)
        v=max(vaux)
        if v > -0.78:
            Luc = obstaculo_unico(w_length,desp,distancias,radio,v)
        else:
            Luc = 0
    else:
        Sraux = []
        for i in range(2,len(d_xkm)-1):
            if i==len(d_xkm)-1:
                print('pause')
            aux = (elev[i]+500*Ce*d_xkm[i]*(dkm-d_xkm[i])-h2)/(dkm-d_xkm[i])
            Sraux.append(aux)
        Srim = max(Sraux)
        #Punto y parámetro de difracción de Bullington.
        db = (h2-h1+Srim*dkm)/(Stim+Srim)
        vb = (h1+Stim*db-(h1*(dkm-db)+h2*db)/dkm)*math.sqrt(0.002*dkm/(w_length
            *db*(dkm-db)))
        Luc = obstaculo_unico(w_length,desp,distancias,radio,vb)

    Lb = Luc + (1-math.e**(-Luc/6))*(10+0.02*dkm)

    return Lb

#Método completo para estimación de pérdidas. Recibe las pérdidas de
Bullington para posteriormente
#aplicar un factor de corrección si fuera necesario.
def metodo_completo(dkm,d_xkm,Ce,obstaculos,elev,h1,h2,w_length,desp,distancias
    ,radio,h_TXm,h_RXm,Lba):

    v1aux=[]
    v2aux=[]
    for i in range (1,len(d_xkm)-1):
        aux = (d_xkm[i]-d_xkm[i-1])*(elev[i]-elev[i-1])
        aux2 = (d_xkm[i]-d_xkm[i-1])*(elev[i]*(2*d_xkm[i]+d_xkm[i-1])+elev[i-1]
            *(d_xkm[i]+2*d_xkm[i-1]))
        v1aux.append(aux)
        v2aux.append(aux2)
    v1 = sum(v1aux)
    v2 = sum(v2aux)
    hstip = (2*v1*dkm-v2)/(dkm**2)
    hsrip = (v2 - v1*dkm)/(dkm**2)
```

```
    #Todos los cálculos y operaciones siguientes corresponden a los
    procedimientos
    #definidos en la norma, en los que se determinan las alturas que tendrán
    TX y RX
    #en los supuestos ficticios.
    hobiaux = []
    aobtaux = []
    aobraux = []
    m = len(d_xkm)
    for i in range(2,m-1):
        aux = elev[i] - (h1*(dkm-d_xkm[i])+h2*d_xkm[i])/dkm
        hobiaux.append(aux)
        aobtaux.append(aux/d_xkm[i])
        aobraux.append(aux/(dkm-d_xkm[i]))
    hobs=max(hobiaux)
    aobt=max(aobtaux)
    aobr=max(aobraux)

    if hobs < 0:
        hstp = hstip
        hsrp = hsrip
    else:
        gt = aobt/(aobt+aobr)
        gr = aobr/(aobt+aobr)
        hstp = hstip - hobs*gt
        hsrp = hsrip - hobst*gr

    if hstp > h1:
        hst = elev[0]
    else:
        hst = hstp

    if hsrp > elev[-1]:
        hsr = elev[-1]
    else:
        hsr = hsrp

    #Finalmente, se determinan cuáles serán estas alturas.
    htsprima = h2 - hst
    hrsprima = h2 - hsr

    #Se definen los vectores elevación y obstáculos a cero, para el supuesto
    de tierra lisa
    obstaculo_liso = [0] * len(obstaculos)
    elev_cero = [0] * len(elev)
    #Se calculan las pérdidas de Bullington para tierra lisa con las nuevas
    alturas
    Lbs = metodo_bullington(dkm,d_xkm,Ce,obstaculo_liso,elev_cero,h1,h2,
        w_length,desp,distancias,radio,h_TXm,h_RXm)
    #Se calculan las pérdidas para una tierra esférica con las nuevas alturas
    Lsph = medio_llano(dkm*1000,1/Ce,h_TXm,h_RXm,htsprima,hrsprima,w_length,
        w_length/3e8,False)

    L = Lba + max(Lsph-Lbs,0)

    return L
```

Cv_sv.py

Este archivo contiene un diccionario con los valores de las integrales de Fresnel[30] C(v) y S(v) a

```
Cv_Sv=dict({"-10.00":[-0.4999,-0.4682],
"-9.99":[-0.4901,-0.4697],
"-9.98":[-0.4812,-0.4742],
"-9.97":[-0.4741,-0.4813],
"-9.96":[-0.4696,-0.4901],
"-9.95":[-0.4680,-0.5000],
"-9.94":[-0.4695,-0.5098],
"-9.93":[-0.4740,-0.5187],
"-9.92":[-0.4810,-0.5258],
"-9.91":[-0.4898,-0.5305],
"-9.90":[-0.4996,-0.5321],
"-9.89":[-0.5095,-0.5308],
"-9.88":[-0.5184,-0.5264],
"-9.87":[-0.5256,-0.5196],
"-9.86":[-0.5304,-0.5108],
"-9.85":[-0.5323,-0.5010],
"-9.84":[-0.5311,-0.4911],
"-9.83":[-0.5270,-0.4821],
"-9.82":[-0.5203,-0.4747],
"-9.81":[-0.5117,-0.4697],
"-9.80":[-0.5019,-0.4676],
"-9.79":[-0.4920,-0.4685],
"-9.78":[-0.4828,-0.4723],
"-9.77":[-0.4753,-0.4788],
"-9.76":[-0.4700,-0.4873],
"-9.75":[-0.4675,-0.4969],
"-9.74":[-0.4680,-0.5069],
"-9.73":[-0.4716,-0.5162],
"-9.72":[-0.4777,-0.5240],
"-9.71":[-0.4860,-0.5296],
"-9.70":[-0.4955,-0.5325],
"-9.69":[-0.5054,-0.5324],
"-9.68":[-0.5149,-0.5293],
"-9.67":[-0.5230,-0.5235],
"-9.66":[-0.5290,-0.5156],
"-9.65":[-0.5324,-0.5062],
"-9.64":[-0.5328,-0.4962],
"-9.63":[-0.5302,-0.4866],
"-9.62":[-0.5249,-0.4782],
"-9.61":[-0.5173,-0.4718],
"-9.60":[-0.5081,-0.4679],
"-9.59":[-0.4982,-0.4669],
"-9.58":[-0.4885,-0.4688],
"-9.57":[-0.4797,-0.4736],
"-9.56":[-0.4728,-0.4808],
"-9.55":[-0.4683,-0.4897],
"-9.54":[-0.4666,-0.4995],
"-9.53":[-0.4679,-0.5094],
"-9.52":[-0.4721,-0.5184],
"-9.51":[-0.4788,-0.5259],
"-9.50":[-0.4873,-0.5310],
"-9.49":[-0.4970,-0.5334],
"-9.48":[-0.5069,-0.5329],
"-9.47":[-0.5163,-0.5294],
"-9.46":[-0.5242,-0.5234],
"-9.45":[-0.5300,-0.5153],
"-9.44":[-0.5332,-0.5059],
"-9.43":[-0.5335,-0.4959],
"-9.42":[-0.5309,-0.4863],
```

partir del parámetro v de difracción.

```
"-9.41":[-0.5256,-0.4779],
"-9.40":[-0.5180,-0.4713],
"-9.39":[-0.5089,-0.4673],
"-9.38":[-0.4990,-0.4661],
"-9.37":[-0.4892,-0.4678],
"-9.36":[-0.4803,-0.4723],
"-9.35":[-0.4731,-0.4791],
"-9.34":[-0.4682,-0.4878],
"-9.33":[-0.4660,-0.4975],
"-9.32":[-0.4667,-0.5074],
"-9.31":[-0.4702,-0.5168],
"-9.30":[-0.4763,-0.5247],
"-9.29":[-0.4844,-0.5305],
"-9.28":[-0.4938,-0.5337],
"-9.27":[-0.5037,-0.5341],
"-9.26":[-0.5134,-0.5317],
"-9.25":[-0.5219,-0.5265],
"-9.24":[-0.5286,-0.5191],
"-9.23":[-0.5330,-0.5102],
"-9.22":[-0.5345,-0.5003],
"-9.21":[-0.5332,-0.4904],
"-9.20":[-0.5291,-0.4814],
"-9.19":[-0.5226,-0.4738],
"-9.18":[-0.5143,-0.4684],
"-9.17":[-0.5047,-0.4656],
"-9.16":[-0.4947,-0.4657],
"-9.15":[-0.4852,-0.4685],
"-9.14":[-0.4769,-0.4740],
"-9.13":[-0.4704,-0.4816],
"-9.12":[-0.4664,-0.4907],
"-9.11":[-0.4651,-0.5006],
"-9.10":[-0.4666,-0.5104],
"-9.09":[-0.4709,-0.5194],
"-9.08":[-0.4775,-0.5269],
"-9.07":[-0.4859,-0.5322],
"-9.06":[-0.4955,-0.5348],
"-9.05":[-0.5055,-0.5347],
"-9.04":[-0.5150,-0.5318],
"-9.03":[-0.5234,-0.5264],
"-9.02":[-0.5299,-0.5188],
"-9.01":[-0.5340,-0.5097],
"-9.00":[-0.5354,-0.4999],
"-8.99":[-0.5340,-0.4900],
"-8.98":[-0.5299,-0.4809],
"-8.97":[-0.5234,-0.4733],
"-8.96":[-0.5151,-0.4678],
"-8.95":[-0.5056,-0.4649],
"-8.94":[-0.4956,-0.4647],
"-8.93":[-0.4860,-0.4672],
"-8.92":[-0.4774,-0.4724],
"-8.91":[-0.4706,-0.4797],
"-8.90":[-0.4661,-0.4886],
"-8.89":[-0.4642,-0.4983],
"-8.88":[-0.4651,-0.5083],
"-8.87":[-0.4687,-0.5176],
"-8.86":[-0.4747,-0.5255],
"-8.85":[-0.4827,-0.5315],
"-8.84":[-0.4920,-0.5351],
"-8.83":[-0.5019,-0.5360],
"-8.82":[-0.5117,-0.5341],
```

```
"-8.81":[-0.5206,-0.5297],
"-8.80":[-0.5280,-0.5229],
"-8.79":[-0.5332,-0.5145],
"-8.78":[-0.5359,-0.5049],
"-8.77":[-0.5359,-0.4949],
"-8.76":[-0.5332,-0.4853],
"-8.75":[-0.5280,-0.4768],
"-8.74":[-0.5207,-0.4700],
"-8.73":[-0.5118,-0.4655],
"-8.72":[-0.5020,-0.4636],
"-8.71":[-0.4921,-0.4643],
"-8.70":[-0.4827,-0.4677],
"-8.69":[-0.4747,-0.4736],
"-8.68":[-0.4684,-0.4813],
"-8.67":[-0.4645,-0.4905],
"-8.66":[-0.4632,-0.5004],
"-8.65":[-0.4647,-0.5103],
"-8.64":[-0.4687,-0.5194],
"-8.63":[-0.4750,-0.5271],
"-8.62":[-0.4831,-0.5329],
"-8.61":[-0.4925,-0.5362],
"-8.60":[-0.5025,-0.5369],
"-8.59":[-0.5123,-0.5350],
"-8.58":[-0.5212,-0.5305],
"-8.57":[-0.5285,-0.5238],
"-8.56":[-0.5339,-0.5153],
"-8.55":[-0.5368,-0.5058],
"-8.54":[-0.5370,-0.4958],
"-8.53":[-0.5347,-0.4862],
"-8.52":[-0.5298,-0.4775],
"-8.51":[-0.5228,-0.4703],
"-8.50":[-0.5142,-0.4653],
"-8.49":[-0.5045,-0.4628],
"-8.48":[-0.4946,-0.4629],
"-8.47":[-0.4850,-0.4656],
"-8.46":[-0.4764,-0.4707],
"-8.45":[-0.4695,-0.4779],
"-8.44":[-0.4647,-0.4866],
"-8.43":[-0.4624,-0.4963],
"-8.42":[-0.4627,-0.5063],
"-8.41":[-0.4656,-0.5158],
"-8.40":[-0.4709,-0.5243],
"-8.39":[-0.4782,-0.5311],
"-8.38":[-0.4870,-0.5357],
"-8.37":[-0.4968,-0.5379],
"-8.36":[-0.5067,-0.5375],
"-8.35":[-0.5163,-0.5345],
"-8.34":[-0.5247,-0.5291],
"-8.33":[-0.5314,-0.5218],
"-8.32":[-0.5360,-0.5129],
"-8.31":[-0.5382,-0.5032],
"-8.30":[-0.5377,-0.4932],
"-8.29":[-0.5348,-0.4837],
"-8.28":[-0.5294,-0.4753],
"-8.27":[-0.5221,-0.4685],
"-8.26":[-0.5133,-0.4638],
"-8.25":[-0.5036,-0.4616],
"-8.24":[-0.4936,-0.4619],
"-8.23":[-0.4841,-0.4648],
"-8.22":[-0.4756,-0.4700],
```

```
"-8.22":[-0.4756,-0.4700],
"-8.21":[-0.4687,-0.4772],
"-8.20":[-0.4638,-0.4859],
"-8.19":[-0.4614,-0.4955],
"-8.18":[-0.4615,-0.5055],
"-8.17":[-0.4641,-0.5151],
"-8.16":[-0.4691,-0.5238],
"-8.15":[-0.4761,-0.5309],
"-8.14":[-0.4847,-0.5360],
"-8.13":[-0.4943,-0.5387],
"-8.12":[-0.5042,-0.5390],
"-8.11":[-0.5139,-0.5367],
"-8.10":[-0.5228,-0.5320],
"-8.09":[-0.5301,-0.5253],
"-8.08":[-0.5356,-0.5170],
"-8.07":[-0.5387,-0.5075],
"-8.06":[-0.5394,-0.4976],
"-8.05":[-0.5376,-0.4877],
"-8.04":[-0.5334,-0.4787],
"-8.03":[-0.5270,-0.4710],
"-8.02":[-0.5190,-0.4651],
"-8.01":[-0.5097,-0.4615],
"-8.00":[-0.4998,-0.4602],
"-7.99":[-0.4899,-0.4615],
"-7.98":[-0.4806,-0.4651],
"-7.97":[-0.4726,-0.4710],
"-7.96":[-0.4662,-0.4787],
"-7.95":[-0.4619,-0.4877],
"-7.94":[-0.4600,-0.4975],
"-7.93":[-0.4606,-0.5074],
"-7.92":[-0.4636,-0.5169],
"-7.91":[-0.4688,-0.5254],
"-7.90":[-0.4760,-0.5323],
"-7.89":[-0.4846,-0.5373],
"-7.88":[-0.4942,-0.5400],
"-7.87":[-0.5042,-0.5402],
"-7.86":[-0.5139,-0.5380],
"-7.85":[-0.5228,-0.5335],
"-7.84":[-0.5304,-0.5270],
"-7.83":[-0.5361,-0.5188],
"-7.82":[-0.5396,-0.5094],
"-7.81":[-0.5408,-0.4995],
"-7.80":[-0.5395,-0.4896],
"-7.79":[-0.5358,-0.4804],
"-7.78":[-0.5300,-0.4722],
"-7.77":[-0.5225,-0.4657],
"-7.76":[-0.5135,-0.4613],
"-7.75":[-0.5038,-0.4591],
"-7.74":[-0.4938,-0.4593],
"-7.73":[-0.4842,-0.4620],
"-7.72":[-0.4755,-0.4668],
"-7.71":[-0.4682,-0.4737],
"-7.70":[-0.4628,-0.4820],
"-7.69":[-0.4595,-0.4914],
"-7.68":[-0.4586,-0.5014],
"-7.67":[-0.4601,-0.5112],
"-7.66":[-0.4638,-0.5205],
"-7.65":[-0.4697,-0.5285],
"-7.64":[-0.4773,-0.5350],
```

```
"-7.63":[-0.4863,-0.5394],
"-7.62":[-0.4960,-0.5416],
"-7.61":[-0.5060,-0.5414],
"-7.60":[-0.5156,-0.5389],
"-7.59":[-0.5244,-0.5341],
"-7.58":[-0.5318,-0.5274],
"-7.57":[-0.5374,-0.5192],
"-7.56":[-0.5409,-0.5098],
"-7.55":[-0.5422,-0.4999],
"-7.54":[-0.5410,-0.4900],
"-7.53":[-0.5376,-0.4807],
"-7.52":[-0.5320,-0.4724],
"-7.51":[-0.5247,-0.4656],
"-7.50":[-0.5160,-0.4607],
"-7.49":[-0.5064,-0.4580],
"-7.48":[-0.4964,-0.4576],
"-7.47":[-0.4867,-0.4595],
"-7.46":[-0.4776,-0.4637],
"-7.45":[-0.4697,-0.4698],
"-7.44":[-0.4635,-0.4777],
"-7.43":[-0.4593,-0.4867],
"-7.42":[-0.4573,-0.4965],
"-7.41":[-0.4575,-0.5064],
"-7.40":[-0.4601,-0.5161],
"-7.39":[-0.4648,-0.5249],
"-7.38":[-0.4714,-0.5323],
"-7.37":[-0.4796,-0.5381],
"-7.36":[-0.4889,-0.5418],
"-7.35":[-0.4987,-0.5433],
"-7.34":[-0.5087,-0.5425],
"-7.33":[-0.5182,-0.5394],
"-7.32":[-0.5267,-0.5343],
"-7.31":[-0.5339,-0.5273],
"-7.30":[-0.5393,-0.5189],
"-7.29":[-0.5426,-0.5095],
"-7.28":[-0.5437,-0.4996],
"-7.27":[-0.5426,-0.4897],
"-7.26":[-0.5392,-0.4803],
"-7.25":[-0.5338,-0.4719],
"-7.24":[-0.5266,-0.4650],
"-7.23":[-0.5181,-0.4599],
"-7.22":[-0.5086,-0.4568],
"-7.21":[-0.4986,-0.4559],
"-7.20":[-0.4887,-0.4573],
"-7.19":[-0.4794,-0.4608],
"-7.18":[-0.4711,-0.4664],
"-7.17":[-0.4643,-0.4736],
"-7.16":[-0.4592,-0.4822],
"-7.15":[-0.4563,-0.4918],
"-7.14":[-0.4555,-0.5017],
"-7.13":[-0.4569,-0.5116],
"-7.12":[-0.4605,-0.5209],
"-7.11":[-0.4660,-0.5292],
"-7.10":[-0.4733,-0.5360],
"-7.09":[-0.4819,-0.5411],
"-7.08":[-0.4914,-0.5441],
"-7.07":[-0.5014,-0.5450],
"-7.06":[-0.5112,-0.5437],
"-7.05":[-0.5206,-0.5402],
```

```
"-7.04":[-0.5290,-0.5347],
"-7.03":[-0.5359,-0.5276],
"-7.02":[-0.5411,-0.5191],
"-7.01":[-0.5444,-0.5096],
"-7.00":[-0.5455,-0.4997],
"-6.99":[-0.5444,-0.4898],
"-6.98":[-0.5411,-0.4803],
"-6.97":[-0.5359,-0.4718],
"-6.96":[-0.5290,-0.4646],
"-6.95":[-0.5207,-0.4591],
"-6.94":[-0.5114,-0.4556],
"-6.93":[-0.5015,-0.4541],
"-6.92":[-0.4915,-0.4548],
"-6.91":[-0.4820,-0.4576],
"-6.90":[-0.4732,-0.4624],
"-6.89":[-0.4657,-0.4690],
"-6.88":[-0.4598,-0.4771],
"-6.87":[-0.4558,-0.4862],
"-6.86":[-0.4538,-0.4960],
"-6.85":[-0.4539,-0.5060],
"-6.84":[-0.4562,-0.5157],
"-6.83":[-0.4605,-0.5247],
"-6.82":[-0.4666,-0.5326],
"-6.81":[-0.4742,-0.5390],
"-6.80":[-0.4831,-0.5436],
"-6.79":[-0.4927,-0.5463],
"-6.78":[-0.5027,-0.5469],
"-6.77":[-0.5125,-0.5453],
"-6.76":[-0.5218,-0.5417],
"-6.75":[-0.5302,-0.5362],
"-6.74":[-0.5372,-0.5291],
"-6.73":[-0.5425,-0.5207],
"-6.72":[-0.5460,-0.5113],
"-6.71":[-0.5474,-0.5015],
"-6.70":[-0.5467,-0.4915],
"-6.69":[-0.5440,-0.4819],
"-6.68":[-0.5393,-0.4731],
"-6.67":[-0.5329,-0.4654],
"-6.66":[-0.5250,-0.4593],
"-6.65":[-0.5161,-0.4549],
"-6.64":[-0.5064,-0.4525],
"-6.63":[-0.4964,-0.4521],
"-6.62":[-0.4866,-0.4538],
"-6.61":[-0.4773,-0.4575],
"-6.60":[-0.4690,-0.4631],
"-6.59":[-0.4620,-0.4702],
"-6.58":[-0.4566,-0.4786],
"-6.57":[-0.4531,-0.4879],
"-6.56":[-0.4515,-0.4978],
"-6.55":[-0.4520,-0.5078],
"-6.54":[-0.4546,-0.5174],
"-6.53":[-0.4590,-0.5264],
"-6.52":[-0.4652,-0.5342],
"-6.51":[-0.4728,-0.5406],
"-6.50":[-0.4816,-0.5454],
"-6.49":[-0.4912,-0.5482],
"-6.48":[-0.5011,-0.5491],
"-6.47":[-0.5110,-0.5479],
"-6.46":[-0.5205,-0.5448],
"-6.45":[-0.5292,-0.5398],
```

```
"-6.44":[-0.5366,-0.5332],
"-6.43":[-0.5426,-0.5252],
"-6.42":[-0.5469,-0.5162],
"-6.41":[-0.5492,-0.5065],
"-6.40":[-0.5496,-0.4965],
"-6.39":[-0.5480,-0.4866],
"-6.38":[-0.5444,-0.4773],
"-6.37":[-0.5391,-0.4689],
"-6.36":[-0.5322,-0.4617],
"-6.35":[-0.5240,-0.4560],
"-6.34":[-0.5148,-0.4520],
"-6.33":[-0.5050,-0.4500],
"-6.32":[-0.4951,-0.4499],
"-6.31":[-0.4852,-0.4518],
"-6.30":[-0.4760,-0.4555],
"-6.29":[-0.4677,-0.4611],
"-6.28":[-0.4606,-0.4681],
"-6.27":[-0.4551,-0.4764],
"-6.26":[-0.4512,-0.4856],
"-6.25":[-0.4493,-0.4954],
"-6.24":[-0.4493,-0.5054],
"-6.23":[-0.4512,-0.5152],
"-6.22":[-0.4550,-0.5244],
"-6.21":[-0.4606,-0.5327],
"-6.20":[-0.4676,-0.5398],
"-6.19":[-0.4759,-0.5454],
"-6.18":[-0.4851,-0.5493],
"-6.17":[-0.4949,-0.5513],
"-6.16":[-0.5048,-0.5514],
"-6.15":[-0.5147,-0.5496],
"-6.14":[-0.5239,-0.5460],
"-6.13":[-0.5324,-0.5406],
"-6.12":[-0.5396,-0.5337],
"-6.11":[-0.5454,-0.5256],
"-6.10":[-0.5495,-0.5165],
"-6.09":[-0.5518,-0.5068],
"-6.08":[-0.5522,-0.4968],
"-6.07":[-0.5508,-0.4869],
"-6.06":[-0.5475,-0.4775],
"-6.05":[-0.5424,-0.4689],
"-6.04":[-0.5358,-0.4614],
"-6.03":[-0.5280,-0.4552],
"-6.02":[-0.5191,-0.4507],
"-6.01":[-0.5095,-0.4479],
"-6.00":[-0.4995,-0.4470],
"-5.99":[-0.4896,-0.4479],
"-5.98":[-0.4800,-0.4507],
"-5.97":[-0.4711,-0.4552],
"-5.96":[-0.4632,-0.4613],
"-5.95":[-0.4566,-0.4688],
"-5.94":[-0.4514,-0.4774],
"-5.93":[-0.4480,-0.4867],
"-5.92":[-0.4464,-0.4966],
"-5.91":[-0.4466,-0.5066],
"-5.90":[-0.4486,-0.5163],
"-5.89":[-0.4524,-0.5256],
"-5.88":[-0.4578,-0.5339],
"-5.87":[-0.4647,-0.5412],
"-5.86":[-0.4728,-0.5470],
"-5.85":[-0.4818,-0.5513],
```

```
"-5.84":[-0.4915,-0.5538],
"-5.83":[-0.5015,-0.5546],
"-5.82":[-0.5114,-0.5535],
"-5.81":[-0.5210,-0.5506],
"-5.80":[-0.5298,-0.5460],
"-5.79":[-0.5377,-0.5400],
"-5.78":[-0.5444,-0.5325],
"-5.77":[-0.5497,-0.5240],
"-5.76":[-0.5533,-0.5147],
"-5.75":[-0.5551,-0.5049],
"-5.74":[-0.5552,-0.4949],
"-5.73":[-0.5535,-0.4851],
"-5.72":[-0.5500,-0.4757],
"-5.71":[-0.5450,-0.4671],
"-5.70":[-0.5385,-0.4595],
"-5.69":[-0.5307,-0.4532],
"-5.68":[-0.5219,-0.4484],
"-5.67":[-0.5125,-0.4453],
"-5.66":[-0.5026,-0.4438],
"-5.65":[-0.4926,-0.4442],
"-5.64":[-0.4828,-0.4462],
"-5.63":[-0.4736,-0.4500],
"-5.62":[-0.4652,-0.4554],
"-5.61":[-0.4578,-0.4621],
"-5.60":[-0.4517,-0.4700],
"-5.59":[-0.4471,-0.4789],
"-5.58":[-0.4442,-0.4884],
"-5.57":[-0.4429,-0.4984],
"-5.56":[-0.4434,-0.5083],
"-5.55":[-0.4456,-0.5181],
"-5.54":[-0.4494,-0.5273],
"-5.53":[-0.4548,-0.5357],
"-5.52":[-0.4616,-0.5430],
"-5.51":[-0.4696,-0.5491],
"-5.50":[-0.4784,-0.5537],
"-5.49":[-0.4879,-0.5567],
"-5.48":[-0.4978,-0.5580],
"-5.47":[-0.5078,-0.5576],
"-5.46":[-0.5176,-0.5556],
"-5.45":[-0.5269,-0.5518],
"-5.44":[-0.5354,-0.5466],
"-5.43":[-0.5428,-0.5400],
"-5.42":[-0.5491,-0.5322],
"-5.41":[-0.5539,-0.5235],
"-5.40":[-0.5572,-0.5140],
"-5.39":[-0.5589,-0.5042],
"-5.38":[-0.5589,-0.4942],
"-5.37":[-0.5572,-0.4844],
"-5.36":[-0.5538,-0.4749],
"-5.35":[-0.5490,-0.4662],
"-5.34":[-0.5427,-0.4584],
"-5.33":[-0.5352,-0.4518],
"-5.32":[-0.5268,-0.4465],
"-5.31":[-0.5175,-0.4427],
"-5.30":[-0.5078,-0.4405],
"-5.29":[-0.4978,-0.4399],
"-5.28":[-0.4879,-0.4410],
"-5.27":[-0.4783,-0.4437],
"-5.26":[-0.4692,-0.4479],
"-5.25":[-0.4610,-0.4536],
```

```
"-5.24":[-0.4538,-0.4605],
"-5.23":[-0.4479,-0.4686],
"-5.22":[-0.4434,-0.4775],
"-5.21":[-0.4403,-0.4870],
"-5.20":[-0.4389,-0.4969],
"-5.19":[-0.4391,-0.5069],
"-5.18":[-0.4409,-0.5167],
"-5.17":[-0.4443,-0.5261],
"-5.16":[-0.4491,-0.5348],
"-5.15":[-0.4553,-0.5427],
"-5.14":[-0.4627,-0.5494],
"-5.13":[-0.4710,-0.5549],
"-5.12":[-0.4802,-0.5589],
"-5.11":[-0.4898,-0.5614],
"-5.10":[-0.4998,-0.5624],
"-5.09":[-0.5098,-0.5617],
"-5.08":[-0.5195,-0.5595],
"-5.07":[-0.5288,-0.5558],
"-5.06":[-0.5373,-0.5506],
"-5.05":[-0.5449,-0.5442],
"-5.04":[-0.5514,-0.5366],
"-5.03":[-0.5567,-0.5281],
"-5.02":[-0.5605,-0.5189],
"-5.01":[-0.5628,-0.5092],
"-5.00":[-0.5636,-0.4992],
"-4.99":[-0.5628,-0.4892],
"-4.98":[-0.5605,-0.4795],
"-4.97":[-0.5567,-0.4703],
"-4.96":[-0.5515,-0.4618],
"-4.95":[-0.5450,-0.4541],
"-4.94":[-0.5375,-0.4476],
"-4.93":[-0.5290,-0.4423],
"-4.92":[-0.5198,-0.4384],
"-4.91":[-0.5101,-0.4360],
"-4.90":[-0.5002,-0.4351],
"-4.89":[-0.4902,-0.4357],
"-4.88":[-0.4804,-0.4378],
"-4.87":[-0.4711,-0.4414],
"-4.86":[-0.4624,-0.4464],
"-4.85":[-0.4546,-0.4526],
"-4.84":[-0.4479,-0.4600],
"-4.83":[-0.4423,-0.4682],
"-4.82":[-0.4380,-0.4773],
"-4.81":[-0.4352,-0.4869],
"-4.80":[-0.4338,-0.4968],
"-4.79":[-0.4339,-0.5067],
"-4.78":[-0.4355,-0.5166],
"-4.77":[-0.4386,-0.5261],
"-4.76":[-0.4431,-0.5350],
"-4.75":[-0.4488,-0.5432],
"-4.74":[-0.4557,-0.5504],
"-4.73":[-0.4636,-0.5566],
"-4.72":[-0.4723,-0.5615],
"-4.71":[-0.4817,-0.5650],
"-4.70":[-0.4914,-0.5671],
"-4.69":[-0.5014,-0.5678],
"-4.68":[-0.5114,-0.5670],
"-4.67":[-0.5211,-0.5648],
"-4.66":[-0.5304,-0.5611],
"-4.65":[-0.5391,-0.5562],
```

```
"-4.64":[-0.5469,-0.5500],
"-4.63":[-0.5538,-0.5427],
"-4.62":[-0.5595,-0.5346],
"-4.61":[-0.5641,-0.5257],
"-4.60":[-0.5672,-0.5162],
"-4.59":[-0.5690,-0.5064],
"-4.58":[-0.5694,-0.4964],
"-4.57":[-0.5683,-0.4864],
"-4.56":[-0.5658,-0.4768],
"-4.55":[-0.5619,-0.4675],
"-4.54":[-0.5568,-0.4590],
"-4.53":[-0.5505,-0.4512],
"-4.52":[-0.5432,-0.4444],
"-4.51":[-0.5350,-0.4387],
"-4.50":[-0.5260,-0.4343],
"-4.49":[-0.5165,-0.4311],
"-4.48":[-0.5067,-0.4293],
"-4.47":[-0.4967,-0.4289],
"-4.46":[-0.4868,-0.4299],
"-4.45":[-0.4771,-0.4323],
"-4.44":[-0.4678,-0.4360],
"-4.43":[-0.4591,-0.4410],
"-4.42":[-0.4512,-0.4471],
"-4.41":[-0.4443,-0.4542],
"-4.40":[-0.4383,-0.4623],
"-4.39":[-0.4336,-0.4711],
"-4.38":[-0.4301,-0.4804],
"-4.37":[-0.4279,-0.4902],
"-4.36":[-0.4270,-0.5001],
"-4.35":[-0.4276,-0.5101],
"-4.34":[-0.4295,-0.5199],
"-4.33":[-0.4327,-0.5294],
"-4.32":[-0.4371,-0.5383],
"-4.31":[-0.4428,-0.5466],
"-4.30":[-0.4494,-0.5540],
"-4.29":[-0.4571,-0.5604],
"-4.28":[-0.4655,-0.5658],
"-4.27":[-0.4746,-0.5700],
"-4.26":[-0.4841,-0.5730],
"-4.25":[-0.4940,-0.5746],
"-4.24":[-0.5040,-0.5749],
"-4.23":[-0.5139,-0.5739],
"-4.22":[-0.5236,-0.5716],
"-4.21":[-0.5329,-0.5680],
"-4.20":[-0.5417,-0.5632],
"-4.19":[-0.5498,-0.5573],
"-4.18":[-0.5570,-0.5504],
"-4.17":[-0.5633,-0.5426],
"-4.16":[-0.5684,-0.5341],
"-4.15":[-0.5725,-0.5249],
"-4.14":[-0.5753,-0.5153],
"-4.13":[-0.5768,-0.5054],
"-4.12":[-0.5771,-0.4955],
"-4.11":[-0.5760,-0.4855],
"-4.10":[-0.5737,-0.4758],
"-4.09":[-0.5701,-0.4665],
"-4.08":[-0.5654,-0.4576],
"-4.07":[-0.5596,-0.4495],
"-4.06":[-0.5528,-0.4422],
"-4.05":[-0.5452,-0.4358],
```

```
"-4.04":[-0.5368,-0.4304],
"-4.03":[-0.5277,-0.4261],
"-4.02":[-0.5182,-0.4230],
"-4.01":[-0.5084,-0.4211],
"-4.00":[-0.4984,-0.4205],
"-3.99":[-0.4885,-0.4211],
"-3.98":[-0.4786,-0.4230],
"-3.97":[-0.4691,-0.4261],
"-3.96":[-0.4601,-0.4303],
"-3.95":[-0.4516,-0.4357],
"-3.94":[-0.4439,-0.4420],
"-3.93":[-0.4370,-0.4492],
"-3.92":[-0.4311,-0.4573],
"-3.91":[-0.4261,-0.4660],
"-3.90":[-0.4223,-0.4752],
"-3.89":[-0.4197,-0.4848],
"-3.88":[-0.4182,-0.4947],
"-3.87":[-0.4180,-0.5047],
"-3.86":[-0.4189,-0.5147],
"-3.85":[-0.4211,-0.5244],
"-3.84":[-0.4244,-0.5338],
"-3.83":[-0.4289,-0.5428],
"-3.82":[-0.4343,-0.5512],
"-3.81":[-0.4408,-0.5588],
"-3.80":[-0.4481,-0.5656],
"-3.79":[-0.4562,-0.5715],
"-3.78":[-0.4649,-0.5764],
"-3.77":[-0.4741,-0.5803],
"-3.76":[-0.4837,-0.5830],
"-3.75":[-0.4936,-0.5845],
"-3.74":[-0.5036,-0.5849],
"-3.73":[-0.5135,-0.5841],
"-3.72":[-0.5233,-0.5822],
"-3.71":[-0.5329,-0.5791],
"-3.70":[-0.5419,-0.5750],
"-3.69":[-0.5505,-0.5698],
"-3.68":[-0.5584,-0.5637],
"-3.67":[-0.5655,-0.5567],
"-3.66":[-0.5718,-0.5489],
"-3.65":[-0.5771,-0.5404],
"-3.64":[-0.5815,-0.5314],
"-3.63":[-0.5848,-0.5220],
"-3.62":[-0.5869,-0.5122],
"-3.61":[-0.5880,-0.5023],
"-3.60":[-0.5880,-0.4923],
"-3.59":[-0.5868,-0.4824],
"-3.58":[-0.5845,-0.4727],
"-3.57":[-0.5811,-0.4633],
"-3.56":[-0.5767,-0.4543],
"-3.55":[-0.5713,-0.4459],
"-3.54":[-0.5650,-0.4381],
"-3.53":[-0.5579,-0.4311],
"-3.52":[-0.5501,-0.4249],
"-3.51":[-0.5416,-0.4196],
"-3.50":[-0.5326,-0.4152],
"-3.49":[-0.5231,-0.4119],
"-3.48":[-0.5134,-0.4097],
"-3.47":[-0.5035,-0.4085],
"-3.46":[-0.4935,-0.4084],
"-3.45":[-0.4835,-0.4094],
```

```
"-3.44":[-0.4738,-0.4114],
"-3.43":[-0.4643,-0.4145],
"-3.42":[-0.4551,-0.4186],
"-3.41":[-0.4465,-0.4237],
"-3.40":[-0.4385,-0.4296],
"-3.39":[-0.4311,-0.4364],
"-3.38":[-0.4246,-0.4439],
"-3.37":[-0.4188,-0.4521],
"-3.36":[-0.4139,-0.4608],
"-3.35":[-0.4100,-0.4700],
"-3.34":[-0.4071,-0.4796],
"-3.33":[-0.4052,-0.4894],
"-3.32":[-0.4043,-0.4994],
"-3.31":[-0.4045,-0.5094],
"-3.30":[-0.4057,-0.5193],
"-3.29":[-0.4079,-0.5290],
"-3.28":[-0.4111,-0.5385],
"-3.27":[-0.4153,-0.5476],
"-3.26":[-0.4204,-0.5562],
"-3.25":[-0.4263,-0.5642],
"-3.24":[-0.4331,-0.5716],
"-3.23":[-0.4405,-0.5783],
"-3.22":[-0.4486,-0.5842],
"-3.21":[-0.4572,-0.5892],
"-3.20":[-0.4663,-0.5933],
"-3.19":[-0.4758,-0.5966],
"-3.18":[-0.4855,-0.5988],
"-3.17":[-0.4954,-0.6001],
"-3.16":[-0.5054,-0.6003],
"-3.15":[-0.5154,-0.5996],
"-3.14":[-0.5253,-0.5979],
"-3.13":[-0.5349,-0.5952],
"-3.12":[-0.5442,-0.5917],
"-3.11":[-0.5531,-0.5872],
"-3.10":[-0.5616,-0.5818],
"-3.09":[-0.5695,-0.5757],
"-3.08":[-0.5767,-0.5688],
"-3.07":[-0.5833,-0.5613],
"-3.06":[-0.5891,-0.5531],
"-3.05":[-0.5941,-0.5445],
"-3.04":[-0.5982,-0.5354],
"-3.03":[-0.6015,-0.5259],
"-3.02":[-0.6038,-0.5162],
"-3.01":[-0.6052,-0.5063],
"-3.00":[-0.6057,-0.4963],
"-2.99":[-0.6053,-0.4863],
"-2.98":[-0.6038,-0.4764],
"-2.97":[-0.6015,-0.4667],
"-2.96":[-0.5983,-0.4572],
"-2.95":[-0.5942,-0.4481],
"-2.94":[-0.5893,-0.4394],
"-2.93":[-0.5836,-0.4312],
"-2.92":[-0.5772,-0.4235],
"-2.91":[-0.5701,-0.4165],
"-2.90":[-0.5624,-0.4101],
"-2.89":[-0.5541,-0.4045],
"-2.88":[-0.5454,-0.3996],
"-2.87":[-0.5362,-0.3956],
"-2.86":[-0.5268,-0.3924],
"-2.85":[-0.5171,-0.3900],
```

```
"-2.84":[-0.5072,-0.3886],
"-2.83":[-0.4972,-0.3880],
"-2.82":[-0.4872,-0.3883],
"-2.81":[-0.4773,-0.3895],
"-2.80":[-0.4675,-0.3915],
"-2.79":[-0.4579,-0.3944],
"-2.78":[-0.4487,-0.3982],
"-2.77":[-0.4397,-0.4027],
"-2.76":[-0.4313,-0.4080],
"-2.75":[-0.4233,-0.4140],
"-2.74":[-0.4158,-0.4207],
"-2.73":[-0.4090,-0.4279],
"-2.72":[-0.4028,-0.4358],
"-2.71":[-0.3973,-0.4441],
"-2.70":[-0.3925,-0.4529],
"-2.69":[-0.3885,-0.4621],
"-2.68":[-0.3853,-0.4715],
"-2.67":[-0.3828,-0.4812],
"-2.66":[-0.3812,-0.4911],
"-2.65":[-0.3805,-0.5011],
"-2.64":[-0.3805,-0.5111],
"-2.63":[-0.3814,-0.5210],
"-2.62":[-0.3831,-0.5309],
"-2.61":[-0.3856,-0.5406],
"-2.60":[-0.3889,-0.5500],
"-2.59":[-0.3930,-0.5591],
"-2.58":[-0.3978,-0.5679],
"-2.57":[-0.4033,-0.5763],
"-2.56":[-0.4094,-0.5842],
"-2.55":[-0.4161,-0.5915],
"-2.54":[-0.4235,-0.5983],
"-2.53":[-0.4313,-0.6045],
"-2.52":[-0.4396,-0.6101],
"-2.51":[-0.4483,-0.6150],
"-2.50":[-0.4574,-0.6192],
"-2.49":[-0.4668,-0.6226],
"-2.48":[-0.4764,-0.6254],
"-2.47":[-0.4862,-0.6273],
"-2.46":[-0.4961,-0.6285],
"-2.45":[-0.5061,-0.6289],
"-2.44":[-0.5161,-0.6286],
"-2.43":[-0.5261,-0.6275],
"-2.42":[-0.5359,-0.6256],
"-2.41":[-0.5455,-0.6230],
"-2.40":[-0.5550,-0.6197],
"-2.39":[-0.5641,-0.6157],
"-2.38":[-0.5729,-0.6110],
"-2.37":[-0.5814,-0.6056],
"-2.36":[-0.5894,-0.5996],
"-2.35":[-0.5970,-0.5931],
"-2.34":[-0.6040,-0.5860],
"-2.33":[-0.6105,-0.5784],
"-2.32":[-0.6165,-0.5704],
"-2.31":[-0.6218,-0.5620],
"-2.30":[-0.6266,-0.5532],
"-2.29":[-0.6306,-0.5440],
"-2.28":[-0.6340,-0.5346],
"-2.27":[-0.6368,-0.5250],
"-2.26":[-0.6388,-0.5152],
"-2.25":[-0.6401,-0.5053],
```

```
"-2.24":[-0.6408,-0.4953],
"-2.23":[-0.6407,-0.4853],
"-2.22":[-0.6399,-0.4754],
"-2.21":[-0.6384,-0.4655],
"-2.20":[-0.6363,-0.4557],
"-2.19":[-0.6335,-0.4461],
"-2.18":[-0.6300,-0.4367],
"-2.17":[-0.6259,-0.4276],
"-2.16":[-0.6212,-0.4188],
"-2.15":[-0.6159,-0.4103],
"-2.14":[-0.6100,-0.4022],
"-2.13":[-0.6036,-0.3945],
"-2.12":[-0.5967,-0.3873],
"-2.11":[-0.5894,-0.3805],
"-2.10":[-0.5816,-0.3743],
"-2.09":[-0.5734,-0.3685],
"-2.08":[-0.5648,-0.3633],
"-2.07":[-0.5560,-0.3587],
"-2.06":[-0.5468,-0.3547],
"-2.05":[-0.5374,-0.3513],
"-2.04":[-0.5278,-0.3484],
"-2.03":[-0.5181,-0.3462],
"-2.02":[-0.5082,-0.3447],
"-2.01":[-0.4982,-0.3437],
"-2.00":[-0.4883,-0.3434],
"-1.99":[-0.4783,-0.3437],
"-1.98":[-0.4683,-0.3447],
"-1.97":[-0.4584,-0.3462],
"-1.96":[-0.4487,-0.3484],
"-1.95":[-0.4391,-0.3511],
"-1.94":[-0.4296,-0.3545],
"-1.93":[-0.4204,-0.3584],
"-1.92":[-0.4115,-0.3629],
"-1.91":[-0.4028,-0.3678],
"-1.90":[-0.3945,-0.3733],
"-1.89":[-0.3865,-0.3793],
"-1.88":[-0.3788,-0.3858],
"-1.87":[-0.3716,-0.3927],
"-1.86":[-0.3648,-0.4000],
"-1.85":[-0.3584,-0.4077],
"-1.84":[-0.3524,-0.4157],
"-1.83":[-0.3470,-0.4241],
"-1.82":[-0.3420,-0.4328],
"-1.81":[-0.3376,-0.4418],
"-1.80":[-0.3336,-0.4509],
"-1.79":[-0.3302,-0.4603],
"-1.78":[-0.3273,-0.4699],
"-1.77":[-0.3250,-0.4796],
"-1.76":[-0.3232,-0.4895],
"-1.75":[-0.3219,-0.4994],
"-1.74":[-0.3212,-0.5094],
"-1.73":[-0.3211,-0.5194],
"-1.72":[-0.3215,-0.5293],
"-1.71":[-0.3224,-0.5393],
"-1.70":[-0.3238,-0.5492],
"-1.69":[-0.3258,-0.5590],
"-1.68":[-0.3283,-0.5687],
"-1.67":[-0.3313,-0.5782],
"-1.66":[-0.3348,-0.5876],
"-1.65":[-0.3388,-0.5968],
```

```
"-1.64":[-0.3433,-0.6057],
"-1.63":[-0.3482,-0.6144],
"-1.62":[-0.3535,-0.6229],
"-1.61":[-0.3593,-0.6310],
"-1.60":[-0.3655,-0.6389],
"-1.59":[-0.3720,-0.6464],
"-1.58":[-0.3790,-0.6536],
"-1.57":[-0.3862,-0.6605],
"-1.56":[-0.3939,-0.6670],
"-1.55":[-0.4018,-0.6731],
"-1.54":[-0.4100,-0.6788],
"-1.53":[-0.4185,-0.6841],
"-1.52":[-0.4272,-0.6890],
"-1.51":[-0.4361,-0.6935],
"-1.50":[-0.4453,-0.6975],
"-1.49":[-0.4546,-0.7011],
"-1.48":[-0.4641,-0.7043],
"-1.47":[-0.4737,-0.7070],
"-1.46":[-0.4834,-0.7093],
"-1.45":[-0.4933,-0.7111],
"-1.44":[-0.5032,-0.7125],
"-1.43":[-0.5131,-0.7134],
"-1.42":[-0.5231,-0.7139],
"-1.41":[-0.5331,-0.7139],
"-1.40":[-0.5431,-0.7135],
"-1.39":[-0.5531,-0.7127],
"-1.38":[-0.5630,-0.7114],
"-1.37":[-0.5728,-0.7097],
"-1.36":[-0.5826,-0.7076],
"-1.35":[-0.5923,-0.7050],
"-1.34":[-0.6018,-0.7021],
"-1.33":[-0.6112,-0.6987],
"-1.32":[-0.6205,-0.6950],
"-1.31":[-0.6296,-0.6908],
"-1.30":[-0.6386,-0.6863],
"-1.29":[-0.6473,-0.6815],
"-1.28":[-0.6558,-0.6763],
"-1.27":[-0.6641,-0.6707],
"-1.26":[-0.6722,-0.6648],
"-1.25":[-0.6801,-0.6587],
"-1.24":[-0.6877,-0.6522],
"-1.23":[-0.6950,-0.6454],
"-1.22":[-0.7021,-0.6383],
"-1.21":[-0.7089,-0.6310],
"-1.20":[-0.7154,-0.6234],
"-1.19":[-0.7217,-0.6156],
"-1.18":[-0.7276,-0.6075],
"-1.17":[-0.7332,-0.5993],
"-1.16":[-0.7385,-0.5908],
"-1.15":[-0.7436,-0.5821],
"-1.14":[-0.7482,-0.5733],
"-1.13":[-0.7526,-0.5643],
"-1.12":[-0.7567,-0.5552],
"-1.11":[-0.7604,-0.5459],
"-1.10":[-0.7638,-0.5365],
"-1.09":[-0.7669,-0.5270],
"-1.08":[-0.7696,-0.5174],
"-1.07":[-0.7721,-0.5077],
"-1.06":[-0.7741,-0.4979],
"-1.05":[-0.7759,-0.4880],
```

```
"-1.04":[-0.7774,-0.4782],
"-1.03":[-0.7785,-0.4682],
"-1.02":[-0.7793,-0.4582],
"-1.01":[-0.7797,-0.4483],
"-1.00":[-0.7799,-0.4383],
"-0.99":[-0.7797,-0.4283],
"-0.98":[-0.7793,-0.4183],
"-0.97":[-0.7785,-0.4083],
"-0.96":[-0.7774,-0.3984],
"-0.95":[-0.7760,-0.3885],
"-0.94":[-0.7744,-0.3786],
"-0.93":[-0.7724,-0.3688],
"-0.92":[-0.7702,-0.3590],
"-0.91":[-0.7676,-0.3494],
"-0.90":[-0.7648,-0.3398],
"-0.89":[-0.7617,-0.3303],
"-0.88":[-0.7584,-0.3208],
"-0.87":[-0.7548,-0.3115],
"-0.86":[-0.7510,-0.3023],
"-0.85":[-0.7469,-0.2932],
"-0.84":[-0.7425,-0.2841],
"-0.83":[-0.7379,-0.2753],
"-0.82":[-0.7331,-0.2665],
"-0.81":[-0.7281,-0.2579],
"-0.80":[-0.7228,-0.2493],
"-0.79":[-0.7174,-0.2410],
"-0.78":[-0.7117,-0.2327]})
```

REFERENCIAS

[1] María Lao Cañadas, «Simulador en Python del efecto de múltiples obstáculos sobre un radioenlace según la Rec. ITU-R P.526».

[2] «Google Elevation API», [En línea]. Available: https://developers.google.com/maps/documentation/elevation/overview

[3] Sector de Radiocomunicación de la ITU, «Rec. P.526 Propagación por difracción», 2013.

[4] Sector de Radiocomunicación de la ITU, «5 Difracción debida a pantallas delgadas, Rec. ITU-R P.526», 2013

[5] «Python», [En línea]. Available: https://es.python.org/

[6] «SRTM», [En línea]. Available: https://www2.jpl.nasa.gov/srtm/

[7] «Elevation», [En línea]. Available: https://pypi.org/project/elevation/

[8] «Estructura de archivos *hgt*», [En línea]. Available: https://librenepal.com/article/reading-srtm-data-with-python/

[9] «Lectura de archivos *hgt*», [En línea], Available: https://github.com/aatishnn/srtm-python

[10] «Open Elevation, Terrain Tiles, Amazon», [En línea]. Available: https://registry.opendata.aws/terrain-tiles/

[11] «Mapzen», [En línea]. Available: https://www.mapzen.com/blog/terrain-tile-service/

[12] «Docker». [En línea]. Available: https://www.docker.com/

[13] «Contenedores de Docker». Available: https://www.docker.com/resources/what-container

[14] Constantino Pérez Vega, «Capítulo 11: Propagación, Sistemas de Telecomunicación», 2007.

[15] «Google Earth», [En línea]. Available: https://www.google.com/intl/es/earth/

[16] «Radio Mobile», [En línea]. Available: https://www.ve2dbe.com/rmonline_s.asp

[17] «Elipsoide de Fresnel», [En línea]. Available: https://es.wikipedia.org/wiki/Zona_de_Fresnel#/media/Archivo:FresnelSVG1.svg

[18] Sector de Radiocomunicación de la ITU, « 4.1 Obstáculo único en arista filo de cuchillo, Figura 8, Rec. ITU-R P.526», 2013.

[19] «Fuentes de la aplicación», [En línea]. Available: https://docs.safe.com/fme/html/FME_Desktop_Documentation/FME_ReadersWriters/terraintilesaws/terraintilesaws.htm

[20] «Fuentes Google», [En línea]. Available: https://developers.google.com/earth-engine/datasets/catalog/USGS_SRTMGL1_003

[21] Sector de Radiocomunicación de la ITU, «3.2 Pérdidas de difracción para cualquier distancia a 10 MHz y frecuencias superiores, Rec. ITU-R P.526», 2013.

[22] Sector de Radiocomunicación de la ITU, «4.1 Obstáculo único en arista filo de cuchillo, Rec. ITU-R P.526», 2013.

[23] Sector de Radiocomunicación de la ITU, «4.2 Obstáculo único de forma redondeada, Rec. ITU-R P.526», 2013.

[24] Sector de Radiocomunicación de la ITU, «4.5 Método para un trayecto terrenal general, Rec. ITU-R P.526», 2013.

[25] Sector de Radiocomunicación de la ITU, «4.5.1 Modelo de Bullington, Rec. ITU-R P.526», 2013.

[26] Sector de Radiocomunicación de la ITU, «4.5.2 Método Completo, Rec. ITU-R P.526», 2013.

[27] Sector de Radiocomunicación de la ITU, «3.1.1.1 Influencia de las características eléctricas de la superficie de la Tierra, Rec. ITU-R P.526», 2013.

[28] Sector de Radiocomunicación de la ITU, « 7 Guía sobre la propagación por difracción, Figura 17, Rec ITU-R P.526», 2013.

[29] «Fórmula Haversine», [En línea]. Available: https://en.wikipedia.org/wiki/Haversine_formula