UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

PROCESADO Y ANÁLISIS DE SEÑALES DE BANDA ANCHA EN TRANSMISIÓN Y REFLEXIÓN EN EL RANGO DE LAS MICROONDAS

TRABAJO FIN DE GRADO

Junio - 2024

AUTOR: Luis Valls Antón

DIRECTOR/ES: Ernesto Ávila Navarro

"Si no puedes sobresalir con talento, triunfa con esfuerzo"

- Dave Weinbaum

AGRADECIMIENTOS

Me gustaría empezar agradeciendo a los departamentos y personas que han hecho este trabajo posible. En especial a Ernesto Ávila, tanto por lo paciente que ha sido conmigo como por todo lo que me ha enseñado y ayudado a lo largo de estos meses.

Por supuesto no podía faltar a gradecer a mi familia, tanto por el cariño como el apoyo que me han brindado a lo largo de estos años y sobretodo al final de esta etapa de mi vida. En especial me gustaría agradecer a mi tío Javi, pues él me motivó a entrar a esta carrera y ha sido mi consejero y guía durante el periodo universitario, sin él esto no habría sido lo mismo. También agradecer a mi abuela Geli por todo el apoyo, motivación y cariño que me ha dado, siempre incitándome a continuar con lo que realmente me gustaba. No me olvido de mis padres, sin ellos esto habría sido realmente imposible, siempre han sido mi mayor fuente de apoyo e inspiración y no dejan de demostrármelo año tras año, de verdad que no podría pedir unos mejores.

Por último me gustaría agradecer a mi abuelita Paquita, que si bien no podrá leer esto ni podrá seguir viendo como crezco, siempre me ha demostrado un amor y apoyo incondicional. Has sido una persona fundamental en mi vida y se que aunque ya no estés aquí siempre estarás a mi lado dándome fuerzas, gracias por todo.

Este trabajo va dedicado a todos vosotros.

RESUMEN

Este Trabajo Fin de Grado se encuadra dentro de las líneas de investigación del Grupo de Investigación de Laboratorio de Microondas de Elche, particularmente en la detección de tumores en cáncer de mama mediante la realización de actividades relacionadas con el procesado de señales y la caracterización dieléctrica de materiales biológicos y biocompatibles en el rango de las microondas.

A lo largo de este proyecto explicaremos las ventajas que puede llegar a tener esta tecnología frente a los métodos empleados hoy en día, para ello analizaremos los resultados que obtenemos para distintos casos de estudio como podrían ser casos de reflexión y transmisión de pulsos en aire o en medios distintos al aire.

Para el desarrollo de este trabajo se han empleado tanto elementos presentes en el laboratorio, como pueden ser antenas de distintos tipos, VNAs o un sistema de imagen médica, como softwares para procesar los datos obtenidos, entre ellos destacamos ADS y Python (empleado en el entorno de Visual Studio Code).

Una vez desarrollado el trabajo vemos que los resultados obtenidos son bastante buenos, llegando en la mayoría de casos a un margen de error muy pequeño, lo que parece indicar que el sistema podrá ser aplicado en un futuro para llevar a cabo mediciones reales.

Palabras clave: antena, microondas, procesado de señal, pulso, Python, microstrip, cable coaxial.

ABSTRACT

This Final Degree Project is part of the research lines of the Microwave Laboratory Research Group of Elche, particularly in the detection of tumors in breast cancer by carrying out activities related to signal processing and dielectric characterization of biological and biocompatible materials in the microwave range.

Throughout this project we will explain the advantages that this technology can have compared to the methods used today, to do so we will analyze the results we obtain for different study cases such as cases of reflection and transmission of pulses in air or a media other than air.

For the development of this work, both elements present in the laboratory have been used, such as antennas of different types, VNAs or a medical imaging system, as well as software to process the data obtained, among them we highlight ADS and Python (used in the Visual Studio Code environment).

Once the work has been carried out, we see that the results obtained are quite good, reaching in most cases a very small margin of error, which seems to indicate that the system can be applied in the future to carry out real measurements.

Keywords: antenna, microwave, signal processing, pulse, Python, microstrip, coaxial cable.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGOS PYTHON DESARROLLADOS

1. CAPÍTULO I: INTRODUCCIÓN

1.1. INTRODUCCIÓN AL TRABAJO

El cáncer de mama supone uno de los mayores desafíos de la salud a nivel mundial en la actualidad, a pesar de los grandes avances tanto en el diagnóstico como en el tratamiento la detección temprana sigue siendo un factor crucial para mejorar la calidad de vida de las pacientes, ya que gracias a esta podríamos lograr que no se llegase a expandir en gran medida, reduciendo así la gravedad de esta patología. El cáncer de mama llegó a ser el tumor más diagnosticado del mundo, superando por primera vez al cáncer de pulmón, según datos publicados en 2021 por el Centro de Investigaciones sobre el cáncer (IARC). En cuanto a la tasa de incidencia, se estiman 132 casos por cada 100.000 habitantes [1]. Según los últimos datos recogidos por la Sociedad Española de Oncología Médica (SEOM), en 2024 se diagnosticarán 36.395 nuevos casos de cáncer de mama, siendo este tipo de tumor más frecuente entre las mujeres en nuestro país por delante del cáncer colorrectal, de pulmón, cuerpo uterino, tiroides y páncreas. Los métodos convencionales como mamografias y resonancias magnéticas son claves con el fin de salvar las vidas de tantas personas, si bien tienen también sus propios riesgos, pues al emplear rayos X en las mamografias te expones a una cantidad pequeña de radiación ionizante, dosis que aunque sea reducida conlleva cierto riesgo, sobre todo en mujeres menores de 40 años. Por otra parte las resonancias no se usan en mujeres que tengan un riesgo promedio ya que puede tener un resultado anormal aun cuando no haya cáncer (falsos positivos) y esto puede dar a lugar a que la mujer realice pruebas o biopsias innecesarias. Debido a esto, la utilización de tecnologías de detección no invasivas y precisas se ha vuelto cada vez más relevante, ocasionando que el uso de las microondas como herramienta para la detección de cáncer de mama haya incrementado de manera considerable estos últimos años, ya que dispone de ciertas ventajas respecto a los otros métodos ya mencionados, tales como son la capacidad para penetrar tejidos biológicos con relativa facilidad y la sensibilidad a las propiedades dieléctricas de los distintos tejidos presentes, lo que permitiría una detección de anomalías de alta

precisión y resolución.

La elección de este tema viene motivada por el interés que he tenido siempre por el ámbito de la salud, que si bien no me ha llevado a estudiar directamente una carrera relacionada con ésta, ha sido siempre un campo muy llamativo para mi, por tanto no dudé cuando se me ofreció la oportunidad de relacionar la carrera que había estudiado con este sector. Por otra parte, el cáncer ya sea de mama o cualquier otro tipo, está presente directa o indirectamente en la vida de la mayoría de personas y por tanto poder ayudar no solo a la persona que lo padece sino también a sus familias y conocidos es un factor que me motiva aún más que el mencionado anteriormente. Al entrar al laboratorio de investigación se me comentó que recientemente hay una creciente necesidad de explorar y desarrollar nuevas tecnologías que ayuden a mejorar la precisión y eficacia de los métodos de diagnóstico ya existentes y que empleando técnicas de microondas se pueden llegar a obtener imágenes de alta resolución y una caracterización de tejidos menos invasiva que métodos convencionales, además de poder identificar patrones distintivos en los datos obtenidos, factor que podría resultar de gran utilidad tanto en el diagnóstico como en la personalización de los tratamientos de cada paciente. Este proyecto como se puede apreciar es muy ambicioso y complejo pero esperamos que con el esfuerzo de todos los que estamos implicados en el proyecto logremos poder contribuir al avance del conocimiento científico en este campo de investigación y aportar herramientas útiles y eficaces en la detección de esta patología. También que los resultados obtenidos puedan llegar a tener un cierto impacto positivo en la práctica médica permitiendo la detección precoz, el seguimiento de la progresión tumoral y el poder monitorizar la respuesta al tratamiento de los pacientes. En este contexto, la tarea que se me asignó dentro dentro del grupo de investigación y el tema del presente trabajo de fin de grado se centrará en el procesado de señales para la detección de cáncer de mama utilizando técnicas de microondas, para la caracterización dieléctrica de materiales biológicos y biocompatibles con el fin de detectar tumores. Por otra parte, se discutirán los desafíos actuales y las futuras perspectivas de esta tecnología en el ámbito de la salud, así como su posible impacto en la mejora de los protocolos existentes de detección y diagnóstico.

1.2. ESTADO DEL ARTE

Como nuestro trabajo fin de grado se encuadra en el uso de técnicas de detección de cáncer de mama en el rango de microondas dedicaremos el siguiente apartado para describir y comentar los métodos más empleados en la actualidad para detectar el cáncer de mama, éstos, como se ha comentado anteriormente son la mamografía y la resonancia magnética. A continuación, explicaremos en que consisten:

1.2.1. MAMOGRAFÍA

En primer lugar tenemos la mamografía o radiografía del seno, la cual es considerada el método de referencia para el cribado mamario y el diagnóstico de cáncer de mama. Se emplea radiación ionizante de rayos X para generar imágenes del tejido mamario. Si bien es un método muy empleado y efectivo no se recomienda para mujeres menores de 50 años o durante el embarazo, debido a problemas de seguridad relacionados con la exposición a esa radiación ionizante. Esto implica que casi el 40 % de las mujeres en Europa de entre 25 y 49 años no pueden emplear la modalidad de detección de cáncer de mama más convencional [2]. Y no es solo por la radiación, durante la mamografía se tiene la necesidad de aplanar la mama lo máximo que se pueda para conseguir una imagen lo más clara posible, proceso que puede resultar muy incomodo para las pacientes. También debemos tener en cuenta otros factores, como son la forma, el tamaño y composición los cuales pueden dificultar el distinguir el tumor en caso de que esté presente. Este factor tiene un mayor impacto en las pacientes más jóvenes pues, suelen contar con una mayor densidad de mama lo que dificulta la penetración de la radiación a través de los tejidos. Es por estos motivos que a pesar de que la mamografía obtenga buenos resultados, sobretodo en mujeres adultas cuya mama tenga una cierta cantidad de contenido graso (permite la penetración de la radiación a través de los tejidos), no es un método que nos pueda servir para todos los casos ni mucho menos cómodo para las pacientes [3]. En la siguiente imagen podemos apreciar el proceso que se lleva a cabo cuando se realiza una mamografía y los resultados que podemos obtener:

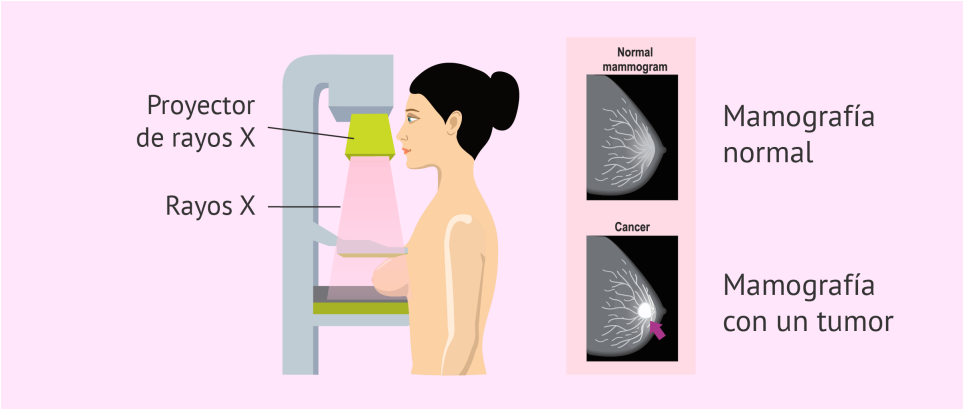


Figura 1.1: Mamografía

1.2.2. RESONANCIA MAGNÉTICA

Por otra parte tenemos la resonancia magnética, esta técnica utiliza un campo magnético y ondas de radio generadas por computadora para crear imágenes detalladas de los órganos y de los tejidos del cuerpo. La mayoría de los aparatos de resonancia magnética son grandes imanes con forma cilíndrica, que producen un potente campo magnético que obliga a los protones en el cuerpo a alinearse con ese campo. Cuando se pulsa una corriente de radiofrecuencia a través de un paciente, los protones son estimulados y giran fuera de equilibrio, luchando contra la fuerza del campo magnético. Cuando se apaga el campo de radiofrecuencia, los sensores son capaces de detectar la energía liberada mientras los protones se realinean con el campo magnético. El tiempo que tardan los protones para realinearse con el campo magnético, así como la cantidad de energía liberada, cambian dependiendo del entorno y la naturaleza química de las moléculas. Los médicos son capaces de identificar la diferencia entre los varios tipos de tejidos basándose en estas propiedades magnéticas. El aparato también puede producir imágenes 3D que nos permiten ver desde diferentes ángulos. En caso de considerarse oportuno se pueden llegar a emplear agentes de contraste para destacar estructuras vasculares y ayudar a caracterizar la inflamación y en nuestro caso los tumores. Los agentes más empleados son derivados de gadolinio, que tienen propiedades magnéticas que afectan los tiempos de relajación de los protones [4].

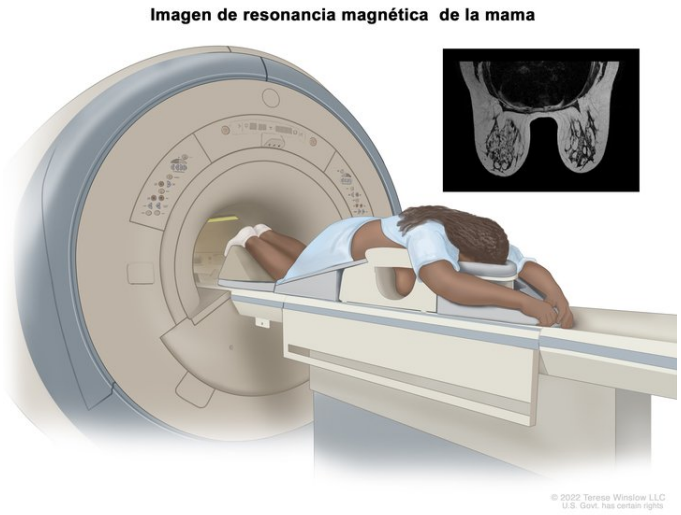


Figura 1.2: Resonancia magnética

1.3. OBJETIVOS

Los principales objetivos y actividades que se llevarán a cabo son los siguientes:

- Revisión bibliográfica de los principales algoritmos de procesado de señal.
- Implementación de algoritmos de procesado de señal para señales tipo RADAR basadas en antenas de banda ancha en el rango de las microondas.
- Implementación y testeo de un set-up para la medida de reflexión y trasmisión de antenas en diferentes escenarios. Evaluación de los resultados de las señales temporales de las antenas tanto en reflexión como en transmisión.
- Medida y análisis de las señales obtenidas por las antenas utilizando modelos o phantoms sintéticos biocompatibles de mama con el sistema de detección de tumores mamarios en el rango de microondas.

1.4. ESTRUCTURA DE LA MEMORIA

En el capítulo I realizamos una introducción al trabajo, hablamos de la situación del cáncer de mama en la actualidad, resaltamos la utilidad de las microondas para combatirlo, se comentan los objetivos que se persiguen y las actividades que realizaremos para cumplirlos.

En el capítulo II se explicarán los fundamentos teóricos del trabajo, qué son las microondas y los parámetros S, sistemas en los que se basan partes del trabajo y se dará una base teórica sobre procesado de señal.

En el capítulo III hablaremos de todos los materiales de los que disponemos en el laboratorio para llevar a cabo las medidas y los programas empleados para simular los resultados.

En el capítulo IV comprobaremos que mediante el análisis de la reflexión de pulsos nos es posible determinar la distancia a la que se encuentra un objeto respecto a una o varias antenas mediante la reflexión de pulsos.

En el capítulo V comprobaremos que mediante el análisis de la transmisión de pulsos nos es posible determinar la distancia a la que se encuentran dos antenas enfrentadas.

En el capítulo VI aplicaremos los desarrollos de los capítulos anteriores para simular un caso real en un sistema de imagen médica por microondas (MICROBIO).

En la bibliografía quedan detalladas todas las fuentes empleadas para la realización y documentación del trabajo.

En los anexos se proporciona y explica el código que se ha creado y empleado para la obtención de los resultados que obtenemos en cada apartado del proyecto.

2. CAPÍTULO II: FUNDAMENTOS TEÓRICOS

Una vez comentada la temática del presente trabajo fin de grado, los principales métodos de los que disponemos en la actualidad y los objetivos que buscamos lograr, pasaremos a hablar de los fundamentos teóricos, es decir en los conceptos más importantes en los que nos basaremos para llevar a cabo nuestro proyecto.

2.1. MICROONDAS

En primer lugar hablaremos de las microondas, estas se corresponden al rango de frecuencias del espectro electromagnético comprendido entre 300 MHz y 300 GHz.

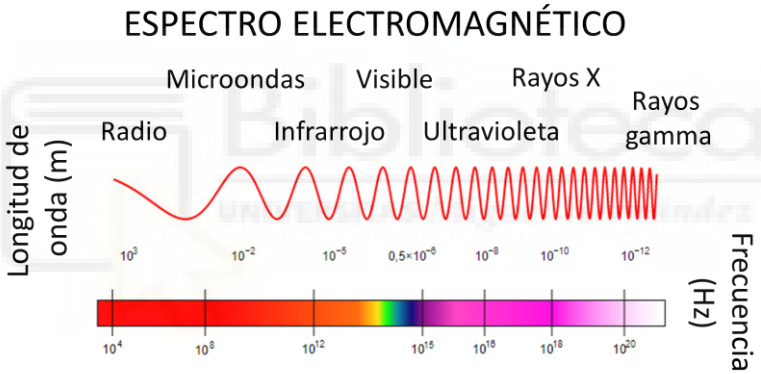


Figura 2.1: Espectro electromagnético

Mediante la imagen anterior [5] podemos determinar que las microondas se encuentran englobadas en las bandas de radiofrecuencia (porción del espectro electromagnético comprendida entre 3 kHz y 300 GHz), específicamente en las de UHF. Como se tratan de ondas a muy alta frecuencia constan de unas longitudes de onda pequeñas, comprendidas en el rango de 1 m a 1 mm, lo que las convierte en opciones muy adecuadas para aplicaciones de radares, telefonía, radiodifusión e incluso médicas. Los motivos por los que emplearemos las microondas en nuestro proyecto se describen en la introducción, nos aportan una gran facilidad a la hora de penetrar tejidos biológicos y también desarrollar sistemas menos invasivos y perjudiciales para el paciente.

2.2. SISTEMA RADAR

Los sistemas RADAR, del inglés "RAdio Detecting And Ranging", son sistemas que emiten ondas electromagnéticas y captan los ecos generados por las reflexiones de estas ondas con el objeto de interés. Mediante este método calculan el tiempo que tardan las ondas en volver y al conocer la velocidad a la que viajan estas ondas son capaces de determinar también la posición del objetivo y viceversa.

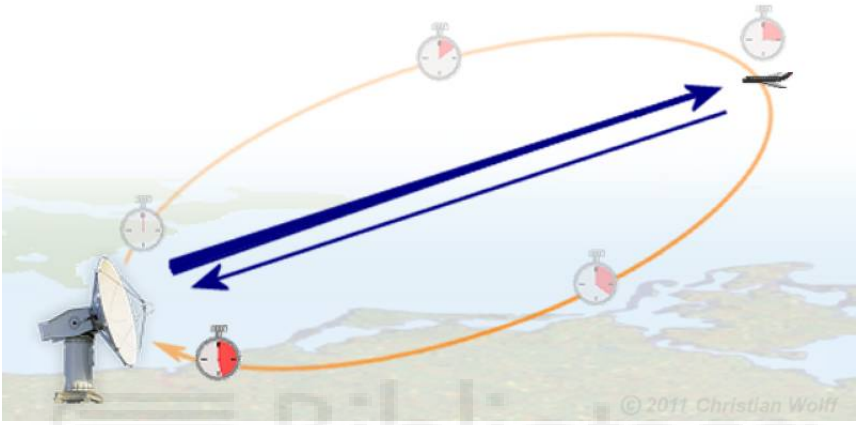


Figura 2.2: Funcionamiento sistema RADAR

2.3. PARÁMETROS S

Los parámetros S también llamados de dispersión o Scattering relacionan ondas incidentes y reflejadas en los puertos del dispositivo [6]. En el caso de disponer de dos puertos son los siguientes:

- S11: refiere a la señal de reflexión en el puerto 1 en comparación con la señal incidente en el puerto 1.
- S12: se refiere a la señal de transmisión en el puerto 1 en relación con la señal incidente en el puerto 2.
- S21: se refiere a la señal de transmisión en el puerto 2 en relación con la señal incidente en el puerto 1.
- S22: se refiere a la señal de reflexión en el puerto 2 en relación con la señal incidente en el puerto 2.

Siendo sus expresiones matemáticas las siguientes:

$$S_{11} = \left.\frac{b_1}{a_1}\right|_{a_2=0} \qquad S_{12} = \left.\frac{b_1}{a_2}\right|_{a_1=0} \qquad S_{21} = \left.\frac{b_2}{a_1}\right|_{a_2=0} \qquad S_{22} = \left.\frac{b_2}{a_2}\right|_{a_1=0}$$

Figura 2.3: Parámetros S

Donde "a"son las señales incidentes y "b"las reflejadas.

Aunque la explicación se hace para el caso en que se dispone de dos puertos, estos pueden ser adaptados para n número de puertos.

2.4. PROCESADO DE SEÑAL

Las señales son mágnitudes físicas (voltaje, corriente, presión sonora, etc) las cuales dependen de una o más variables independientes (espacio, tiempo, etc) [7]. Hay distintos tipos de señales:

- Discretas: señales las cuales están definidas solo en instantes discretos de tiempo, es decir, cuya variable o variables independientes sólo pueden tomar un conjunto de valores finito y por lo tanto el valor de la señal sólo está definido para ese conjunto de valores.
- Continuas: señales definidas en un dominio continuo, es decir, cuya variable o variables independientes pueden tomar cualquier valor real y por tanto la señal está definida para sucesiones continuas de la variable independiente.
- Determinísticas: son aquellas que se pueden describir mediante una función matemática exacta.
- Aleatorias: son aquellas que contienen componentes impredecibles y se describen estadísticamente.

El procesado de señal es la disciplina encargada de estudiar y desarrollar técnicas de tratamiento, análisis y clasificación de señales de naturaleza diversa (señales de

audio, biológicas, de comunicación), todo esto con el fin de mostrar los datos que medimos de tal manera que nos sea posible ver cosas que no podemos apreciar directamente.Para los ingenieros de microondas que trabajan en circuitos distribuidos, componentes separados por cables, líneas de transmisión o guías de ondas, la respuesta en el dominio del tiempo (TD) proporciona información sobre los atributos del circuito, así como métodos para mejorar los resultados de medición, eliminado así las imperfecciones causadas por los accesorios y equipos de prueba que están separados en el tiempo del dispositivo a medir. Si bien una red se caracteriza matemáticamente por su función de transferencia, la respuesta en frecuencia de una red proporciona la respuesta físicamente medible de una red, utilizando señales sinusoidales como estímulo y medición de la respuesta como cambios de magnitud y fase en las señales de estímulo. El análisis de Fourier es ideal para representar la respuesta física y puede proporcionar un análisis útil de una red. Sin embargo, los sistemas de medición son limitados pues solo se puede medir un número de puntos de frecuencia finito en anchos de banda específicos, por lo que cualquier interpretación de las mediciones debe estar incluida en estas limitaciones [8].

Debido a estas propiedades, el procesado de señal abarca un amplio grupo de disciplinas, como podrían ser la medicina, telecomunicaciones, exploración espacial, procesamiento y compresión multimedia, entre otras. En vistas a futuro, vemos que el rol que cumple el procesado de señal se vuelve cada vez más importante, pues como hemos mencionado hay un creciente número de aplicaciones y sistemas que requieren de algoritmos muy sofisticados, requiriendo una constante evolución y expansión de esta rama de conocimiento [9].

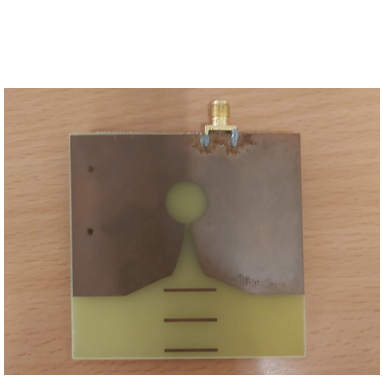
3. CAPÍTULO III: MATERIALES Y MÉTODOS

En este capítulo se describirán todos los materiales, equipos y programas que emplearemos a lo largo del proyecto, realizando una descripción de los mismos y explicando su función dentro del proyecto:

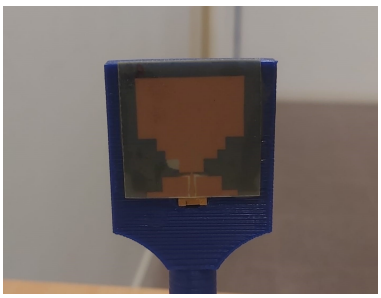
3.1. ANTENAS

Son dispositivos de un puerto capaces de convertir energía guiada en energía radiada, conforman la principal interfaz aérea en cualquier sistema de comunicaciones al ser las encargadas de transmitir y recibir señales. Como ya hemos mencionado en el apartado 2.3 mediante los parámetros S podemos describir como se comportan las señales reflejadas y transmitidas en los diferentes puertos de un sistema, en nuestro caso concreto mediante el empleo de alguno de estos parámetros como son el S11 o el S21 podremos establecer relaciones que nos permitirán determinar la distancia a la que se encuentran ciertos elementos respecto a nuestras antenas.

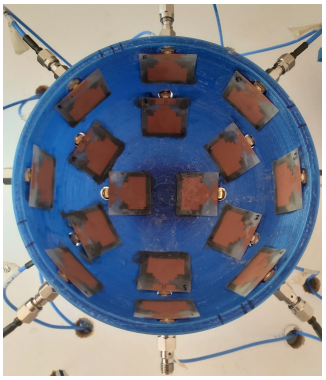
Los modelos de antena que emplearemos a lo largo del trabajo son los siguientes: antena Vivaldi y antenas monopolo de banda ancha.



(a) Antena Vivaldi



(b) Antena monopolo



(c) Antenas monopolo

Figura 3.1: Antenas empleadas

3.2. VECTOR NETWORK ANALYZER (VNA)

Un Analizador Vectorial de Redes, o VNA, es un instrumento de medición que se utiliza para caracterizar dispositivos pasivos como cables, atenuadores y filtros para dispositivos activos como amplificadores y convertidores. En nuestro caso concreto nos será de gran utilidad pues lo emplearemos para medir los parámetros S o de dispersión de las distintas antenas que usaremos a lo largo del proyecto, y exportar estos datos medidos como archivos s1p o s2p (dependiendo del número de puertos), para que posteriormente los analicemos empleando nuestro código de Python.

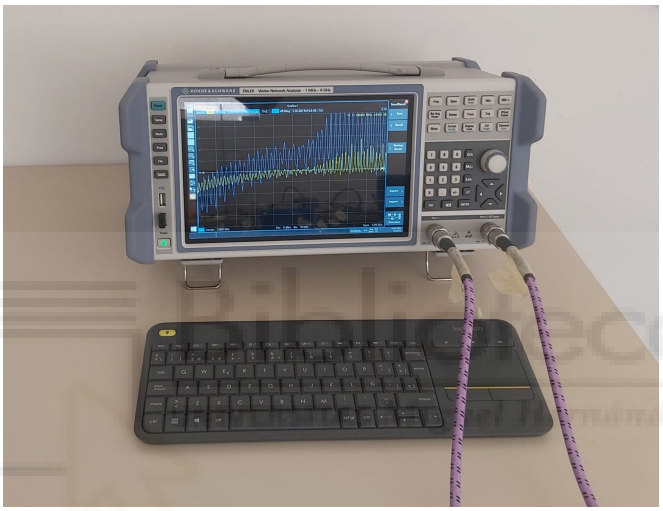


Figura 3.2: VNA

El VNA es quizás el instrumento electrónico más preciso utilizado en mediciones de RF y microondas. Los VNA modernos pueden medir señales de alta y baja potencia con mayor precisión que cualquier otro sensor de potencia, y pueden medir la ganancia en frecuencia de un dispositivo electrónico con un rendimiento rastreable a mediciones de dimensiones físicas. Es por eso que previamente a realizar cualquier medición, debemos asegurarnos de calibrar adecuadamente nuestro sistema, para ello haremos uso de un kit de calibración ZV-Z135 de la marca Rohde & Schwarz como el que se puede apreciar a continuación.

(a) Modelo kit de calibración (b) Calibrador

Figura 3.3: Kit de calibración VNA

Mediante el empleo de este kit de calibración conseguiremos unas medidas mucho más precisas y exactas al eliminar errores que no son implícitos de nuestras medidas como podrían ser la desadaptación de impedancias o las pérdidas producidas en cables y conectores. Constan de varios estándares como son:

- Open: conector en abierto.
- Short: conector en cortocircuito.
- Load: resistencia de 50 Ω(generalmente).
- Through: nos permite una conexión directa entre los puertos del VNA.

Usando la función de Carta de Smith del VNA podremos comprobar que se cumplen estos estándares. Una vez se termine la calibración la podremos guardar y emplear cuando queramos, aunque para evitar posibles errores llevaremos a cabo varias calibraciones a lo largo del proyecto.

3.3. CABLE COAXIAL

Este tipo de cable es el más idóneo para la transmisión de señales de frecuencia elevada o Radio Frecuencia (RF). Las partes que lo componen son las siguientes:

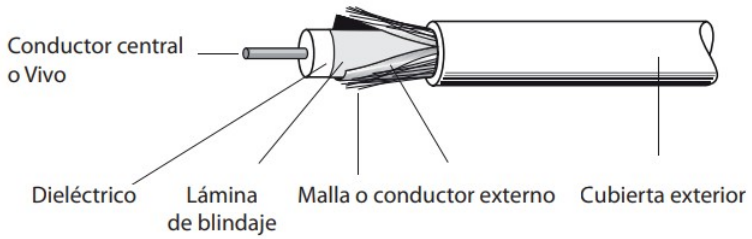


Figura 3.4: Cable coaxial

- Vivo o conductor central: es el encargado de transmitir la señal transmitida, está compuesto por un único hilo o varios trenzados, de cobre, cobre estañado, cobre plateado, aluminio cobreado o acero cobrado.
- Dieléctrico: material de una elevada resistividad cuya función es aislar el vivo del blindaje.
- Lámina de blindaje: cubierta de cobre o aluminio que junto a la malla conforma el apantallamiento del cable coaxial.
- Malla o conductor externo: trenzado realizado con hilos finos (husos) de cobre , cobre estañado, cobre plateado, aluminio cobreado o acero cobreado. Al estar conectada a masa absorbe el ruido electromagnético externo impidiendo que alcance al vivo. A mayor trenzado de la malla más calidad tendrá el cable.
- Cubierta: aislante que protege al cable de agentes externos. Entre la cubierta y la malla, algunos cables disponen un lámina antimigratoria que tiene por objeto evitar que los aditivos de la cubierta y la humedad migren al interior del cable, evitando así el deterioro de sus características.

  Sus parámetros más importantes son:

  - ϵr: permitividad relativa del sustrato.
  - Di: diámetro interno del cable.

- Do: diámetro externo del cable.
- L: longitud del cable.
- Rho: resistividad del metal en relación con el cobre. Se emplea para los cálculos de pérdidas.
- Tanδ: tangente de pérdidas del dieléctrico. Se emplea para los cálculos de pérdidas.

Tanto la imagen como la información están sacadas de [10].

3.4. LÍNEAS MICROSTRIP

Una línea de transmisión microstrip consiste en una fina tira conductora colocada en un lado de un sustrato dieléctrico, que tiene en el otro lado del dieléctrico un plano de tierra. [11]

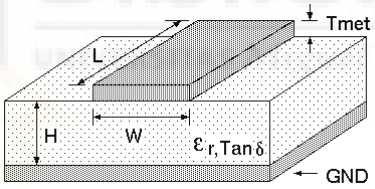


Figura 3.5: Línea microstrip

- ϵo: permitividad en el vacío.
- ϵr: permitividad relativa del sustrato.
- ϵeff: se calcula como la raíz cuadrada de ϵr.
- H: espesor dieléctrico del sustrato.
- L: longitud de la línea de transmisión.
- W: ancho de la línea de transmisión.
- Tmet: espesor del metal. Influye tanto en Z0 como en las pérdidas de la línea.

- Rho: resistividad del metal en relación con el cobre. Empleada para cálculos de pérdidas.
- Rough: rugosidad de la superficie metálica. Es la desviación rms de la superficie del conductor respecto de un plano. Una superficie rugosa aumenta las pérdidas en la línea.
- Tanδ: tangente de pérdidas del dieléctrico. Empleada para cálculos de pérdidas.

Tanto la imagen como la información están sacadas de [12].

3.5. CABLES DE RF

Elementos conductores empleados para transmitir señales de RF de un dispositivo a otro, minimizando las pérdidas de señal, interferencias y reflexiones, garantizando así una transmisión de alta calidad. Disponemos de distintos tipos:

- Cable coaxial.
- Par trenzado (blindado o sin blindaje).
- Fibra óptica.

Mediante su empleo podremos conectar el VNA a las antenas empleadas para realizar las medidas. Deberemos tener en cuenta que a la hora de realizar la calibración del sistema, esta deberá ser realizada al final de los mismos para evitar añadir distancias que no se corresponden a nuestras medidas. Nosotros en el laboratorio dispones de cables coaxiales del modelo FL086-24SM+ de la empresa Mini-Circuits.

Figura 3.6: Cable coaxial de RF

3.6. CONECTORES DE RF

Elemento necesario para poder conectar los cables de RF con nuestras antenas u otros dispositivos. Asegurando una transmisión eficiente y mínima pérdida de señal en el rango de RF. Están específicamente optimizados para manejar señales de alta frecuencia sin degradar su integridad. Hay distintos tipos, como son:

- Conector BNC (Bayonet Neill-Concelman).
- Conectore SMA (SubMiniature version A).
- Conector N.
- Conector TNC (Threaded Neill-Concelman).
- Conector UHF.

En nuestro caso emplearemos los conectores SMA que son de los que disponemos en el laboratorio:

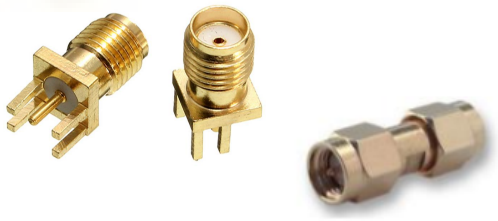


Figura 3.7: Conectores de RF

Una vez comentados todos los elementos y componentes que usaremos a lo largo del proyecto, realizaremos una descripción de los diferentes softwares que se usarán y los motivos de elección.

3.7. ADVANCED DESIGN SYSTEM (ADS)

Software de Keysight Technologies que nos permite realizar simulaciones circuitales en el rango de las microondas y por tanto estimar las propiedades de los elementos de interés. En nuestro caso concreto se usará para realizar las simulaciones de los casos ideales de reflexión y transmisión de señales entre antenas y exportar estos resultados en archivos de formato .s1p y .s2p para que puedan ser leidos por nuestro código.

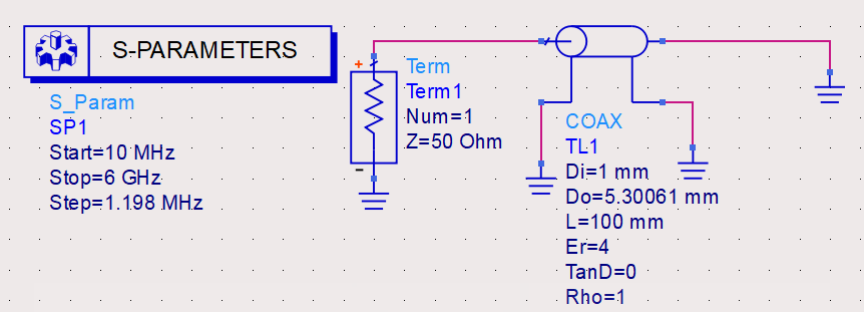


Figura 3.8: Advanced Desing System

3.8. VISUAL STUDIO CODE

Editor de código fuente desarrollado por Microsoft que nos permite editar, depurar y compilar código, en nuestro caso lo usaremos como entorno para el lenguaje de programación que emplearemos , el cual será Python.

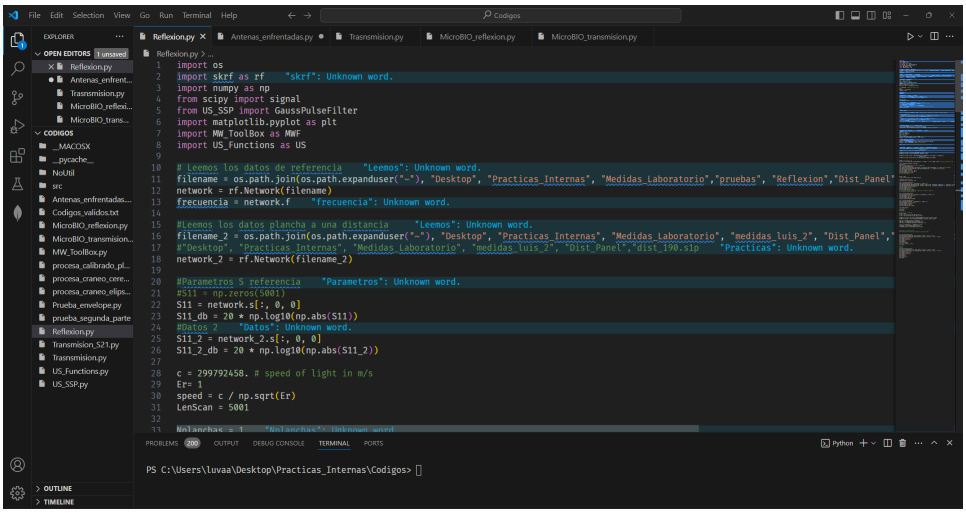


Figura 3.9: Visual Studio Code

3.9. PYTHON

Emplearemos el lenguaje de programación Python, ya que dispone de algunas librerías que nos facilitarán mucho las tareas que llevaremos a cabo a lo largo del proyecto, algunas de estas librerías son: skrf, numpy, matplotlib y scipy, cuyas funciones quedan descritas en los ANEXOS al final del trabajo.

Figura 3.10: Python

3.10. EXCEL

Excel es un programa que permite editar hojas de cálculo realizando cálculos, gráficas y tablas desarrollado por Microsoft. Nosotros lo emplearemos para recoger los datos de las medidas y comparar los datos que deseamos obtener con los que obtenemos mediante las medidas.

Figura 3.11: Excel

En cuanto a los métodos podemos decir: Se fueron realizando distintos códigos en Python para medir los parámetros S de una antena en distintas situaciones y comprobar que se podía medir la distancia en cada uno de los casos.

3.11. ANÁLISIS DE PULSOS

Los pulsos gaussianos se emplean comúnmente en aplicaciones de comunicación, óptica y acústica debido a su excelente localización de tiempo y frecuencia, característica que los hace ideales para detectar señales que pueden estar ocultas por el ruido [13]. Con el fin de trabajar de la manera más adecuada generaremos distintos pulsos Gaussianos y analizaremos cuál de todas las posibilidades se adapta de una mejor manera a nuestro caso concreto, por otra parte realizaremos las derivadas de alguno de ellos para comprobar si mediante el empleo de alguna de estas obtenemos mejores resultados que los pulsos en si. Con el fin de estudiar la viabilidad de las derivadas del pulso Gaussiano haremos uso de los polinomios de Hermite, los cuales son una secuencia de polinomios ortogonales que surgen frecuentemente en el análisis matemático y nos permiten obtener una expresión analítica de cada una de las derivadas. La ecuación que emplearemos para conseguir cada una de las derivadas del pulso en el dominio del tiempo es la siguiente:

$$T_n = (-1)^n * \frac{1}{(\sqrt{2} * \sigma)^2} * H_n(\frac{t}{\sqrt{2} * \sigma}) * G(t) \tag{3.1}$$

Donde n es el orden de la derivada (en nuestro caso particular estará comprendido entre 1 y 7), σ se obtiene a partir de la siguiente expresión:

$$\sigma = \sqrt{\frac{\tau^2}{2}} \tag{3.2}$$

G(t) es nuestro pulso Gaussiano y Hn es el polinomio de Hermite del orden de la derivada, cuya expresión podemos obtener en la en la imagen que se muestra a continuación:

8.95 The Hermite polynomials $H_n(x)$

8.950 Definition

1. $H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n}\left(e^{-x^2}\right)$ SM 567(14)

   or

2. $H_n(x) = 2^n x^n - 2^{n-1}\binom{n}{2}x^{n-2} + 2^{n-2}\cdot 1\cdot 3\cdot\binom{n}{4}x^{n-4} - 2^{n-3}\cdot 1\cdot 3\cdot 5\cdot\binom{n}{6}x^{n-6} + \ldots$ MO 105a

3.[10] $H_0(x) = 1$

4.[10] $H_1(x) = 2x$

5.[10] $H_2(x) = 4x^2 - 2$

6.[10] $H_3(x) = 8x^3 - 12x$

7.[10] $H_4(x) = 16x^4 - 48x^2 + 12$

8.[10] $H_5(x) = 32x^5 - 160x^3 + 120x$

9.[10] $H_6(x) = 64x^6 - 480x^4 + 720x^2 - 120$

10.[10] $H_7(x) = 128x^7 - 1344x^5 + 3360x^3 - 1680x$

11.[10] $H_8(x) = 256x^8 - 3584x^6 + 13440x^4 - 13440x^2 + 1680$

8.951 The integral representation:

$$H_n(x) = \frac{2^n}{\sqrt{\pi}}\int_{-\infty}^{\infty}(x+it)^n e^{-t^2}\,dt \qquad \text{MO 106a}$$

Functional relations

8.952 Recursion formulas:

1. $\frac{d\,H_n(x)}{dx} = 2n\,H_{n-1}(x)$ SM 569(22)

2. $H_{n+1}(x) = 2x\,H_n(x) - 2n\,H_{n-1}(x)$ SM 570(23)

   For the orthogonality, see **7.374** 1 and **8.904**.

3.[10] $n\,H_n(x) = -n\,H'_{n-1}(x) + x\,H'_n(x)$ MS 5.6.2

Figura 3.12: Polinomios de Hermite

Haciendo uso de la ecuación 3.1, la cual nos proporciona la expresión general de las derivadas de pulso Gaussiano, la expresión de los polinomios de Hermite representada en la imagen anterior (3.12) obtenida de [14] y de nuestro código generaremos las siete primeras derivadas de un pulso Gausiano y las representaremos en el dominio del tiempo:

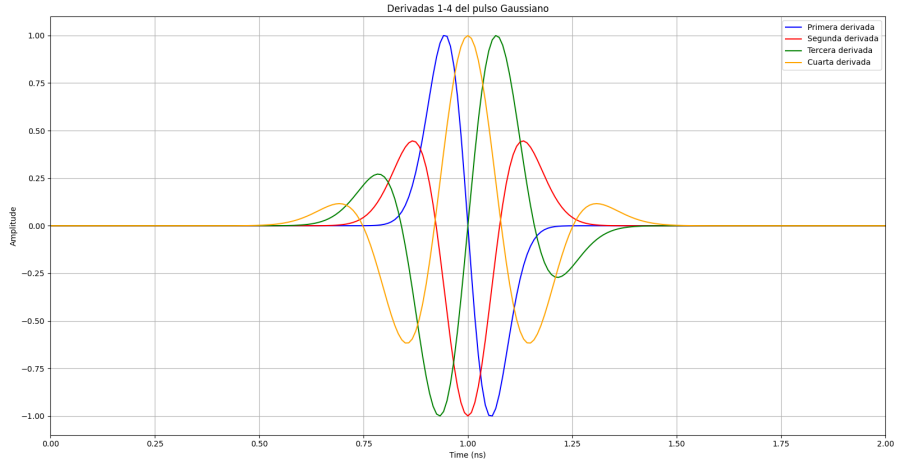


Figura 3.13: Derivadas 1-4 del pulso Gaussiano

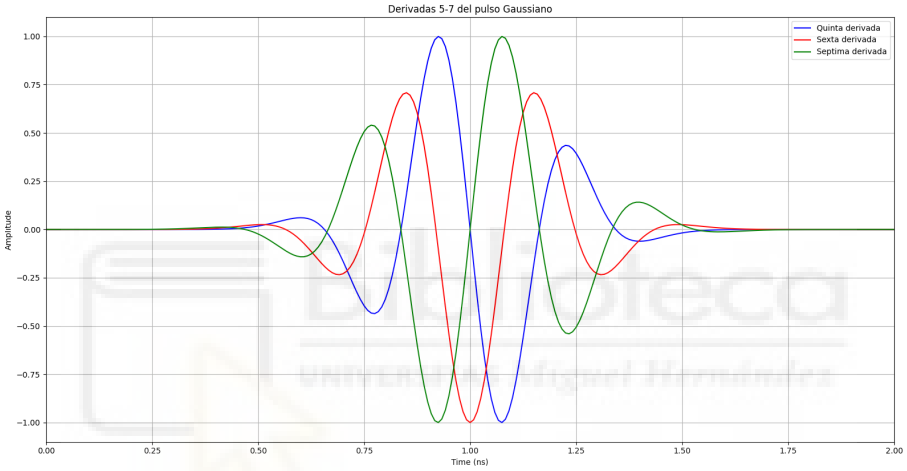


Figura 3.14: Derivadas 5-7 del pulso Gaussiano

Los datos empleados para la generación de cada una de las derivadas con el fin de que en el dominio de la frecuencia estén centradas justo en la mitad de nuestro ancho de banda (3GHz) han sido los siguientes:

Variables	**Primera**	**Segunda**	**Tercera**	**Cuarta**	**Quinta**	**Sexta**	**Séptima**
n	1	2	3	4	5	6	7
Amp $(\times 10^{8})$	50	50	50	50	50	50	50
τ $(\times 10^{-12})$	77.5	107.5	130	152.5	170	185	200

Tabla 3.1: Valores para generación de derivadas

Una vez que hemos generado todos los pulsos de interés los representaremos en una misma gráfica en el dominio de la frecuencia, viendo así de una forma comparativa cuál es el que cubre de una mejor manera nuestro ancho de banda, de esta forma podremos descartar aquellos pulsos y derivadas que no se ajusten completamente a nuestras especificaciones de ancho de banda y seleccionar el más indicado:

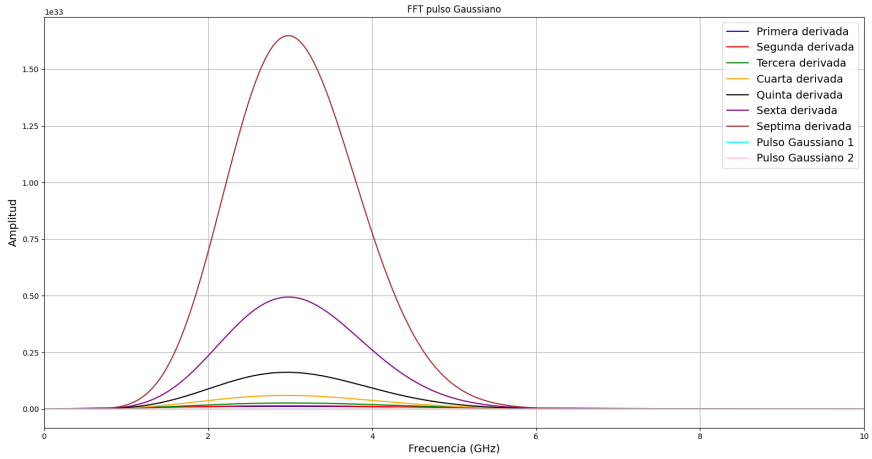


Figura 3.15: Pulsos frente al ancho de banda

Esta gráfica en sí no nos resulta muy útil ya que al no tener todos los pulsos la misma amplitud no se puede apreciar de manera clara cuál de todas las opciones es la que cubre mejor todo el ancho de banda, por lo tanto normalizaremos los pulsos y los volveremos a representar:

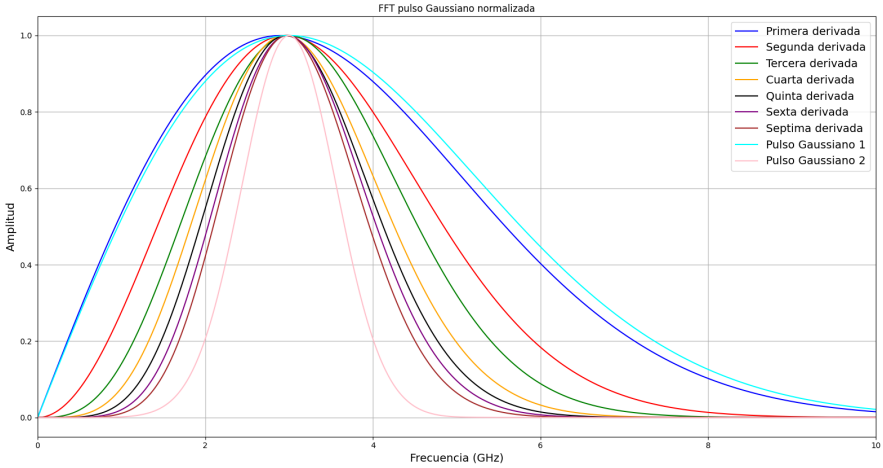


Figura 3.16: Pulsos normalizados frente al ancho de banda

Por lo que podemos apreciar el pulso más indicado es el representado en azul claro y por tanto será el que usaremos por ahora (aunque también usaremos otros en apartado concretos al adecuarse de una mejor manera a otro tipo de situaciones).

4. CAPÍTULO IV: ANÁLISIS DE PULSOS EN REFLEXIÓN

4.1. INTRODUCCIÓN

En este capítulo se pretende demostrar que es posible determinar la posición a la que se encuentra una plancha metálica situada a una distancia determinada de una o varias antenas mediante la reflexión de pulsos, para ello, como se ha mencionado anteriormente se analizarán los parámetros S obtenidos, más concretamente haremos énfasis en los parámetros S11 y S22, ya que estos nos aportaran información relativa a la señal que es reflejada de vuelta a las antenas por la plancha metálica. Mediante esta información nos será posible determinar el tiempo que tarda la onda en llegar y reflejarse en la plancha y por tanto, la distancia a la que se encuentra de cada antena.

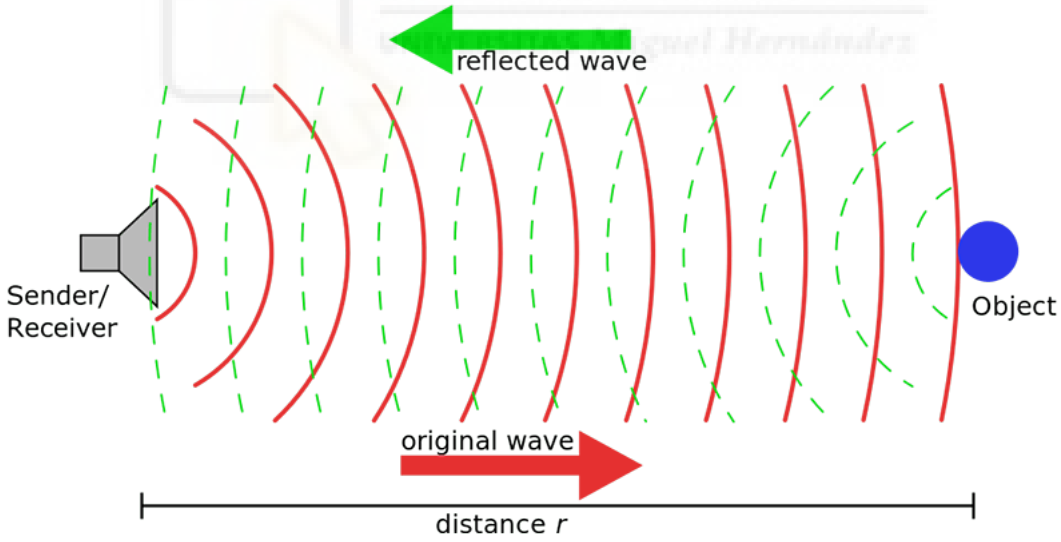


Figura 4.1: Ilustración reflexión de pulsos

Con fines aclaratorios se ha diseñado un diagrama de flujo en el cual se describe el proceso seguido para la creación del código de python en este apartado concreto del trabajo y brevemente explicaremos los aspectos más relevantes respectivos al mismo.

En primer lugar, se obtienen los datos frecuenciales de los parámetros S11 de los ficheros s1p. Para ello hacemos uso de la función network incluida en la librería skrf, más concretamente de sus funciones "network.f" con la que podemos acceder a las frecuencias a las que se han medido o simulado los parámetros S y "network.s" para acceder a los parámetros S en si, indicando entre corchetes el que nos interesa, siendo el S11 [:,0,0], el S12[:,0,1], el S21[:,1,0] y el S22[:,1,1]:

```
#Obtenemos datos referencia
network = rf.Network(filename)
frecuencia = network.f

#Obtenemos datos a una distancia
network_2 = rf.Network(filename_2)

#Donde filename y filename_2 sería la ruta a los archivos
   ↪ s1p en cuestión

#Leemos datos S11 referencia
#S11 = np.zeros(5001)
S11 = network.s[:, 0, 0]
S11_db = 20 * np.log10(np.abs(S11))
#Leemos datos S11 a una distancia
S11_2 = network_2.s[:, 0, 0]
S11_2_db = 20 * np.log10(np.abs(S11_2))

```

Código 4.1: Obtención de parámetros S11

Nuestro eje de frecuencias está limitado entre 10 MHz y 6 GHz, nosotros generaremos un nuevo eje comprendido entre 0 y 120 GHz:

```
Fs = 120e9
DF = frecuencia[1]-frecuencia[0]
new_frequency = np.linspace(0, Fs, len(frecuencia))
F_axis = np.arange(0, Fs, DF)
nfft = len(F_axis)

```

Código 4.2: Nuevo eje de frecuencias

Seguiremos con la creación de una ventana Tukey, la cual suavizará los lóbulos de la señal cerca de sus puntos iniciales y finales, eliminará reflexiones parásitas y reducirá la fuga espectral en el dominio de la frecuencia al realizar transformadas de Fourier, mejorando así la precisión de nuestro sistema y por otra parte, generaremos los espectros rellenos con ceros de simetría hermitiana de las señales:

```
MyWin = US.makeWindow(SortofWin='tukey', WinLen=LenScan,
   ↪ param1=0.25, param2=1, Span=0, Delay=0)

if np.mod(nfft,2)==1:
MyWin_ext = np.concatenate(( MyWin, np.zeros((int(
   ↪ (nfft+1)/2 )-LenScan)) ))
MyWin = np.concatenate((MyWin_ext,
   ↪ np.flipud(MyWin_ext[:-1])))
else:
MyWin_ext = np.concatenate(( MyWin,
   ↪ np.zeros((int(nfft/2)-LenScan)) ))
MyWin = np.concatenate((MyWin_ext, np.flipud(MyWin_ext)))

S_ref = MWF.makeSpectrum(S11_ref, F_axis, frecuencia)
S_DataMatrix = MWF.makeSpectrum(np.squeeze(S11_plancha),
   ↪ F_axis, frecuencia)

```

Código 4.3: Creación ventana Tukey y espectros de las señales

Realizaremos un filtrado de las señales utilizando una forma de onda de pulso con el fin de evitar señales parásitas y elementos externos a nuestro sistema , aplicaremos la ventana Tukey creada en el apartado anterior y finalmente podremos realizar la transformada inversa de nuestras señales, obteniendo así nuestros datos en el dominio del tiempo:

```
ref_in_time, borrar = MWF.adaptedFilter(S_ref, Pulse,
   ↪ XCR=True, Aligned=True,
Wiener=False)
winref_in_time = np.real(np.fft.ifft(S_ref * MyWin))

DataMatrix_in_time = np.zeros((Nplanchas, Nantenas, nfft))
WinMatrix_in_time = np.zeros((Nplanchas, Nantenas, nfft))

DataMatrix_in_time, borrar = MWF.adaptedFilter(S_DataMatrix,
   ↪ Pulse,XCR=True, Aligned=True,
Wiener=False)
WinMatrix_in_time=
   ↪ np.real(np.fft.ifft(np.squeeze(S_DataMatrix) * MyWin))

```

Código 4.4: Filtrado, aplicación ventana Tukey e IFFT de las señales

Una vez diseñado el código demostraremos su funcionamiento, en primer lugar enfrentándolo a casos ideales como son sistemas sencillos simulados en el programa ADS y posteriormente comprobaremos si sigue siendo válido frente a situaciones reales.

4.2. CASOS IDEALES

Como se ha comentado anteriormente, previamente a realizar las correspondientes medidas en el laboratorio llevaremos a cabo una serie de casos de prueba con el fin de demostrar que nuestro código funciona correctamente. Para crear estos casos haremos uso del programa de simulación ADS y de su herramienta "linecalc", la cual se muestra a continuación:

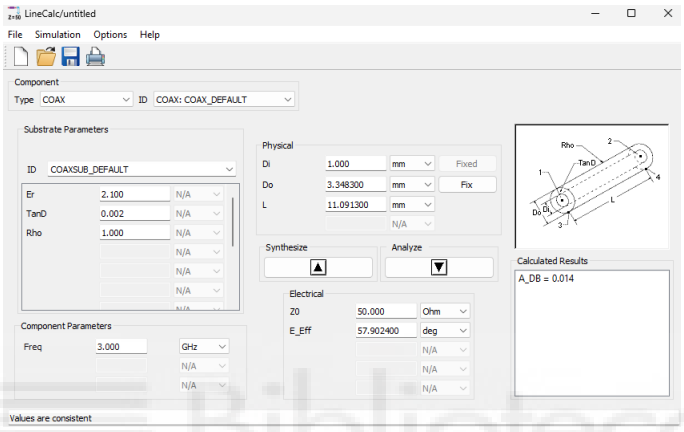
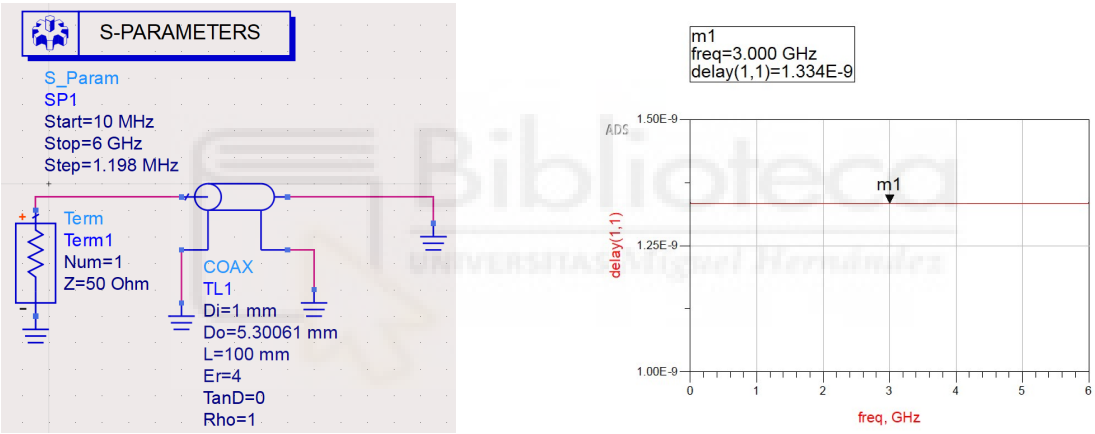


Figura 4.2: Herramienta Linecalc

Gracias a esta herramienta podremos diseñar sistemas sencillos, ya que fijando valores como son la ϵr del medio, la tangente de pérdidas, la frecuencia (en nuestro caso la central de nuestro sistema 3GHz) y la impedancia carácterística obtenemos las dimensiones respectivas de elementos que nos pueden servir de prueba como son cables coaxiales o líenas microstrip. Empleando estos dos elementos hemos decidido analizar el comportamiento de nuestro código frente a las siguientes situaciones: cable coaxial acabado en abierto, cable coaxial acabado en corto, línea microstrip acabada en abierto y línea microstrip acabada en corto. La comprobación que llevaremos a cabo es simple, no buscaremos determinar una distancia sino que ésta tendrá un valor fijo y lo que buscaremos es comprobar que podemos determinar distintos valores de ϵr. Haciendo nuevamente uso de la herramienta "linecalc" mantendremos todos los campos con unos valores constantes y variaremos únicamente el campo "Er", al sintetizar los datos obtendremos las dimensiones del diámetro interno (Di) y el diámetro externo (Do) en el caso del coaxial o el ancho (W) en el caso de la línea microstrip, que junto con la longitud de fijemos serán los datos que necesi-

taremos para crear cada caso. Una vez abramos la ventana de simulación de ADS podremos representar varios de interés, pero el dato que realmente necesitamos es el retardo de grupo, este valor nos dará la información relativa al tiempo que tarda la señal en atravesar el medio en el que se encuentra y ser reflejada. Exportaremos los datos generados como archivos s1p y los analizaremos empleando el código que se ha diseñado para este apartado del proyecto. La distancia que fijaremos para estos casos de ejemplo será de 100 mm e iremos variando el valor de la ϵr (y por tanto los valores de Di y Do o W) en un rango de valores entre 1 y 10, obteniendo el retardo de grupo para cada caso. Los resultados obtenidos han sido los siguientes:

4.2.1. CABLE COAXIAL ACABADO EN CORTO



(a) Esquema ADS (b) Retardo de grupo

Figura 4.3: Ejemplo cable coaxial acabado en corto en ADS (ϵr=4)

Haciendo uso del código que hemos diseñado, el cual se ha descrito anteriormente y está completamente desarrollado en el ANEXO 2, obtenemos el siguiente resultado:

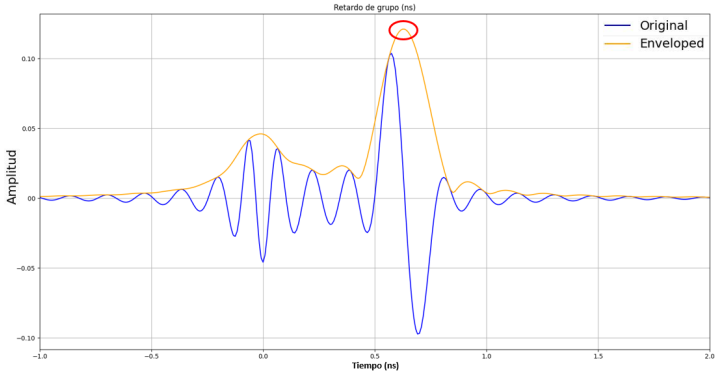


Figura 4.4: δt obtenida en Python para ϵr=1

Observando la posición donde se encuentra el máximo (señalado en rojo) obtenemos el valor de δt o retardo de grupo. Aplicando este mismo método para el resto de casos obtendremos unos valores de δt esperados (obtenidos mediante ADS) y medidos (obtenidos mediante código de Python), los cuales recogeremos en una tabla con el fin de ver el error que se está cometiendo y que posteriormente usaremos para comprobar la viabilidad del método que estamos empleado:

ϵr	**δt esperado (ns)**	**δt obtenido (ns)**	**Error (%)**
1	0.6671	0.638	4.3622
2.1	0.9668	0.94	2.772
4	1.334	1.31	1.7991
5	1.492	1.468	1.6086
10	2.11	2.085	1.1848

Tabla 4.1: Retardos de grupo cable coaxial acabado en corto en reflexión

Como se puede comprobar estamos midiendo el tiempo de forma correcta pues el error que se comete es pequeño y por tanto, podemos proceder al cálculo de ϵr.

En primer lugar calculamos el tiempo que tarda en llegar la señal:

$$t = \frac{\delta t}{2} = 0{,}319ns \tag{4.1}$$

El valor que obtenemos de δt incluye tanto el tiempo de ida de la señal desde el emisor hasta el objeto reflector como el tiempo de vuelta desde el objeto reflector hasta el receptor y por tanto, como a nosotros solo nos interesa el tiempo hasta que la señal se refleja, únicamente debemos tomar en consideración el correspondiente a la ida, que se corresponde con la mitad del tiempo total, es decir, δt/2. A partir de este tiempo y fijando una distancia, por ejemplo 100 mm, podremos hallar la velocidad de la señal en función del medio (según el valor de ϵr).

$$v = \frac{d}{t} = \frac{100 \times 10^{-3}}{0{,}319 \times 10^{-9}} = 313{,}4796238 * 10^{6} m/s \tag{4.2}$$

La relación entre la permitividad del medio ϵ y la velocidad de la onda (v) se deriva de la ecuación de onda para un medio material:

$$v = \frac{1}{\sqrt{\epsilon * \mu}} \tag{4.3}$$

Sabiendo que $\epsilon = \epsilon$r * ϵo, que $\mu = \mu$o en un medio no magnético y que

$$c = \frac{1}{\sqrt{\epsilon o * \mu o}} \tag{4.4}$$

La fórmula específica para la velocidad en un medio que obtenemos es la siguiente:

$$v = \frac{c}{\sqrt{\epsilon r}} \tag{4.5}$$

Despejando ϵr y tomando la velocidad de la luz como $c = 3 \times 10^8$ m/s, obtenemos:

$$\epsilon r = (\frac{c}{v})^2 = (\frac{3 * 10^8}{313{,}4796238 * 10^6})^2 = 0{,}9518 \tag{4.6}$$

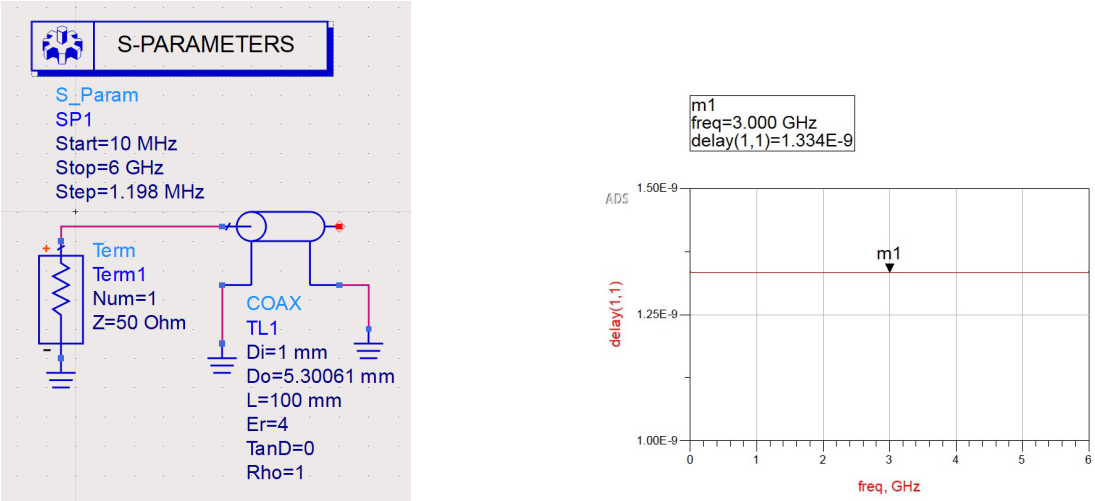
Aplicando el mismo procedimiento para el resto de valores obtenidos:

ϵr esperado	**ϵr obtenido**	**Error (%)**
1	0.9158	8.42
2.1	1.9881	5.329
4	3.8612	3.47
5	4.8488	3.024
10	9.7813	2.187

Tabla 4.2: Valores de ϵr para cable coaxial acabado en corto en reflexión

Vemos que los resultados que obtenemos son acordes a lo que se espera, pudiéndose deber el error que obtenemos a inexactitudes a la hora de ajustar el código o factores que no podemos controlar. El procedimiento que seguiremos será el mismo para el resto de casos, por tanto no volveremos a realizar todos los cálculos pero sí que indicaremos los valores de δt obtenidos y los cálculos finales que realicemos.

4.2.2. CABLE COAXIAL ACABADO EN ABIERTO



(a) Esquema ADS

(b) Retardo de grupo

Figura 4.5: Ejemplo cable coaxial acabado en abierto

En el caso del cable coaxial acabado en abierto los valores del retardo de grupo son los siguientes:

ϵr	δt esperado (ns)	δt obtenido (ns)	Error (%)
1	0.6671	0.638	4.3622
2.1	0.9668	0.94	2.772
4	1.334	1.31	1.7991
5	1.492	1.468	1.6086
10	2.11	2.085	1.1848

Tabla 4.3: Retardos de grupo cable coaxial acabado en abierto en reflexión

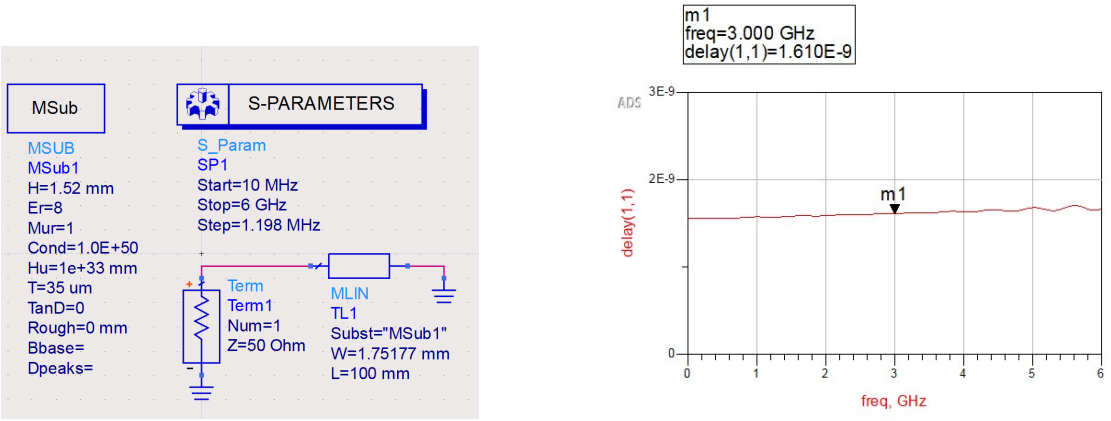
Siguiendo el mismo procedimiento descrito en el caso anterior obtendremos los valores de ϵr:

ϵr esperado	ϵr obtenido	Error (%)
1	0.9158	8.42
2.1	1.9881	5.329
4	3.8612	3.47
5	4.8488	3.024
10	9.7813	2.187

Tabla 4.4: Valores de ϵr para cable coaxial acabado en abierto en reflexión

Una vez analizados los casos relacionados al cable coaxial, pasaremos a analizar los relacionados con la línea microstrip.

4.2.3. LÍNEA MICROSTRIP ACABADA EN CORTO



(a) Esquema ADS (b) Retardo de grupo

Figura 4.6: Ejemplo línea microstrip acabada en corto

En el caso de la línea microstrip acabada en corto los valores del retardo de grupo son los siguientes:

ϵr	δt esperado (ns)	δt obtenido (ns)	Error (%)
1	0.6671	0.638	4.3622
2	0.8857	0.86	2.902
4.4	1.239	1.217	1.7756
8	1.61	1.596	0.8696
10	1.779	1.767	0.6745

Tabla 4.5: Retardos de grupo línea microstrip acabado en corto en reflexión

En el caso de líneas microstrip deberemos calcular el valor de ϵeff en vez de ϵr y comparar este valor con el que obtenemos de la herramienta linecalc de ADS en el apartado "Calculated Results":

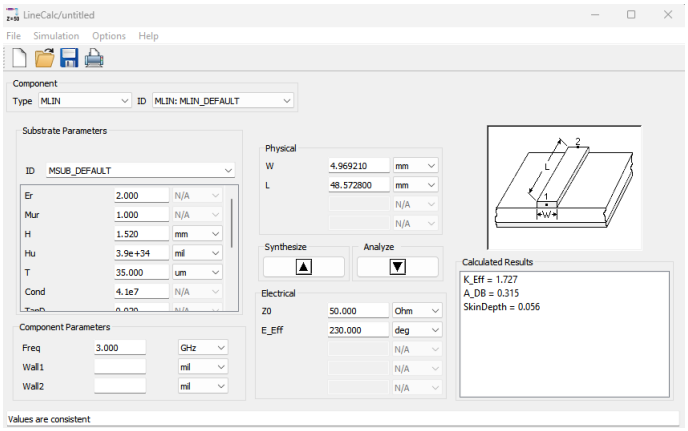


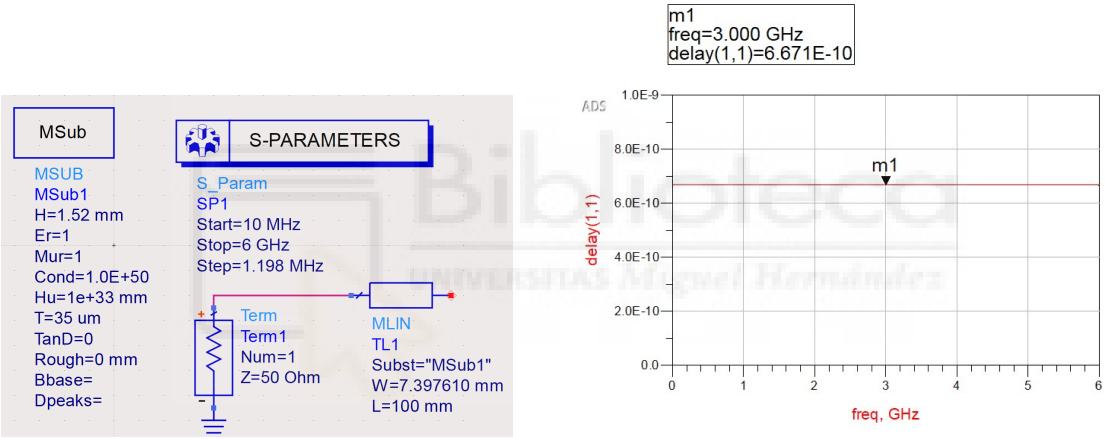
Figura 4.7: Valor de ϵeff obtenida de linecalc

El procedimiento sigue siendo el mismo que en los casos anteriores y por lo tanto obtenemos que:

ϵr	ϵeff esperada	ϵeff obtenida	Error (%)
1	1	0.9158	8.42
2	1.727	1.6641	4.7453
4.4	3.365	3.3324	0.9688
8	5.603	5.7312	2.2881
10	6.711	7.0252	4.6819

Tabla 4.6: Valores de ϵr para línea microstrip acabada en corto en reflexión

4.2.4. LÍNEA MICROSTRIP ACABADA EN ABIERTO



(a) Esquema ADS (b) Retardo de grupo

Figura 4.8: Ejemplo línea microstrip acabada en abierto

Finalmente los valores del retardo de grupo obtenidos para el caso de línea microstrip acabada en abierto son:

ϵr	δt esperado (ns)	δt obtenido (ns)	Error (%)
1	0.6671	0.638	4.3622
2	0.8857	0.86	2.902
4.4	1.239	1.217	1.7756
8	1.61	1.596	0.8696
10	1.779	1.767	0.6745

Tabla 4.7: Retardos de grupo línea microstrip acabada en abierto en reflexión

Con los que obtenemos los siguientes valores de ϵeff:

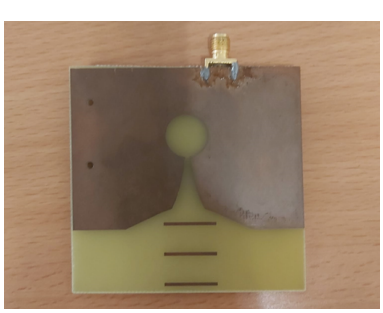
ϵr	**ϵeff esperada**	**ϵeff obtenida**	**Error (%)**
1	1	0.9158	8.42
2	1.727	1.6641	4.7453
4.4	3.365	3.3324	0.9688
8	5.603	5.7312	2.2881
10	6.711	7.0252	4.6819

Tabla 4.8: Valores de ϵr para línea microstrip acabada en abierto en reflexión

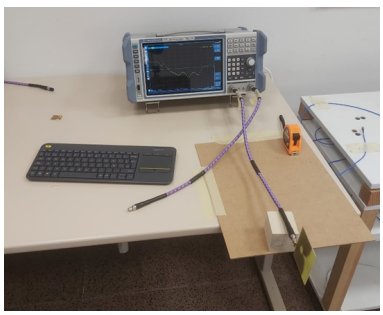
En vista de los resultados obtenidos podemos afirmar que el programa diseñado es capaz de procesar los datos de manera correcta (salvo por un pequeño error) y por tanto, podemos pasar a analizar los casos reales. En estos trataremos de determinar el tiempo que tarda la señal en reflejarse y al conocer el valor de la velocidad de la luz y la ϵr del medio en el que nos encontramos, seremos también capaces de determinar la distancia a la que se encuentra el elemento reflector (plancha metálica) de las antenas. Para demostrarlo analizaremos dos casos: en el primero emplearemos una antena vivladi y una plancha metálica y en el segundo dos antenas de banda ancha y otra plancha metálica.

4.3. CASOS REAL 1: EMPLEANDO ANTENA VIVALDI

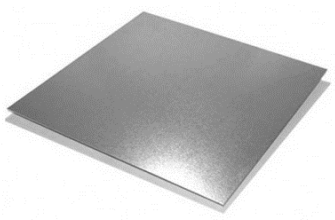
Una vez demostrado que los valores que obtenemos son válidos para los casos ideales, analizaremos la viabilidad de nuestro método cuando nos enfrentamos a casos reales. Para ello haremos uso del código que se encuentra descrito en el ANEXO 2 (resumido brevemente al principio de este capítulo) y de los materiales de los que disponemos en el laboratorio. El equipo que emplearemos para la realización de esta prueba se muestra a continuación:



(a) Antena Vivaldi



(b) VNA



(c) Plancha metálicas

Figura 4.9: Materiales empleados

4.3.1. TOMA DE MEDIDAS

Como ya hemos mencionado anteriormente el método que empleamos se basa en medir el tiempo de vuelo de una señal reflejada y conociendo la velocidad a la que se propaga la onda por el medio calcular la distancia correspondiente a ese tiempo. Este principio o método es el mismo que se emplea en aplicaciones de tipo RADAR (con las respectivas diferencias de cada sistema). Una vez comentado la finalidad de estos casos, pasaremos a describir el proceso que seguiremos a la hora de realizar la toma de medidas de pulsos en reflexión, para ayudar a la explicación haremos uso de la siguiente imagen donde podemos apreciar como se realizó una de las medidas en el laboratorio.

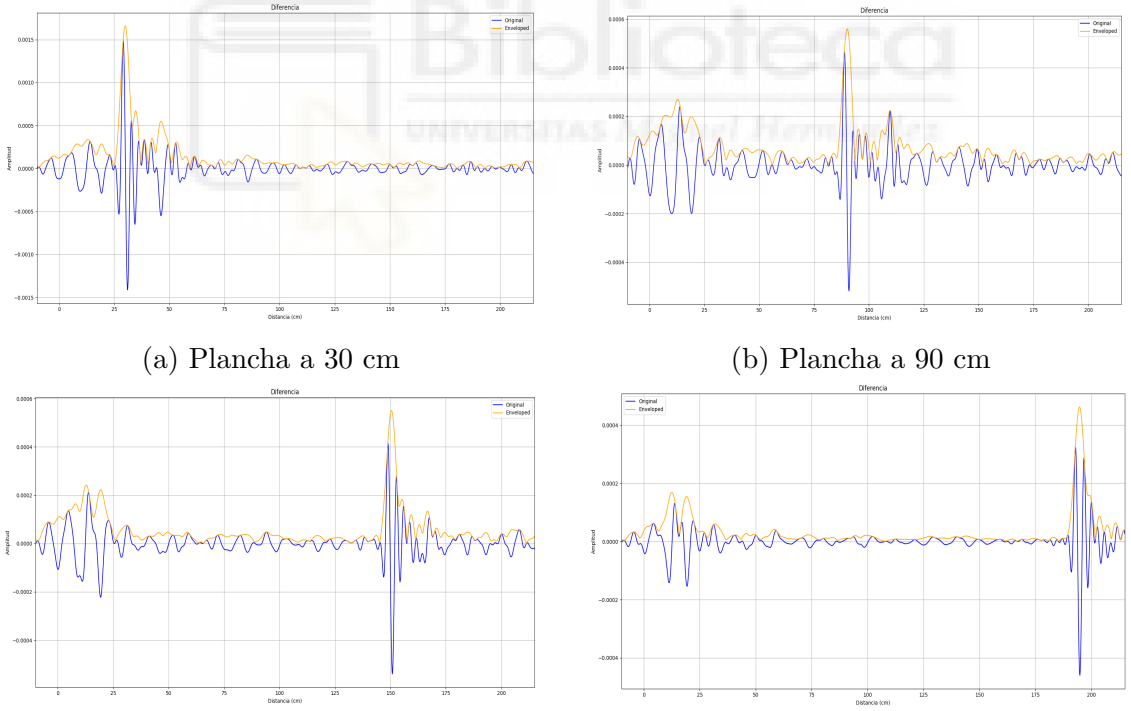
Figura 4.10: Proceso medida reflexión

Conectamos la antena Vivaldi al VNA, colocamos la plancha metálica a una distancia de la antena, intentando que estas estén lo más alineadas posible, ayudándonos de un metro vamos midiendo varias distancias y variando la posición de la plancha metálica. Con la ayuda del VNA obtendremos los parámetros S11 de la antena a las distintas distancias medidas y estos los valores podrán ser exportados como archivos s1p para poder usarlos en el programa de Python diseñado. A la hora de analizar los resultados tendremos en cuenta dos casos: medidas tomadas a "grandes distancias" y medidas tomadas a "pequeñas distancias". Esto se debe a lo que llamamos "zona de ruido", a grandes distancias en principio no deberíamos tener ningún problema para poder determinar la distancia a la que se encuentra la plancha, ya que se

encuentra lo suficientemente lejos de cualquier interferencia, sin embargo a pequeñas distancias se puede dar el caso de que se mezcle nuestra señal de interés (posición de la plancha) con interferencias de la propia antena que no somos capaces de eliminar, imposibilitando el poder determinar la distancia a la que se encuentra la plancha metálica. En principio este ruido podría eliminarse casi en su totalidad haciendo la diferencia de la señal medida respecto a una de referencia, la cual solo detectaría los elementos del laboratorio. Las mediciones realizadas han sido a 30, 50, 70, 90, 110, 130, 150, 170, 190 y 195 cm aproximadamente. Si hacemos uso del código de Python creado obtenemos los siguientes resultados.

4.3.2. RESULTADOS

Algunos de las distancias obtenidas mediante nuestro código de Python se muestran a continuación:



(a) Plancha a 30 cm

(b) Plancha a 90 cm

(c) Plancha a 150 cm

(d) Plancha a 195 cm

Figura 4.11: Variación distancia plancha metálica

Como hemos comentado anteriormente el máximo de nuestra señal se correspondería con la distancia a la que se encuentra la plancha de la antena. Por una parte, vemos que somos capaces de determinar los cambios de posición de la plancha y por otra

vemos a medida que alejamos la plancha metálica de la antena , el máximo de nuestra señal se va alejando también de la referencia. A continuación, se muestra una tabla comparativa en la cual podremos ver la diferencia entre los valores esperados y los valores que hemos obtenido mediante nuestro programa de Python (medido), además del error que estamos cometiendo.

Esperado	**Medido**	**Error (%)**
30	30.04	0.133
50	49.9	0.2
70	69.51	0.7
90	90.12	0.133
110	110.86	0.782
130	130.22	0.169
150	150.45	0.3
170	168.57	0.84
190	191.93	1.016
195	194.68	0.164

Si representamos la diferencia de estas distancias en una gráfica:

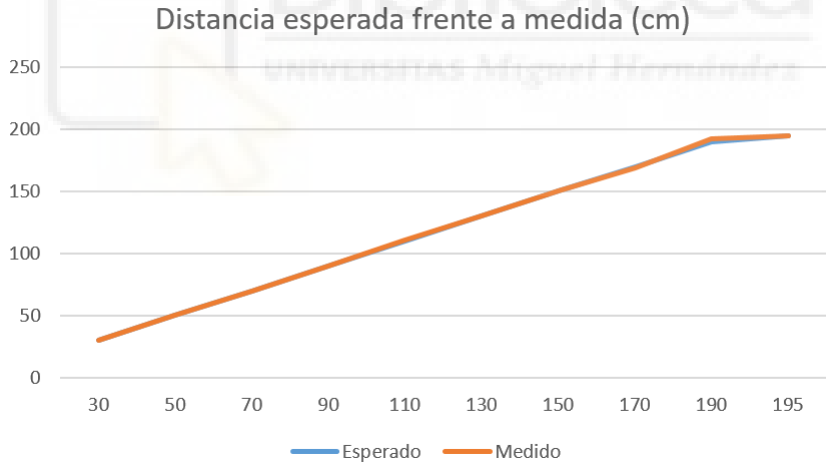


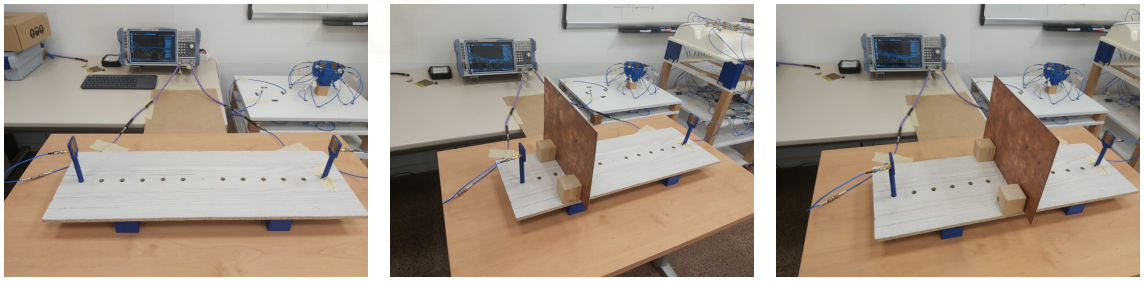
Figura 4.12: Gráfico lineal distancia esperada frente a medida

En vista de los valores que obtenemos podemos afirmar que para este primer caso nos ha sido posible determinar la distancia a la que se encuentra la plancha metálica de la antena. Como es de esperar se comente un cierto error, pero este no es muy elevado, pudiéndose deber a inexactitudes de la cinta métrica usada, un error humano a la hora de tomar las medidas o la necesidad de mayor precisión a la hora de procesar los datos.

4.4. CASO REAL 2: EMPLEANDO ANTENAS DE BANDA ANCHA

4.4.1. INTRODUCCIÓN

Una vez comprobado que podemos medir la distancia a la que se encuentra un objeto de una antena haciendo uso del parámetro S11 que obtenemos de la reflexión de una señal generada por la antena y reflejada por el objeto, pasaremos a un caso un poco más complejo pero que se rige bajo el mismo fundamento. Siguiendo un procedimiento análogo al caso anterior, demostraremos que podemos determinar la posición de un objeto situado entre dos antenas enfrentadas fijas. Para lograr este fin, mediremos los parámetros S11 y S22 de las antenas (pues estos nos aportaran la información relacionada a la reflexión de nuestra señal en ambos puertos) respecto a una plancha metálica situada entre ambas, de la misma forma que en el caso anterior iremos moviéndola a distintas distancias, comprobando en primer lugar, si podemos detectar los cambios de posición y en segundo, la distancia a la que se encuentra de cada una de las antenas en el momento que se realiza la medición.



(a) Antenas enfrentadas (b) Adición plancha metálica (c) Desplazamiento

Figura 4.13: Materiales empleados

El código que empleamos es el mismo que para el caso anterior, siendo la única diferencia que en este caso debemos de cargar archivos s2p al disponer de dos puertos (antenas) y por tanto, deberemos asignar más variables y procesarlas, pues estamos empleando tanto parámetros S11 como S22, estos cambios en el código están reflejados en el ANEXO 2.

4.4.2. TOMA DE MEDIDAS

Como ya hemos comentado anteriormente disponemos de un sistema que se basa en dos antenas enfrentadas fijas separadas una distancia aproximada de 60 cm y de una plancha metálica. Mediante el empleo de este sistema pretendemos medir los parámetros S11 y S22, y de forma análoga al capítulo anterior determinar la distancia a la que se encuentra la plancha de cada una de las antenas.

El procedimiento que seguiremos para llevar a cabo las medidas se describe a continuación: inicialmente pondremos la plancha metálica a una distancia próxima a una de las antenas, nosotros escogeremos la que está situada a la izquierda en la Figura 5.1 a) a la cual llamaremos antena 1, pues esta está conectada al puerto 1 del VNA y de esta forma evitaremos cualquier confusión, el VNA realizará las medidas y exportaremos los datos como archivos s2p para que puedan ser empleados por nuestro código. A partir de estos datos generaremos dos subgráficas, como las que se pueden apreciar en las figuras 4.14, 4.15 y 4.16, la subgráfica de arriba está generada a partir del S11 y por tanto nos indicará la distancia a la que se encuentra la plancha de la antena 1 y la de abajo está generada a partir del S22 y por tanto nos indicará la distancia a la que se encuentra la plancha de la antena 2. Lo que esperamos obtener en este primer caso son que el pico o máximo de la primera señal, generada a partir del S11, se encuentre muy cerca del origen (específicamente a la distancia a la que pongamos la plancha de la antena) y el máximo de la segunda señal, generada a partir del S22, se encuentre alejado (a una distancia complementaria de los 60 cm). A medida que vayamos moviendo la plancha metálica nos alejaremos de la antena 1 y por tanto, el primer pulso se desplazará hacia la derecha (alejándose del origen/antena 1), por consiguiente, nos estaremos acercando a la antena 2 y por tanto el segundo pulso se desplazará hacia la izquierda (menor distancia respecto a la plancha). Cuando la plancha esté situada a la mitad de distancia, ambos pulsos deberán estar aproximadamente en la misma posición pues ambas antenas deberían encontrarse a la misma distancia de la plancha, demostrando así que somos capaces de determinar la posición y distancia de la misma.

4.5. RESULTADOS

A continuación se presentan los resultados obtenidos para los casos descritos anteriormente:

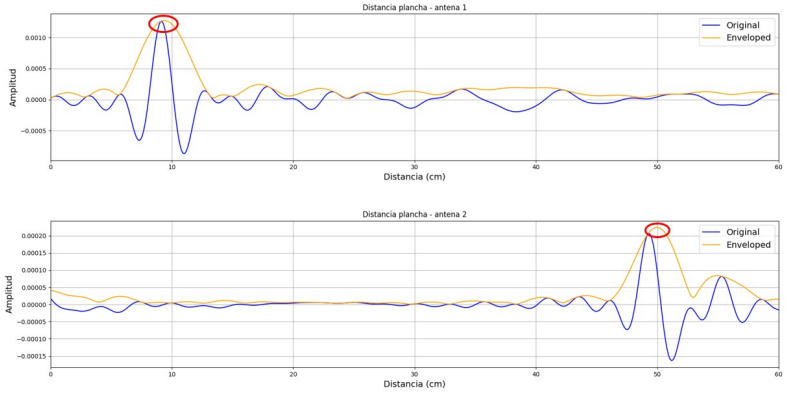


Figura 4.14: Distancia 10 cm respecto antena 1

Como podemos observar al poner la plancha metálica a 10 cm de la antena de referencia obtenemos un máximo en 9.43 cm y otro en 49.902 cm, al sumar ambas distancias obtenemos 59.332 cm. Con lo cual podemos afirmar que podemos determinar la distancia de ambas antenas a la plancha de manera correcta aunque tenemos un pequeño error de 1 cm a la hora de realizar las mediciones. En los siguientes casos veremos el comportamiento de los máximos a medida que vamos alejando la plancha metálica de la antena de referencia.

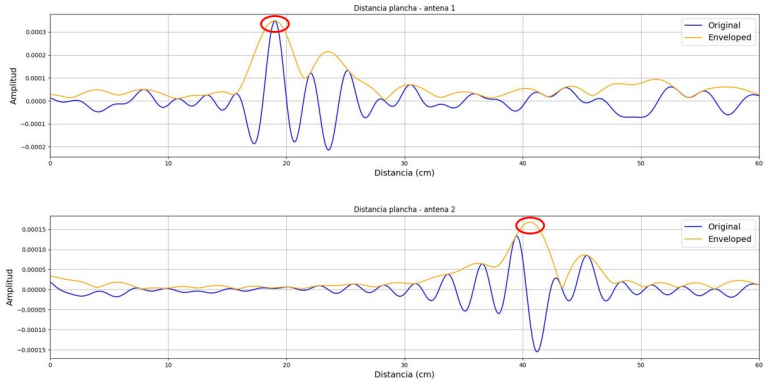


Figura 4.15: Distancia 20 cm respecto antena 1

En este segundo caso vemos que el máximo que antes estaba en 10 cm ahora se obtiene en 19.048 cm y que el que estaba en 59.332 cm ahora está en 40.658 cm, esto tiene sentido ya que hemos alejado la plancha de la antena de referencia y está más

cerca de la antena complementaria, además al sumar ambas distancias obtenemos 59.706 cm, distancia que concuerda con la establecida de 60 cm. Vemos que el error se ha reducido respecto al caso anterior, probablemente por una mayor precisión a la hora de realizar la medición.

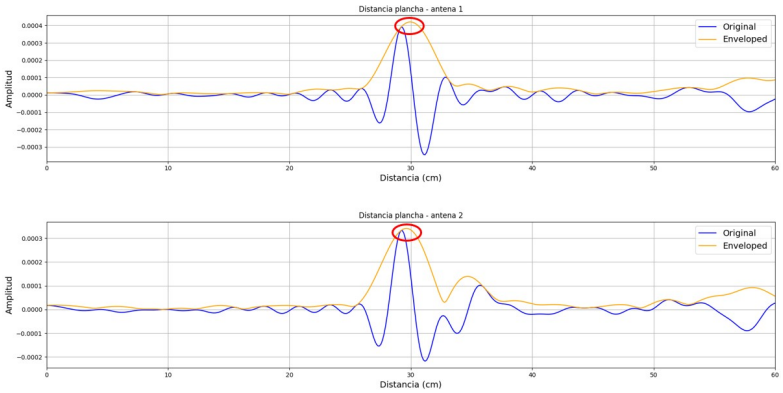


Figura 4.16: Distancia 30 cm respecto antena 1

Al disponer la plancha metálica a 30 cm de la antena de referencia la distancia entre ambas antenas con respecto a la plancha es la misma (las dos deben estar aproximadamente 30 cm). La antena de referencia está a 29.916 cm mientras que la complementaria a 29.666 cm, si sumamos las distancias obtenemos 59.582 cm. Y por tanto, podemos afirmar que las mediciones son correctas y podemos pasar a realizar las mediciones de los parámetros S21.

Los casos que hemos mostrado son únicamente algunos ejemplos, los cuales hemos considerado de mayor interés, en la tabla que se muestra a continuación hemos recogido los valores obtenidos para el resto de casos que se midieron:

Dist plancha-Antena 1 (teórica)	**Antena 1**	**Antena 2**
2.5	2.185	57.52
5	4.683	55.148
7.5	6.557	53.025
10	9.43	49.902
12.5	11.553	47.903
15	14.052	45.53
17.5	16.55	42.907
20	19.048	40.658
22.5	21.92	37.785
25	24.42	35.287
27.5	26.792	32.914
30	29.916	29.666

Como podemos observar hemos obtenido valores bastante buenos para todos los casos, pues el error máximo que cometemos está en torno a 1 cm y la suma de las distancias de las dos antenas es de aproximadamente los 60 cm que se esperan. Por tanto, podemos finalizar diciendo que somos capaces de determinar la distancia a la que se encuentra la plancha de cada una de las antenas y que siendo más exhaustivo a la hora de tomar las medidas y realizar los debidos ajustes en el código podríamos llegar a obtener resultados todavía más precisos.

5. CAPÍTULO V: ANÁLISIS DE PULSOS EN TRANSMISIÓN

5.1. INTRODUCCIÓN

En los capítulos anteriores describimos el método de reflexión de pulsos, el cual vimos que resultaba ser bastante eficaz y nos permitía obtener unos resultados más que razonables a la hora de determinar distancias para casos sencillos, pero a medida que los casos se van haciendo un poco más complejos, como podría ser el caso de cuando se disponen dos antenas enfrentadas, no nos era posible continuar empleando este método. Por tanto, en el siguiente capítulo llevaremos a cabo el estudio y análisis del método de transmisión de pulsos y comprobaremos si con este método podemos abarcar los casos que no nos eran posibles mediante el método de reflexión.

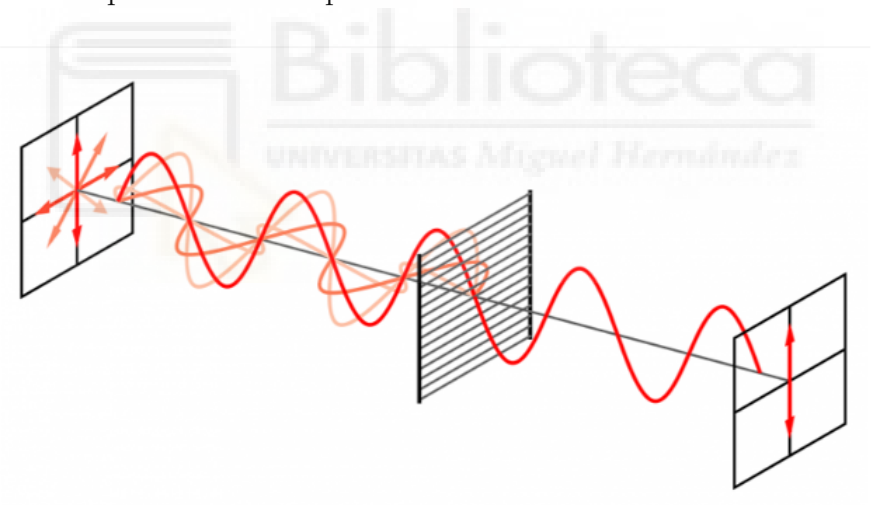


Figura 5.1: Transmisión de pulso

Nuevamente con fines aclaratorios y de igual manera que se hizo en los capítulos anteriores, hemos diseñado un diagrama de flujo que describirá el proceso seguido para la creación del código en este apartado del proyecto y se darán algunas explicaciones de los aspectos más relevantes del mismo.

En primer lugar, obtenemos los datos frecuenciales de los parámetros S21 de los ficheros s2p. Para ello hacemos uso de las funciones "network.f" y "network.s", al querer obtener los parámetros S21 al usar la función network.s pondremos entre corchetes [:,1,0]:

```
1
2       #Obtenemos datos parámetros S
3       network = rf.Network(filename)
4       frecuencia = network.f
5       #Donde filename es la ruta donde se encuentran nuestros
           ↪ archivos s2p
6       S_params = network.s
7
8       #Leemos los parámetros S
9       S21 = network.s[:, 1, 0]
10      S21_db = 20 * np.log10(np.abs(S21))
11
12      #Inicalizamos variable y le pasamos los valores
13      S21_ref = np.zeros((Nantenas, LenScan), dtype=complex)
14      S21_ref = S21
15
```

Código 5.1: Obtención de parámetros S21

Los dos siguientes fragmentos del código se corresponden con la generación del nuevo eje de frecuencias y del enventanado Tukey. Teniendo en cuenta que seguiremos exactamente el mismo procedimiento descrito anteriormente podríamos no incluirlos, pues podría resultar repetitivo, pero como tratamos de realizar una descripción independiente de cada código sí que los incluiremos. Y por tanto, el código correspondiente a la generación del nuevo eje de frecuencias es el siguiente:

```
1       Fs = 120e9
2       DF = frecuencia[1]-frecuencia[0]
3       new_frequency = np.linspace(0, Fs, len(frecuencia))
4       F_axis = np.arange(0, Fs, DF)
5       nfft = len(F_axis)
6       t_axis = np.arange(nfft) / Fs
7
```

Código 5.2: Generamos nuevo eje de frecuencias y tiempo

Y el correspondiente a la generación del enventanado Tukey es el siguiente:

```
    #Ventana tukey

    MyWin = US.makeWindow(SortofWin='tukey', WinLen=LenScan,
       ↪ param1=0.25, param2=1, Span=0, Delay=0)
    if np.mod(nfft,2)==1:
    MyWin_ext = np.concatenate(( MyWin, np.zeros((int(
       ↪ (nfft+1)/2 )-LenScan)) ))
    MyWin = np.concatenate((MyWin_ext,
       ↪ np.flipud(MyWin_ext[:-1])))
    else:
    MyWin_ext = np.concatenate(( MyWin,
       ↪ np.zeros((int(nfft/2)-LenScan)) ))
    MyWin = np.concatenate((MyWin_ext, np.flipud(MyWin_ext)))

```

Código 5.3: Generamos ventana Tukey

Como ya hemos mencionado realizaremos un filtrado de las señales utilizando una forma de onda de pulso con el fin de evitar señales parásitas y elementos externos a nuestro sistema, aplicaremos la ventana Tukey creada en el apartado anterior y finalmente podremos realizar la transformada inversa de nuestras señales, obteniendo así nuestros datos en el dominio del tiempo:

```
S_ref = MWF.makeSpectrum(S21_ref, F_axis, frecuencia)
ref_in_time, borrar = MWF.adaptedFilter(S_ref, Pulse, XCR=True,
      ↪ Aligned=True,
Wiener=False) #
vo = (S_ref[0,:] * Pulse_in)

OffSet = 0 # offset in mm
dis_axis = (np.arange(ref_in_time.shape[1]))  * speed / (Fs) * 10
      ↪ 00 - OffSet


#Calculamos la IFFT  de Vi(f) y Vo(f)
Vit = np.fft.ifft(Pulse_in)
Vot = np.real(np.fft.ifft(vo) * MyWin)


```

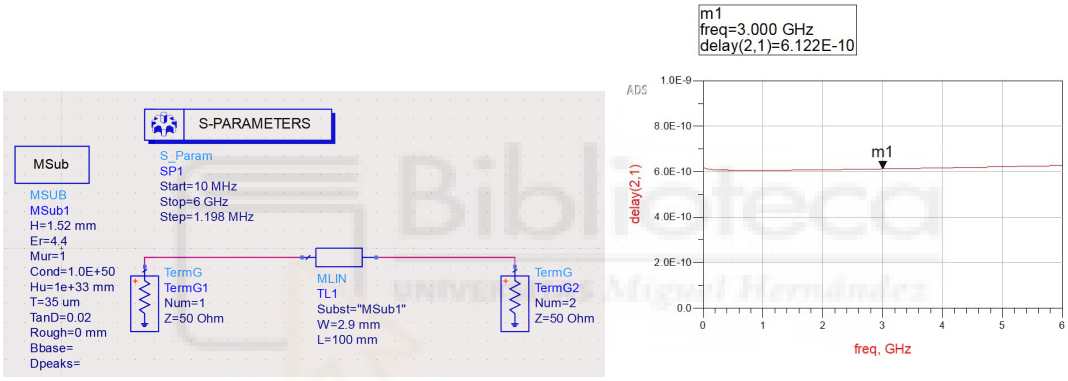
Código 5.4: Generación del espectro, cálculo de Vo(f), generación del eje de distancias e IFFT de Vo(f)

Una vez comentadas las partes más relevantes del código pasaremos a analizar unos casos de prueba para demostrar el buen funcionamiento del mismo.

5.2. CASOS IDEALES

Al igual que en el caso de la reflexión de pulsos primero debemos realizar algunas comprobaciones con casos sencillos e ideales con el fin de adecuar nuestro programa y comprobar que realmente las mediciones que vamos a realizar son correctas. Los casos a estudiar serán de nuevo un cable coaxial y línea microstrip con distintos valores de ϵr y los métodos que emplearemos serán exactamente los mismos que los descritos en los anteriores casos.

5.2.1. LÍNEA MICROSTRIP



(a) Esquema ADS (b) Retardo de grupo

Figura 5.2: Ejemplo línea microstrip en ADS

Los valores de retardo de grupo obtenidos para los distintos valores de ϵr son los que se muestran a continuación:

ϵr	δt esperada (ns)	δt obtenida (ns)	Error (%)
2	0.4379	0.4499	2.74
4.4	0.6122	0.62457	2.02
8	0.7955	0.8083	1.609
10	0.8783	0.8917	1.5257

El procedimiento es análogo al caso de las medidas en reflexión, con la única diferencia de que ahora sí que empleamos el valor completo de δt, pues será el tiempo que tarda la señal en transmitirse a lo largo de nuestro sistema. Por ejemplo, si tomamos

el retardo de grupo correspondiente a $\epsilon r = 2$ y una L = 100 mm obtenemos:

$$v = \frac{d}{t} = \frac{100 \times 10^{-3}}{0{,}4499 \times 10^{-9}} = 222{,}2716159 * 10^{6} m/s$$

Si tomamos la velocidad de la luz como $c = 3 \times 10^{8}$ m/s, y empleamos la siguiente ecuación:

$$v = \frac{c}{\sqrt{\epsilon eff}} \tag{5.1}$$

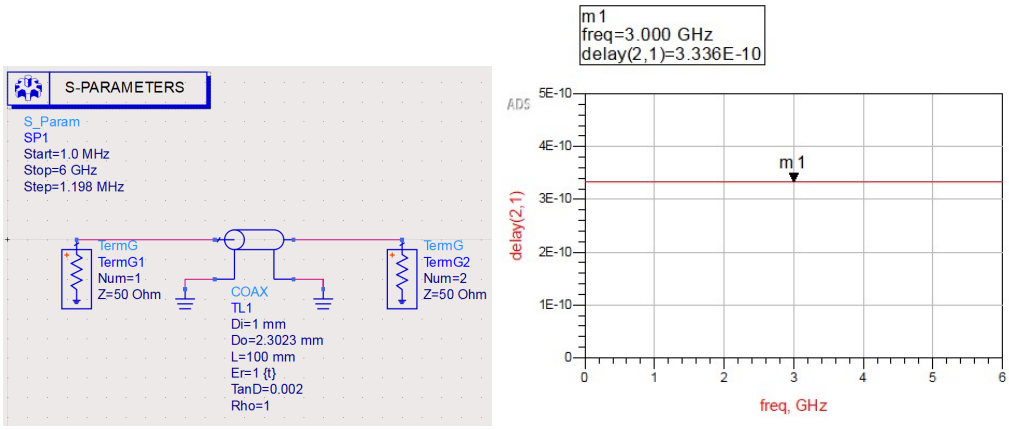
Despejando ϵeff y tomando la velocidad de la luz como $c = 3 \times 10^{8}$ m/s, obtenemos:

$$\epsilon eff = (\frac{c}{v})^2 = (\frac{3 * 10^{8}}{222{,}2716159 * 10^{6}})^2 = 1{,}8192$$

Repitiendo el mismo proceso para el resto de casos obtenemos:

ϵr	ϵeff esperada	ϵeff obtenida	Error (%)
2	1.727	1.8192	5.3387
4.4	3.325	3.5059	5.4406
8	5.536	5.872	6.069
10	6.711	7.1463	6.4864

5.2.2. CABLE COAXIAL



(a) Esquema ADS (b) Retardo de grupo

Figura 5.3: Ejemplo cable coaxial en ADS

Los retardos de grupo obtenidos en este caso han sido los siguientes:

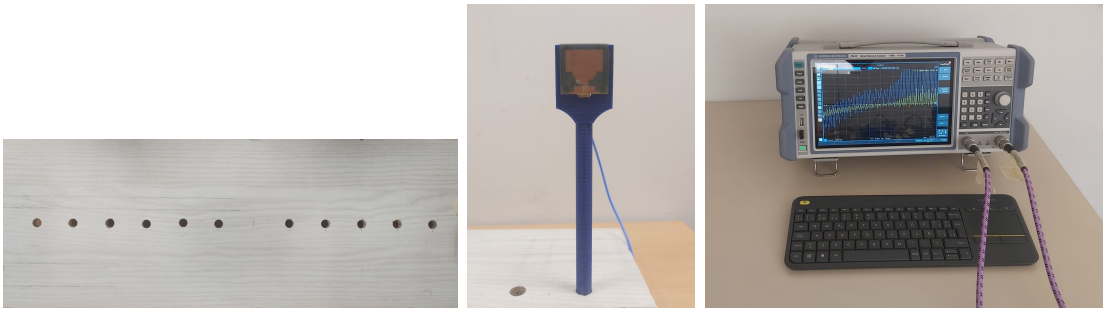
ϵr	**δt esperada (ns)**	**δt obtenida (ns)**	**Error (%)**
1	0.3336	0.34167	2.42
2.1	0.4834	0.4917	1.717
4	0.6671	0.675	1.184
5	0.7459	0.75	0.5497
10	1.055	1.0583	0.3128

Empleando los valores de δt que obtenemos calculamos el valor de ϵr :

ϵr esperada	**ϵr obtenida**	**Error (%)**
1	1.049	4.9
2.1	2.173	3.476
4	4.095	2.375
5	5.055	1.1
10	10.066	0.66

5.3. CASOS REALES

Tras haber realizado las pruebas correspondientes y en vista de los resultados obtenidos hemos podido comprobar que nuestros método es válido y por tanto, podremos medir la distancia en casos reales de transmisión. Para ello haremos uso del código que se encuentra descrito en el ANEXO 4 (resumido brevemente al principio de este capítulo) y de los materiales que disponemos en el laboratorio. El equipo que emplearemos se muestra a continuación:

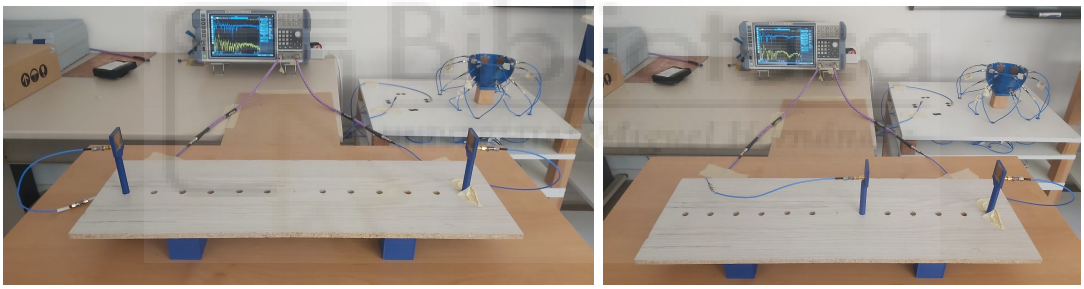


(a) Soporte de antenas (b) Antena (c) VNA

Figura 5.4: Materiales empleados

5.3.1. TOMA DE MEDIDAS

El procedimiento que seguiremos para realizar la toma de medidas se describe a continuación. Inicialmente dispondremos las antenas a la máxima distancia que nos permita el soporte, este soporte consiste en una tabla de madera de unos 60 cm con agujeros cada 5 cm en ambos sentidos respecto al centro de la misma, y trataremos de medir la distancia a la que se encuentran empleando los parámetros S21 que obtenemos mediante el VNA. Una vez tomada esta medida pasaremos a la siguiente, sacaremos la antena de donde se encuentra y la dispondremos en el siguiente agujero, si continuamos realizando este proceso sucesivamente iremos acercando las antenas cada vez más, hasta que alcancemos la distancia mínima permitida (en torno a los 5 cm). Mediante este método pretendemos en primer lugar ser capaces de detectar las variaciones de posición de las antenas y en segundo lugar la distancia real a la que se encuentran.



(a) Situación inicial (b) Desplazamiento de antena

Figura 5.5: Toma de medidas en transmisión

Una vez explicado el procedimiento que seguiremos para la toma de medidas y el código que se empleará para procesar los datos que obtengamos de estas, lo único que falta es analizar los resultados que obtenemos para casos reales. Los casos planteados son los siguientes: antenas en campo lejano, antenas a distancias medias y antenas en campo próximo.

5.3.2. RESULTADOS

Caso 1: Antenas en campo lejano (60 cm)

Para este primer caso alejaremos las antenas la máxima distancia que nos permita el soporte que emplearemos, la cual se corresponde con 60 cm. En la imagen que se muestra a continuación podemos diferenciar dos subgráficas, la primera representa el pulso de entrada Vi (señal transmitida), mientras que la segunda el pulso de salida Vo (señal recibida):

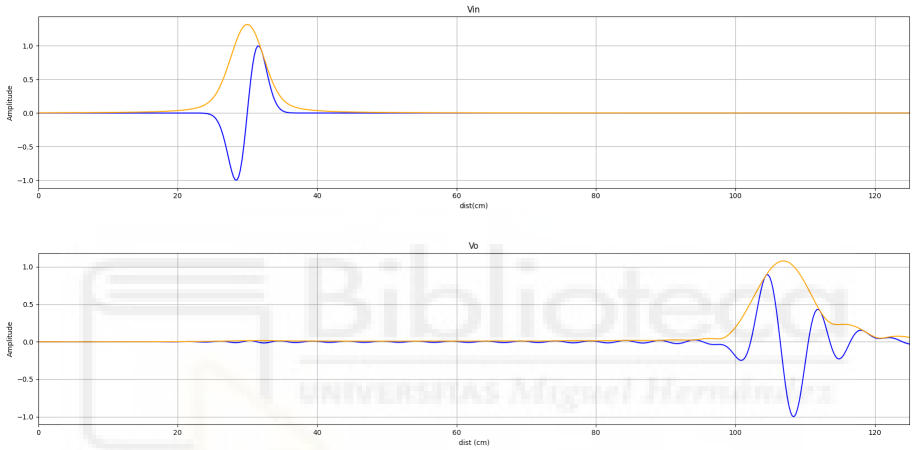
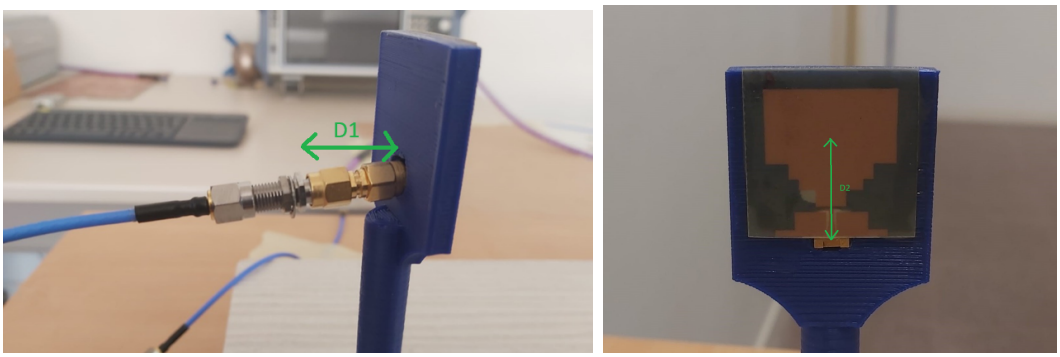


Figura 5.6: Transmisión a máxima distancia 60 cm

Para determinar la distancia entre las antenas lo único que debemos hacer es realizar la diferencia de distancias entre los máximos de ambas señales. Haciendo uso de nuestro código podemos determinar que el máximo de Vi se encuentra en 29.98 cm, mientras que el de Vo se encuentra en 106.93 cm, por lo tanto al realizar la diferencia obtendríamos que la distancia entre máximos es de 76.95 cm, lo que en un principio puede parecer un error muy grande al esperar un valor sobre los 60 cm. Tras realizar más pruebas con distintos casos llegamos a la conclusión de que esta distancia de error era común en todos ellos y por tanto debía deberse a un factor que no estábamos teniendo en cuenta a la hora de realizar las medidas. Se realizaron diversos cálculos de distancias de los distintos elementos del sistema hasta que dimos con los causantes de la diferencia en las distancias:



(a) Distancia añadida por conector (b) Tramo de antena

Figura 5.7: Distancias añadidas

Esta diferencia de 16.95 cm respecto al valor esperado se debe en parte a que a la hora de hacer la calibración, esta se hizo al principio o entrada de los conectores y no directamente en la antena y por tanto como se muestra en la figura 6.7 a) estamos añadiendo una distancia que no se debe tener en cuenta a la hora de realizar las medidas. Esta distancia es de unos 3.7 cm por cada conector, lo que implicaría añadir 7.4 cm a nuestro sistema, pero es que además estos conectores por dentro están formados por teflón, por tanto para saber la distancia real que están añadiendo deberemos multiplicar esta distancia por la ϵr del material (ϵr=2.1), obteniendo una distancia de 10.724 cm. Este no es el único error que debemos tener en cuenta pues si observamos la figura 6.7 b) vemos que hay un pequeño tramo de 1.65 cm desde que la señal sale del vivo de la antena hasta que llega a la zona donde se radia. Si a distancia le multiplicamos la ϵr del material de la antena obtenemos que la distancia que añaden las antenas son 6.017 cm. Sumando ambos errores obtendríamos que estamos añadiendo aproximadamente 16.74 cm, lo cual más o menos se corresponde con la diferencia de 16.95 cm que cometemos en nuestras medidas y por tanto si a la distancia que hemos obtenido antes le quitamos la distancia añadida por los conectores y el tramo de antena obtenemos que la distancia real entre las dos antenas es de 60.21 cm. Una vez corregida esta distancia podemos representar los pulsos de forma adecuada:

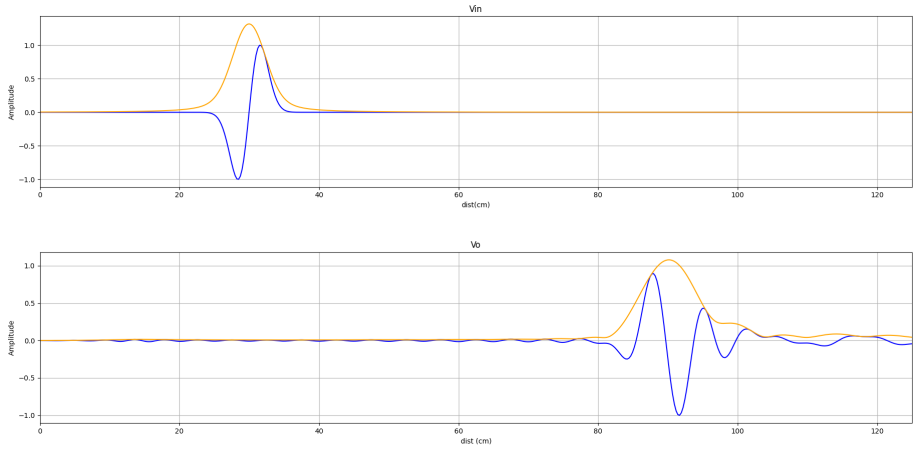


Figura 5.8: Distancia 60 cm corregida

Caso 2: Antenas a distancias medias (35 cm y 25 cm)

Para estos siguientes casos acercaremos la antena un poco, como por la disposición de los agujeros del soporte nos es imposible colocarla justo a la mitad de la distancia máxima (30 cm) la colocaremos lo más próxima posible a esa distancia (35 y 25 cm). Como podemos comprobar a continuación, para el caso de 35 cm somos capaces de detectar el cambio de posición de la antena:

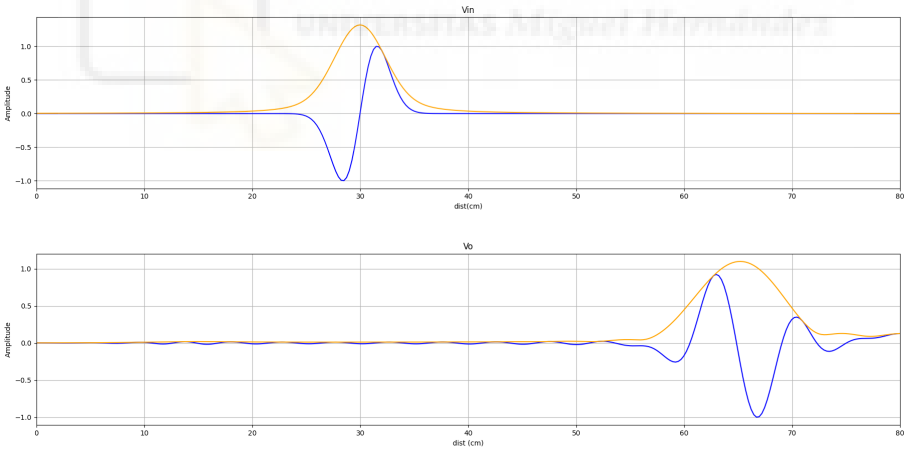


Figura 5.9: Transmisión a medias distancias 35 cm

Y la distancia que obtenemos mediante la diferencia de posición de máximos también es correcta, pues 65.202 (respectivo a Vo) menos 29.98 (respectivo a Vi) se corresponde con una distancia de 35.222 cm.

Como ya hemos comentado la otra medida que se tomó para el caso de medias distancias es la de 25 cm de distancia, el resultado obtenido se muestra a continuación:

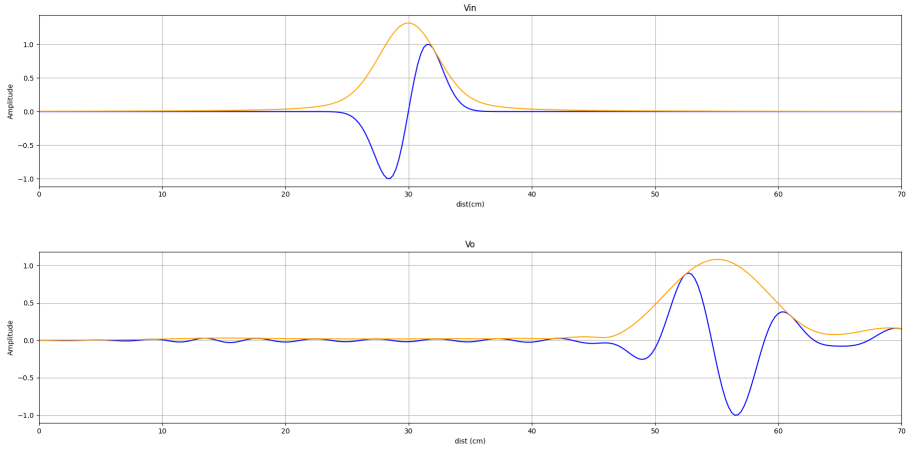


Figura 5.10: Transmisión a medias distancias 25 cm

A simple vista se podría determinar que es correcto pues la posición del máximo parece estar desplazada 10 cm con respecto al caso anterior. Para comprobarlo realizamos la diferencia de posición de los máximos, los cuales se encuentran en 54.96 y 29.98 respectivamente, lo que supone una distancia de 24.98 cm.

Caso 3: Antenas en campo próximo (15 cm)

Finalmente, tenemos el caso de campo próximo, para realizar este estudio acercamos todo lo que nos permitía el soporte físico las dos antenas. Los resultados obtenidos se muestran a continuación:

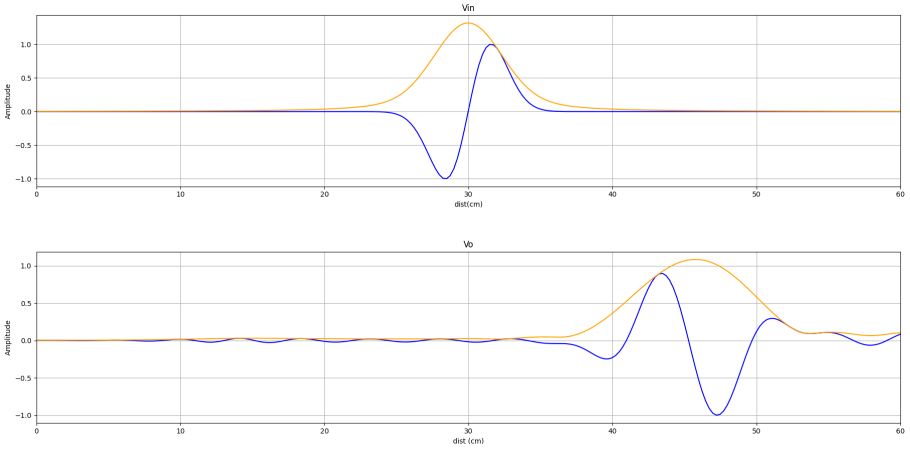


Figura 5.11: Transmisión a mínima distancia 15 cm

Siguiendo el mismo planteamiento que en los casos anteriores obtenemos el valor de la posición de los máximos de cada pulso, que se corresponden con 45.716 y 29.98, y haciendo su diferencia obtenemos la distancia a la que se encuentran las antenas, empleando estos valores obtenemos 15.736 cm.

Los casos previamente planteados eran los más representativos pues cubríamos todos los rangos de distancias, pero se realizaron más medidas a parte de estas para tener una gran muestra y poder verificar así el buen funcionamiento de nuestro código. En la tabla que se muestra a continuación hemos recogido los resultados obtenidos para todo el resto de distancias medidas, el procedimiento que hemos seguido ha sido el mismo que el descrito en los casos anteriores:

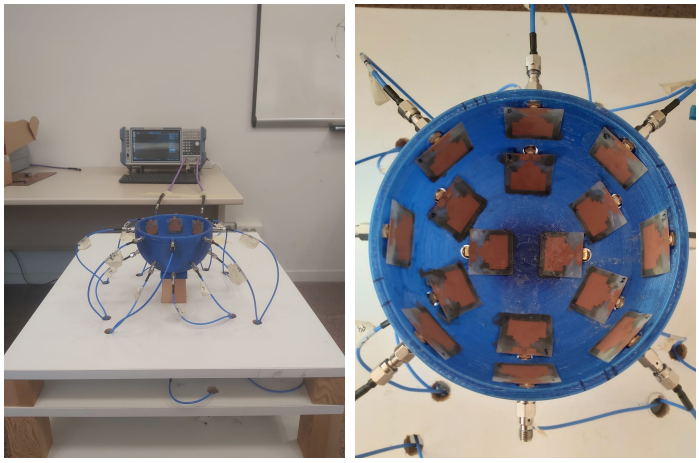
Distancia esperada (cm)	**Distancia obtenida (cm)**	**Error (%)**
60	60.206	0.3433
55	55.21	0.3818
50	50.213	0.426
45	45.466	1.036
40	40.22	0.55
35	35.223	0.637
25	24.98	0.08
20	20.483	2.415
15	15.736	4.91

Para concluir este capítulo y en vista de los resultados obtenidos podemos determinar que nos ha sido posible determinar la distancia entre un par de antenas enfrentado empleando el método de transmisión de pulsos. También que el error que cometemos no es muy grande (no es mayor de 1 cm en ningún caso) y por tanto, las medidas se están tomando correctamente y el código que hemos diseñado para llevar a cabo el procesado de los datos es válido.

6. CAPÍTULO VI: ANÁLISIS DE PULSOS EN SISTEMA DE IMAGEN MÉDICA POR MICROONDAS, MICROBIO

6.1. INTRODUCCIÓN

Los resultados obtenidos en anteriores capítulos nos permiten afirmar que nos es posible calcular distancias a las que se encuentran distintos elementos, sin embargo, hay un factor que ha sido crucial para la obtención de estos resultados y este es que el medio por el cual se transmiten las señales es homogéneo, particularmente aire. Estos casos no terminan de ser "realistas" pues en aplicaciones reales (médicas u otros ámbitos) no suele cumplirse esa condición (los elementos de estudio están formados por distintos tejidos y materiales con distintas propiedades y estructuras) y por tanto, debemos comprobar la viabilidad de nuestro sistema y métodos en casos donde el sistema no es homogéneo. Por estos motivos el siguiente capítulo se empleará para llevar a cabo un caso que podríamos considerar "real", en el cual someteremos los desarrollos llevados a cabo en capítulos anteriores a medios que no son homogéneos, comprobando si somos capaces de determinar la distancia a la que se encuentran los distintos elementos o el tiempo que tarda la señal en propagarse a lo largo del sistema. El dispositivo empleado es el siguiente:



(a) Sistema empleado (b) Interior del sistema

Figura 6.1: Sistema microBIO

Como se puede observar se trata de una cavidad repleta de antenas de un diámetro aproximado de 13.2 cm (distancia entre dos antenas enfrentadas)y como ya hemos dicho, el objetivo que perseguimos es el mismo que anteriores capítulos, demostrar que podemos determinar la distancia a la que se encuentran los distintos elementos que introduzcamos o el tiempo que tarda la señal en propagarse a lo largo de este sistema, empleando las medidas de dos antenas que se encuentren enfrentadas. En nuestro caso, llevamos a cabo las medidas con dos pares de antenas enfrentadas distintas, las antenas 1-5 y 9-13.

Una vez comentado el dispositivo que emplearemos para realizar las medidas hablaremos de los elementos y medios que emplearemos, en el laboratorio disponemos de los siguientes:

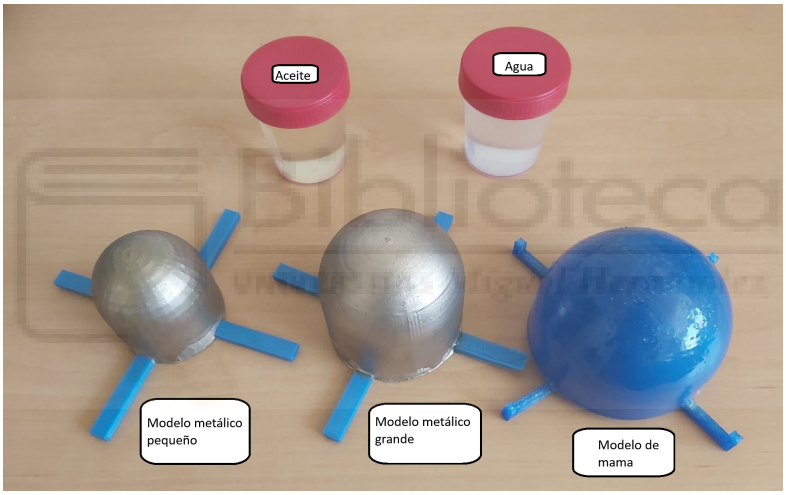


Figura 6.2: Elementos disponibles

- Modelo metálico grande y pequeño: no se emplearán como tal para realizar medidas, pero al pasar nuestros código de un sistema a otro distinto debemos realizar algunos ajustes en el mismo y por tanto, emplearemos estos elementos cuyas propiedades nos facilitarán el trabajo.
- Modelo de mama: se realizarán dos medidas distintas respectivas a este modelo, en la primera la referencia será el vacío y veremos si somos capaces de detectar la mama en si y en la segunda emplearemos la este modelo como referencia e introduciremos dentro distintos materiales. Hecho de PLA ($\epsilon r = 2.88 \approx 3$).
- Aceite: material que introduciremos dentro de la mama con $\epsilon r \approx 5$.

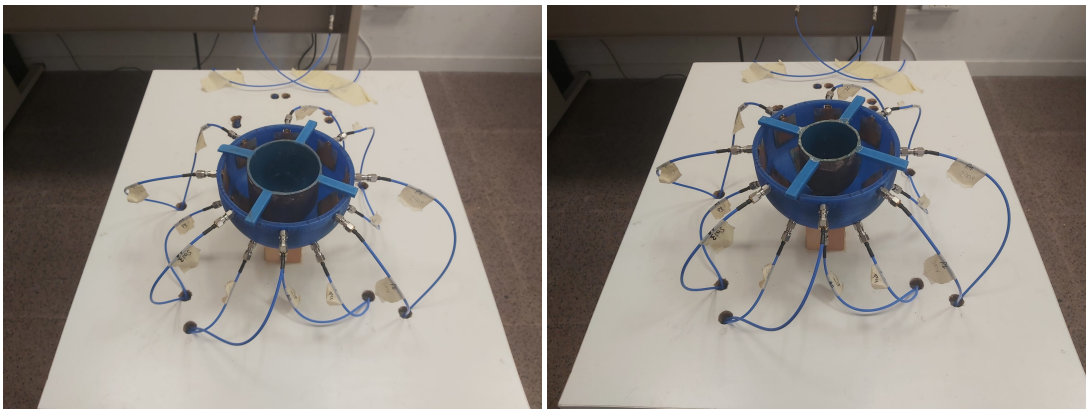
- Agua: material que introduciremos dentro de la mama con $\epsilon r \approx 80$.

Cabe destacar que nos habría gustado poder realizar las medidas de algún material cuyo valor de ϵr se encontrase entre los del aceite y el agua, pero tras mucho tiempo pensando no logramos encontrar ningún material de "fácil obtención" cuyas propiedades se adecuasen a nuestras necesidades. Una vez comentados los materiales de los que disponemos en el laboratorio pasaremos a comentar los casos de estudio tanto en reflexión como en transmisión.

6.2. REFLEXIÓN

6.2.1. REFLEXIÓN DE MODELOS METÁLICOS

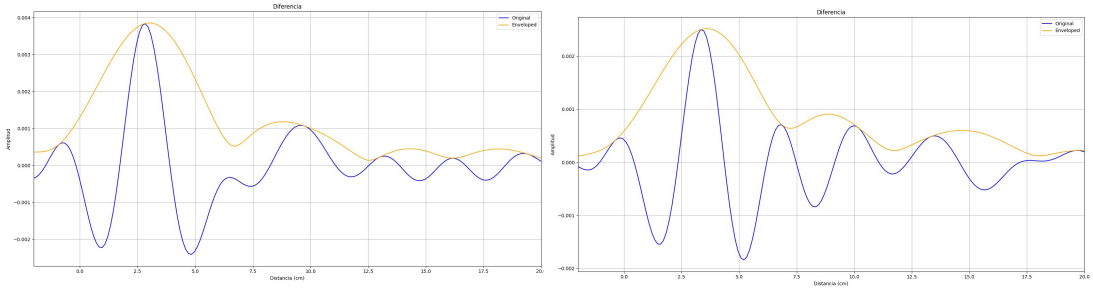
Como ya se ha comentado en el apartado anterior las medidas con los modelos metálicos no forman parte en si de los resultados pues únicamente se emplearán para realizar los ajustes necesarios en el código para poder tomar las medidas de forma correcta al pasar de un equipo a otro. El código empleado es el mismo que en el capítulo IV (descrito en el ANEXO II) y el único cambio que debemos realizar en el valor de offset de las antenas. Para ello deberemos saber las distancias a las que se encuentran los elementos en el laboratorio, haciendo uso de una cinta métrica obtenemos que la distancia que hay entre una de las antenas y el borde del modelo metálico grande es de 2.85 cm y en el caso del modelo metálico pequeño es de 3.6 cm.



(a) Modelo metálico grande (b) Modelo metálico pequeño

Figura 6.3: Modelos metálicos

Una vez conociendo la posición del máximo ajustaremos el offset de nuestro sistema obteniendo:



(a) Posición modelo metálico grande (b) Posición modelo metálico pequeño

Figura 6.4: Modelos metálicos

Las posiciones de los máximos de las señales se obtienen en 2.961 y 3.585cm respectivamente. Y por tanto, una vez realizados los ajustes necesarios nuestro código ya está ajustado para este sistema y podemos proceder a realizar las medidas de los casos planteados anteriormente.

6.2.2. REFLEXIÓN MAMA VACÍA

El primer caso que analizaremos será la medida de los parámetros de reflexión cuando el modelo de mama se encuentra vacío. Esta medida es importante ya que tendrá dos finalidades, primeramente, como en los resto de casos para comprobar si mediante el uso de los parámetros S11 podemos determinar la distancia a la que se encuentran el borde del modelo y la antena de referencia (antena 1) y en segundo lugar, servir como "referencia" para los siguientes casos, pues para poder introducir los medios que hemos descrito en la introducción del capítulo el modelo de mama deberá estar presente y no debemos confundir las reflexiones que puede generar este elemento con las genere nuestro medio. A continuación, podemos apreciar nuestro sistema al introducir el modelo de mama:

Figura 6.5: Modelo de mama vacío

Nuevamente, haciendo uso de la cinta métrica determinamos que la distancia a la que se encuentran la antena y el borde del modelo es de 2.1 cm. El resultado que hemos obtenido empleando nuestro código ajustado ha sido el siguiente:

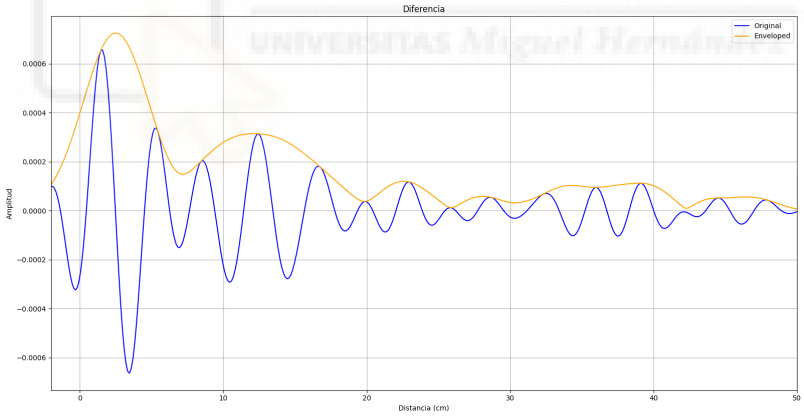
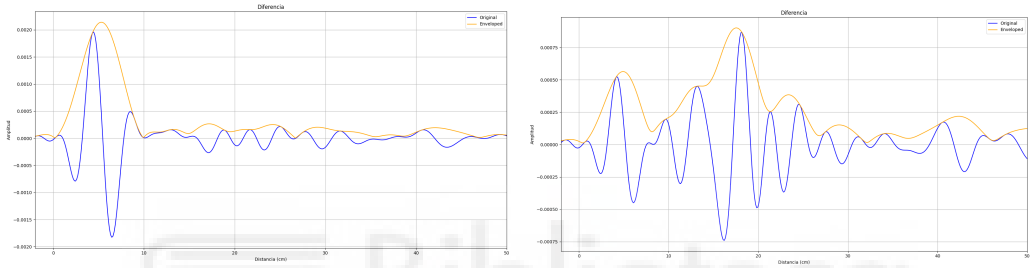


Figura 6.6: Posición modelo mama vacía

Esta vez la posición del máximo se obtiene a 2.461 cm. Lo que implicaría que estamos cometiendo un cierto error aunque no muy grande, pues este no llega a ser ni de medio centímetro y por tanto se puede considerar un error “esperable”.

6.2.3. REFLEXIÓN MAMA CON DISTINTOS MATERIALES

Una vez comprobado que somos capaces de detectar el modelo de mama introduciremos dentro de este botes que contienen líquidos con distinto valor de ϵr, concretamente agua (ϵr = 80) y aceite (ϵr = 5). Para el caso de medidas de reflexión este no será un factor que afectará demasiado pues lo que realmente nos interesa es la distancia a la que se produce el rebote de la señal con el borde del bote que contiene el líquido y por tanto, las propiedades del mismo no deberían ser muy significativas, pero en transmisión sí pues la señal deberá atravesar el bote. Los resultados que obtenemos se muestran a continuación:



(a) Usando agua (b) Usando aceite

Figura 6.7: Posición borde del bote

En la primera de las dos imágenes podemos ver el resultado obtenido en el caso de poner agua dentro del bote, se puede apreciar únicamente un máximo y el resto de la señal muy atenuada, esto se debe a que el agua absorbe mucha energía (su ϵr es bastante alta, sobre 80), especialmente a altas frecuencias (región de microondas). Este máximo se encuentra aproximadamente sobre los 4.9 cm, que se pueden corresponder con los 4.35 cm que obtenemos con la ayuda del metro y teóricamente. Por otra parte, en la segunda imagen podemos ver el resultado obtenido al poner aceite del envase. Se pueden apreciar dos máximos, el primero está situado a 4.9 cm y se corresponde con el borde del bote, mientas que el segundo se encuentra sobre los 17.5 cm y este se podría deber un elemento de nuestro sistema que no ha sido eliminado, como podría ser la antena situada al otro extremo, pues como podremos ver en el siguiente apartado la distancia a la que se encontraría analíticamente (ya que físicamente no puede ser superior de 13.2 cm) sería en torno a los 19.2 cm y podría entrar en el margen de error de nuestro sistema.

Una vez comprobado que podemos realizar las medidas en reflexión en el sistema MICROBIO de manera correcta solo falta comprobar su comportamiento empleando el método de transmisión de pulsos para los mismos casos.

6.3. TRANSMISIÓN

6.3.1. TRANSMISIÓN MAMA VACÍA

Volveremos a colocar únicamente el modelo de mama vacío pero la toma de medidas no será tan directa como en el caso de reflexión, pues ahora la señal atravesará todo nuestro sistema, en este caso particular el sistema no cambia demasiado por lo que no será un factor que altere mucho los resultados, pero en los siguientes casos veremos la importancia que tiene. Con fines aclaratorios se ha diseñado un esquema de distancias a los distintos elementos que la señal que tiene que atravesar en el sistema, el cual se muestra a continuación:

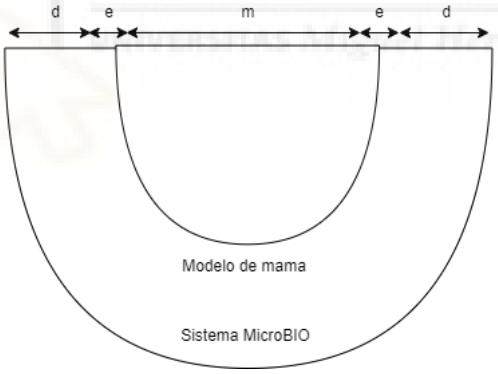


Figura 6.8: Explicación distancias modelo de mama vacío

La distancia “d” se corresponde con la distancia de 2.1 cm que hay entre cualquier antena y el borde del modelo de mama, la distancia “e” se corresponde con el grosor del modelo de mama que son 0.3 cm y finalmente la distancia ”m”que es el diámetro interno del modelo de mama, aproximadamente 8.4 cm. Como no todo el medio es aire y hay tramos cuya ϵr es mayor que la de éste lo que esperamos es que la señal se retrase y por tanto analíticamente es como si tuviese que recorrer más distancia:

$$distancia = 2 * d * \sqrt{\epsilon r_1} + 2 * e * \sqrt{\epsilon r_2} + m * \sqrt{\epsilon r_1} \tag{6.1}$$

Y por tanto, teóricamente obtenemos que:

$$distancia = 2 * 2{,}1 * \sqrt{1} + 2 * 0{,}3 * \sqrt{3} + 8{,}4 * \sqrt{1} = 13{,}64cm \tag{6.2}$$

Si hacemos uso de nuestro código obtenemos:

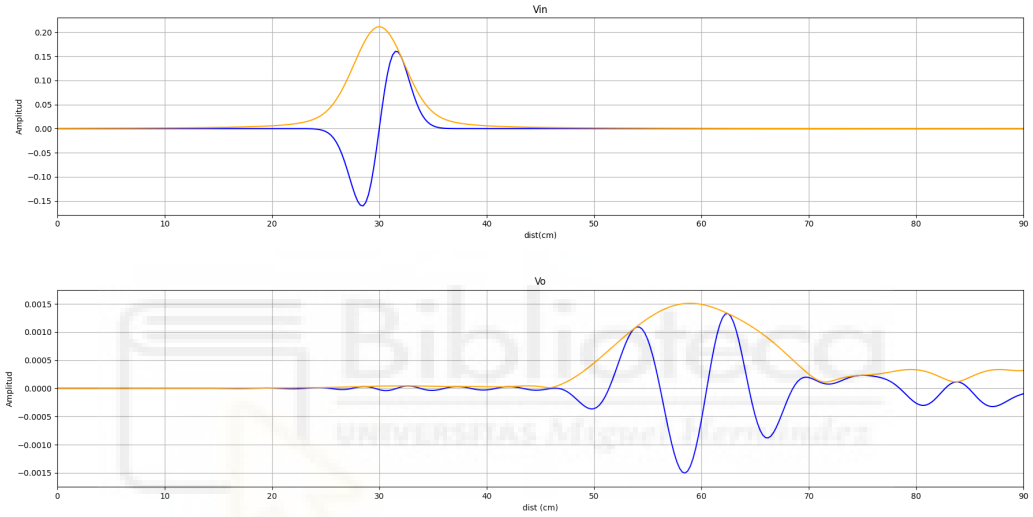


Figura 6.9: Transmisión mama vacía

Cuyo máximo del pulso de entrada está situado en 29.98 cm y el del pulso de salida en 58.6. La diferencia entre máximos hace una distancia de 28.62 cm a los cuales al igual que en el capítulo V debemos restarle su correspondiente exceso de distancia añadida por conectores y las propias antenas. Para este caso concreto el exceso se corresponde con 13.26 cm, al restarlo al valor obtenido anteriormente obtenemos una distancia de 15.36 cm, este resultado si bien presenta un cierto error es bastante bueno y probablemente podría mejorarse si empleásemos un método distinto a la envolvente para obtener la posición del máximo de Vo, pues como se puede apreciar esta comienza a no ser la opción más óptima debido a la gran amplitud que presenta en este caso.

6.3.2. TRANSMISIÓN MAMA CON DISTINTOS MATERIALES

Como ya hemos comentado en el apartado anterior para estos casos la toma de medidas no será tan sencilla pues la señal deberá atravesar todo nuestro sistema, el cual está formado por distintos materiales y por tanto la velocidad a la que se transmite la señal a través de estos medios no es la de la luz ($\epsilon r \neq 1$), este efecto ya ocurría en mucha menor medida en el caso anterior (solo los pequeños tramos que suponen el grosor del modelo de mama) y la ϵr de algunos materiales varía de forma más rápida y brusca que la de otros con la frecuencia. Particularmente, en los materiales dispersivos como puede ser el agua, las variaciones en la velocidad de propagación causarán que diferentes componentes frecuenciales del pulso viajen a diferentes velocidades, lo cual resultaría en una dispersión temporal del pulso. En el caso del aceite pasará en mucha menos medida y será prácticamente despreciable. De igual forma que en el anterior caso hemos diseñado un esquema orientativo:

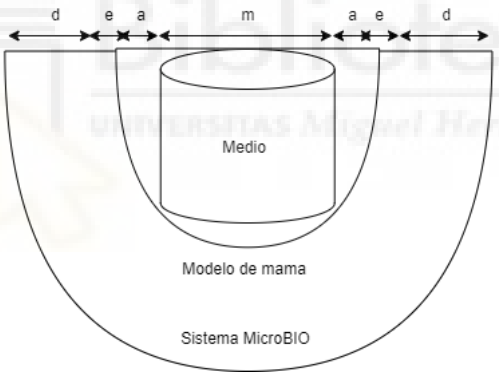


Figura 6.10: Explicación distancias

La distancia “d” se corresponde con la distancia de 2.1 cm que hay entre cualquier antena y el borde del modelo de mama, la distancia “e” se corresponde con el grosor del modelo de mama que son 0.3 cm, la distancia “a” se corresponde a la distancia de 1.95 cm que hay entre el borde interno del modelo de mama y el borde del bote que contiene el medio, finalmente la distancia ”m”que es el diámetro del bote que contiene al medio que son 4.5 cm. Tal y como hemos hecho en el apartado anterior calcularemos las nuevas distancias relativas a la transmisión por estos medios.

En el caso del aceite:

$$distancia = 2 * d * \sqrt{\epsilon r_1} + 2 * e * \sqrt{\epsilon r_2} + 2 * a * \sqrt{\epsilon r_1} + m * \sqrt{\epsilon r_3} \quad (6.3)$$

$$distancia = 2 * 2{,}1 * \sqrt{1} + 2 * 0{,}3 * \sqrt{3} + 2 * 1{,}95 * \sqrt{1} + 4{,}5 * \sqrt{5} = 19{,}20cm \quad (6.4)$$

Y para el caso del agua:

$$distancia = 2 * 2{,}1 * \sqrt{1} + 2 * 0{,}3 * \sqrt{3} + 2 * 1{,}95 * \sqrt{1} + 4{,}5 * \sqrt{80} = 49{,}39cm \quad (6.5)$$

Los resultados que obtenemos empleando nuestro código se muestran a continuación:

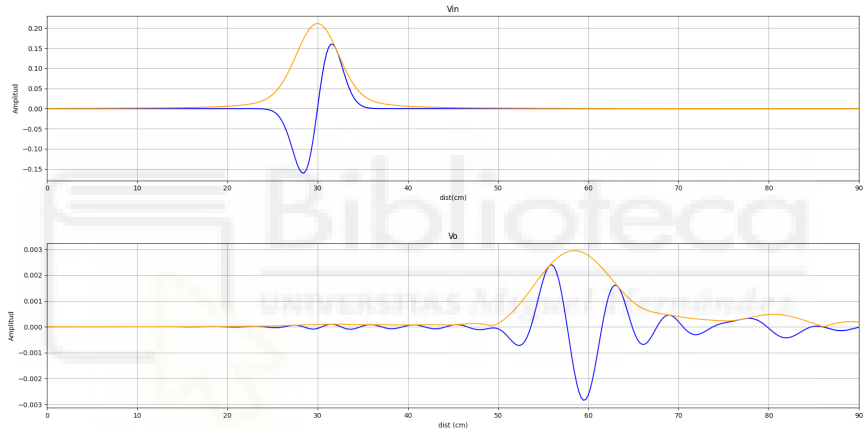


Figura 6.11: Transmisión con medio aceite

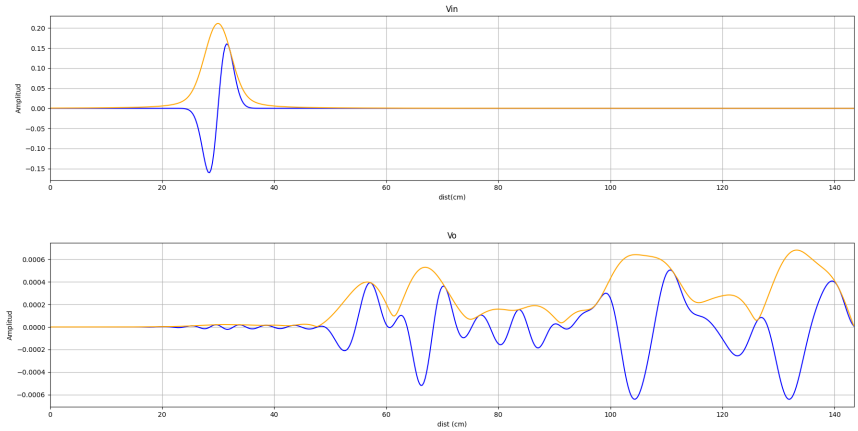
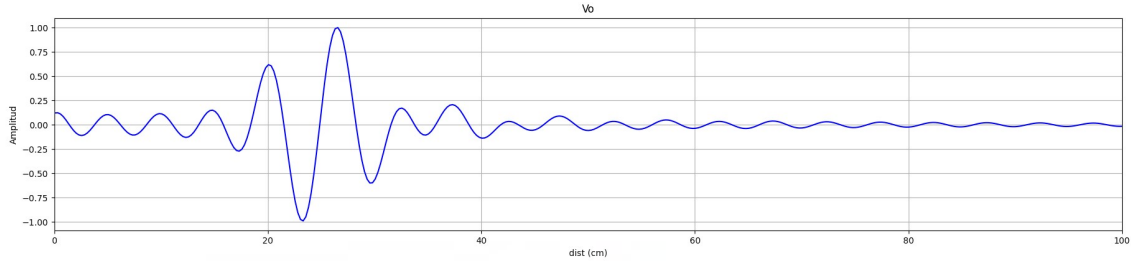
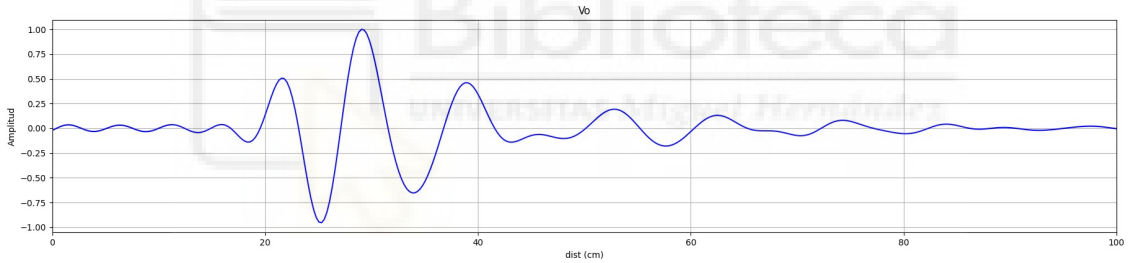


Figura 6.12: Transmisión con medio agua

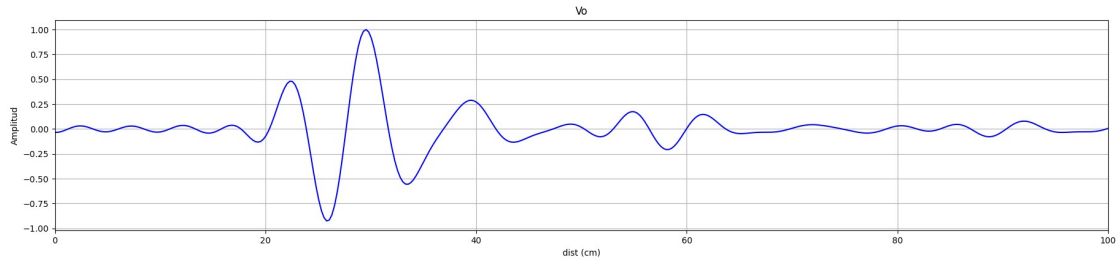
Vemos que para el caso del aceite, donde ϵr es pequeña podríamos llegar a sacar conclusiones sobre la distancia, pero en el del agua, donde la ϵr es mucho mayor nuestro método deja de ser eficaz.Por tanto, ya que estos casos no nos proporcionan los resultados que esperábamos y no disponemos de más materiales para hacer pruebas en casos intermedios llevaremos a cabo una serie de simulaciones en el programa HFSS junto a mi compañera del laboratorio Maya Moreno. En estas pruebas nuestro sistema diseñado consistirá en dos antenas separadas 20 cm, justo a la mitad de distancia entre estas se dispondrá un medio (correspondiente a 1 cm de la distancia) al cual podremos ir cambiando el valor de ϵr. Estos son los resultados obtenidos:



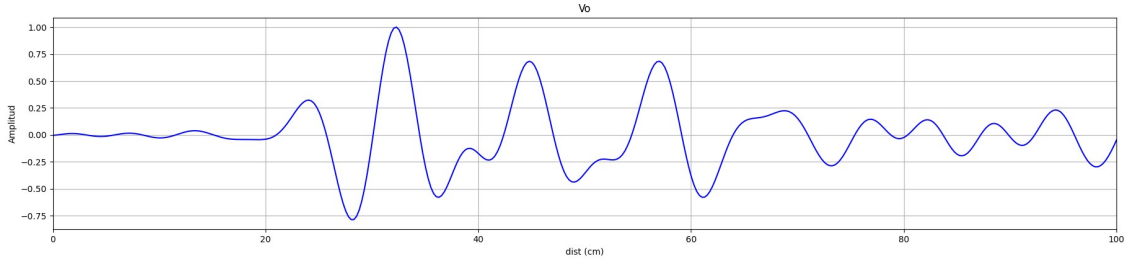
(a) ϵr = 1



(b) ϵr = 5

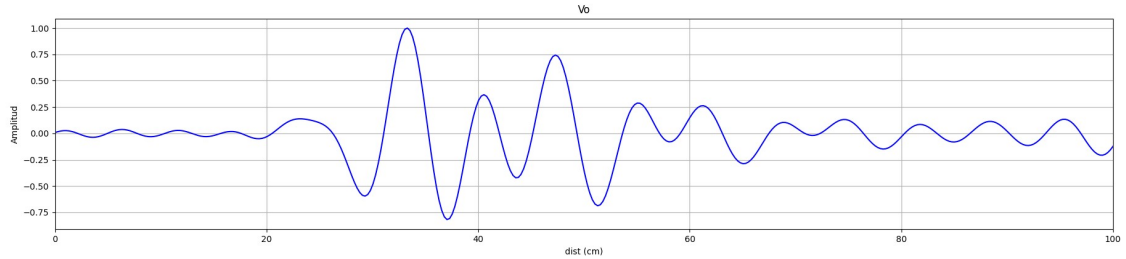


(c) ϵr = 10

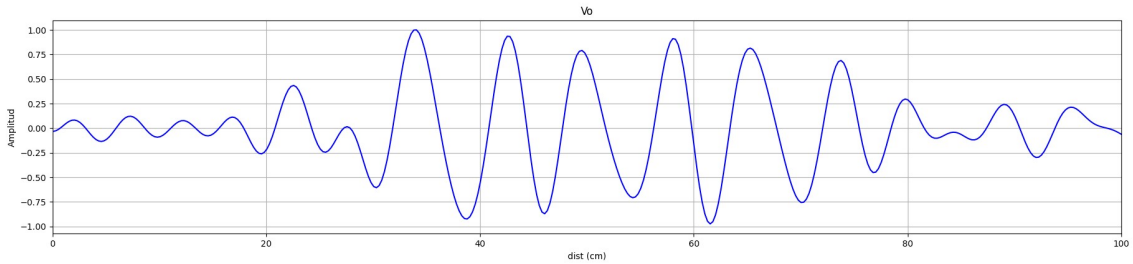


(d) ϵr = 40

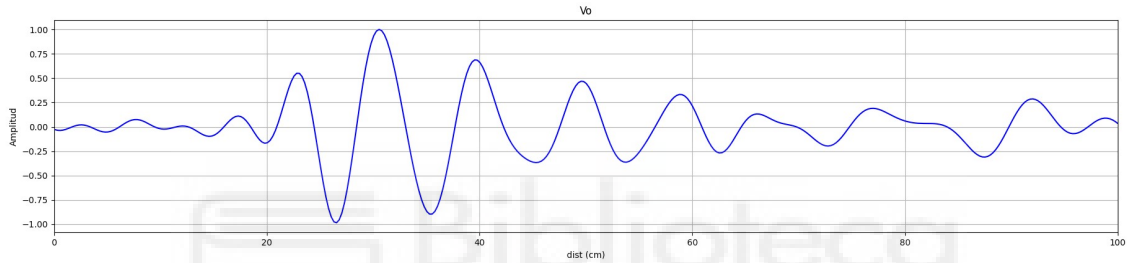
Figura 6.13: Posición borde del bote (Parte 1)



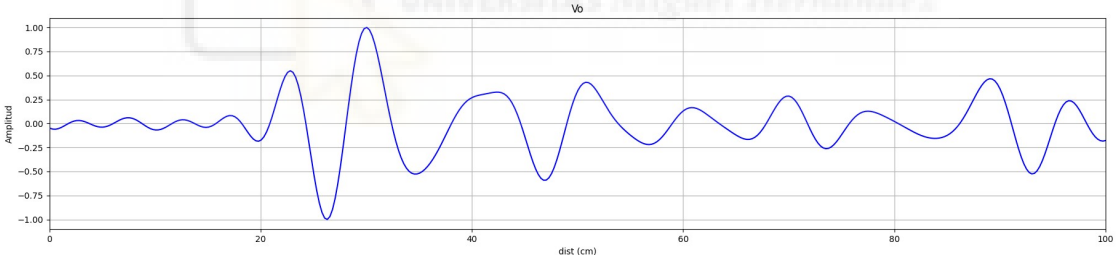
(a) $\epsilon r = 50$



(b) $\epsilon r = 60$



(c) $\epsilon r = 70$



(d) $\epsilon r = 80$

Figura 6.14: Posición borde del bote (Parte 2)

Nota: a diferencia de la mayoría de los casos planteados en anteriores apartados para la obtención de estos resultados ya se ha realizado la resta de posición del pulso inicial y del offset introducido por el sistema.

Como se puede ver, los métodos empleamos en capítulos anteriores no son válidos para todos los casos, en los primeros, (de ϵr 1 a 40) sí que podríamos llegar a determinar la distancia, pero a medida que aumenta el valor de ϵr (de 50 a 80) nos es imposible seguir empleando el método de la envolvente debido a la aparición de muchas interferencias, ajenas a nuestros sistema en la zona de interés y por tanto, nos es imposible determinar la posición del máximo del pulso. Lo más probable es que el método que estamos empleando para adaptar los datos a nuestro sistema sea únicamente válido cuando el valor de ϵr no es muy alto o que necesitemos realizar algún método de procesado adicional para poder obtener resultados válidos.

Como parece que nuestro sistema podría ser válido para valores de ϵr baja, realizamos un análisis de los cuatro primeros casos. Como hemos visto en anteriores casos si representamos los resultados en distancia obtenemos valores que están por encima de lo físicamente establecido lo cual no tendría sentido. Sin embargo, lo que sí que podemos realizar es un análisis en el dominio del tiempo y calcular los incrementos de tiempo que esperaríamos a medida que aumenta ϵr:

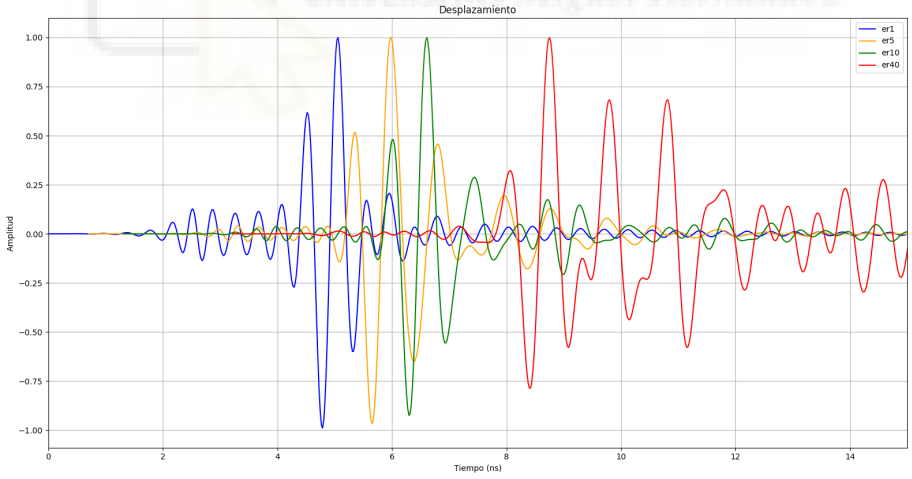


Figura 6.15: Casos válidos en el dominio del tiempo

Para llevar a cabo el análisis cogeremos un punto común de los 4 pulsos en el cual se aprecie el desplazamiento, este podría ser el máximo del primer lóbulo de subida que podemos encontrar en la figura 6.15 y que señalamos en la siguiente imagen:

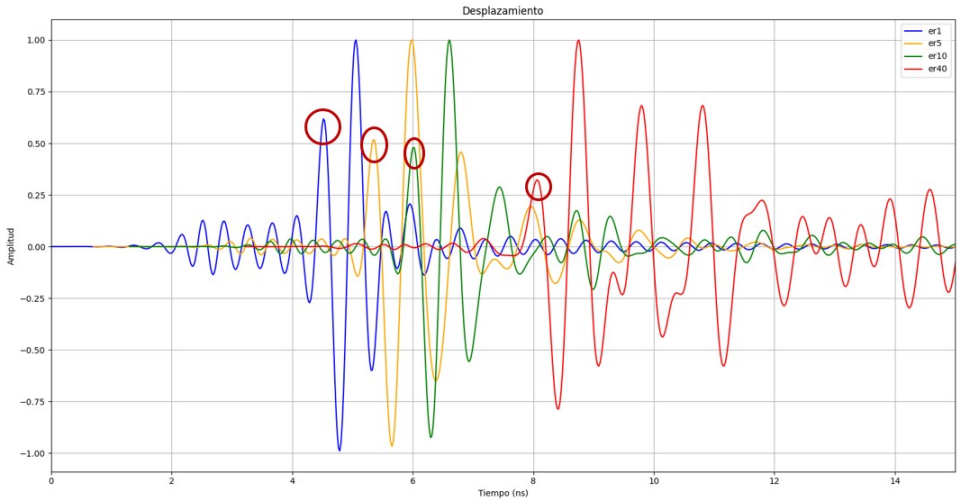


Figura 6.16: Zona de interés

Vemos que el lóbulo sí que se está desplazando a medida que aumenta el valor de ϵr, este cambio en distancia no era tan apreciable como ahora en el tiempo, ya que como hemos comentado anteriormente el sistema son dos antenas separadas 20 cm y el medio distinto del aire se encuentra a la mitad de distancia, cubriendo únicamente 1 cm de esos 20. Aunque el valor de ϵr aumente mucho el incremento que supone en distancia no es tan visible como en tiempo. Ahora bien, una vez demostrado que se desplaza falta ver si ese desplazamiento es el esperado. Para ello haremos debemos obtener los valores esperados y calculados, para los calculados bastará con hacer la diferencia tiempo de los puntos señalados en la figura 6.16, los cuales se encuentran en 4.53, 5.355, 6.015 y 8.068, y para los esperados haremos uso de la siguiente expresión:

$$tiempo = \frac{distancia}{v} = \frac{distancia}{c} * \sqrt{\epsilon r} \tag{6.6}$$

Los resultados que obtenemos se muestran a continuación:

	De ϵr = 1 a 5	**De ϵr = 5 a 10**	**De ϵr = 10 a 40**
Incremento_esperado (ns)	0.8243	0.609	2.116
Incremento_calculado (ns)	0.805	0.68	2.053

Tabla 6.1: Distancias medidas y calculadas simulación transmisión

Viendo los resultados obtenidos podemos afirmar que nuestro código es capaz de calcular de manera correcta las distancias, el tiempo que tarda la señal en propagarse o por lo menos los incrementos de tiempo en los casos en los que ϵr no es demasiado grande y que ha medida que esta va aumentando se nos dificulta mucho esta labor hasta que llega un punto que nos es imposible.

7. CONCLUSIONES Y LÍNEAS FUTURAS

Para dar fin a este trabajo fin de grado extraeremos las conclusiones a las que hemos llegado con la realización del proyecto y estableceremos las limitaciones o problemas que hemos encontrado durante la realización del mismo, dando lugar así líneas futuras de los aspectos que se podrían mejorar con el fin de obtener unos mejores resultados.

7.1. CONCLUSIONES

Las principales conclusiones que podemos sacar tras la realización de este proyecto es que hemos conseguido llevar a cabo un análisis de pulsos tanto en reflexión como en transmisión de forma exitosa sobre casos sencillos y que hemos podido realizar un buen análisis sobre la reflexión y transmisión en un caso de mayor complejidad (sistema MICROBIO) siendo este último caso el que tiene un mayor margen de mejora.

En el laboratorio montamos diferentes set-ups en los que empleamos diferentes tipos de antenas y elementos para llevar a cabo mediciones de los parámetros S de estas antenas, los cuales nos aportan información relacionada a la transmisión y reflexión de las señales. Mediante el empleo del lenguaje Python y el entorno Visual Studio Code llevamos a cabo el procesado de los datos y representación de los resultados. En cada capítulo se llevo a cabo una introducción en la que explicábamos los propósitos que se tenían en el mismo acompañado de un diagrama de flujo orientativo del código que íbamos a emplear. Con el fin de realizar los ajustes necesarios en cada caso realizamos unas simulaciones de casos ideales de gran sencillez gracias al software de ADS. Posterior a esto analizamos los resultados que obteníamos al emplear el código (ya ajustado) a los casos que medimos del laboratorio. Para estos casos obtuvimos muy buenos resultados tanto para reflexión como para transmisión y por tanto quisimos comprobar los resultados que podíamos obtener al someter nuestros códigos a casos de mayor complejidad.

Tras realizar las correcciones necesarias a los códigos los sometimos a diferentes escenarios empleando el sistema de imagen médica por microondas "MICROOBIO" y distintos elementos presentes en el laboratorio. Los resultados que obtuvimos en el caso de reflexión fueron bastante buenos consiguiendo errores relativamente bajos. Por otra parte, para el caso de transmisión los resultados empeoraron un poco, si bien somos capaces de realizar un correcto análisis cuando ϵr no era demasiado grande, para los casos donde esta condición no se cumplía nos resultaba imposible.

7.2. LÍNEAS FUTURAS

Durante el desarrollo del proyecto hemos afrontado diversas situaciones, la mayoría de ellas han sido nuevas pues me estaba introduciendo en un campo completamente nuevo para mí como es el procesado de señal y estaba empleando un lenguaje de programación que tampoco había empleado nunca. Esto a ocasionado que haya ciertos aspectos a mejorar en un futuro. A continuación se describirán estas líneas futuras:

- Depuración del código de Python: cuando se empezó el proyecto no tenía conocimientos de este lenguaje de programación, a medida que fui desarrollando los distintos códigos me di cuenta de la necesidad de emplear una mayor cantidad de funciones o crear librerías personalizadas con las procesos y operaciones de utilidad para conseguir un código más legible y entendible para gente que lo quiera emplear para sus proyectos.

- Emplear mejores antenas: las antenas que hemos empleado fueron diseñadas para este tipo de funciones pero como cualquier otra antena tiene sus defectos. Empleando otros equipos podríamos ver si obtenemos mejores resultados.

- Realización de medidas en más materiales: para disponer de una mayor muestra de datos debemos aplicar nuestros procesos a la mayor cantidad de casos posibles, pero esta tarea no es sencilla pues debemos encontrar materiales que se adapten a nuestras necesidades tanto físicas como dieléctricas y por tanto no se han podido estudiar todos los casos deseados.

- Queda pendiente la búsqueda de más materiales que se puedan emplear en el laboratorio para tener una mayor muestra de medida y mejorar la forma de procesar los datos para los casos de transmisión cuando el valor de ϵr es muy grande.

BIBLIOGRAFÍA

[1] GEICAM. «El cáncer de mama en España: situación actual.» (2024), dirección: https://www.geicam.org/sala-de-prensa/el-cancer-de-mama-en-espana.

[2] ©. Mammowave, «The first ultra-high sensitive breast imaging device based on non-ionizing safe microwave.»

[3] S. Kwon y S. Lee, «Recent Advances in Microwave Imaging for Breast Cancer Detection,» *International Journal of Biomedical Imaging*, vol. 2016, T. K. Bera, ed., págs. 1-26, 2016, ISSN: 1687-4196. DOI: 10.1155/2016/5054912.

[4] M. Por y Kocak, «RM ponderada en T1 Resonancia magnética,» 2021.

[5] Nueva Escuela Mexicana. «Espectro Electromagnético.» (2023), dirección: https://nuevaescuelamexicana.sep.gob.mx/detalle-ficha/37100/.

[6] E. Á. Navarro, *Apuntes de Sistemas Electrónicos de Comunicaciones*, Apuntes de clase, Universidad Miguel Hernández, Asignatura: Sistemas Electrónicos de Comunicaciones, 2024.

[7] I. Roig, J. Gosálbez, C. Ramón, M. Ricós y L. Vergara Domínguez, *Señales y sistemas. Teoría y pronlemas*, Apuntes de clase, Universidad Politécnica de Valencia, Asignatura: Señales y sistemas, 2024.

[8] J. P. Dunsmore, *Handbook of microwave component measurements: with advanced VNA techniques*. Wiley, 2012.

[9] A. V. Oppenheim, *Discrete-time signal processing*. Pearson Education India, 1999.

[10] Televés. «Guia de producto 2024 y Catálogo Do It Yourself.» (2024), dirección: https://www.televes.com/es/catalogos.

[11] José Alberto Bava y Auerlio Juan Sanz. «Amplificadores - Capítulo 3.» (1999), dirección: https://catedra.ing.unlp.edu.ar/electrotecnia/sistcom/Amplificadores/Capitulo3.pdf.

[12] Daniel R. McMahill. «Microstrip analysis/synthesis calculator.» (2010), dirección: https://mcalc.sourceforge.net.

[13] Tamoghna Das. «Create a gauss pulse using scipy.signal gausspulse.» (2023), dirección: https://www.tutorialspoint.com/create-a-gauss-pulse-using-scipy-signal-gausspulse.

[14] I. S. Gradštejn, I. M. Ryzhik, D. Zwillinger y V. H. Moll, eds., *Table of integrals, series, and products*, Eighth edition. Waltham, MA: Academic Press, 2015, 1133 págs., Includes bibliographical references and indexes. - Print version record, ISBN: 0123849349.

ANEXOS

A lo largo del trabajo hemos hecho uso de distintos códigos según el capítulo en el que nos encontrábamos y el tipo de medida que teníamos que llevar a cabo. Debido a que se emplearon distintos códigos, en cada ocasión se explicó el procedimiento seguido (mediante el diagrama de flujo) y se presentaron los aspectos más relevantes de cada código de forma resumida. Los siguientes cuatro anexos recogen todos los códigos completos que hemos empleado para llevar a cabo el procesado de los datos de cada caso, desde la generación de pulsos y derivadas, el análisis de reflexión de pulsos, el análisis de transmisión de pulsos y las medidas en el sistema MICROBIO.

A. ANEXO I: GENERACIÓN PULSOS Y DERIVADAS

Este anexo servirá como guía para ver como se han generado todos los pulsos y derivadas que se estudiaron y emplearon a lo largo del proyecto. Particularmente, con la ayuda de este código podremos generar las gráficas 3.13, 3.14, 3.15 y 3.16, correspondientes al apartado 3.9. A continuación, se describirá por una parte, los datos empleados para la generación de cada pulso y derivada (Amplitud, to, tau) y por otra su generación en si, pues los pulsos gaussianos los podemos generar directamente pero para generar las derivadas deberemos usar las expresiones de los polinomios de Hermite, las cuales fueron planteadas en la figura 3.12.

Cabe destacar que aunque los códigos que se muestran en los Anexos aparecen separados por partes es únicamente con fines comprensivos y de claridez, para que cualquiera de los códigos funcione correctamente debe ser usado en su plenitud.

```
    import os
    import skrf as rf
    import numpy as np
    from math import sqrt
    from scipy import signal
    from US_SSP import GaussPulseFilter
    import matplotlib.pyplot as plt
    import MW_ToolBox as MWF
    import US_Functions as US
    import sys

    sys.path.append('C:\\Users\\luvaa\\AppData\\Roaming\\
    Python\\Python312\\site-packages')

    # Leemos los datos de referencia
    filename = os.path.join(os.path.expanduser("~"), "Desktop",
        ↪ "Practicas_Internas", "Medidas_Laboratorio","pruebas",
        ↪ "Transmision", "15_cm.s2p")

    network = rf.Network(filename)
    frecuencia = network.f

    c = 299792458.
    Er= 1
    speed = c / np.sqrt(Er)
    LenScan = 5001

    Nplanchas = 1
    Nantenas = 1

    Fs = 120e9  # Nueva frecuencia de muestreo de 0 Hz a 120 GHz
    DF = frecuencia[1]-frecuencia[0]
    new_frequency = np.linspace(0, Fs, len(frecuencia))
    F_axis = np.arange(0, Fs, DF)
    nfft = len(F_axis)
    t_axis = np.arange(nfft) / Fs


    Amp = 50 * 10**8
    tau = 60 * 10**(-12)
    paso = tau / 100
    tin = 7 * tau
    Pulse = MWF.makePulse(Amp, tau, tin, nfft, Fs)
    Pulse_F = np.fft.fft(Pulse, nfft)


```

Código A.1: Lectura de valores, asignación de constantes, creación nuevo eje de frecuencias

```
    # Pulso gaussiano modificado
    Pulse_G = signal.gausspulse(t_axis, fc=3e9, bw=0.5, bwr=-6,
        ↪ tpr=-60)
    Pulse_GF = np.fft.fft(Pulse_G, nfft)  # Pulse spectra

    #Datos pulso
    tau = 300* 10**(-12)
    Amp = 50 * 10**8
    to = 1*1e-9
    sigma = np.sqrt(tau**2/2)
    #Datos primera derivada del pulso
    tau_1 = 77.5 * 10**(-12)
    Amp_1 = 50 * 10**8
    to_1 = 1*1e-9
    sigma_1 = np.sqrt(tau_1**2/2)
    #Datos segunda derivada del pulso
    tau_2 = 107.5 * 10**(-12)
    Amp_2 = 50 * 10**8
    to_2 = 1*1e-9
    sigma_2 = np.sqrt(tau_2**2/2)
    #Datos tercera derivada del pulso
    tau_3 = 130 * 10**(-12)
    Amp_3 = 50 * 10**8
    to_3 = 1*1e-9
    sigma_3 = np.sqrt(tau_3**2/2)
    #Datos cuarta derivada del pulso
    tau_4 = 152.5 * 10**(-12)
    Amp_4 = 50 * 10**8
    to_4 = 1*1e-9
    sigma_4 = np.sqrt(tau_4**2/2)
    #Datos quinta derivada del pulso
    tau_5 = 170 * 10**(-12)
    Amp_5 = 50 * 10**8
    to_5 = 1*1e-9
    sigma_5 = np.sqrt(tau_5**2/2)
    #Datos sexta derivada del pulso
    tau_6 = 185 * 10**(-12)
    Amp_6 = 50 * 10**8
    to_6 = 1*1e-9
    sigma_6 = np.sqrt(tau_6**2/2)
    #Datos septima derivada del pulso
    tau_7 = 200 * 10**(-12)
    Amp_7 = 50 * 10**8
    to_7 = 1*1e-9
    sigma_7 = np.sqrt(tau_7**2/2)

```

Código A.2: Datos para todas derivadas de los pulsos

Al generar las distintas derivadas de un pulso debemos ajustar sus valores con el fin de que estén centradas en el punto de interés, nosotros buscamos que estén centradas en 3 GHz, probando distintos valores los mejores resultados se obtuvieron con los mostrados en el código anterior.

Una vez establecemos los valores generamos un pulso por derivada y su TF.

```
    #Pulso gaussiano
    Pulse = Amp*np.exp(-((t_axis-to)**2)/(2*(sigma**2)))
    Pulse_F = np.fft.fft(Pulse)

    Pulse_1 = Amp_1*np.exp(-((t_axis-to_1)**2)/(2*(sigma_1**2)))
    Pulse_1_F = np.fft.fft(Pulse_1)

    Pulse_2 = Amp_2*np.exp(-((t_axis-to_2)**2)/(2*(sigma_2**2)))
    Pulse_2_F = np.fft.fft(Pulse_2)

    Pulse_3 = Amp_3*np.exp(-((t_axis-to_3)**2)/(2*(sigma_3**2)))
    Pulse_3_F = np.fft.fft(Pulse)

    Pulse_4 = Amp_4*np.exp(-((t_axis-to_4)**2)/(2*(sigma_4**2)))
    Pulse_4_F = np.fft.fft(Pulse_1)

    Pulse_5 = Amp_5*np.exp(-((t_axis-to_5)**2)/(2*(sigma_5**2)))
    Pulse_5_F = np.fft.fft(Pulse_2)

    Pulse_6 = Amp_6*np.exp(-((t_axis-to_6)**2)/(2*(sigma_6**2)))
    Pulse_6_F = np.fft.fft(Pulse_1)

    Pulse_7 = Amp_7*np.exp(-((t_axis-to_7)**2)/(2*(sigma_7**2)))
    Pulse_7_F = np.fft.fft(Pulse_2)

    #Datos pulso A
    tau_A = 75 * 10**(-12)
    Amp_A = 50 * 10**8
    to_A = 1*1e-9
    sigma_A = np.sqrt(tau_A**2/2)
    #Pulso A
    Pulse_A =
        ↪ Amp*(t_axis-to_A)*np.exp(-((t_axis-to_A)**2)/(tau_A**2))
    Pulse_AF = np.fft.fft(Pulse_A)
    Pulse_norm_A = Pulse_A/np.max(Pulse_A)
    Pulse_norm_AF = np.fft.fft(Pulse_norm_A)
    Pulse_norm_A_F =
        ↪ np.abs(Pulse_norm_AF)/np.max(np.abs(Pulse_norm_AF))
    #Datos pulso E
    tau_E = 400 * 10**(-12)
    Amp_E = 50 * 10**8
    to_E = 1*1e-9
    sigma_E = np.sqrt(tau_E**2/2)
    fo = 3*1e9
    #Pulso E
    Pulse_E = (Amp_E*np.exp(-(t_axis-to_E)**2/(tau_E)**2))*np.cos(
    (2*np.pi*fo)*(t_axis-to_E))
    Pulse_EF = np.fft.fft(Pulse_E)
    Pulse_norm_E = Pulse_E/np.max(Pulse_E)
    Pulse_norm_EF = np.fft.fft(Pulse_norm_E)
    Pulse_norm_E_F =
        ↪ np.abs(Pulse_norm_EF)/np.max(np.abs(Pulse_norm_EF))

```

Código A.3: Generación de todos los pulsos y sus transformadas de Fourier

Obtenemos la derivada de cada pulso mediante los polinomios de Hermite.

```
    #Generamos las derivadas del pulso Gaussiano

    n1 = 1
    n2 = 2
    n3 = 3
    n4 = 4
    n5 = 5
    n6 = 6
    n7 = 7

    H1 = 2*(t_axis-to_1)/(np.sqrt(2)*sigma_1)
    H2 = 4*(((t_axis-to_2)/(np.sqrt(2)*sigma_2))**2)-2
    H3 = 8*(((t_axis-to_3)/(np.sqrt(2)*sigma_3))**3)-12*
    ((t_axis-to_3)/(np.sqrt(2)*sigma_3))
    H4 = 16*(((t_axis-to_4)/(np.sqrt(2)*sigma_4))**4)-48*
    (((t_axis-to_4)/((np.sqrt(2)*sigma_4)))**2)+12
    H5 = 32*(((t_axis-to_5)/((np.sqrt(2)*sigma_5)))**5)-160*
    (((t_axis-to_5)/((np.sqrt(2)*sigma_5)))**3)+120*
    ((t_axis-to_5)/((np.sqrt(2)*sigma_5)))
    H6 = 64*(((t_axis-to_6)/((np.sqrt(2)*sigma_6)))**6)-480*
    (((t_axis-to_6)/((np.sqrt(2)*sigma_6)))**4)+720*
    (((t_axis-to_6)/((np.sqrt(2)*sigma_6)))**2)-120
    H7 = 128*(((t_axis-to_7)/((np.sqrt(2)*sigma_7)))**7)-1344*
    (((t_axis-to_7)/((np.sqrt(2)*sigma_7)))**5)+3360*
    (((t_axis-to_7)/((np.sqrt(2)*sigma_7)))**3)-1680*
    ((t_axis-to_7)/((np.sqrt(2)*sigma_7)))

    Primera_derivada_num =
        ↪ ((-1)**n1)*(1/(np.sqrt(2)*sigma_1)**2)*H1*Pulse_1
    Primera_derivada_num_F = np.fft.fft(Primera_derivada_num,nfft)
    Primera_derivada_num_norm =
        ↪ Primera_derivada_num/np.max(Primera_derivada_num)
    Primera_derivada_num_norm_F =
        ↪ np.abs(Primera_derivada_num_F)/np.max(
    np.abs(Primera_derivada_num_F))

    Segunda_derivada_num =
        ↪ ((-1)**n2)*(1/(np.sqrt(2)*sigma_2)**2)*H2*Pulse_2
    Segunda_derivada_num_F = np.fft.fft(Segunda_derivada_num,nfft)
    Segunda_derivada_num_norm =
        ↪ Segunda_derivada_num/np.max(np.abs(Segunda_derivada_num))
    Segunda_derivada_num_norm_F =
        ↪ np.abs(Segunda_derivada_num_F)/np.max(
    np.abs(Segunda_derivada_num_F))

    Tercera_derivada_num =
        ↪ ((-1)**n3)*(1/(np.sqrt(2)*sigma_3)**2)*H3*Pulse_3
    Tercera_derivada_num_F = np.fft.fft(Tercera_derivada_num,nfft)
    Tercera_derivada_num_norm =
        ↪ Tercera_derivada_num/np.max(Tercera_derivada_num)
    Tercera_derivada_num_norm_F =
        ↪ np.abs(Tercera_derivada_num_F)/np.max(
    np.abs(Tercera_derivada_num_F))

```

Código A.4: Generación de las tres primeras derivadas del pulso Gaussiano

```
    Cuarta_derivada_num =
        ↪ ((-1)**n4)*(1/(np.sqrt(2)*sigma_4)**2)*H4*Pulse_4
    Cuarta_derivada_num_F = np.fft.fft(Cuarta_derivada_num,nfft)
    Cuarta_derivada_num_norm =
        ↪ Cuarta_derivada_num/np.max(Cuarta_derivada_num)
    Cuarta_derivada_num_norm_F =
        ↪ np.abs(Cuarta_derivada_num_F)/np.max(
    np.abs(Cuarta_derivada_num_F))

    Quinta_derivada_num =
        ↪ ((-1)**n5)*(1/(np.sqrt(2)*sigma_5)**2)*H5*Pulse_5
    Quinta_derivada_num_F = np.fft.fft(Quinta_derivada_num,nfft)
    Quinta_derivada_num_norm =
        ↪ Quinta_derivada_num/np.max(Quinta_derivada_num)
    Quinta_derivada_num_norm_F =
        ↪ np.abs(Quinta_derivada_num_F)/np.max(
    np.abs(Quinta_derivada_num_F))

    Sexta_derivada_num =
        ↪ ((-1)**n6)*(1/(np.sqrt(2)*sigma_6)**2)*H6*Pulse_6
    Sexta_derivada_num_F = np.fft.fft(Sexta_derivada_num,nfft)
    Sexta_derivada_num_norm =
        ↪ Sexta_derivada_num/np.max(np.abs(Sexta_derivada_num))
    Sexta_derivada_num_norm_F =
        ↪ np.abs(Sexta_derivada_num_F)/np.max(
    np.abs(Sexta_derivada_num_F))

    Septima_derivada_num =
        ↪ ((-1)**n7)*(1/(np.sqrt(2)*sigma_7)**2)*H7*Pulse_7
    Septima_derivada_num_F = np.fft.fft(Septima_derivada_num,nfft)
    Septima_derivada_num_norm =
        ↪ Septima_derivada_num/np.max(Septima_derivada_num)
    Septima_derivada_num_norm_F =
        ↪ np.abs(Septima_derivada_num_F)/np.max(
    np.abs(Septima_derivada_num_F))

```

Código A.5: Generación de las siguientes cuatro derivadas

Una vez hemos generado todas las derivadas del pulso, lo único que falta es representarlas (con fines comparativos se representaron los derivadas normalizadas ya que se pueden apreciar mejor los resultados), en nuestro caso lo hemos hecho en el dominio del tiempo, tal y como se puede observar en el siguiente fragmento del código.

```
    #Representación de las 7 derivadas en el dominio del tiempo
    plt.figure(figsize=(10, 6))
    plt.plot(t_axis*1e9, Primera_derivada_num_norm, label='Primera
        ↪ derivada', color = 'blue')
    plt.plot(t_axis*1e9, Segunda_derivada_num_norm, label='Segunda
        ↪ derivada', color = 'red')
    plt.plot(t_axis*1e9, Tercera_derivada_num_norm, label='Tercera
        ↪ derivada', color = 'green')
    plt.plot(t_axis*1e9, Cuarta_derivada_num_norm, label='Cuarta
        ↪ derivada', color = 'orange')
    plt.xlabel('Time (ns)')
    plt.ylabel('Amplitud')
    plt.title('Derivadas 1-4 del pulso Gaussiano')
    plt.legend()
    plt.grid(True)
    plt.xlim(0,2)
    plt.show()


    plt.figure(figsize=(10, 6))
    plt.plot(t_axis*1e9, Quinta_derivada_num_norm, label='Quinta
        ↪ derivada', color = 'blue')
    plt.plot(t_axis*1e9, Sexta_derivada_num_norm, label='Sexta
        ↪ derivada', color = 'red')
    plt.plot(t_axis*1e9, Septima_derivada_num_norm, label='Septima
        ↪ derivada', color = 'green')
    plt.xlabel('Time (ns)')
    plt.ylabel('Amplitud')
    plt.title('Derivadas 5-7 del pulso Gaussiano')
    plt.legend()
    plt.grid(True)
    plt.xlim(0,2)
    plt.show()

```

Código A.6: Representación de las derivadas en el dominio del tiempo

Pero también somos capaces de representar las derivadas en el dominio de la frecuencia, tanto normalizadas como no normalizadas, como se puede apreciar en el siguiente fragmento de código.

```
    #Representamos las derivadas en el dominio de la frecuencia
    plt.figure(figsize=(10, 6))
    plt.plot(F_axis/1e9, np.abs(Primera_derivada_num_F),
        ↪ label='Primera derivada', color = 'blue')
    plt.plot(F_axis/1e9, np.abs(Segunda_derivada_num_F),
        ↪ label='Segunda derivada', color = 'red')
    plt.plot(F_axis/1e9, np.abs(Tercera_derivada_num_F),
        ↪ label='Tercera derivada', color = 'green')
    plt.plot(F_axis/1e9, np.abs(Cuarta_derivada_num_F),
        ↪ label='Cuarta derivada', color = 'orange')
    plt.plot(F_axis/1e9, np.abs(Quinta_derivada_num_F),
        ↪ label='Quinta derivada', color = 'black')
    plt.plot(F_axis/1e9, np.abs(Sexta_derivada_num_F),
        ↪ label='Sexta derivada', color = 'purple')
    plt.plot(F_axis/1e9, np.abs(Septima_derivada_num_F),
        ↪ label='Septima derivada', color = 'brown')
    plt.plot(F_axis/1e9, np.abs(Pulse_AF), label='Pulso Gaussiano
        ↪ 1', color = 'cyan')
    plt.plot(F_axis/1e9, np.abs(Pulse_EF), label='Pulso Gaussiano
        ↪ 2', color = 'pink')
    plt.xlabel('Frecuencia (GHz)')
    plt.ylabel('Amplitud')
    plt.title('FFT pulso Gaussiano')
    plt.legend()
    plt.grid(True)
    plt.xlim(0,10)
    plt.show()

    #Derivadas normalizadas en el dominio de la frecuencia
    plt.figure(figsize=(10, 6))
    plt.plot(F_axis/1e9, np.abs(Primera_derivada_num_norm_F),
        ↪ label='Primera derivada', color = 'blue')
    plt.plot(F_axis/1e9, np.abs(Segunda_derivada_num_norm_F),
        ↪ label='Segunda derivada', color = 'red')
    plt.plot(F_axis/1e9, np.abs(Tercera_derivada_num_norm_F),
        ↪ label='Tercera derivada', color = 'green')
    plt.plot(F_axis/1e9, np.abs(Cuarta_derivada_num_norm_F),
        ↪ label='Cuarta derivada', color = 'orange')
    plt.plot(F_axis/1e9, np.abs(Quinta_derivada_num_norm_F),
        ↪ label='Quinta derivada', color = 'black')
    plt.plot(F_axis/1e9, np.abs(Sexta_derivada_num_norm_F),
        ↪ label='Sexta derivada', color = 'purple')
    plt.plot(F_axis/1e9, np.abs(Septima_derivada_num_norm_F),
        ↪ label='Septima derivada', color = 'brown')
    plt.plot(F_axis/1e9, np.abs(Pulse_norm_A_F), label='Pulso
        ↪ Gaussiano 1', color = 'cyan')
    plt.plot(F_axis/1e9, np.abs(Pulse_norm_E_F), label='Pulso
        ↪ Gaussiano 2', color = 'pink')
    plt.xlabel('Frecuencia (GHz)')
    plt.ylabel('Amplitud')
    plt.title('FFT pulso Gaussiano normalizada')
    plt.legend()
    plt.grid(True)
    plt.xlim(0,10)
    plt.show()
```

Código A.7: Representación de las derivadas en el dominio de la frecuencia

B. ANEXO II: CÓDIGO REFLEXIÓN DE PULSOS

Este anexo sirve como aclaración de todos los procesos seguidos en la parte de simulación del capítulo III: análisis de pulsos en reflexión. En este anexo aportaremos todo el código empleado durante esta fase del trabajo, diferencias del código entre caso reales y también explicaciones de funciones que hemos desarrollado para la medición de pulsos en reflexión. Algunas de las funciones empleadas forman parte de librerías personalizadas aportadas por compañeros del equipo de investigación del laboratorio y por tanto no son públicas, pero se explicará su funcionamiento y utilidad. Una vez aclarado el contenido del anexo pasaremos a hablar de la librerías empleadas, en primer lugar tenemos las que nos proporciona Python, como son: skrf (la cual nos proporciona un objeto para una red de microondas de N puertos), numpy (nos permite crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas), matplotlib (para la generación de gráficos en dos dimensiones, a partir de datos contenidos en listas o arrays) y scipy (para cálculos matemáticos complejos o problemas científicos, como por ejemplo generación de pulsos gaussianos), por otra parte tenemos las librerias perosnalizadas: MW_Toolbox (incluye funciones para generar el espectro de las señales), US_Functions (para realizar el enventanado de las señales) y US_SSP.

De nuevo resaltar que aunque el código esté separado por partes es únicamente con fines de comprensión del mismo.

En primer lugar se presentará el código empleado en el primer caso reaal “empleando antena Vivaldi”.

En primer lugar, importamos las librerías descritas al principio del anexo. Hacemos uso de la función network incluida en la librería skrf con el fin de obtener los datos frecuenciales de los parámetros S11 de los ficheros s1p. Establecemos constantes básicas como son "c" la velocidad de la luz y "Er" la permitividad dieléctrica del medio (aire) o el número de puntos "LenScan". Inicializamos a cero unas variables y les pasamos los valores de los parámetros S medidos.

```
    import os
    import skrf as rf
    import numpy as np
    import matplotlib.pyplot as plt
    import MW_ToolBox as MWF
    import US_Functions as US
    from scipy import signal
    from US_SSP import GaussPulseFilter

    # Leemos los datos de referencia
    filename = os.path.join(os.path.expanduser("\~"), "Desktop",
        ↪ "Practicas_Internas", "Medidas_Laboratorio","pruebas",
        ↪ "Reflexion","Dist_Panel", "referencia.s1p")
    network = rf.Network(filename)
    frecuencia = network.f

    #Leemos los datos plancha a una distancia
    filename_2 = os.path.join(os.path.expanduser("~"), "Desktop",
        ↪ "Practicas_Internas", "Medidas_Laboratorio", "pruebas",
        ↪ "Reflexion","Dist_Panel", "dist_70.s1p")
    network_2 = rf.Network(filename_2)

    #Parametros S referencia
    #S11 = np.zeros(5001)
    S11 = network.s[:, 0, 0]

    #Datos 2
    S11\_2 = network\_2.s[:, 0, 0]
    c = 299792458.
    Er= 1, speed = c / np.sqrt(Er)
    LenScan = 5001, Nplanchas = 1, Nantenas = 1

    #Inicalizamos y pasamos los valores
    S11_ref = np.zeros((Nantenas, LenScan), dtype=complex)
    S11_plancha = np.zeros((Nantenas, LenScan),dtype=complex)
    S11_ref = S11
    S11_plancha = S11_2

```

Código B.8: Lectura de valores y asignación de constantes

Generamos un nuevo eje de frecuencias, el cual en vez de ir de 10 MHz a 6 GHz se expandirá interpolando entre 0 y 120 GHz. Generaremos un enventanado que servirá para suavizar los lóbulos de la señal cerca de sus puntos iniciales y finales, eliminará reflexiones parásitas y reducirá las posibles fugas espectrales en el dominio de la frecuencia al realizar transformadas de Fourier, mejorando así la precisión de nuestro sistema. Mediante las dos últimas líneas de este fragmento generaremos los espectros rellenos con ceros de simetría hermitiana de las señales.

```
1
2     #Creamos nuevo eje de frecuencias entre 0 y 120 GHz
3     Fs = 120e9  # Nueva frecuencia de muestreo
4     DF = frecuencia[1]-frecuencia[0]
5     new_frequency = np.linspace(0, Fs, len(frecuencia))
6     F_axis = np.arange(0, Fs, DF)
7     nfft = len(F_axis)
8
9     #Ventana tukey
10    MyWin = US.makeWindow(SortofWin='tukey', WinLen=LenScan,
          ↪ param1=0.25, param2=1, Span=0, Delay=0)
11
12    if np.mod(nfft,2)==1:
13    MyWin_ext = np.concatenate(( MyWin, np.zeros((int( (nfft+1)/2
          ↪ )-LenScan)) ))
14    MyWin = np.concatenate((MyWin_ext, np.flipud(MyWin\_ext[:-1])))
15    else:
16    MyWin_ext = np.concatenate(( MyWin,
          ↪ np.zeros((int(nfft/2)-LenScan)) ))
17    MyWin = np.concatenate((MyWin_ext, np.flipud(MyWin_ext)))
18
19    #Gerenamos espectro de las dos señales además de interpolar
          ↪ con zeros
20    S_ref = MWF.makeSpectrum(S11_ref, F_axis, frecuencia)
21    S_DataMatrix = MWF.makeSpectrum(np.squeeze(S11_plancha),
          ↪ F_axis, frecuencia)
22
```

Código B.9: Generación nuevo eje de frecuencias, aplicación de enventanado Tukey, interpolación de valores y generación de espectro

NOTA: La función "makeSpectrum" forma parte de las librerías personalizadas y se encarga de crear el espectro de la señal, expandiendo el espectro existente mediante relleno de ceros. Recibe como entradas una matriz 2D de valores complejos (espectros de antenas), el eje de frecuencia deseado con relleno de ceros y el eje de frecuencia de la señal adquirida, y retorna como salida el espectro relleno con ceros.

A continuación, mediante el empleo de una de las funciones de las que disponemos en nuestras librerías personaliza realizaremos un filtrado de las señales (una vez les aplicamos el enventanado) a las que posteriormente les realizaremos su transformada inversa de Fourier.

```
    # Pulso Gaussiano
    Amp = 50 * 10**8
    tau = 60 * 10**(-12)
    paso = tau / 100
    tin = 7 * tau
    Pulse = MWF.makePulse(Amp, tau, tin, nfft, Fs)
    Pulse_F = np.fft.fft(Pulse, nfft)

    # Pulso gaussiano modificado
    t_axis = np.arange(nfft) / Fs
    Pulse_G = signal.gausspulse(t_axis, fc=3e9, bw=0.5, bwr=-6,
        ↪ tpr=-60)
    Pulse_GF = np.fft.fft(Pulse_G, nfft)

    Flim = 10e9 # limite superior del eje de frecuencias para
        ↪ plotear

    #Realizamos un filtrado de las señales y hacemos las
        ↪ transformadas inversas
    ref_in_time, borrar = MWF.adaptedFilter(S_ref, Pulse,
        ↪ XCR=True, Aligned=True,
    Wiener=False)
    winref_in_time = np.real(np.fft.ifft(S_ref * MyWin))

    DataMatrix_in_time = np.zeros((Nplanchas, Nantenas, nfft))
    WinMatrix_in_time = np.zeros((Nplanchas, Nantenas, nfft))

    DataMatrix_in_time, borrar = MWF.adaptedFilter(S_DataMatrix,
        ↪ Pulse, XCR = True, Aligned = True, Wiener=False)
    WinMatrix_in_time=
        ↪ np.real(np.fft.ifft(np.squeeze(S_DataMatrix) * MyWin))

    Antenna = 0
    s_ref = np.roll(ref_in_time, Delay, axis=1)
    time_axis = dis_axis /Fs

```

Código B.10: Generación de pulso Gaussiano, filtrado de señales y transformadas inversas

NOTA: La función "adaptedFilter" forma parte de las librerías personalizadas y se encarga de filtrar las señales de entrada (como matriz) utilizando una forma de onda de pulso. Recibe como entrada una matriz 2D de valores complejos (espectro de antenas), matriz de flotación (forma de onda de pulso en el dominio del tiempo), un booleano que si es verdadero (predeterminado) usa correlación cruzada en el dominio

de la frecuencia, de lo contrario, producto en frecuencia (convolución), otro booleano que si es verdadero (predeterminado), alinea la forma de onda del pulso a cero para evitar retrasos y finalmente otro booleano que si es verdadero (predeterminado), usa la forma de onda de pulso como filtro de Wiener y devuelve una matriz 2D que es la señal filtrada en el dominio de la frecuencia y otra matriz 2D que es la señal filtrada en el dominio del tiempo.

```
    Delay = 5000
    s_ref = np.roll(ref\_in\_time, Delay, axis=1)
    ws_ref = np.roll(winref\_in\_time, Delay, axis=1)
    DataMatrix_in_time = np.roll(np.squeeze(DataMatrix\_in\_time),
        ↪ Delay, axis=0)
    WinMatrix_in_time = np.roll(np.squeeze(WinMatrix\_in\_time),
        ↪ Delay, axis=0)

    Max_ind = np.argmax(np.abs(s_ref[Antenna, :]))
    wMax_ind = np.argmax(np.abs(ws_ref[Antenna, :]))
    Max_ind_e = np.argmax(US.envelope(s_ref[Antenna, :]))

    #Resta de señales para eliminar parte común
    resta = s_ref[Antenna,:] - DataMatrix_in_time
    enveloped = np.abs(signal.hilbert(s_ref))
    enveloped_resta = np.abs(signal.hilbert(resta))

    plt.figure(figsize=(10,6))
    plt.plot(dis_axis/10-622, resta, label ='Original', color =
        ↪ 'blue')
    plt.plot(dis_axis/10-622, enveloped_resta, label ='Enveloped',
        ↪ color = 'orange')
    plt.xlabel('Distancia (cm)')
    plt.ylabel('Amplitud')
    plt.title('Diferencia')
    plt.legend()
    plt.grid(True)
    plt.xlim(-10,200)
    plt.show()

    plt.figure(figsize=(10,6))
    plt.plot(t_axis*1e9-40.891, resta, label ='Original', color =
        ↪ 'blue')
    plt.plot(t_axis*1e9-40.891, enveloped_resta, label
        ↪ ='Enveloped', color = 'orange')
    plt.xlabel('Tiempo (ns)')
    plt.ylabel('Amplitud')
    plt.title('Diferencia')
    plt.legend()
    plt.grid(True)
    plt.xlim(-1,4)
    plt.show()

```

Código B.11: Resta de señales para eliminar ruido, envolvente mediante transformada de Hilbert y ploteo final

Como ya hemos comentado en el proyecto el código empleado para el segundo caso "EMPLEANDO ANTENAS DE BANDA ANCHA" es prácticamente igual que el descrito para el primer caso, con la variación de que en este estamos trabajando tanto con los parámetros S11 como S22 y por tanto deberemos realizar algunos cambios en el mismo para poder procesar todos los datos que emplearemos.

En primer lugar, al tener dos tipos de datos (S11 y S22) debemos disponer del doble de variables.

```
1
2     #Leemos los datos de referencia
3     filename = os.path.join(os.path.expanduser("~"), "Desktop",
          ↪ "Practicas_Internas", "Medidas_Laboratorio",
          ↪ "medidas_luis_2", "Face2Face","Nuevas","referencia.s2p")
4     network = rf.Network(filename)
5     frecuencia = network.f
6
7     #Leemos los datos de la plancha a una distancia
8     filename_2 = os.path.join(os.path.expanduser("~"), "Desktop",
          ↪ "Practicas_Internas", "Medidas_Laboratorio",
          ↪ "medidas_luis_2", "Face2Face","Nuevas","30_cm.s2p")
9     network_2 = rf.Network(filename_2)
10
11    #Parámetros S referencia
12    S11 = network.s[:, 0, 0]
13    S11_db = 20 * np.log10(np.abs(S11))
14    S22 = network.s[:, 1, 1]
15    S22_db = 20 * np.log10(np.abs(S22))
16
17    #Datos antena
18    S11_2 = network_2.s[:, 0, 0]
19    S11_2_db = 20 * np.log10(np.abs(S11_2))
20    S22_2 = network_2.s[:, 1, 1]
21    S22_2_db = 20 * np.log10(np.abs(S22_2))
22
23    c = 299792458. # speed of light in m/s
24    Er= 1, speed = c / np.sqrt(Er)
25    LenScan = 5001, Nplanchas = 1, Nantenas = 1
26
27    #Inicalizamos y pasamos los valores
28    S11_ref = np.zeros((Nantenas, LenScan), dtype=complex)
29    S22_ref = np.zeros((Nantenas, LenScan), dtype=complex)
30    S11_antena = np.zeros((Nantenas, LenScan),dtype=complex)
31    S22_antena = np.zeros((Nantenas, LenScan),dtype=complex)
32    S11_ref = S11
33    S22_ref = S22
34    S11_antena = S11_2
35    S22_antena = S22_2
36
```

Código B.12: Lectura de datos y asignación de constantes

La generación del nuevo eje de frecuencias y del enventanado se realiza de la misma forma que el caso anterior la única diferencia en este aparatado es que debemos generar el espectro de la señal tanto para el caso del S11 como para el S22.

```
    #Creamos nuevo eje de frecuencias entre 0 y 120 GHz
    Fs = 120e9  #Nueva frecuencia de muestreo de 0 Hz a 120 GHz
    DF = frecuencia[1]-frecuencia[0]
    new_frequency = np.linspace(0, Fs, len(frecuencia))
    F_axis = np.arange(0, Fs, DF)
    nfft = len(F_axis)

    #Ventana tukey

    MyWin = US.makeWindow(SortofWin='tukey', WinLen=LenScan,
        ↪ param1=0.25, param2=1, Span=0, Delay=0)

    if np.mod(nfft,2)==1:
    MyWin_ext = np.concatenate(( MyWin, np.zeros((int( (nfft+1)/2
        ↪ )-LenScan)) ))
    MyWin = np.concatenate((MyWin_ext, np.flipud(MyWin_ext[:-1])))
    else:
    MyWin_ext = np.concatenate(( MyWin,
        ↪ np.zeros((int(nfft/2)-LenScan)) ))
    MyWin = np.concatenate((MyWin_ext, np.flipud(MyWin_ext)))

    #Gerenamos espectro de las dos señales además de interpolar
        ↪ con zeros
    S_ref = MWF.makeSpectrum(S11_ref, F_axis, frecuencia)
    S2_ref = MWF.makeSpectrum(S22_ref, F_axis, frecuencia)
    S_DataMatrix = MWF.makeSpectrum(np.squeeze(S11_antena),
        ↪ F_axis, frecuencia)
    S2_DataMatrix = MWF.makeSpectrum(np.squeeze(S22_antena),
        ↪ F_axis, frecuencia)

```

Código B.13: Generación nuevo eje de frecuencias, aplicación de enventanado Tukey, interpolación de valores y generación de espectro

El siguiente paso era realizar el filtrado de las señales (después de aplicarles en enventanado) y realizar sus transformadas inversas de Fourier, este paso sigue igual pero debe hacerse dos veces, una para cada tipo de dato.

```
    #Generamos pulso Gaussiano para el procesado de la señal
    Amp = 50 * 10**8
    tau = 60 * 10**(-12)
    paso = tau / 100
    tin = 7 * tau
    Pulse = MWF.makePulse(Amp, tau, tin, nfft, Fs)
    Pulse_F = np.fft.fft(Pulse, nfft)

    #Pulso gaussiano modificado
    t_axis = np.arange(nfft) / Fs
    Pulse_G = signal.gausspulse(t\_axis, fc=3e9, bw=0.5, bwr=-6,
        ↪ tpr=-60)
    Pulse_GF = np.fft.fft(Pulse_G, nfft)
    Flim = 10e9

    #Realizamos un filtrado de las señales y hacemos las
        ↪ transformadas inversas
    #Para parámetros S11
    ref_in_time, borrar = MWF.adaptedFilter(S_ref, Pulse,
        ↪ XCR=True, Aligned=True,
    Wiener=False)
    winref_in_time = np.real(np.fft.ifft(S_ref * MyWin))
    DataMatrix_in_time = np.zeros((Nplanchas, Nantenas, nfft))
    WinMatrix_in_time = np.zeros((Nplanchas, Nantenas, nfft))
    DataMatrix_in_time, borrar = MWF.adaptedFilter(S_DataMatrix,
        ↪ Pulse,XCR=True, Aligned=True, Wiener=False)
    WinMatrix_in_time=
        ↪ np.real(np.fft.ifft(np.squeeze(S_DataMatrix) * MyWin))

    #Para parametros S22
    ref_in_time_2, borrar = MWF.adaptedFilter(S2_ref, Pulse,
        ↪ XCR=True, Aligned=True,
    Wiener=False)
    winref_in_time_2 = np.real(np.fft.ifft(S2_ref * MyWin))
    DataMatrix_in_time_2 = np.zeros((Nplanchas, Nantenas, nfft))
    WinMatrix_in_time\_2 = np.zeros((Nplanchas, Nantenas, nfft))

    DataMatrix_in_time_2, borrar =
        ↪ MWF.adaptedFilter(S2_DataMatrix, Pulse,XCR=True,
        ↪ Aligned=True, Wiener=False)
    WinMatrix_in_time_2=
        ↪ np.real(np.fft.ifft(np.squeeze(S2_DataMatrix) * MyWin))

    OffSet = 0 # offset en mm
    dis_axis = (np.arange(ref_in_time.shape[1]))  * speed / (2 *
        ↪ Fs) * 1000 - OffSet
    dis_axis_2 = (np.arange(ref_in_time\_2.shape[1]))  * speed /
        ↪ (2 * Fs) * 1000 - OffSet

```

Código B.14: Generación de pulso Gaussiano, filtado de señales y transformadas inversas

Realizamos la resta de señales en ambos casos con el fin de eliminar ruido y elementos comunes. También generaremos la envolvente de la señal, la cual nos facilitará la labor de encontrar y determinar el valor de la posición del máximo de la señal.

```
    Antenna = 0
    time_axis = dis_axis / Fs * 1e6
    time_axis_2 = dis_axis\_2 / Fs * 1e6

    Delay = 5000
    s_ref = np.roll(ref_in_time, Delay, axis=1)
    ws_ref = np.roll(winref_in_time, Delay, axis=1)

    s2_ref = np.roll(ref_in_time_2, Delay, axis=1)
    ws_ref_2 = np.roll(winref_in_time_2, Delay, axis=1)

    DataMatrix_in_time = np.roll(np.squeeze(DataMatrix_in_time),
        ↪ Delay, axis=0)
    WinMatrix_in_time = np.roll(np.squeeze(WinMatrix_in_time),
        ↪ Delay, axis=0)

    DataMatrix_in_time_2 =
        ↪ np.roll(np.squeeze(DataMatrix_in_time_2), Delay, axis=0)
    WinMatrix_in_time_2 = np.roll(np.squeeze(WinMatrix_in_time_2),
        ↪ Delay, axis=0)

    Max_ind = np.argmax(np.abs(s_ref[Antenna, :]))
    wMax_ind = np.argmax(np.abs(ws_ref[Antenna, :]))
    Max_ind_e = np.argmax(US.envelope(s_ref[Antenna, :]))
    Max_ind_2 = np.argmax(np.abs(s2_ref[Antenna, :]))\\
    wMax_ind_2 = np.argmax(np.abs(ws_ref_2[Antenna, :]))
    Max_ind_e_2 = np.argmax(US.envelope(s2_ref[Antenna, :]))

    #Resta de señales para elimnar el ruido
    resta = DataMatrix_in_time - s_ref[Antenna,:]
    resta_2 = DataMatrix_in_time_2 - s2_ref[Antenna,:]

    #Envolvente de la señal
    enveloped = np.abs(signal.hilbert(s_ref))
    enveloped_2 = np.abs(signal.hilbert(s2_ref))
    enveloped_resta = np.abs(signal.hilbert(resta))
    enveloped_resta_2 = np.abs(signal.hilbert(resta_2))


```

Código B.15: Generación de pulso Gaussiano, filtado de señales y transformadas inversas

Lo único que falta es representar los resultados que obtenemos tanto para el caso del S11 como del S22. Para ello haremos una gráfica con dos subgráficas, la primera mostrará los resultados obtenidos para el S11 y la segunda para el S22.

```
1
2  max_position_1 = np.argmax(enveloped_resta)
3  max_x_value_1 = dis_axis[max_position_1]-Offset_dist
4
5  max_position_2 = np.argmax(enveloped_resta_2)
6  max_x_value_2 = dis_axis[max_position_2]-Offset_dist
7
8  print("Posición del máximo de S11 (en cm):", max_x_value_1/10)
9  print("Posición del máximo de S22 (en cm):", max_x_value_2/10)
10
11
12 plt.figure(figsize=(10, 6))
13 plt.subplot2grid((2, 1), (0, 0))
14 plt.plot(((dis_axis-Offset_dist)/10), resta, label='Original',
       ↪ color='blue')
15 plt.plot(((dis_axis-Offset_dist)/10), enveloped_resta,
       ↪ label='Enveloped', color='orange')
16 plt.xlabel('Distancia (cm)')
17 plt.ylabel('Amplitud')
18 plt.title('Distancia plancha - antena 1')
19 plt.legend()
20 plt.grid(True)
21 plt.xlim(0, 60)
22
23 plt.subplot2grid((2, 1), (1, 0))
24 plt.plot(((dis_axis_2-Offset_dist)/10), resta_2, label='Original',
       ↪ color='blue')
25 plt.plot(((dis_axis_2-Offset_dist)/10), enveloped_resta_2,
       ↪ label='Enveloped', color='orange')
26 plt.xlabel('Distancia (cm)')
27 plt.ylabel('Amplitud')
28 plt.title('Distancia plancha - antena 2')
29 plt.legend()
30 plt.grid(True)
31 plt.xlim(0, 60)
32 plt.tight_layout()
33
34 plt.show()
35
```

Código B.16: Ploteo final

C. ANEXO III: CÓDIGO TRANSMISIÓN DE PULSOS

Este anexo sirve como aclaración de todos los procesos seguidos en el capítulo V: análisis de pulsos en transmisión. Aportando todo el código empleado durante el desarrollo de esta fase del proyecto y las explicaciones necesarias de las funciones y métodos empleados.

El código comienza de la misma forma que el de análisis de pulsos en reflexión, importamos las librerías, cargamos los archivos de los que obtenemos los datos, leemos los datos y asignamos constantes y variables con la única diferencia de que ahora emplearemos los parámetros S21.

```
import os
import skrf as rf
import numpy as np
from math import sqrt
from scipy import signal
from US_SSP import GaussPulseFilter
import matplotlib.pyplot as plt
import MW_ToolBox as MWF
import US_Functions as US
import sys
sys.path.append('C:\\Users\\luvaa\\AppData\\Roaming\\Python\\
Python312\\site-packages')

# Leemos los datos de referencia
filename = os.path.join(os.path.expanduser("~"), "Desktop",
    ↪ "Practicas_Internas", "Medidas_Laboratorio","pruebas",
    ↪ "Transmision", "25_cm.s2p")

network = rf.Network(filename)
frecuencia = network.f

#Parametros S referencia
S21 = network.s[:, 1, 0]
S21_db = 20 * np.log10(np.abs(S21))

c = 299792458.
Er= 1, speed = c / np.sqrt(Er)
LenScan = 5001
Nplanchas = 1
Nantenas = 1

#Inicalizamos y pasamos los valores
S21_ref = np.zeros((Nantenas, LenScan), dtype=complex)
S21_ref = S21

```

Código C.17: Lectura de valores y asignación de constantes

De igual forma que ne los casos anteriores debemos generar un nuevo eje de frecuencias que vaya de 0 a 120 GHz y un enventanado para suavizar las señales.

```
    Fs = 120e9  # Nueva frecuencia de muestreo de 0 Hz a 120 GHz
    DF = frecuencia[1]-frecuencia[0]
    new_frequency = np.linspace(0, Fs, len(frecuencia))
    F_axis = np.arange(0, Fs, DF)
    nfft = len(F_axis)
    t_axis = np.arange(nfft) / Fs

    #Ventana tukey

    MyWin = US.makeWindow(SortofWin='tukey', WinLen=LenScan,
        ↪ param1=0.25, param2=1, Span=0, Delay=0)

    if np.mod(nfft,2)==1:
    MyWin_ext = np.concatenate(( MyWin, np.zeros((int( (nfft+1)/2
        ↪ )-LenScan)) ))
    MyWin = np.concatenate((MyWin_ext, np.flipud(MyWin_ext[:-1])))
    else:
    MyWin_ext = np.concatenate(( MyWin,
        ↪ np.zeros((int(nfft/2)-LenScan)) ))
    MyWin = np.concatenate((MyWin_ext, np.flipud(MyWin_ext)))

```

Código C.18: Generación nuevo eje de frecuencias y de enventanado Tukey

El pulso de salida "Vo" se obtendrá como el producto de los parámetros S21 por un pulso de entrada que generaremos nosotros, para la selección de este pulso realizamos varias pruebas con las formas de las que disponemos en el ANEXO I, comprobando con cuál obteníamos un máximo más claro, a continuación se muestra cuales fueron los datos empleados para la generación del pulso.

```
    tau = 75 * 10**(-12)
    Amp = 50 * 10**8
    to = 1*1e-9
    sigma = np.sqrt(tau**2/2)

    #Pulso Generado
    Pulse_A = Amp*(t_axis-to)*np.exp(-((t_axis-to)**2)/(tau**2))
    Pulse_AF = np.fft.fft(Pulse_A)
    Pulse_norm_A = Pulse_A/np.max(Pulse_A)
    Pulse_norm_AF = np.fft.fft(Pulse_norm_A)
    Pulse_norm_A_F =
        ↪ np.abs(Pulse_norm_AF)/np.max(np.abs(Pulse_norm_AF))

```

Código C.19: Generación del pulso de entrada para la obtención de Vo

Seleccionamos cuál será nuestro pulso de entrada, generamos el espectro de la señal y le aplicamos el filtrado. Una vez realizados los pasos anteriores ya podemos obtener Vo como el producto del pulso de entrada por el S21. Generamos el eje de distancias y realizamos las transformadas inversas de Fourier a Vin y Vo, obteniéndolas en el dominio del tiempo.

```
Pulse_in = Pulse_AF   #Si se definen otros pulsos cambiar pulso entrada
Pulse_norm =  Pulse_norm_A   #Pulso de entrada normalizado

S_ref = MWF.makeSpectrum(S21_ref, F_axis, frecuencia)
ref_in_time, borrar = MWF.adaptedFilter(S_ref, Pulse, XCR=True, Aligned=True,
Wiener=False) #
vo = (S_ref[0,:] * Pulse_in)

OffSet = 0 # offset in mm
dis_axis = (np.arange(ref_in_time.shape[1]))  * speed / (Fs) * 1000  - OffSet


#Calculamos la IFFT  de Vi(f)
Vit = np.fft.ifft(Pulse_in)

#Calculamos la IFFT  de Vo(f) obteniendo Vo(t)

Vot = np.real(np.fft.ifft(vo) * MyWin)
Vot_norm = Vot/np.max(np.abs(Vot))

```

Código C.20: Selección pulso de entrada, generación espectro de la señal y filtrado, generación eje de distancias y obtención de Vit y Vot

Una vez en este punto ya disponemos tanto del pulso de entrada como el de salida y por tanto lo único que faltaría sería representarlos en una gráfica. Previamente a esto llevaremos a cabo una serie de procesos previos para facilitar la obtención de los resultados.

En primer lugar emplearemos la transformada Hilbert para obtener la envolvente de las señales, facilitando la obtención del máximo y usamremos la función "argmax" de la librería numpy para obtener la posición del máximo, mediante el uso de los "print" podremos sacar por consola las posiciones de los máximos, cuya resta será la distancia que buscamos.

```
#Representamos la envolvente
enveloped_vin = np.abs(signal.hilbert(Pulse_norm))
enveloped_vot = np.abs(signal.hilbert(Vot_norm))

# Encuentra las posiciones de los máximos
vi_max_position = np.argmax(enveloped_vin)
vo_max_position = np.argmax(enveloped_vot)

# Obtiene los valores de las posiciones en el eje x
        ↪ correspondientes a los máximos
vi_max_x_value = dis_axis[vi_max_position]
vo_max_x_value = dis_axis[vo_max_position]
resta_max = dis_axis[vo_max_position] - dis_axis[vi_max_position]

print("Posición del máximo de Vi(t) (en cm):", vi_max_x_value/10)
print("Posición del máximo de Vo(t) (en cm):", vo_max_x_value/10)
print("Distancia (cm):", resta_max/10 )


```

Código C.21: Generación nuevo eje de frecuencias, aplicación de enventanado Tukey, interpolación de valores y generación de espectro

Como se explico en el apartado 5.3.2 hubo que realizar unas correcciones a las distancias obtenidas pues estábamos añadiendo tramos que no ser correspondían con nuestras medidas sino con elementos de nuestros sistema. Esas correcciones se reflejan a continuación.

```
#Aplicando corrección conectores y tramo de antena (cm)
conector = 3.7 * sqrt(2.1)
tramo_antena = 1.65 * sqrt(3.325)
error = 2*conector + 2* tramo_antena
print("Distancia corregida(cm):", (resta_max/10)-error )

```

Código C.22: Corrección aplicada a las distancias obtenidas

Una vez terminadas las correcciones y procesos solo nos falta representar las señales finales en una gráfica (divida en dos subgráficas).

```
plt.figure(figsize=(10, 6))
plt.subplot2grid((2, 1), (0, 0))
plt.plot(dis_axis/10, Pulse_norm, label= 'Vin', color = 'blue')
plt.plot(dis_axis/10, enveloped_vin, label= 'Enveloped Vin', color
        ↪ = 'orange')
plt.xlabel('dist(cm)')
plt.ylabel('Amplitude')
plt.title('Vin')
plt.grid(True)
plt.xlim(0,60)

plt.subplot2grid((2, 1), (1, 0))
plt.plot((dis_axis/10)-error, Vot_norm, label= 'Vo', color =
        ↪ 'blue')
plt.plot((dis_axis/10)-error, enveloped_vot, label= 'Enveloped
        ↪ Vo', color = 'orange')
plt.xlabel('dist (cm)')
plt.ylabel('Amplitude')
plt.title('Vo')
plt.xlim(0,125)    #Iremos reduciendo el eje x a medida que
        ↪ reducimos la distancia
plt.grid(True)
plt.tight_layout()

```

Código C.23: Representación final de las señales