Rivero Sánchez Gabriel Gerardo

Software interactivo para la simulación del patrón de radiación de una antena Helicoidal

2026

Universidad Autónoma de Querétaro

Facultad de Ingeniería

Software interactivo para la simulación del patrón de radiación de una antena Helicoidal

Tesis

Que como parte de los requisitos para obtener el Grado de

Licenciado en

Ingeniería Física.

Presenta

Rivero Sánchez Gabriel Gerardo

Dirigido por:
Dr. Aldrin Melitón Cervantes Contreras.

Querétaro, Qro. Mayo de 2026.

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Licenciatura en Ingeniería Física

Software interactivo para la simulación del patrón de radiación de una antena Helicoidal

Tesis

Que como parte de los requisitos para obtener el Grado de

Licenciado en Ingeniería Física.

Presenta

Rivero Sánchez Gabriel Gerardo

Dirigido por

Dr. Aldrin Melitón Cervantes Contreras

Dr. Aldrin Melitón Cervantes Contreras.
Presidente

Dr. Marco Antonio Aceves Fernández.
Secretario

Dr. Josué de Jesús Trejo Alonso.
Vocal

Dr. Alberto Hernández Almada.
Vocal

Centro Universitario, Querétaro, Qro. México
Fecha de aprobación por el Consejo Universitario Mayo de 2026.

Dedicatoria

Este trabajo está dedicado especialmente a mi familia (Judith, Belén, Gerardo, Yaqueline y Pueblito), por formarme en los valores de la perseverancia, la gratitud, el respeto y la resiliencia.

A mis amigos (Daniel, Emilio, Miguel, Vanessa, Isabel, Marín y Luis), por su compañía y atención a lo largo de años difíciles.

A la propia vida y a mi etapa en la física, que me enseñaron la ética personal, la humildad de reconocer mi propia ignorancia y la obsesión por sumergirme en lo desconocido con el fin de alcanzar el conocimiento.

"Si quieres encontrar los secretos del universo, piensa en términos de energía, frecuencia y vibración".

— **Nikola Tesla**

Agradecimientos

Quiero agradecer a la Universidad Autónoma de Querétaro por brindarme la oportunidad de realizar este trabajo. Agradezco especialmente al director de esta tesis, el Dr. Aldrin Melitón Cervantes Contreras, por el seguimiento, las observaciones y las correcciones realizadas a lo largo del desarrollo del proyecto. Asimismo, agradezco a los profesores Dr. Aceves Fernández, Dr. Trejo Alonso y Dr. Almada por fungir como sinodales de este trabajo.

Resumen

El presente trabajo analiza la formación de los patrones de radiación de una antena helicoidal a partir del estudio de las ondas electromagnéticas. Con el objetivo de facilitar la comprensión de este fenómeno, se desarrolló software especializado que permite la visualización y análisis del comportamiento del patrón de radiación bajo distintas condiciones.

Abstract

This thesis analyzes the formation of radiation patterns in a helical antenna based on the study of electromagnetic waves. In order to facilitate understanding of this phenomenon, specialized software was developed that allows visualization and analysis of radiation pattern behavior under different conditions.

Índice

Índice de figuras

Índice de cuadros

1.. Introducción

1.1. Descripción del problema.

Normalmente, resulta complejo visualizar el cómo se ve la emisión de radiación. Se requiere de herramientas visuales que normalmente hacen uso de softwares que requiere de un pago, normalmente excesivo, para poder hacer un análisis investigativo en lo que se refiere al electromagnetismo. Por lo que, se requiere ampliamente una herramienta de amplio alcance para la comunidad estudiantil para que se pueda entender de manera visual e intuitiva el funcionamiento de las antenas (en este caso las Helicoidales).

Para la actualidad académica se necesita de una herramienta que no limite la comprensión de este tipo de fénomenos que son de suma importancia en esta era tecnológica. El problema a abordar radica en poder analizar un entorno gráfico (con ayuda de python) para observar de manera tridimensional el espectro de radiación de una antena Helicoidal para dar una alternativa de análisis en la didáctica de este tipo de comportamiento.

1.2. Justificación.

Lo que motivó su ejecución e investigación fue el hecho de que los softwares existentes en el mercado para la explicación de patrones de radiación no proporcionan una comprensión clara de los fundamentos que hay detrás de los resultados. La razón principal de este trabajo es ofrecer un código de libre acceso y uso, que pueda ser modificado por cualquier persona según sus necesidades, además de explicar el proceso necesario para llegar a este tipo de solución. Otro de sus fines es que pueda ser utilizado como una herramienta de visualización y análisis de fenómenos electromagnéticos. Por esta razón, se decidió abordar el tema con mayor rigor académico, con la finalidad de aportar una contribución relevante a la didáctica de las antenas.

La elección de una antena helicoidal, y no de otro tipo, se debe a su gran valor dentro del campo de las telecomunicaciones, debido a sus características geométricas y físicas. Su análisis resulta conveniente, ya que permite integrar diversos conceptos del electromagnetismo, como la propagación de ondas, la geometría tridimensional y el comportamiento de la radiación. Por ello, contar con una herramienta visual facilita su comprensión. Su propósito no solo es didáctico, sino también permitir observar de manera clara cómo se desarrollan estos procesos.

Por otro lado, es importante señalar que el desarrollo de un software de código abierto es de gran relevancia, ya que permite realizar modificaciones y adaptaciones según las necesidades de cada usuario. Esto no pretende desvalorizar el trabajo de los softwares existentes en el mercado ni de aquellos de libre uso. Sin embargo, contar con un desarrollo propio reduce costos, evita el pago de licencias y elimina la necesidad de someterse a las políticas de cada empresa.

Además, este tipo de herramienta permite su optimización e integración con nuevos modelos, dependiendo del tipo de antena que se desee analizar. A diferencia de algunos

softwares de libre uso, que no explican el trasfondo de los diseños ni el funcionamiento de la teoría electromagnética, esta propuesta busca aportar claridad en dichos procesos.

De esta manera, se establece un cimiento hacia la independencia tecnológica, tanto para los estudiantes como para cualquier persona interesada en comprender y aprender estos conceptos.

1.3. Planteamiento.

Se propone hacer un estudio de antenas, con la finalidad de llegar al tipo Helicoidal, en el cual se pueda comprender la geometría y emisión del espectro de radiación. Por lo que se plantea desarrollar un software que permita el aprendizaje práctico con la inserción de parámetros físicos. Se requiere que este modelo visualice y analice el patrón de radiación de la antena de manera tridimensional con el uso de herramientas de programación en Python.

1.4. Hipótesis.

El desarrollo de un software interactivo permitirá simular el patrón de radiación de una antena helicoidal en función de sus parámetros, obteniendo resultados consistentes con la teoría electromagnética y facilitando su análisis y comprensión en un entorno académico.

1.5. Objetivo General.

Desarrollar una simulación del espectro de radiación de una antena helicoidal que permita insertar parámetros mediante una interfaz gráfica interactiva en Python, con el fin de facilitar la enseñanza y el análisis del diseño en telecomunicaciones.

1.6. Objetivos Específicos.

- Analizar la literatura de electromagnetismo hasta el desarrollo de las antenas helicoidales.
- Desarrollar una simulación que permita visualizar el comportamiento del campo radiado.
- Representar el patrón de radiación con ayuda de Python.
- Comparar la simulación teórica con los resultados obtenidos de cálculos propios.
- Diseñar una interfaz gráfica que permita modificar los parámetros de la antena y observar su efecto en tiempo real.

2.. Antecedentes

2.1. Sobre radiación.

El estudio de la radiación electromagnética constituye la base para comprender el funcionamiento de las antenas, ya que describe cómo la energía es generada, propagada y distribuida en el espacio. En este contexto, es necesario introducir los conceptos fundamentales que permiten caracterizar las ondas electromagnéticas, así como su clasificación dentro del espectro y los distintos tipos de radiación. Estos elementos proporcionan el marco teórico indispensable para analizar el comportamiento del campo radiado en aplicaciones de telecomunicaciones.

2.1.1. Frecuencia y longitud de onda.

Para describir cualquier onda electromagnética es necesario definir dos magnitudes fundamentales: la frecuencia y la longitud de onda. La frecuencia f representa el número de oscilaciones completas (ciclos) que realiza la onda por unidad de tiempo y se mide en hercios (Hz), donde 1 Hz equivale a un ciclo por segundo. La longitud de onda λ es la distancia mínima entre dos puntos de la onda que se encuentran en el mismo estado de vibración (por ejemplo, entre dos crestas consecutivas).

Ambas magnitudes están relacionadas con la velocidad de propagación de la onda. En el vacío, todas las ondas electromagnéticas viajan a la velocidad de la luz $c = 3 \times 10^8\,\mathrm{m/s}$ (Young and Freedman, 2018), cumpliendo la relación expresada en la ecuación 1:

$$c = f\lambda \tag{1}$$

Esta ecuación muestra que frecuencia y longitud de onda son inversamente proporcionales: a mayor frecuencia, menor longitud de onda, y viceversa.

Como ejemplos cotidianos, un router de Wi-Fi opera típicamente en frecuencias de 2.4 GHz o 5 GHz, mientras que una estación de radio FM transmite alrededor de 0.1 GHz (Greiner, 2015).

La Unión Internacional de Telecomunicaciones (ITU) clasifica las bandas de frecuencia utilizadas en tecnología y comunicaciones, como se muestra en la Tabla 1.

Tabla 1: Clasificación de las bandas de frecuencia según la ITU (Rajib, 2025).

Nombre	Abreviatura inglesa	Banda ITU	Frecuencias	Longitud de onda
Extra baja frecuencia	ELF	1	Inferior a 3 Hz	más de 100 000 km
Súper baja frecuencia	SLF	2	3–30 Hz	100 000–10 000 km
Ultra baja frecuencia	ULF	3	30–300 Hz	10 000–1 000 km
Muy baja frecuencia	VLF	4	300–3 000 Hz (3–30 kHz)	1 000–100 km
Baja frecuencia	LF	5	30–300 kHz	100–1 km
Media frecuencia	MF	6	300–3 000 kHz	1 km–100 m
Alta frecuencia	HF	7	3–30 MHz	100–10 m
Muy alta frecuencia	VHF	8	30–300 MHz	10–1 m
Ultra alta frecuencia	UHF	9	300–3 000 MHz	1 m–100 mm
Súper alta frecuencia	SHF	10	3–30 GHz	100–10 mm
Extra alta frecuencia	EHF	11	30–300 GHz	10–1 mm
Por encima de los 300 GHz	–	–	Mayor a 300 GHz	menos de 1 mm

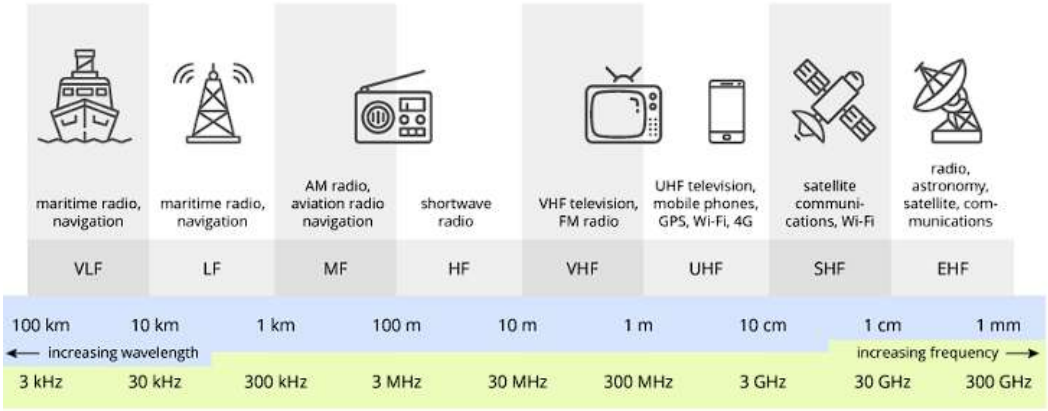


Figura 1: Usos de frecuencia en la tecnología (Side, 2021).

La Tabla 1 muestra la clasificación de las frecuencias y sus correspondientes longitudes de onda, mientras que la Figura 1 ilustra algunas aplicaciones tecnológicas de estas bandas.

2.1.2. Espectro electromagnético.

La radiación electromagnética abarca un amplio rango de frecuencias y longitudes de onda conocido como espectro electromagnético. Este espectro se divide en distintas regiones que, de menor a mayor frecuencia (o de mayor a menor longitud de onda), incluyen: ondas de radio, microondas, infrarrojo, luz visible, ultravioleta, rayos X y rayos gamma. Cada región posee propiedades características y aplicaciones específicas en ciencia y tecnología (Fontal et al., 2005).

Como se observa en la Figura 2, el espectro electromagnético se extiende desde ondas de radio (baja frecuencia, gran longitud de onda) hasta rayos gamma (alta frecuencia, longitud de onda muy pequeña). La luz visible, que es la región perceptible

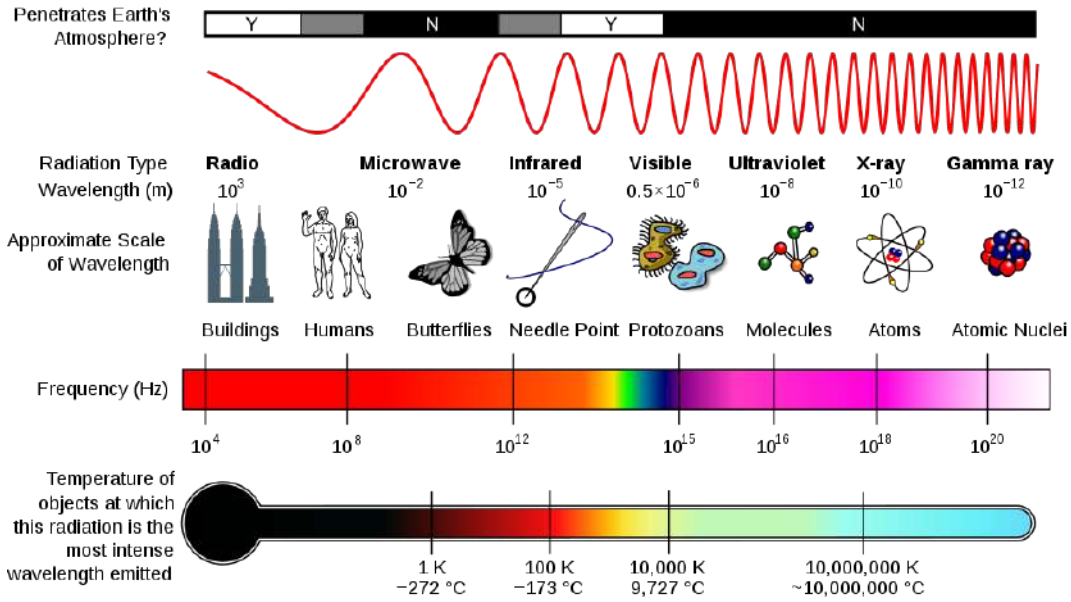


Figura 2: Espectro electromagnético (Voices and Voices, 2024).

2.1.3. Tipos de radiación.

La radiación se puede clasificar en tres tipos principales según la naturaleza de las partículas o fotones emitidos (Hoyt, 1958):

a. **Radiación** α: Consiste en núcleos de helio (dos protones y dos neutrones) emitidos a altas velocidades. Tienen baja capacidad de penetración y alta capacidad de ionización.

b. **Radiación** β: Son electrones (o positrones) de alta velocidad emitidos desde el núcleo de átomos radiactivos. Tienen mayor poder de penetración que las partículas α.

c. **Radiación** γ: Son fotones de alta energía, similares a los rayos X pero de origen nuclear. Corresponden a la radiación electromagnética más energética del espectro.

Finalmente, para la producción de ondas electromagnéticas es fundamental recordar que una carga eléctrica en reposo crea un campo eléctrico, mientras que una carga en movimiento (acelerada) crea un campo magnético y, cuando oscila, produce radiación electromagnética que se propaga en el espacio (Pérez Navarro, 2011).

2.2. Antenas.

En esta sección se abordan los fundamentos de las antenas, elementos esenciales en el estudio y aplicación de la radiación electromagnética. Se presentan los conceptos

básicos, su funcionamiento, así como sus principales características y aplicaciones en sistemas de comunicación (Kraus, 1950; Pozar, 2012; Balanis, 2005).

2.2.1. Conceptos básicos.

Kraus (1950) y Pozar (2012) ofrecen las siguientes definiciones útiles:

- La línea de transmisión es un dispositivo para transmitir o ganar energía de radiofrecuencia de un punto a otro.
- La guía de onda es cuando la onda pasa a través de una larga línea de forma unidimensional y no se difunde fuera del espacio.
- La onda estacionaria es la interferencia entre la onda incidente y la reflejada.
- El resonador es el dispositivo que genera ondas estacionarias.
- La cavidad del resonador es el flujo guiado de entrada y salida; no interviene en una sección corta de circuito de guía de onda.

Mientras que Huidobro (2013) establece las siguientes definiciones:

- Ancho de banda es un margen de las frecuencias.
- Directividad se puede describir como la potencia que tiene la onda en su dirección.
- La ganancia es la directividad aplicada pero a una distancia R.
- El rendimiento de la antena se ve en la relación entre potencia de radiación y la potencia total.
- La impedancia es la relación entre voltaje y corriente de una onda viajera.
- La anchura del haz es un intervalo angular en el que la densidad de potencia radiada es igual a la mitad de la máxima.
- La polarización es una dirección que traza la geometría del $\vec{E}$.

2.2.2. ¿Cómo se forma una antena y cuáles son sus regiones?

Podemos definir a una antena como aquella estructura transicional entre el espacio libre y un dispositivo de guía (la cual adopta la forma de una línea coaxial o un tubo, como se muestra en las Figuras 3 y 4). La antena sirve tanto para recibir como para transmitir energía electromagnética (Balanis, 2005).

Como se muestra en la Figura 3, el proceso de radiación inicia en una fuente de radiofrecuencia que genera una onda viajera a través de una línea de transmisión. Al llegar a la región de transición, correspondiente a la antena, esta onda deja de

estar confinada y se convierte en una onda que se propaga en el espacio libre. En esta zona, la energía guiada se transforma en radiación electromagnética, lo que permite la transmisión de señales a largas distancias.

Por otro lado, la Figura 4 ilustra las distintas regiones espaciales asociadas a una antena. En la región cercana o de Fresnel, los campos eléctricos y magnéticos presentan variaciones complejas y aún no forman un patrón de radiación definido. A medida que la distancia aumenta, se alcanza la región lejana o de Fraunhofer, donde las ondas se comportan de manera más uniforme y el patrón de radiación se estabiliza, permitiendo su análisis mediante aproximaciones más simples. La frontera entre ambas regiones depende de las dimensiones características de la antena y de la longitud de onda de operación (Kraus, 1950).

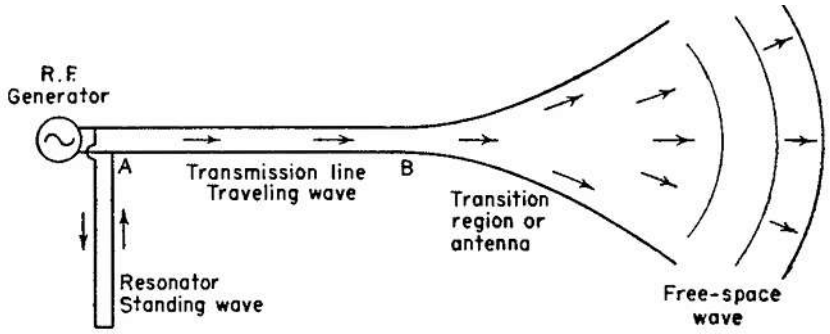


Figura 3: La antena es una región de transición entre una guía de onda de transmisión lineal y el espacio libre de onda (Kraus, 1950).

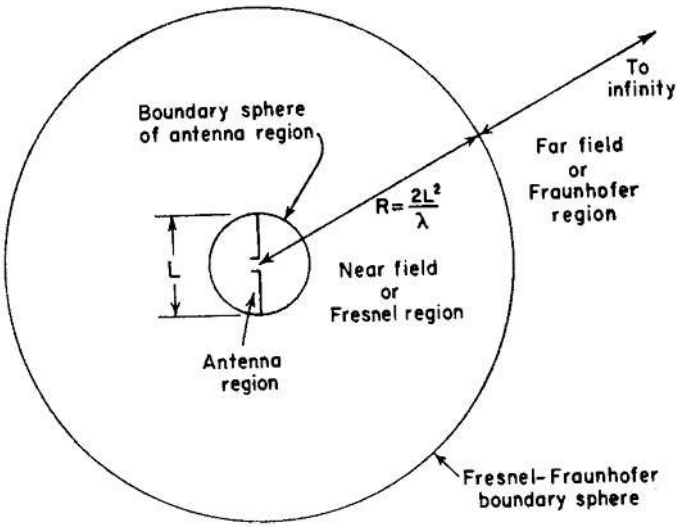


Figura 4: Región de antena, región de Fresnel y región de Fraunhofer (Kraus, 1950).

Las condiciones necesarias para la obtención de radiación son las siguientes. El mecanismo de radiación ocurre gracias al uso de cables conductores que serán el medio

por el cual las cargas eléctricas fluyen. Para que exista la radiación se debe hacer uso de la aceleración o desaceleración de cargas, o hacer que las cargas tengan una oscilación armónica (Balanis, 2005):

a. Si la carga no se desplaza, no hay radiación.

b. Si la carga se desplaza con velocidad uniforme:

   - No hay radiación si es en un alambre recto e infinito.
   - Sí hay radiación si es en un alambre curvado, doblado, discontinuo o trunco.

c. Hay radiación si la carga oscila en movimiento armónico.

2.2.3. ¿Cómo sería una antena ideal?

La antena ideal es la de tipo isotrópica, que transmite energía de forma uniforme en todas direcciones. Sin embargo, en la teoría no es realizable, ya que existen pérdidas y no existe la transmisión perfecta. Se usa como concepto de referencia para evaluar las antenas (Kraus, 1950).

Esta antena hipotética, en el espacio libre, irradia energía de manera uniforme en todas direcciones. El flujo de la energía radiada por unidad de tiempo y de área se conoce como el vector de Poynting, expresado en la ecuación 2 (Kraus, 1950):

$$\vec{S} = \vec{E} \times \vec{H} \tag{2}$$

2.2.4. Sobre satélites.

Otra de las utilidades en las telecomunicaciones es el GPS, que se describe como una red de 24 satélites en órbitas de dos revoluciones por día; cuatro satélites comparten órbita y todos se posicionan a 55 grados de la línea ecuatorial. Operan a 1.575 GHz debido a que el agua en la atmósfera absorbe la radiación, por lo que deben superar los 1.228 GHz. Los satélites se ubican a cerca de los 35 786 km encima de la superficie (GPS, 2023; Kaplan, 2017), como se observa en las Figuras 5 y 6.

Como se observa en la Figura 5, las antenas en el espacio forman parte de sistemas satelitales que permiten la transmisión y recepción de señales electromagnéticas hacia la Tierra. Estas antenas, generalmente acopladas a satélites, están diseñadas para operar en condiciones de vacío y radiar energía de manera eficiente hacia regiones específicas del planeta, lo que posibilita aplicaciones como telecomunicaciones, navegación y monitoreo terrestre.

Por otro lado, la Figura 6 ilustra la trayectoria orbital de un satélite alrededor de la Tierra, así como la geometría involucrada en la transmisión de señales. En este contexto, los sistemas GPS se componen de una constelación de satélites distribuidos en órbitas específicas, lo que permite que un receptor en la superficie terrestre determine su posición mediante la recepción simultánea de señales de múltiples satélites. Este

Figura 5: Cómo se ve una antena en el espacio (NASA, 2023).

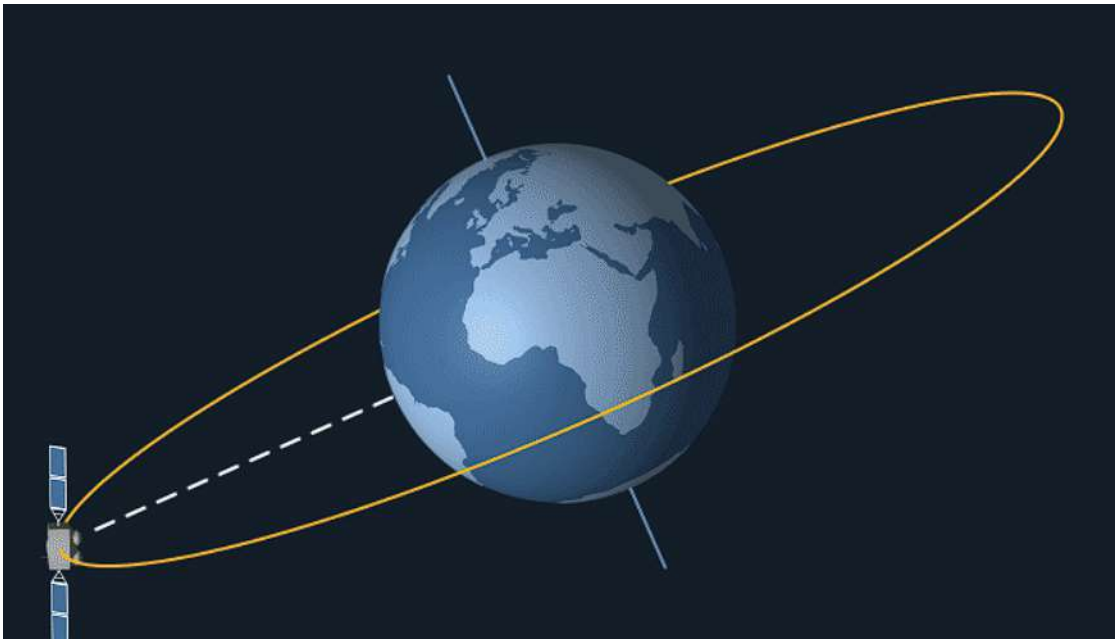


Figura 6: Órbita de una antena satelital (NASA, 2023).

proceso requiere al menos cuatro señales para obtener las tres coordenadas espaciales y la corrección temporal del sistema, lo que garantiza una localización precisa (GPS, 2023; Kaplan, 2017).

2.2.5. ¿Qué es el patrón de radiación?

Como se muestra en la Figura 7, el análisis del patrón de radiación de una antena se realiza comúnmente en un sistema de coordenadas esféricas, que permite describir la dirección de propagación de los campos electromagnéticos. En este sistema, la intensidad de radiación depende de la dirección, lo que da lugar a distribuciones no uniformes de energía en el espacio.

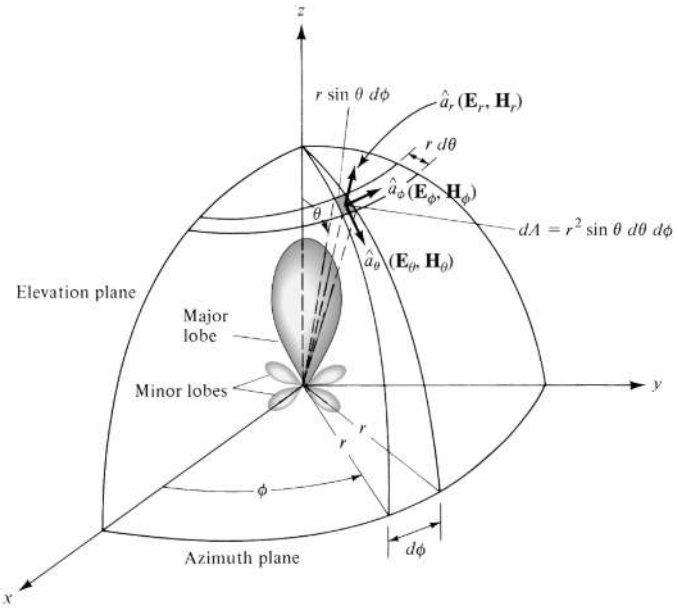


Figura 7: Sistema de coordenadas para análisis de antenas (Balanis, 2005).

Por otro lado, la Figura 8 presenta la forma típica de un patrón de radiación, donde se identifican los lóbulos principales y secundarios. El lóbulo principal indica la dirección en la que la antena radia la mayor cantidad de energía, mientras que los lóbulos secundarios representan radiación no deseada en otras direcciones. Asimismo, se definen parámetros importantes como el ancho de haz a media potencia (HPBW) y el ancho entre primeros nulos (FNBW), los cuales permiten caracterizar la directividad y el desempeño de la antena (Balanis, 2005).

El patrón de radiación es la forma en que una antena distribuye su potencia en el espacio y se define tomando como referencia una antena isótropa, que radia igual en todas direcciones. La relación entre la potencia radiada por la antena y la de la isótropa en una dirección se denomina ganancia directiva, expresada en dBi, cuyo valor máximo se conoce simplemente como ganancia (Fuentes, 2007).

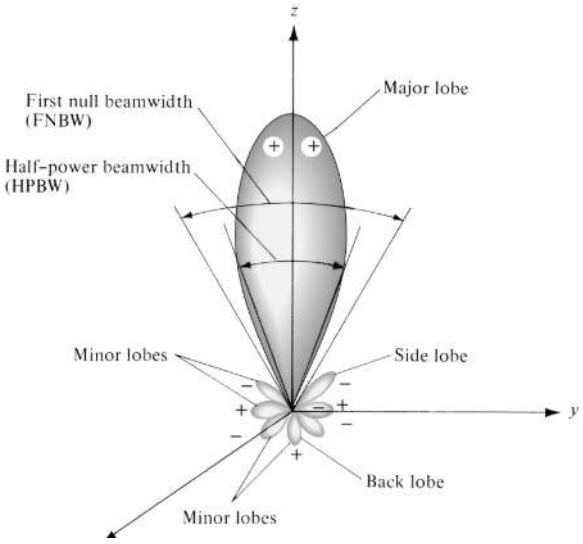


Figura 8: Lóbulos de radiación y anchos de haz de un patrón de antena (Balanis, 2005).

Normalmente, lo que se describe como patrón de radiación es la gráfica de campo eléctrico y magnético. Existen varias obtenciones de los campos dependiendo del tipo de configuraciones que se tengan. Algunos ejemplos son el de las fuentes isotrópicas y el del dipolo, que se forma como una dona.

Fuentes (2007) presenta los vectores de Poynting para este tipo de fuentes. Para una fuente isotrópica, el vector de Poynting promedio se expresa en la ecuación 3:

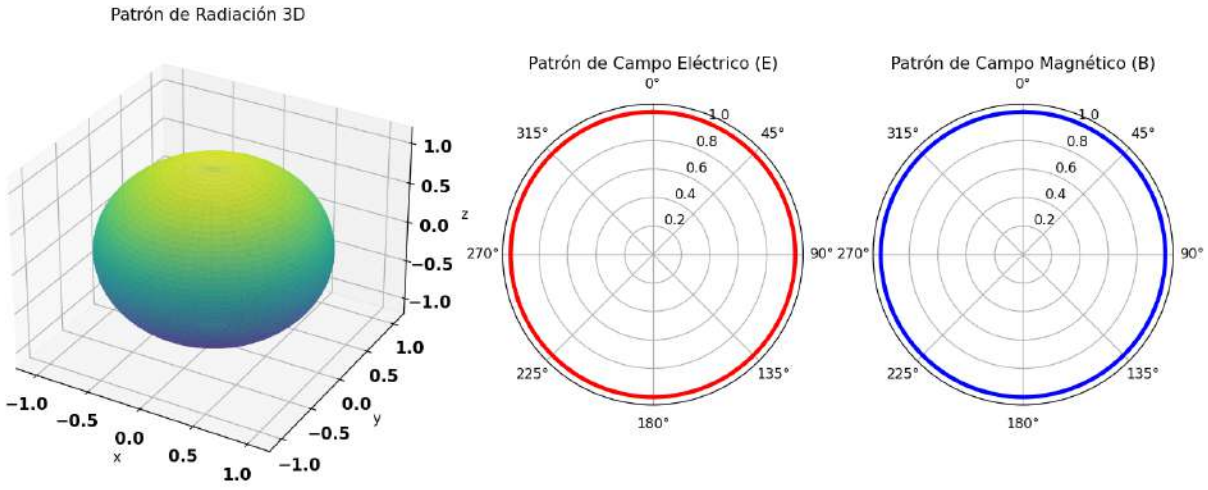


Figura 9: Patrón de radiación de una fuente isotrópica en 3D programada en Python (autoría propia).

$$\langle \vec{S} \rangle_{\text{iso}} = \frac{P_t}{4\pi r^2}\,\hat{r} \tag{3}$$

Para un dipolo eléctrico, el vector de Poynting promedio se expresa en la ecuación 4:

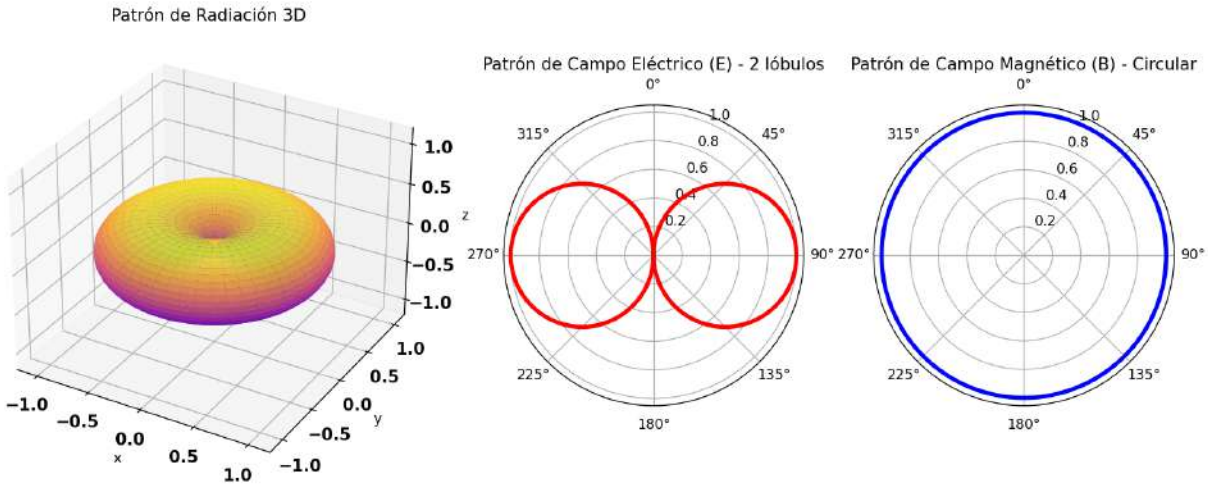


Figura 10: Patrón de radiación de un dipolo en 3D programada en Python (autoría propia).

$$\langle \vec{S} \rangle_{\text{Dipolo}} = \frac{3}{2} \frac{P_t}{4\pi r^2} \sin^2\theta \, \hat{r} \tag{4}$$

Las Figuras 9 y 10 muestran los patrones de radiación tridimensionales obtenidos mediante simulación en Python para una fuente isotrópica y un dipolo eléctrico, respectivamente. En el caso de la fuente isotrópica (Figura 9), el patrón de radiación es esféricamente simétrico, lo que indica que la potencia radiada se distribuye por igual en todas las direcciones. Esto concuerda con la ecuación 3, cuya magnitud depende únicamente de la distancia r y no de los ángulos.

El vector de Poynting y el campo eléctrico son usados de la misma manera para hacer los patrones de radiación, siempre que r sea lo suficientemente grande para estar en la zona lejana del campo eléctrico (Kraus, 1950; Balanis, 2005). La razón matemática del porqué se hace uso de los campos eléctricos se fundamenta en la ecuación 5 (Griffiths, 2013):

$$\vec{S}_{\text{rad}} = \frac{1}{\mu_o c} E_{\text{rad}}^2 \, \hat{r} \tag{5}$$

2.2.6. ¿Cuál es la importancia y el uso de la radiación?

Uno de los aparatos que usamos a diario y hace uso de la radiación electromagnética en la actualidad es el microondas.

- El horno de microondas, percibido desde 1946 como un accidente por parte de Percy Spencer al estar trabajando en investigaciones con el magnetrón, quien

notó que una barra de chocolate que tenía se había derretido. Funciona a partir de ondas electromagnéticas con longitud de onda entre 10^{-2} m y 10^{-3} m, donde las sustancias que tengan dipolos eléctricos verán oscilar sus partículas y se convertirán en calor. Funcionan con el magnetrón que convierte la energía eléctrica en ondas electromagnéticas (Fuentefría et al., 2011).

Otro ejemplo importante es la iluminación; las lámparas, los focos y el alumbrado hacen uso de la radiación electromagnética en el espectro visible (CONUEE, 2021).

El uso de los artefactos médicos es otro ámbito a considerar de importancia:

- La Resonancia Magnética, usada para observar estructuras internas del cuerpo con el uso de fuertes campos magnéticos y ondas de radiofrecuencia. Tiene un imán principal en el sistema de alrededor de 1.5 T a 3 T (Peña, 2025).

- Los rayos X, donde dependiendo la densidad del material por la que pasa se observan diferentes tonos en escala de grises. Huesos y metales aparecen en blanco; aire en pulmones en negro; grasa y músculos en diferentes tonalidades grisáceas (Mayo Clinic, sf).

- Los láseres de uso médico son utilizados para cirugía plástica, tratamientos de daño solar, tratamiento de acné, rejuvenecimiento, depilación láser, eliminación de tatuajes y tratamiento de lesiones vasculares (Martínez-Carpio and Trelles, 2010).

Para la ciencia y tecnologías actuales se pueden mencionar los aceleradores del CERN, computadoras y radares:

- Es importante señalar al físico Henri Becquerel, quien en 1896 descubrió la radiactividad de algunos átomos que emitían energía a partir de partículas y fotones. Se usó la descripción de radiación α, β y γ (Andrade, 1971).

- Rutherford, Marsden y Geiger usaron en 1911 las partículas α como proyectiles para verificar el modelo atómico de Thompson. La construcción de aceleradores requiere de una fuente de voltaje, fuente de iones y electrodos; se basa en la interacción de campos eléctricos sobre la carga de las partículas emitida por la fuente de iones (Andrade, 1971).

Además, el enfoque principal de esta tesis se encuentra en las comunicaciones, dado que se trabaja con antenas. Sus aplicaciones abarcan desde el internet inalámbrico hasta la radio, la televisión, los satélites y el GPS (Matan, 2023):

- Las antenas, así como el wifi, son capaces de recibir y emitir señales que se basan en unos y ceros. El funcionamiento de cualquier antena no es diferente (Brain et al., 2004).

- Las antenas son dispositivos diseñados para recibir o emitir ondas; su uso va también en los teléfonos, mandos y routers. El dipolo tiene la capacidad de captar energía para amplificarla (Huidobro, 2013).

- La energía que debe transmitirse debe serlo con la mayor eficiencia posible para el acoplamiento más óptimo. Se conoce que la línea de transmisión sirve para guiar los campos (Aznar et al., 2004).

2.3. Análisis matemático de la radiación.

La siguiente sección desarrolla la formulación matemática de la radiación electromagnética generada por una carga en trayectoria helicoidal. Para ello, se introduce en primer lugar el concepto de tiempo retardado. Posteriormente, se emplean los potenciales retardados para establecer la relación entre la dinámica de la carga y los campos radiados, lo que finalmente conduce a la expresión de la potencia emitida por el sistema.

2.3.1. Tiempo retardado.

En esta sección se siguen las explicaciones y formulaciones presentadas por Griffiths (2013) en su obra *Introduction to Electrodynamics* y por Pérez Navarro (2011).

Empecemos con que un dipolo no es más que la unión de una carga positiva y otra negativa a cierta distancia, donde se define al momento dipolar en la ecuación 6:

$$\vec{p} = q\,\vec{d} \tag{6}$$

Lo siguiente a mencionar es el tiempo retardado. Al desplazarse la carga, la información electromagnética no se transmite de manera instantánea, sino que se propaga con velocidad finita c. Sea $\mathcal{R}$ la distancia entre la posición de la fuente en el instante de emisión y el punto de observación. Entonces, el tiempo que tarda la señal en propagarse está dado por la ecuación 7:

$$t_{\text{prop}} = \frac{\mathcal{R}}{c} \tag{7}$$

De esta manera, si t es el tiempo de observación y t_r el instante de emisión (tiempo retardado), ambos se relacionan como se expresa en la ecuación 8:

$$t_r = t - \frac{\mathcal{R}}{c} \tag{8}$$

2.3.2. Potenciales retardados.

El potencial escalar temporal se formula en la ecuación 9:

$$\Phi(\mathbf{r}, t) = \frac{1}{4\pi\epsilon_0} \int_V \frac{\rho(\mathbf{r}', t_r)}{|\mathbf{r} - \mathbf{r}'|}\, dV' \tag{9}$$

Mientras que el potencial vectorial se expresa en la ecuación 10:

$$\mathbf{A}(\mathbf{r},t) = \frac{\mu_0}{4\pi}\int_V \frac{\mathbf{J}(\mathbf{r}',t_r)}{|\mathbf{r}-\mathbf{r}'|}\,dV' \tag{10}$$

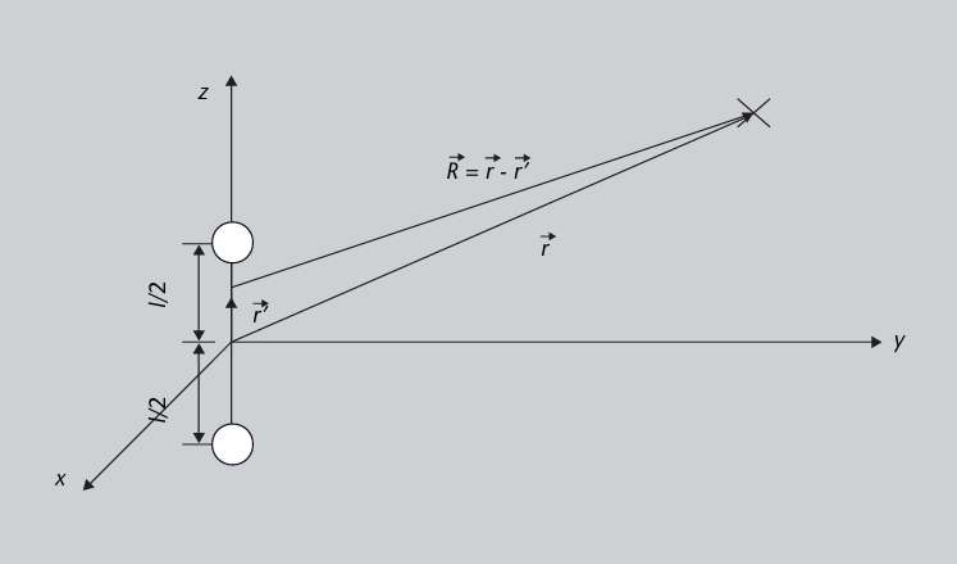


Figura 11: Parámetros para campos retardados (Pérez Navarro, 2011).

Al hablar de cargas puntuales, se llega a los potenciales de Liénard-Wiechert. El potencial vectorial se expresa en la ecuación 11 y el potencial escalar en la ecuación 12:

$$\mathbf{A}(\mathbf{r},t) = \frac{\mu_0}{4\pi}\frac{q\,\mathbf{v}}{(1-\hat{\boldsymbol{\mathcal{R}}}\cdot\boldsymbol{\beta})\mathcal{R}} = \frac{\mathbf{v}}{c^2}\,\phi \tag{11}$$

$$\phi(\mathbf{r},t) = \frac{kq}{(1-\hat{\boldsymbol{\mathcal{R}}}\cdot\boldsymbol{\beta})\mathcal{R}} = \frac{kqc}{\mathcal{R}c-\boldsymbol{\mathcal{R}}\cdot\mathbf{v}} \tag{12}$$

Gracias al uso de la Figura 11 se puede deducir de buena manera el potencial escalar temporal y vectorial.

2.3.3. ¿Cómo se trabaja la radiación?

Con el uso de los potenciales de Liénard-Wiechert es que se puede deducir la fórmula de Larmor, expresada en la ecuación 13:

$$P = \frac{\mu_0 q^2 a^2}{6\pi c} \tag{13}$$

Donde q es la carga, a es la aceleración de la carga, μ_0 es la permeabilidad magnética en el vacío, c es la velocidad de la luz y P corresponde a la potencia radiada.

Primero debemos saber que existen dos maneras de generar radiación:

- Oscilar.
- Acelerar.

En el estudio de la radiación electromagnética es conveniente distinguir tres regiones espaciales, como se muestra en la Figura 12. Para ello, se define r como la distancia desde el centro de la fuente hasta el punto de observación, mientras que d representa la dimensión característica de la región donde se encuentra distribuida la carga (o el radio de la trayectoria en nuestro caso).

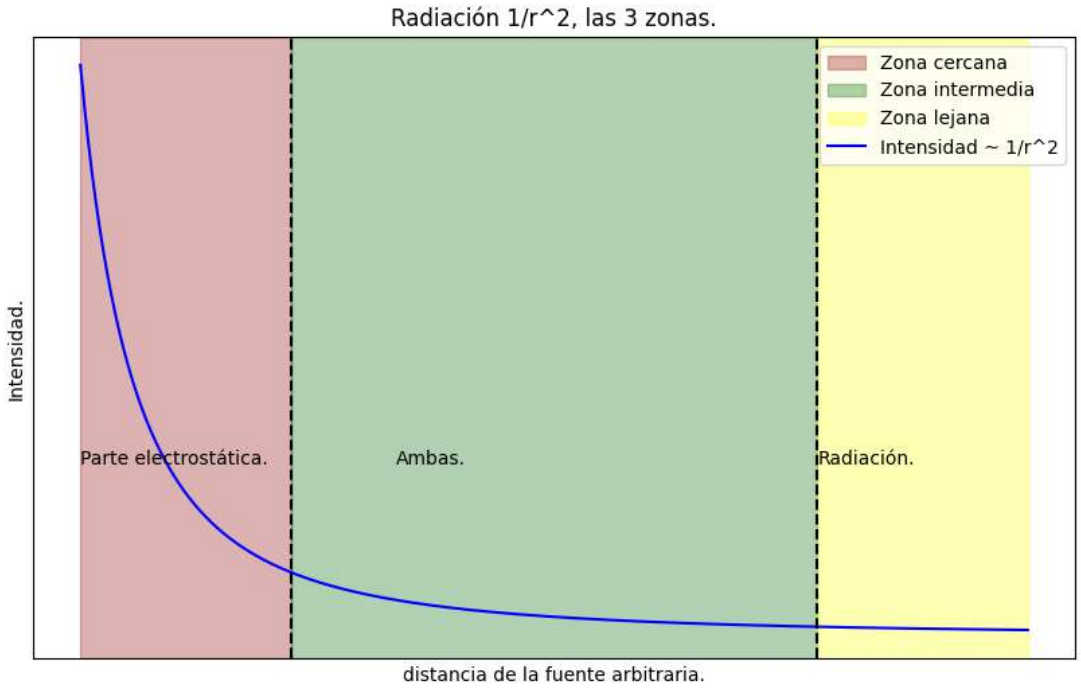


Figura 12: Zonas de radiación en electromagnetismo (autoría propia).

La primera corresponde a la región cercana o zona electrostática, en la cual se cumple que $d \ll r$. En esta región predominan los campos cuasiestáticos.

La segunda región, conocida como zona intermedia o de inducción, satisface la condición $d \ll r \sim \lambda$. Aunque no es el foco principal de este trabajo, es importante mencionarla ya que representa la transición entre los campos cercanos y radiados.

Finalmente, la tercera región corresponde a la zona lejana o de radiación, donde se cumple que $d \ll \lambda \ll r$. En esta zona predominan los campos radiados, los cuales decrecen como $1/r$ en amplitud y como $1/r^2$ en intensidad, siendo esta la región de mayor interés en este análisis.

Manejando la radiación debida a una fuente arbitraria de carga, es necesario definir claramente las distancias involucradas. Usando la Figura 11 como referencia y la ley de cosenos, se deduce la relación geométrica fundamental expresada en la ecuación **??**:

- La distancia desde el punto de observación hasta un elemento de carga dq ubicado en la fuente es $\mathcal{R} = |\mathbf{r} - \mathbf{r}'|$.
- Aquí, $\mathbf{r}$ es el vector de posición desde el origen (centro de la fuente) hasta el punto de observación.
- $\mathbf{r}'$ es el vector de posición desde el origen hasta el elemento de carga dq dentro de la fuente.
- Por lo tanto, $\mathcal{R}$ es la distancia total que debe recorrer la información electromagnética (y es la que se usa en el tiempo retardado $t_r = t - \mathcal{R}/c$).
- Es crucial notar que esta $\mathcal{R}$ es una variable dinámica que depende de $\mathbf{r}$ y $\mathbf{r}'$, y no debe confundirse con el radio constante R_0 de la trayectoria helicoidal definido anteriormente.

Se obtienen las fórmulas en coordenadas esféricas expresadas en las ecuaciones 14, 15 y 16 (Griffiths, 2013):

$$\mathbf{E} = \frac{\mu_0}{4\pi r}\,\ddot{p}\,\sin\theta\,\hat{\boldsymbol{\theta}} \tag{14}$$

$$\mathbf{B} = \frac{\mu_0}{4\pi c r}\,\ddot{p}\,\sin\theta\,\hat{\boldsymbol{\phi}} \tag{15}$$

$$\mathbf{S} = \frac{\mu_0}{16\pi^2 c}(\ddot{p})^2\left(\frac{\sin\theta}{r}\right)^2\hat{\mathbf{r}} \tag{16}$$

2.3.4. Potencia

Para una carga puntual en movimiento arbitrario, el campo eléctrico radiado está dado por la ecuación 17 (Griffiths, 2013):

$$\mathbf{E}_{\text{rad}} = \frac{q}{4\pi\epsilon_0}\frac{\mathcal{R}}{(\mathbf{r}\cdot\mathbf{u})^3}[\mathbf{r}\times(\mathbf{u}\times\mathbf{a})] \tag{17}$$

2.4. Parámetros Fundamentales de una Antena

Se introducen los parámetros fundamentales que permiten caracterizar el comportamiento de una antena desde el punto de vista físico e ingenieril. Se comienza con los conceptos de directividad y ganancia, los cuales describen cómo se distribuye la potencia radiada en el espacio y qué tan eficiente es la antena al concentrarla en una dirección específica. Posteriormente, se analiza el ancho de haz de media potencia (HPBW), que

permite cuantificar la apertura angular del lóbulo principal de radiación. También se estudian la eficiencia y las pérdidas, destacando los mecanismos físicos que limitan el rendimiento real de una antena. Asimismo, se aborda la relación entre frecuencia y longitud de onda, esencial para comprender el funcionamiento en distintas bandas del espectro electromagnético. Finalmente, se incluyen conceptos clave como la impedancia, que determina la correcta transferencia de energía, y la polarización, que describe la orientación del campo eléctrico radiado y su impacto en la transmisión y recepción de señales.

2.4.1. Directividad y ganancia.

Para poder llegar a este concepto se necesita primero un concepto elemental ya mencionado: la potencia radiada. Para fuentes isotrópicas en electrodinámica se conoce como la integral de la ecuación **??**. Se transforma en la ecuación 18 cuando la fuente es isotrópica (Griffiths, 2013; Kraus, 1950):

$$W = P_r \, 4\pi r^2 \tag{18}$$

Se llega al concepto de intensidad de radiación, que es la potencia por unidad de ángulo sólido, expresada en la ecuación 19:

$$U = P_r r^2 \tag{19}$$

Para una fuente isótropa, de la ecuación 18 y 19 se obtiene la ecuación 20:

$$W = 4\pi U_o \tag{20}$$

Donde U_o es la potencia por radián cuadrado. La intensidad de radiación del hemisferio norte de una esfera se conoce como U_m; la del hemisferio sur es cero. La intensidad de radiación se reescribe en la ecuación 21 (Kraus, 1950):

$$W = \int_0^{2\pi} \int_0^{\pi/2} U_m \sin\theta \, d\theta \, d\phi = 2\pi U_m \tag{21}$$

Al igualar las ecuaciones 20 y 21 se obtiene la Directividad expresada en la ecuación 22:

$$D = \frac{U_m}{U_o} \tag{22}$$

Una vez que ya se conoce la Directividad, es necesario conocer la Ganancia, se define en la ecuación 23 (Kraus, 1950):

$$G_0 = \frac{\text{intensidad máxima de radiación de la antena en estudio}}{\text{intensidad de radiación isotrópica (sin pérdidas), misma potencia de entrada}} \tag{23}$$

Poniendo al valor máximo de intensidad de radiación subjetiva de una antena como U'_m y al de máxima eficiencia como U_m, e implementando un factor de eficiencia de radiación k, se llega a la ecuación 24:

$$U'_m = k\,U_m, \qquad 0 \leq k \leq 1 \tag{24}$$

De donde se obtiene la relación entre ganancia y directividad en la ecuación 25:

$$G_o = \frac{U'_m}{U_o} = \frac{k\,U_m}{U_o} = kD \tag{25}$$

Expresados en decíbeles, la directividad y ganancia toman la forma de las ecuaciones 26 y 27:

$$D\,(\text{dB}) = 10\log_{10} D \tag{26}$$

$$G\,(\text{dB}) = 10\log_{10} G \tag{27}$$

2.4.2. Ancho de haz de media potencia (HPBW).

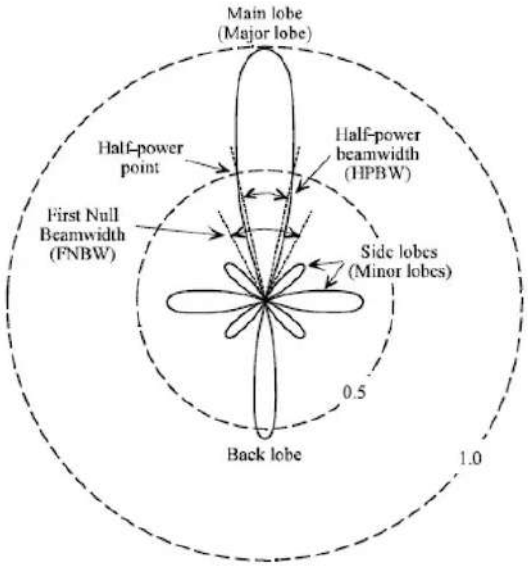


Figura 13: Descripción de lóbulos en el patrón de radiación World (2023).

Para poder desglosar este tema, lo más fundamental es describir lo que se está tratando en el patrón de radiación. ¿Qué incluye cada uno? (World, 2023), se muestra en la Figura 13:

- Lóbulo principal. Aquí se describe la principal dirección en que una antena radia; existe mayor potencia.

- Lóbulo menor. Todo aquel que no sea el principal.
- Lóbulo lateral. Es un lóbulo que existe en una dirección no prevista.
- Lóbulo posterior. Es el lóbulo que está en el lado opuesto del principal.

El HPBW es un término expresado en grados que describe la medida del lóbulo principal en puntos de media potencia; es la separación angular en la que la radiación disminuye al 50 % del pico al haz principal (RF, 2023).

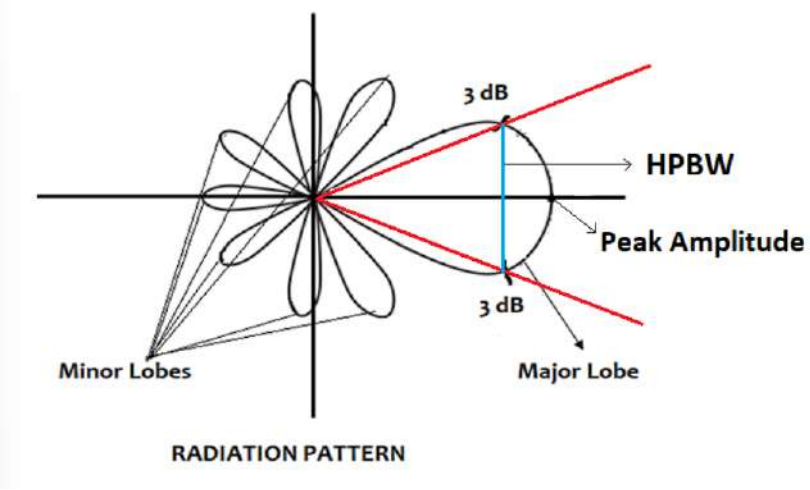


Figura 14: ¿Cómo se ve HPBW? (RF, 2023).

En la Figura 14 se muestra la representación del ancho de haz de media potencia (HPBW, por sus siglas en inglés: Half Power Beam Width). En dicha gráfica se observa el lóbulo principal del patrón de radiación, cuya máxima amplitud corresponde a la dirección donde la antena irradia mayor potencia. Los puntos señalados a -3 dB respecto al máximo indican el nivel en el cual la potencia se reduce al 50 % de su valor máximo, ya que una disminución de 3 dB equivale a la mitad de la potencia en escala logarítmica. El HPBW se define entonces como la separación angular entre estos dos puntos de media potencia, delimitando la región donde se concentra la mayor parte de la energía radiada. Este parámetro es fundamental para caracterizar la directividad de la antena, ya que un menor ancho de haz implica una mayor concentración de potencia en una dirección específica.

Como se observa en la Figura 15, la representación en decíbeles permite identificar la variación de la potencia con el ángulo θ. En particular, el nivel de -3 dB delimita los puntos de media potencia respecto al máximo del lóbulo principal, lo cual permite determinar el HPBW. Este concepto no debe confundirse con el ancho de banda, el cual describe el rango de frecuencias en el que una antena opera de manera eficiente y se mide en Hz (Y, 2024).

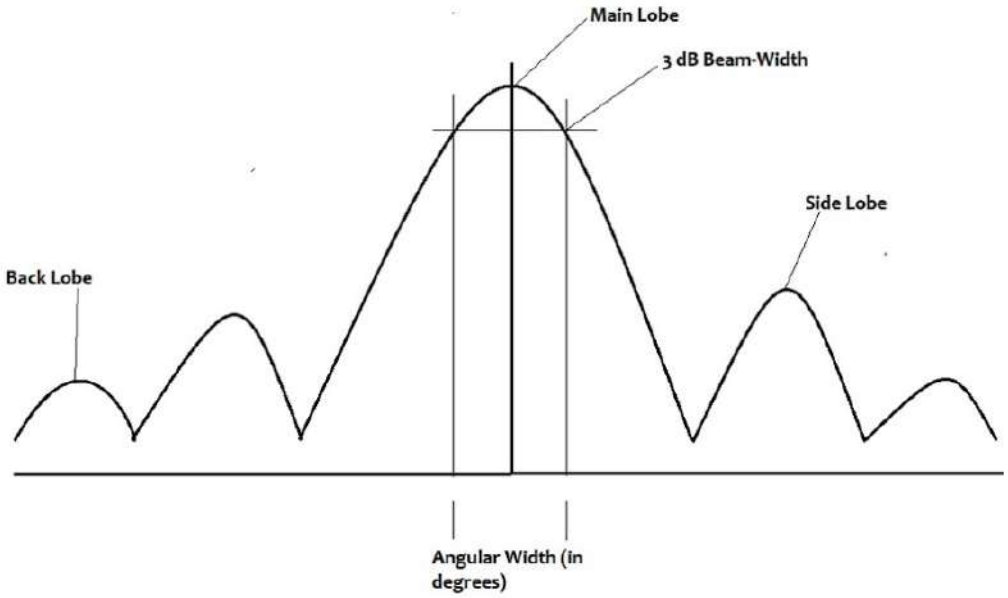


Figura 15: Intensidad de radiación en función del ángulo θ en decibeles, mostrando la referencia de -3 dB para determinar el HPBW (RF, 2023).

2.4.3. Eficiencia y pérdidas.

La relación de potencia radiada se define en la ecuación 28:

$$\eta = \frac{P_r}{P_{\text{en}}} \tag{28}$$

Esta expresión representa un porcentaje del 0 al 100 %, donde el 0 indica que no radia y el 100 indica el mejor funcionamiento(Matan, 2024).

El porcentaje de eficiencia también puede descomponerse en sus factores de pérdida, como se expresa en la ecuación 29:

$$\eta = \eta_{\text{cond}}\, \eta_{\text{die}} \tag{29}$$

Combinando las ecuaciones 22, 25 y 29, la ganancia que no incluye la pérdida de impedancia se calcula con las ecuaciones 30 y 31:

$$G = \eta \cdot D \tag{30}$$

$$G_R = \eta_R \cdot G \tag{31}$$

Para poder explicar el concepto de tangente de pérdida primero se necesita definir la constante dieléctrica compleja en la ecuación 32 (Blattenberger, 2025):

$$\epsilon = \epsilon_R + j\epsilon_I \tag{32}$$

Entonces la tangente de pérdida se define como un factor de disipación en la ecuación 33:

$$\tan\delta = \frac{\epsilon_I}{\epsilon_R} \tag{33}$$

Esta relación representa la proporción de energía perdida por ciclo respecto a la energía almacenada. A menor valor, menores pérdidas.

2.4.4. Impedancia.

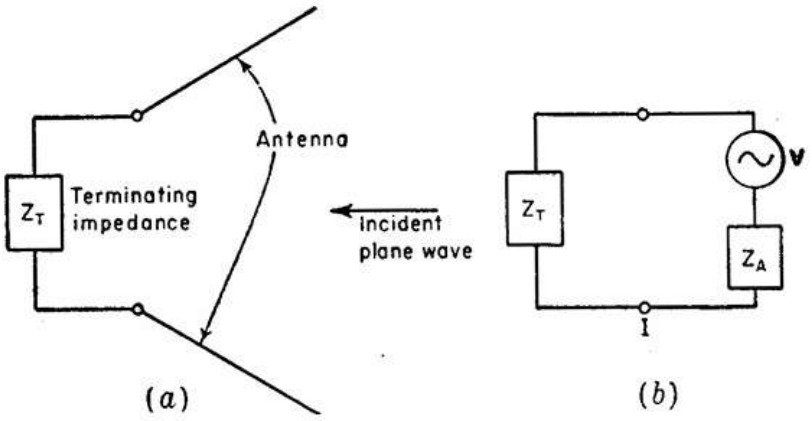


Figura 16: Diagrama de una antena con impedancia Z_T de incidencia de onda plana sobre la antena y su circuito correspondiente (Kraus, 1950).

En el circuito de la Figura 16, la antena obtiene su potencia desde la incidencia de una onda y la dirige hacia la terminal de la impedancia Z_T. La antena se reemplaza por un generador equivalente con un voltaje equivalente y una impedancia de la antena Z_A. La corriente resultante se expresa en la ecuación 34 (Kraus, 1950):

$$I = \frac{V}{Z_T + Z_A} \tag{34}$$

Las impedancias se consideran números complejos, expresados en las ecuaciones 35 y 36:

$$Z_T = R_T + jX_T \tag{35}$$

$$Z_A = R_A + jX_A \tag{36}$$

Donde X es la autoreactancia y la resistencia R_A de la antena tiene dos partes: la resistencia de radiación y la resistencia de pérdida, como se expresa en la ecuación 37:

$$R_A = R_r + R_I \tag{37}$$

2.4.5. Polarización

La polarización se define como aquella curva trazada por el campo eléctrico instantáneo radiado por una antena en un plano perpendicular en dirección radial. Se caracteriza por el radio axial, el sentido de la rotación y el ángulo de inclinación τ (Balanis, 2005). El ángulo transmitido τ_t y el recibido τ_m se relacionan en la ecuación 38:

$$\tau_t = 180^\circ - \tau_m \tag{38}$$

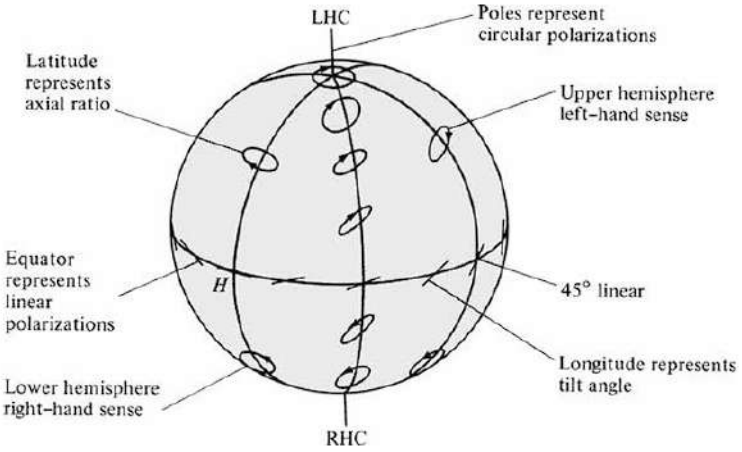
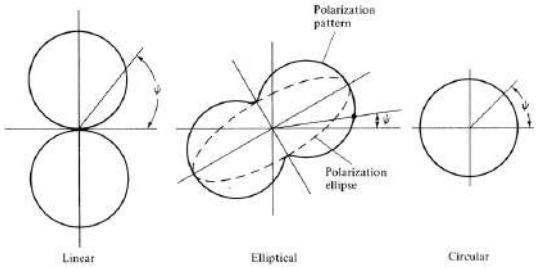


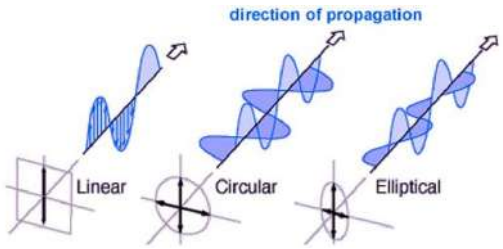
Figura 17: Representación de la polarización en la esfera de Poincaré (Balanis, 2005).

En la Figura 17 se puede observar cómo la polarización de una antena se puede apreciar mejor en la superficie de una esfera de Poincaré (Balanis, 2005). Si se usa un punto para representar la polarización de la onda incidente y otro para la antena receptora, el ángulo entre ambos es usado para determinar las pérdidas por polarización.

Mientras que en la Figura 18 se observa la forma de las polarizaciones más comunes (Hardesty, 2024), se describen algunos tipos comunes:



(a) Tipos de polarización más comunes (Balanis, 2005).



(b) Otra forma de ver las polarizaciones (Hardesty, 2024).

Figura 18: Tipos de polarización.

- Polarización lineal: oscilación de E en el plano horizontal o vertical.
- Polarización oblicua: oscilación de 45° respecto al plano horizontal o vertical.
- Polarización circular: onda que gira a medida que la señal es propagada.
- Polarización elíptica: E se propaga en forma de hélice elíptica.

Si vamos a una parte más matemática, se manejan a los campos eléctricos en dos coordenadas. Tomando las coordenadas x e y, el campo se expresa en las ecuaciones 39 y 40 (Saleh, 2019; Hecht, 2017):

$$E(x,t) = E_0 \exp[i(kx - \omega t)] \tag{39}$$

$$E(y,t) = E_0 \exp[i(ky - \omega t)] \tag{40}$$

Si la polarización es lineal solo existe una de estas ecuaciones. A otro grado, se toma la forma de las ecuaciones 41 y 42:

$$E_x(z,t) = \mathrm{Re}\{E_0 \cos\alpha \, \exp[i(kz - \omega t)]\} \tag{41}$$

$$E_y(z,t) = \mathrm{Re}\{E_0 \sin\alpha \, \exp[i(kz - \omega t)]\} \tag{42}$$

Un término importante para la clasificación de las polarizaciones es el vector de Jones (Collett, 2005), que se expresa como la ecuación 43:

$$\mathbf{E} = \begin{bmatrix} E_x \\ E_y \end{bmatrix} \tag{43}$$

Las polarizaciones circulares se expresan en las ecuaciones 44 y 45:

$$E_x(z,t) = \mathrm{Re}\{E_0 \exp[i(kz - \omega t)]\} = |E_0| \cos(kz - \omega t) \tag{44}$$

$$E_y(z,t) = \mathrm{Re}\{\pm i \, E_0 \exp[i(kz - \omega t)]\} = \mp |E_0| \sin(kz - \omega t) \tag{45}$$

Si va a la derecha se usa signo menos; si va a la izquierda se usa signo más. Al usar el vector de Jones de la ecuación 43, se llega a:

$$\frac{\mathbf{E}}{E_x} = \begin{bmatrix} 1 \\ \pm i \end{bmatrix} \tag{46}$$

Para las polarizaciones elípticas se tienen las ecuaciones 47 y 48:

$$E_x(z,t) = \mathrm{Re}\{E_{0x} \exp[i(kz - \omega t)]\} = |E_{0x}| \cos(kz - \omega t - \theta_x) \tag{47}$$

$$E_y(z,t) = \mathrm{Re}\{E_{0y} \exp[i(kz - \omega t)]\} = |E_{0y}| \cos(kz - \omega t - \theta_y) \tag{48}$$

Y el vector de Jones de la ecuación 43 se reescribiría como la ecuación 49:

$$\frac{\mathbf{E}}{E_x} = \begin{bmatrix} 1 \\ \dfrac{E_{0y}}{E_{0x}} \exp\{i\,[\theta_x(t) - \theta_y(t)]\} \end{bmatrix} \tag{49}$$

3.. Fundamentación teórica

3.1. ¿Por qué se emiten ondas electromagnéticas en la naturaleza?

La emisión de ondas electromagnéticas tiene su origen en las transiciones de energía que ocurren a nivel atómico y molecular. De acuerdo con la mecánica cuántica, los electrones sólo pueden ocupar niveles de energía discretos, por lo que al pasar de un estado a otro intercambian energía en forma de radiación electromagnética. Este proceso explica fenómenos fundamentales como la emisión de luz y permite entender mecanismos como la absorción, la emisión espontánea y la emisión estimulada, los cuales describen cómo la materia interactúa con los campos electromagnéticos.

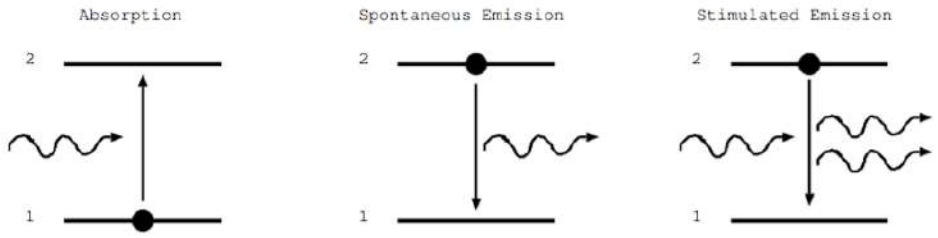


Figura 19: Absorción, emisión espontánea, emisión estimulada (Mitofsky, 2025).

Si bien este tema no es estrictamente necesario para el desarrollo del trabajo, sí es fundamental para el entendimiento. La naturaleza de los átomos y moléculas contiene vibraciones que en mecánica cuántica se conocen como niveles de energía. Los saltos de energía que van de un estado excitado al estado fundamental, o viceversa como se ve en la ecuación 50 (Saleh, 2019):

$$\Delta E = \hbar\nu \tag{50}$$

donde ν es la frecuencia con la que el átomo vibra.

La mayor parte de la luz se emite por vibraciones atómicas (Mitofsky, 2025). Como se ilustran en la Figura 19:

- El primer fenómeno es la emisión espontánea: un electrón decae al estado fundamental por sí solo y emite un fotón.
- El segundo es la absorción: un electrón en estado fundamental interactúa con un campo electromagnético, absorbe el fotón y transita a un estado excitado.
- El tercero es la emisión estimulada: un electrón ya excitado interactúa con un campo electromagnético; el fotón estimula la transición al estado fundamental y hace que se emita un segundo fotón con la misma frecuencia, dirección, fase y polarización. Este mecanismo es la base de los láseres.

3.2. La antena helicoidal.

En esta sección se describe la configuración física de la antena helicoidal, poniendo énfasis en la geometría de su trayectoria. Esta descripción es fundamental, ya que permite modelar el movimiento de la carga y establecer las bases para el análisis de los campos electromagnéticos que genera. Por otro lado, esta antena se destaca por su capacidad de operar con polarización circular y generar patrones de radiación altamente directivos en ciertas condiciones. Su diseño se basa en una geometría en forma de hélice, cuyas dimensiones están directamente relacionadas con la longitud de onda de operación. A continuación, se describen los principios de funcionamiento, componentes y condiciones que permiten optimizar su desempeño.

3.2.1. ¿Cómo es su geometría?

Se inicia con la parametrización diferenciable (Kuhnel, 2015), expresada en la ecuación 51:

$$\alpha(t) = \big(a\cos(\omega t),\ b\sin(\omega t),\ vt\big), \qquad t \in \mathbb{R} \tag{51}$$

Su límite queda dentro de un cilindro trazado por la ecuación 52:

$$x^2 + y^2 = a^2 \tag{52}$$

A esta superficie se le denomina helicoide, y su parametrización está dada por la ecuación 53:

$$\mathbf{x}(u,v) = (v\cos u,\ v\sin u,\ au), \qquad 0 < u < 2\pi,\ -\infty < v < \infty \tag{53}$$

Los coeficientes de la primera forma fundamental asociados a esta parametrización se presentan en la ecuación 54:

$$E(u,v) = v^2 + a^2, \qquad F(u,v) = 0, \qquad G(u,v) = 1 \tag{54}$$

Estos coeficientes permiten estudiar las propiedades métricas de la superficie. En particular, E y G están relacionados con la variación de los vectores tangentes en las direcciones de u y v, mientras que $F = 0$ indica que dichas direcciones son ortogonales entre sí, lo que refleja la naturaleza reglada y ortogonal de la parametrización del helicoide. En la Figura 20 se muestra la representación gráfica de esta superficie.

3.2.2. ¿Qué campos se obtuvieron?

Para describir la geometría de la trayectoria helicoidal de una carga en movimiento, es necesario definir la magnitud $r(t)$, la cual representa la distancia radial desde el origen hasta la posición instantánea de la partícula. Para una trayectoria helicoidal, la posición de la partícula puede descomponerse en un movimiento circular en el plano transversal

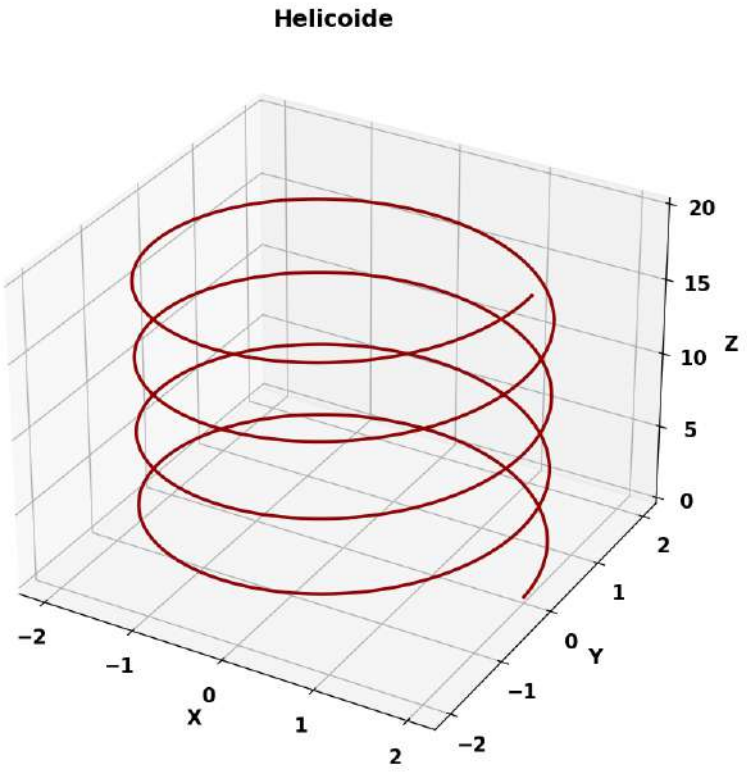


Figura 20: Representación gráfica del helicoide con la superficie reglada generada por una línea recta que se desplaza helicoidalmente alrededor de un eje (autoría propia).

y un movimiento uniforme a lo largo del eje longitudinal. La expresión general de la distancia radial se presenta en la ecuación 55:

$$r(t) = \sqrt{R_0^2 \cos^2(\omega t) + R_0^2 \sin^2(\omega t) + (vt)^2} \tag{55}$$

Simplificando la expresión de la ecuación 55 se obtiene la ecuación 56:

$$r(t) = \sqrt{R_0^2 + (vt)^2} \tag{56}$$

Esta expresión muestra que la distancia radial resulta de la combinación de un radio constante R_0 asociado al movimiento circular y un término creciente en el tiempo debido al desplazamiento longitudinal. A partir de la ecuación 56, se obtienen sus derivadas temporales, expresadas en las ecuaciones 57 y 58:

$$\dot{r}(t) = \frac{v^2 t}{\sqrt{R_0^2 + (vt)^2}} \tag{57}$$

$$\ddot{r}(t) = \frac{v^2 R_0^2}{(R_0^2 + v^2 t^2)^{3/2}} \tag{58}$$

El momento dipolar eléctrico del sistema se expresa en la ecuación 59:

$$\vec{P}(t) = q\, r(t)\, \hat{r} = q\sqrt{R_0^2 + (vt)^2}\, \hat{r} \tag{59}$$

La segunda derivada temporal del momento dipolar, que es la cantidad relevante en la radiación, resulta en la ecuación 60:

$$\vec{P}''(t) = \frac{q\, v^2 R_0^2}{(R_0^2 + v^2 t^2)^{3/2}}\, \hat{r} \tag{60}$$

A partir de la formulación de Larmor para radiación de dipolos (Griffiths, 2013), se obtienen los campos eléctrico y magnético en la zona de radiación. El campo eléctrico se expresa en la ecuación 61 y el campo magnético en la ecuación 62:

$$\vec{E} = -\frac{\mu_o}{4\pi} \frac{q\, v^2 R_0^2}{(R_0^2 + v^2 t^2)^{3/2}} \frac{\sin\theta}{r}\, \hat{\theta} \tag{61}$$

$$\vec{B} = \frac{\mu_o}{4\pi c} \frac{q\, v^2 R_0^2}{(R_0^2 + v^2 t^2)^{3/2}} \frac{\sin\theta}{r}\, \hat{\phi} \tag{62}$$

Las ecuaciones 61 y 62 muestran que los campos radiados dependen de la aceleración del sistema (contenida en la ecuación 60) y presentan la dependencia angular característica $\sin\theta$, propia de la radiación dipolar. El flujo de energía electromagnética se describe mediante el vector de Poynting en la ecuación 63:

$$\vec{S} = \frac{\mu_o}{16\pi^2 c} \frac{q^2 v^4 R_0^4}{(R_0^2 + v^2 t^2)^3} \frac{\sin^2\theta}{r^2}\, \hat{r} \tag{63}$$

Finalmente, integrando la ecuación 63 sobre toda la superficie sólida, se obtiene la potencia radiada en la ecuación 64:

$$P = \frac{\mu_o}{6\pi c} \frac{q^2 v^4 R_0^4}{(R_0^2 + v^2 t^2)^3} \tag{64}$$

3.2.3. ¿Para qué está diseñada una antena helicoidal?

El mejor funcionamiento de una antena helicoidal corresponde al primer modo de transmisión (Kraus, 1950), mediante la condición de la ecuación 65:

$$0.75 < \frac{C}{\lambda} < 1.25, \qquad S < \frac{\lambda}{4} \tag{65}$$

Las características que describen qué longitud de onda puede transmitirse se expresan mediante la aproximación de la ecuación 66:

$$C \approx \lambda \tag{66}$$

Para la aproximación más simple se hace un barrido de frecuencias (normalmente de 100 MHz a 3 GHz), verificando que se cumpla la condición de la ecuación 65 para que la

frecuencia sea óptima. Cabe aclarar que la frecuencia óptima no es con la que opera la antena, sino la frecuencia con la que opera mejor si se la alimenta con ella. Fuera de ese rango se puede operar con cualquier frecuencia, pero se obtendrán patrones diferentes. Para saber con qué frecuencia opera una antena determinada se requiere conectar un analizador de redes para la medición de las resonancias, o bien construir el circuito sintonizador que sirve como filtro pasa-bandas.

3.2.4. Componentes.

Los componentes del circuito de una antena son los siguientes:

- Fuente: Ya tiene una frecuencia determinada (en México es de 60 Hz).
- Receptor: Recibe señal inducida.
- Circuito de adaptación (RLC): Este circuito es útil para ajustar la impedancia; lo que se busca es que la señal se emita como onda electromagnética en la antena y no se regrese por el circuito. Esto se logra con la condición de la ecuación 67:

$$Z_{\text{antena}} + Z_{\text{adaptación}} \approx Z_{\text{transmisor}} \tag{67}$$

  - Bobinas (L): aportan la reactancia inductiva.
  - Capacitores (C): aportan la reactancia capacitiva.
  - Resistencias (R): controlan el ancho de banda.

- Antena: Puede ser cualquiera, dependiendo de lo que se desee lograr.
- Alimentación: Es la conexión de la antena con la fuente (normalmente una guía de onda con una impedancia característica de $50\,\Omega$–$75\,\Omega$).

3.2.5. Circuito de un Sintonizador.

Cuando se trata de corriente alterna, el voltaje y la corriente del sistema no son constantes. El voltaje y la corriente se expresan en las ecuaciones 68 y 69:

$$V(t) = V_o \sin(2\pi f t) \tag{68}$$

$$I(t) = I_o \sin(2\pi f t) \tag{69}$$

Para averiguar la frecuencia se usa un osciloscopio y se observa el periodo para deducir la ecuación 70:

$$f = \frac{1}{T} \tag{70}$$

Es importante señalar que en el momento de transmisión de la antena existen varias frecuencias en la propia corriente. Normalmente, para filtrarlas se emplean circuitos RLC. La impedancia total del sistema se expresa en la ecuación 71:

$$Z = R_{\text{sistema}} + Z_{\text{antena}} + jX \tag{71}$$

Se tienen varios conceptos eléctricos con sus fórmulas (ALLELCO, 2025; Alexander, 2018). La capacitancia está dada por la ecuación 72:

$$C = \frac{Q}{V} \tag{72}$$

Donde Q es la carga del capacitor. La inductancia viene dada por la ecuación 73:

$$L = \frac{\Phi}{I} \tag{73}$$

donde Φ es el flujo magnético. La resistencia se calcula con la ley de Ohm en la ecuación 74:

$$R = \frac{V}{I} \tag{74}$$

Para un circuito RLC en serie, la impedancia compleja se expresa en la ecuación 75:

$$Z(\omega) = R + j\left(\omega L - \frac{1}{\omega C}\right) \tag{75}$$

La parte imaginaria de la impedancia es la reactancia, expresada en la ecuación 76:

$$X(\omega) = \omega L - \frac{1}{\omega C} \tag{76}$$

Si se conocen los valores de L y C, así como la reactancia X, la ecuación 76 conduce a la ecuación diferencial 77:

$$\omega^2 LC - \omega XC - 1 = 0 \tag{77}$$

cuya solución es la ecuación 78:

$$\omega = \frac{XC \pm \sqrt{(XC)^2 + 4LC}}{2LC} \tag{78}$$

Una vez conocida la frecuencia angular a partir de la ecuación 78, la frecuencia se calcula con la ecuación 79:

$$f = \frac{\omega}{2\pi} \tag{79}$$

Lo que se busca en una antena es una resonancia eficiente, que sólo se logra cuando el sistema está en resonancia, condición expresada en la ecuación 80:

$$X_L = X_C \tag{80}$$

Si queremos producir longitudes de onda más largas, se requieren resonancias más bajas, y viceversa. Cuando $X = 0$, la solución de la ecuación 77 conduce primero a la frecuencia angular de resonancia en la ecuación 81:

$$\omega_o = \frac{1}{\sqrt{LC}} \tag{81}$$

y, de manera equivalente, a la frecuencia de resonancia en la ecuación 82:

$$f_o = \frac{1}{2\pi\sqrt{LC}} \tag{82}$$

Para conocer si la reactancia es la adecuada, se expresa la impedancia en forma polar en la ecuación 83:

$$Z = |Z|\, e^{j\psi}, \qquad X = |Z| \sin\psi, \quad R = |Z| \cos\psi \tag{83}$$

Sabiendo la impedancia y la resistencia del sistema, la reactancia puede deducirse de la ecuación 83 como se muestra en la ecuación 84:

$$X = \pm\sqrt{|Z|^2 - R^2} \tag{84}$$

3.2.6. Relaciones de longitudes de onda.

Todas las antenas gozan de un ancho de banda que pueden transmitir o recibir, por lo que existe un rango ideal para cada tipo. En el caso de la antena helicoidal (Kraus, 1950), para el primer modo de transmisión conforme a la condición de la ecuación 65:

- $C/\lambda < 0.75$: la antena opera en modo normal (radiación broadside). La corriente en la hélice está en fase. Similar a un dipolo; baja ganancia; polarización casi lineal. Usado para comunicaciones locales.

- $0.75 < C_\lambda < 1.25$ con $S < \lambda/4$: la antena opera en modo axial (radiación end-fire). Corriente en desfase progresivo; máxima ganancia; se forma un haz en la dirección del eje. Polarización circular. Puede ser usado para satélites.

- $C/\lambda > 1.25$: el patrón se degrada y aparecen lóbulos múltiples. La radiación se dispersa, hay pérdida de directividad; baja eficiencia de radiación; polarización elíptica o irregular.

3.3. Radiación y modos de transmisión en las hélices.

El comportamiento electromagnético de una antena helicoidal puede analizarse a partir de dos enfoques complementarios: la propagación de ondas a lo largo de la estructura (modo de transmisión) y la forma en que esta energía es radiada al espacio (modo de radiación). La relación entre ambos permite comprender cómo la geometría de la hélice determina el tipo de patrón radiado, la polarización y la directividad. En particular, el estudio de estos modos resulta fundamental para identificar las condiciones bajo las cuales la antena presenta un desempeño óptimo.

El modo de transmisión se usa cuando una onda electromagnética es propagada en una hélice infinita dentro de una guía de onda ideal. Por otro lado, el modo de radiación describe el patrón de radiación en campo lejano de la antena helicoidal (Kraus, 1950). Como se mencionó previamente, existen dos modos principales: el modo normal, asociado a R_0, y el modo axial, asociado a R_1.

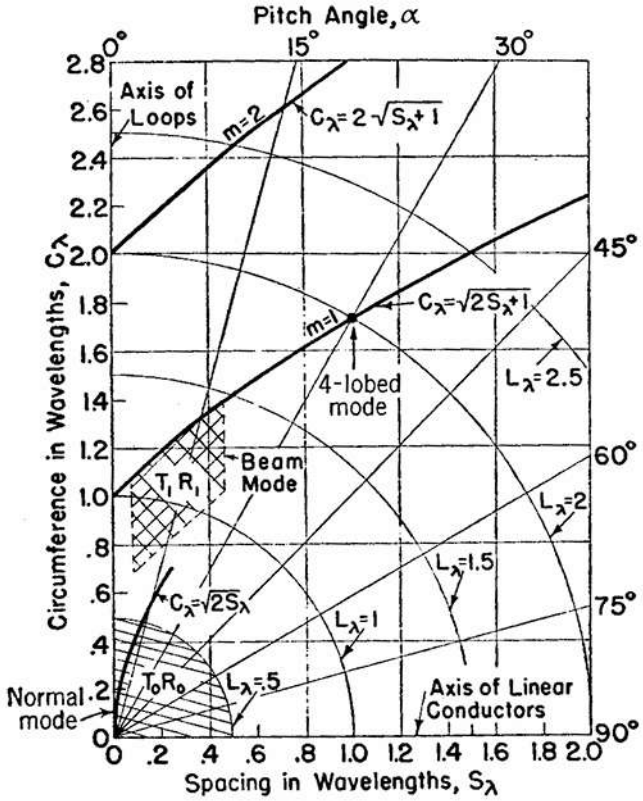


Figura 21: Regiones de los diferentes modos de operación (Kraus, 1950).

En la Figura 21 se presenta el diagrama propuesto por Kraus (1950), el cual permite identificar los distintos modos de operación de la antena en función de dos parámetros normalizados: la circunferencia respecto a la longitud de onda ($C_\lambda = C/\lambda$) y la separación entre espiras ($S_\lambda = S/\lambda$). El eje horizontal corresponde a S_λ, mientras que el eje vertical representa C_λ.

Dentro de este diagrama, la región inferior izquierda, identificada como T_0R_0, corresponde al modo normal de operación. En esta zona, la corriente en la hélice se encuentra prácticamente en fase, por lo que el patrón de radiación es similar al de un dipolo, con baja directividad y polarización principalmente lineal.

En contraste, la región central, correspondiente a T_1R_1, describe el modo axial. En este caso, la corriente presenta un desfase progresivo a lo largo de la hélice, lo que da lugar a un patrón de radiación altamente directivo en la dirección del eje (*end-fire*), con polarización circular. Esta es la región de mayor interés en aplicaciones prácticas.

Las demás regiones del diagrama representan transiciones hacia comportamientos más complejos, donde aparecen múltiples lóbulos y la predicción del patrón de radiación se vuelve más difícil.

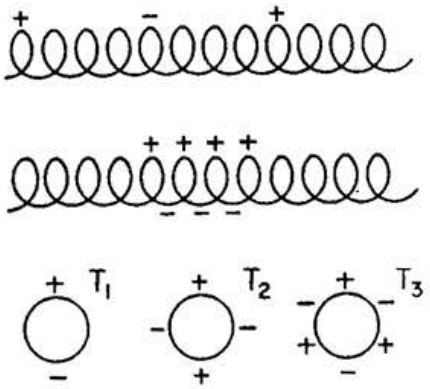


Figura 22: Distribuciones de carga en la helicoide (Kraus, 1950).

La Figura 22 complementa este análisis al mostrar cómo se distribuyen las cargas a lo largo de la hélice. Estas distribuciones pueden interpretarse como arreglos de dipolos, cuadripolos u órdenes superiores, dependiendo de la geometría y de la relación entre la longitud de onda, la circunferencia y el espaciamiento entre espiras. Cuando las contribuciones longitudinales y circunferenciales de la corriente se encuentran en proporciones adecuadas (particularmente cuando $0.75 < C_\lambda < 1.25$, conforme a la condición de la ecuación 65), se favorece la formación de un patrón bien definido.

Los valores designados con el subíndice 1 corresponden al caso en que la circunferencia es del orden de una longitud de onda ($m = 1$), cuyo patrón es de polarización circular. Existen dos campos calculados: uno para ϕ (plano a la página) y otro para θ (normal a la página); bajo los parámetros de T_1R_1 las diferencias son mínimas, lo que da lugar a la polarización circular. En general, para $m \geq 2$, los patrones presentan múltiples lóbulos o características cónicas, como se puede ver en la Figura 23.

El campo de la helicoide es en general elípticamente polarizado; sólo bajo ciertos parámetros la polarización que se presenta es circular o lineal. Cabe señalar que los modos T_0 se atenúan rápidamente, mientras que los modos de onda para T_1 son relativamente constantes en amplitud.

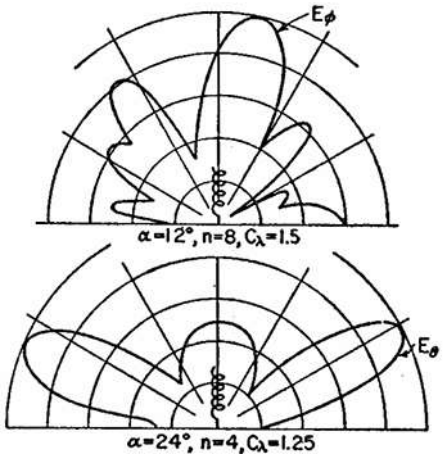


Figura 23: Patrones multilóbulos y cónicos (Kraus, 1950).

3.4. Deducción del patrón de radiación.

La deducción del patrón de radiación de la antena helicoidal se fundamenta en el modelado de la estructura como un arreglo de fuentes elementales, lo que permite describir su comportamiento en el campo lejano mediante herramientas de teoría de antenas. En este contexto, resulta esencial emplear el principio de multiplicación de patrones, el cual establece una relación directa entre el patrón de una fuente individual y el factor de arreglo asociado a su distribución espacial.

3.4.1. El principio de multiplicación.

Para llegar a la deducción del patrón de radiación se debe saber que la antena tiene varias posibilidades de ser tratada: como una fuente puntual, como una apertura o, de manera general, como un arreglo de fuentes puntuales.

Se llega al **principio de multiplicación de patrones**, cuyo enunciado es: "El patrón de campo total de un arreglo de fuentes no isotrópicas pero similares es el producto del patrón de la fuente individual y el patrón de un arreglo de fuentes puntuales isotrópicas, cada una ubicada en el centro de fase de la fuente individual y teniendo la misma amplitud y fase relativas, mientras que el patrón de fase total es la suma de los patrones de fase de la fuente individual y del arreglo de fuentes puntuales isotrópicas" (Kraus, 1950).

Para explicar este concepto se toma el caso de **dos fuentes no isotrópicas pero similares**, donde F es designada al arreglo de fuentes puntuales isotrópicas y f a una fuente individual. Esto viene del principio de superposición de campos, expresado en la ecuación 85:

$$E_{\text{total}} = E_1 + E_2 + \cdots + E_n \tag{85}$$

El campo de una fuente no isotrópica individual se expresa en la ecuación 86:

$$\vec{E}_{\text{individual}} = f(\theta,\phi)\exp\big[j f_p(\theta,\phi)\big] \tag{86}$$

El campo del arreglo de fuentes isotrópicas se expresa en la ecuación 87:

$$E_{\text{arreglo}} = \sum_{i=1}^{N} a_i\, e^{j\delta_i}\, e^{j\vec{k}\cdot\vec{r}_i} = F(\theta,\phi)\exp\big[j F_p(\theta,\phi)\big] \tag{87}$$

Al aplicar el principio de multiplicación a las ecuaciones 86 y 87, el campo total resulta en la ecuación 88:

$$E_{\text{total}} = f(\theta,\phi)\, F(\theta,\phi)\, \exp\big[f_p(\theta,\phi) + F_p(\theta,\phi)\big] \tag{88}$$

Gracias a la ecuación 88, se puede llegar sin problemas al patrón producido por un arreglo deseado. Si las fuentes son **disímiles**, entonces no se puede aplicar el principio de multiplicación de patrones y los campos deben sumarse en cada ángulo.

3.4.2. Arreglo lineal de n fuentes isotrópicas con misma amplitud y espacio.

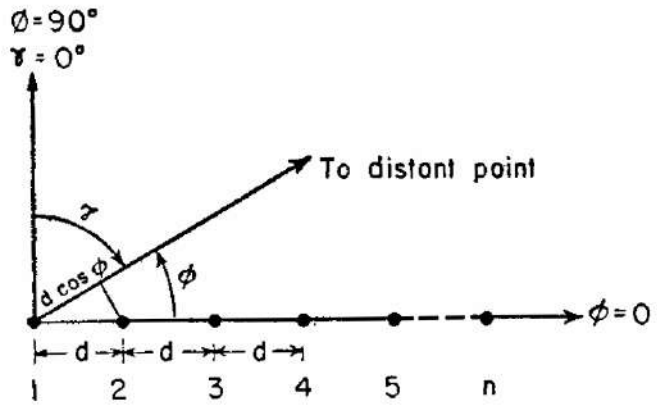


Figura 24: Arreglo lineal de n fuentes isotrópicas (Kraus, 1950).

Se llega a este tema porque es la mejor descripción que se puede tener de los componentes de las antenas helicoidales (Kraus, 1950). En la Figura 24 se puede observar en qué consiste el arreglo. El campo eléctrico a una distancia lejana y con dirección ϕ se toma como la serie geométrica de la ecuación 89:

$$E = 1 + e^{j\psi} + e^{2j\psi} + \cdots + e^{j(n-1)\psi} \tag{89}$$

Con una pequeña diferencia de fase dada por δ, la fase de los campos es $\psi = d_r\cos\phi + \delta$. Las amplitudes de los campos se toman como iguales. Si se multiplica la ecuación 89 por $e^{j\psi}$ se obtiene la ecuación 90:

$$Ee^{j\psi} = e^{j\psi} + e^{2j\psi} + \cdots + e^{jn\psi} \tag{90}$$

Al restar la ecuación 89 de la ecuación 90 y operar, se llega a la ecuación 91:

$$E = \frac{1 - e^{jn\psi}}{1 - e^{j\psi}} \tag{91}$$

Reescribiendo la ecuación 91 mediante la identidad $e^{j\theta} - e^{-j\theta} = 2j \sin\theta$, y centrando la fase en el arreglo, se llega a la ecuación clave (Kraus, 1950):

$$E = \frac{\sin(n\psi/2)}{\sin(\psi/2)} \tag{92}$$

3.4.3. ¿Cómo se llega a la antena helicoidal?

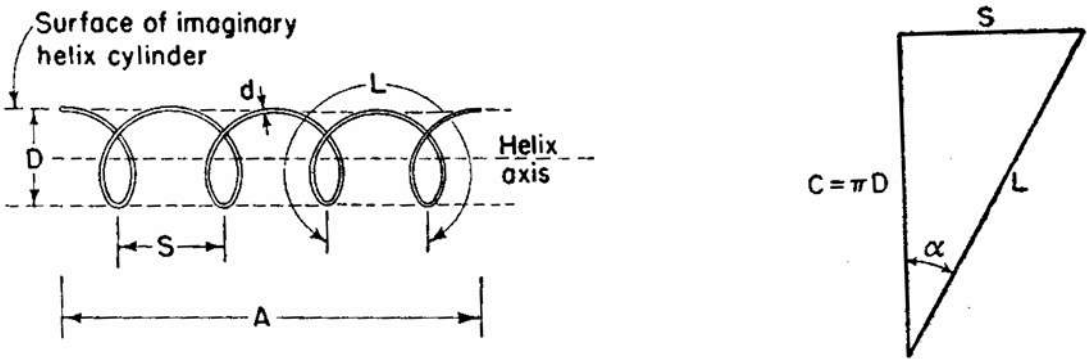


Figura 25: Descripción geométrica de la helicoide (Kraus, 1950).

La antena helicoidal es el conjunto de las antenas de tipo loop y lineal. Tiene dos tipos de radiación (Kraus, 1950):

a. Modo de radiación axial: el campo es máximo en la dirección del eje de la hélice con polarización circular. Es la representación más práctica del patrón de radiación.

b. Modo de radiación normal: el campo es máximo en una dirección normal al eje de la hélice; el patrón es aproximado al de un dipolo (Balanis, 2005).

Las dimensiones de la helicoide están dadas de la siguiente manera: D es el diámetro de la hélice; $C = \pi D$ es la circunferencia; S es el espaciado entre vueltas; α es el ángulo de paso ($S/\pi D$); L es la longitud de una vuelta; n es el número de vueltas; $A = nS$ es la longitud axial; y d es el diámetro del filamento. Cuando $\alpha = 0°$ la antena es de tipo loop; cuando $\alpha = 90°$ es de tipo lineal (Kraus, 1950).

Para continuar con el entendimiento de la antena helicoidal es necesario el concepto de arreglo *end-fire* (Balanis, 2005), que concentra la radiación a lo largo del eje del arreglo. Su factor de fase se expresa en la ecuación 93:

$$\psi = kd\cos\theta + \beta \tag{93}$$

donde $k = 2\pi/\lambda$, d es la distancia entre elementos, β es la diferencia de fase progresiva entre elementos y θ es el ángulo de observación. Si se desea que el máximo ocurra hacia adelante ($\theta = 0$), de la ecuación 93 se obtiene:

$$\beta = -kd \tag{94}$$

Otro concepto clave es la condición de Hansen-Woodyard, propuesta en 1938 e introducida por Balanis (2005), que se aplica cuando se requiere suprimir los lóbulos traseros y mejorar la directividad. La condición es la ecuación 95:

$$\beta = -\left(kd + \frac{\pi}{N}\right) \tag{95}$$

Esta condición asegura que el máximo desplazado cumpla la ecuación 96 y que el lóbulo trasero quede suprimido según la ecuación 97:

$$|\psi|_{\theta=0^\circ} = \frac{\pi}{N} \tag{96}$$

$$|\psi|_{\theta=180^\circ} = \pi \tag{97}$$

3.4.4. Caso 1 del patrón de la helicoide.

Recordando la ecuación 92, que es la forma no normalizada del factor de arreglo de n fuentes isotrópicas, y aplicando la condición de Hansen-Woodyard de la ecuación 95 para mayor directividad, la máxima magnitud ocurre en $\psi = -\pi/n$, lo que conduce al patrón normalizado de la ecuación 98:

$$E_{\text{nom}} = \sin\left(\frac{\pi}{2n}\right) \frac{\sin(n\psi/2)}{\sin(\psi/2)} \tag{98}$$

Usando el principio de multiplicación de la ecuación 88, se hace el producto de la ecuación 98 por el campo de la vuelta individual $E_{\text{ind}} = \cos\phi$. El patrón de la helicoide queda expresado en la ecuación 99 (Kraus, 1950):

$$E = \sin\left(\frac{\pi}{2n}\right) \frac{\sin(n\psi/2)}{\sin(\psi/2)} \cos\phi \tag{99}$$

Para conocer el factor de fase en la antena helicoidal, se adapta la ecuación 93 a la helicoide, donde $S_r = 2\pi S/\lambda$, obteniendo la ecuación 100:

$$\psi = 2\pi\left(S_\lambda \cos\phi - \frac{L_\lambda}{p}\right) \tag{100}$$

donde p es la velocidad relativa de fase (v/c), siendo v la velocidad a lo largo del conductor y c la velocidad de la luz. Usando la condición *end-fire* ($\psi = -2\pi m$) con máxima directividad ($\phi = 0$) en la ecuación 100, se llega a la ecuación 101:

$$m + S_\lambda = \frac{L_\lambda}{p} \tag{101}$$

En la ecuación 101, m representa los modos de radiación: $m = 0$ corresponde a la aproximación de un dipolo; $m = 1$ al modo de radiación con polarización circular ($\lambda \approx C$); $m = 2$ cuando $\lambda \approx 2C$; y así sucesivamente. La relación geométrica de la hélice se expresa en la ecuación 102:

$$L^2 = \pi^2 D^2 + S^2 \tag{102}$$

Al reducir la ecuación 101 a $m = 1$ y $p = 1$, se obtiene la relación para la radiación en modo axial, de donde se deducen las ecuaciones 103 y **??**:

$$D_\lambda = \frac{\sqrt{2S_\lambda + 1}}{\pi}, \qquad C_\lambda = \sqrt{2S_\lambda + 1} \tag{103}$$

3.4.5. ¿Por qué existen diferentes condiciones para deducir la velocidad de fase?

Para el modo T_1R_1, se tiene plantea la velocidad de fase a partir del principio *end-fire*, donde los campos están en fase, obteniendo la ecuación 104 (Kraus, 1950):

$$p = \frac{1}{\sin\alpha + (\cos\alpha / C_\lambda)} \tag{104}$$

Al aplicar el principio de Hansen y Woodyard de la ecuación 95 sobre la ecuación 104, se obtiene la ecuación 105:

$$p = \frac{1}{\sin\alpha + \dfrac{2n+1}{2n}\dfrac{\cos\alpha}{C_\lambda}} \tag{105}$$

Otro caso para la velocidad de fase se obtiene midiendo el ángulo ϕ_0 en el campo lejano con ψ_0. Partiendo de $\psi = -(2\pi m + \psi_0)$ y usando la ecuación 101 con $m = 1$, la velocidad de fase queda como la ecuación 106:

$$p = \frac{L_\lambda}{S_\lambda \cos\phi_0 + 1 + (\psi_0 / 2\pi)} \tag{106}$$

La ecuación 106 corresponde al primer caso nulo de la medición del patrón de radiación.

Existe una cuarta relación para la velocidad de fase, apropiada para el primer modo de transmisión, que se obtiene aplicando condiciones de frontera para un conductor helicoidal en coordenadas cilíndricas a través de dos puntos c y d, como se ve en la Figura 26. Bajo estas condiciones, la velocidad de fase se expresa en la ecuación 107:

$$p = \frac{C_\lambda}{m\cos\alpha + hR\sin\alpha} \tag{107}$$

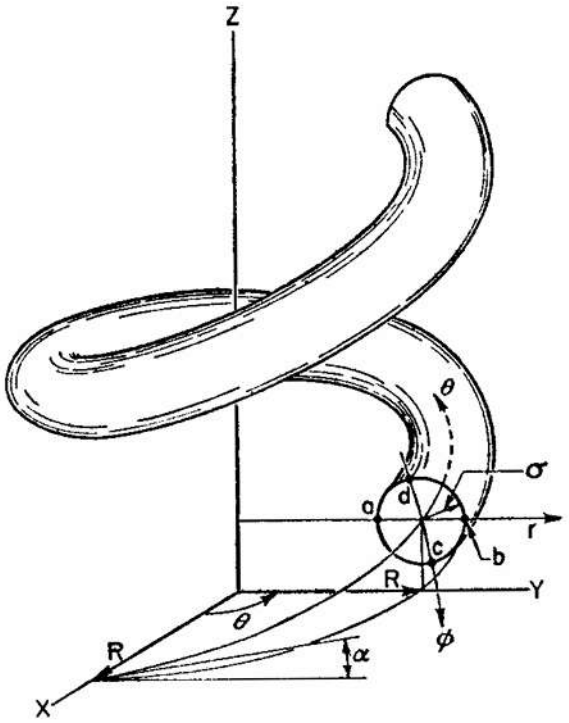


Figura 26: Helicoide con condiciones de frontera (Kraus, 1950).

con el parámetro hR definido en la ecuación 108:

$$hR = \tan\alpha \, \frac{m \, J_m^2(kR)}{J_{m-1}(kR) \, J_{m+1}(kR)} \tag{108}$$

donde m está dado por el orden de transmisión, R es el radio de la hélice y J son las funciones de Bessel de argumento kR. El argumento se define en la ecuación 109:

$$kR = \sqrt{C_\lambda^2 - (hR)^2} \tag{109}$$

3.4.6. Caso 2 del patrón de la helicoide.

Por otro lado, el tratamiento de las antenas para el modo T_1R_1 considera $m = 1$ en la ecuación 101 (Kraus, 1950):

$$p = \frac{L_\lambda}{S_\lambda + 1} \tag{110}$$

que se reduce, usando la ecuación 104, a:

$$p = \frac{1}{\sin\alpha + (\cos\alpha / C_\lambda)} \tag{111}$$

Esto indica que se radia en el modo axial ($\frac{3}{4} < C_\lambda < \frac{4}{3}$). Sin embargo, esto no es suficiente para que el resultado graficado concuerde con el calculado, por lo que se opta por más consideraciones. Al usar la condición de Hansen-Woodyard de la ecuación 95 (Kraus, 1950):

$$\psi = -\left(2\pi m + \frac{\pi}{n}\right) \tag{112}$$

Si $\phi = 0$, se obtiene la expresión intermedia de la ecuación 113:

$$p = \frac{L_\lambda}{S_\lambda + m + (1/2n)} \tag{113}$$

Poniendo $m = 1$ en la ecuación 113, se llega a la ecuación 114:

$$p = \frac{L_\lambda}{S_\lambda + (2n+1)/(2n)} \tag{114}$$

Al juntar y simplificar las ecuaciones 100 y 93, se tiene la ecuación 115 para el factor de fase del modo T_1R_1:

$$\psi = 360^\circ \left[S_\lambda(1 - \cos\phi) + \frac{1}{2n}\right] \tag{115}$$

Dependiendo de las condiciones deseadas (ya sea de manera generalizada o específica con T_1R_1), la ecuación 115 permite obtener el patrón de radiación para las condiciones $12^\circ < \alpha < 15^\circ$, $\frac{3}{4} < C_\lambda < \frac{4}{3}$ y $n > 3$. El patrón de radiación completo se expresa en la ecuación 116:

$$E = \sin\left(\frac{90^\circ}{n}\right) \frac{\sin(n\psi/2)}{\sin(\psi/2)} \cos\phi \tag{116}$$

Ya con esas consideraciones se tienen los dos casos de patrón de la helicoide, con las variaciones de las ecuaciones 115 y 100 aplicadas en la ecuación 116.

3.4.7. Variaciones en la impedancia.

Cuando $C_\lambda < 2/3$ la impedancia terminal resulta muy sensible a cambios en la frecuencia. Cuando se está en el orden 1 y $2/3 < C_\lambda < 4/3$, la impedancia terminal es constante en función de la frecuencia. En general, la impedancia en modo axial de la antena helicoidal tiene un valor de 100 a 200 ohmios y viene dada por la ecuación 117 (Kraus, 1950):

$$R = 140\, C_\lambda \quad [\text{ohmios}] \tag{117}$$

que es $\pm 20\,\%$ aproximadamente de las mediciones empíricas cuando se aplica a las condiciones $12^\circ < \alpha < 15^\circ$, $2/3 < C_\lambda < 4/3$ y $n > 3$.

3.4.8. Caso 3 del tratamiento de la helicoide.

Con lo anterior se puede hacer una tercera aproximación del patrón de radiación para los circuitos RLC. Si se considera que para el modo axial de transmisión 1 la resistencia va de 100 a 200 ohmios y que $R = 140\, C_\lambda$ de la ecuación 117, aproximando $C_\lambda \approx 1$, con las condiciones $12° < \alpha < 15°$, $2/3 < C_\lambda < 4/3$ y $n > 3$, se pueden generar variaciones de patrones y frecuencias debidas a la naturaleza del patrón de radiación con un circuito RLC, haciendo uso de las ecuaciones 115 y 116 como base de la simulación.

4.. Metodología

4.1. Enfoque general de la metodología

El presente trabajo se desarrolla como un enfoque mixto que consta del análisis teórico de la conducta electromagnética de las antenas helicoidales y, después, de la aplicación computacional de los hallazgos mediante simuladores interactivos. El propósito de esta investigación es adoptar los modelos matemáticos obtenidos en los capítulos anteriores y traducirlos en herramientas visuales que permitan estudiar el comportamiento del modelo en cuestión de manera intuitiva y reproducible. En síntesis, consta de dos hitos por hacer. La primera parte implica la revisión y sistematización del marco contextual: geometría helicoidal, consideraciones sobre el funcionamiento en modo axial, función de velocidad de fase y el modelo general del patrón de radiación propuesto por MSV; los cuales se pueden recuperar, de los avatares previos, en los capítulos anteriores. En la segunda, se naturaliza todo el marco teórico al generar módulos interactivos de software donde sea posible manipular los parámetros físicos y observar el comportamiento en tiempo real. Se idearon cuatro módulos para ello, cada uno enfocado en un caso específico de funcionamiento.

Para todos los propósitos, el módulo uno se encarga de la simulación general del patrón de radiación para cualquier combinación posible de parámetros. El segundo restringe el análisis al primer modo de transmisión axial. El tercero incorpora el diseño desde una perspectiva circuital mediante un modelo RLC equivalente. El cuarto integra el análisis circuital con la simulación del patrón del primer modo de transmisión.

Esto representa el módulo más completo. Este diseño modular permite la comparación de resultados entre casos, la extrapolación del efecto de variables aisladas y la validación cualitativa de los resultados con respecto a las expresiones teóricas. La progresión de la complejidad desde los módulos más simples ayuda en la comprensión gradual del sistema a los futuros usuarios o investigadores.

4.2. Herramientas computacionales empleadas

Para la implementación de los simuladores se empleó el lenguaje de programación Python, elegido por su amplio ecosistema de librerías científicas y su capacidad para integrar cálculo numérico, visualización gráfica y desarrollo de interfaces en un mismo entorno.

Las librerías principales utilizadas se describen a continuación:

- `NumPy` proporciona las estructuras de arreglos y las rutinas de cálculo numérico que sostienen toda la evaluación matemática de los modelos, incluyendo funciones trigonométricas vectorizadas y operaciones de álgebra lineal.
- `SciPy`, específicamente su submódulo de funciones especiales, se empleó para evaluar las funciones de Bessel de primera especie que intervienen en el cálculo de la velocidad de fase.

- `Matplotlib` fue la librería central para la generación de gráficos, tanto en representaciones polares del patrón de radiación como en proyecciones tridimensionales de la helicoide. A través de su módulo de animación se generaron las secuencias de cuadros que conforman los archivos GIF de visualización dinámica.

- `Tkinter` se utilizó para la construcción de las interfaces gráficas de usuario, permitiendo al operador ingresar parámetros mediante campos de texto y controles deslizantes, y visualizar los resultados sin necesidad de modificar el código fuente.

- `Matplotlib.widgets` complementó la interacción dentro de las propias figuras, habilitando controles como potenciómetros deslizantes integrados en la ventana gráfica.

La combinación de estas herramientas permite la generación de gráficos tridimensionales, animaciones y la manipulación interactiva de parámetros, cubriendo de manera integral los requerimientos de visualización planteados en los objetivos del trabajo.

4.3. Arquitectura general del simulador

En la Figura 27 y el Pseudocódigo 1, se describe en modo de diagrama de flujo cómo es que trabaja la ventana priincipal del software . En ella se ha de notar la elección por parte del usuario para la resolución del problema de la helicoide de 4 maneras distintas, donde cada una de las opciones contiene una variedad de lo que se pretende hacer, entre ellas la ejecución de la simulación, la generación del gif, el muestreo de los cálculos obtenidos, la limpieza del gráfico en caso de que el usuario desee hacer una nueva visualización con la introducción de otros parámetros, calcular los parámetros y dibujar la helicoide.

El desarrollo del software se estructuró en módulos independientes, cada uno encargado de un tipo específico de simulación. Esta decisión de diseño obedece a criterios de claridad, mantenibilidad y extensibilidad: al mantener cada caso de uso en un módulo propio se facilita la identificación y corrección de errores, la reutilización de funciones auxiliares comunes y la incorporación de nuevas funcionalidades en el futuro.

De manera general, todos los simuladores siguen la misma lógica de funcionamiento, articulada en las etapas que se describen a continuación.

a. Definición de parámetros de entrada: El usuario proporciona, a través de la interfaz gráfica, los valores de los parámetros físicos que caracterizan la antena helicoidal: ángulo de paso α, circunferencia C, número de vueltas, modo de operación y, según el módulo, parámetros eléctricos adicionales como voltaje, corriente y carga del capacitor. Cada campo incluye restricciones de rango que impiden la introducción de valores físicamente inválidos.

b. Cálculo de parámetros geométricos y electromagnéticos: A partir de las entradas, se evalúan las relaciones derivadas que caracterizan completamente la geometría

helicoidal: radio R, diámetro D, espaciado entre vueltas S, longitud de arco L, altura máxima h y los parámetros normalizados C_λ, S_λ y L_λ, conforme a las ecuaciones 118 a 126.

c. Evaluación del modelo matemático del patrón de radiación: Con los parámetros calculados se evalúa la función de velocidad de fase y, a partir de ella, la expresión del patrón de radiación correspondiente al módulo en cuestión. En esta etapa se aplican las restricciones físicas necesarias, como la limitación de la velocidad de fase a valores no superiores a la velocidad de la luz.

d. Generación de la geometría de la helicoide: Se calculan las coordenadas tridimensionales de la curva helicoidal parametrizada, que representa la estructura física de la antena. Esta geometría sirve tanto como referencia visual como punto de partida para la animación del patrón de radiación a lo largo del eje de propagación.

e. Visualización y animación de resultados: Los datos calculados se presentan mediante gráficas polares del patrón, representaciones tridimensionales de la helicoide y animaciones en formato GIF que ilustran la evolución del patrón conforme la señal recorre la antena. Los valores numéricos relevantes se despliegan en la interfaz como texto formateado.

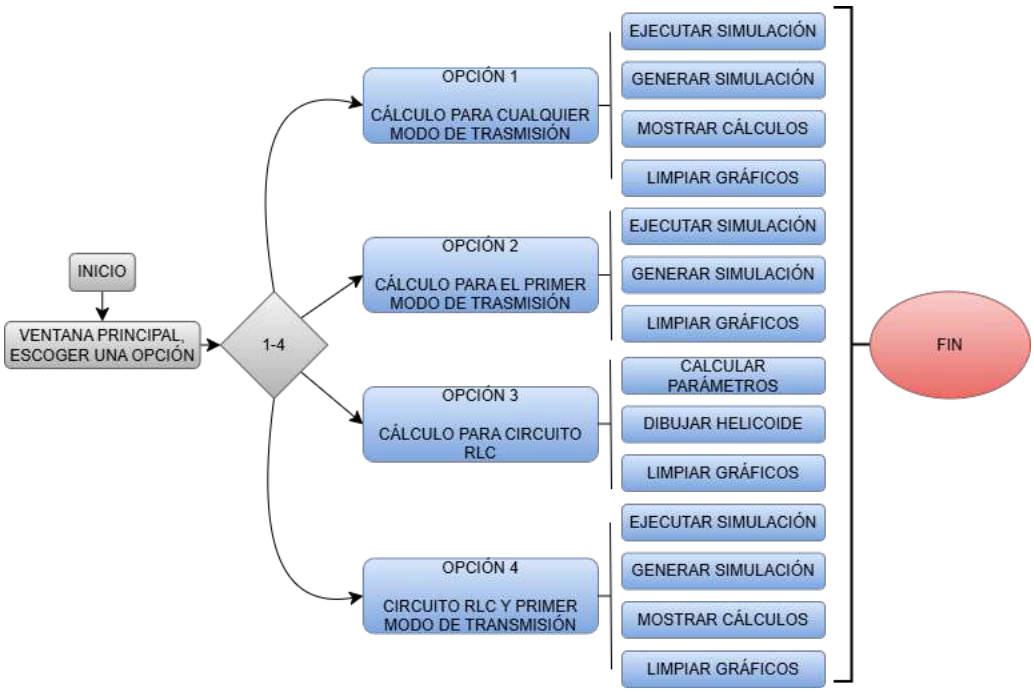


Figura 27: Diagrama de flujo acerca del software (autoría propia).

```
INICIO

```

```
Mostrar ventana principal

Mostrar menu con 4 opciones:
    1. Simulacion completa
    2. Primer modo de transmision
    3. Circuito RLC
    4. Simulacion acoplada RLC

Leer seleccion del usuario

SI opci0n = 1:
    Ejecutar modulo de simulacion general

SI opcion = 2:
    Ejecutar modulo del primer modo de transmision

SI opcion = 3:
    Ejecutar modulo de circuito RLC

SI opcion = 4:
    Ejecutar modulo acoplado RLC
FINALIZAR
```

Código 1: Pseudocódigo del software (autoría propia).

Esta estructura común permite reutilizar funciones de cálculo y visualización entre módulos, y facilita la extensión del software hacia nuevos casos de operación.

4.4. Simulación del patrón de manera general

En la Figura 28 y el Pseudocódigo 2, se observa el diagrama de flujo de la primera opción que describe la simulación del patrón de radiación de manera general, donde se introduce los parámetros $0° < \alpha < 90°$, C, factor, $n > 0$ y m. Se valida la introducción, se hace el cálculo geométrico, se calcula la velocidad de fase a partir de un bessel de primera especie donde se pone la condición de que $p < 1$ y el modo de transmisión m que se introduce en la ecuación ψ. A partir de esto ya se puede hacer la ejecución de la simulación, la generación del gif, el muestreo de los cálculos obtenidos, la limpieza del gráfico.

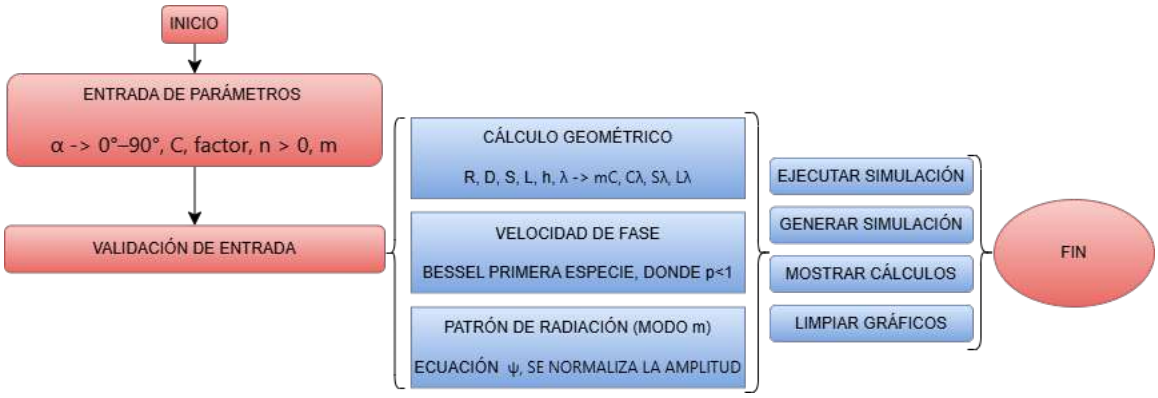


Figura 28: Diagrama de flujo acerca de la opción 1 (autoría propia).

```
INICIO

Leer parametros de entrada:
    alpha, C, factor, n, m

Validar rangos fisicos:
    0 < alpha < 90
    C > 0
    n > 0
    m > 0

SI los parametros no son validos:
    Mostrar mensaje de error
    FINALIZAR

Calcular parametros geometricos:
    R, D, S, L, h

Calcular longitud de onda:
    lambda approx mC

Calcular parametros normalizados:
    Clambda, Slambda, Llambda

Evaluar velocidad de fase:
    Usar funciones de Bessel

SI p > 1:
    p = 1

Evaluar patron de radiacion:
    Usar ecuacin psi

Normalizar amplitud

Mostrar grafica tridimensional
```

```
37
38 SI usuario selecciona Ejecutar Simulacion:
39     Mostrar patron de radiacion
40
41 SI usuario selecciona Generar Simulacion:
42     Crear GIF
43
44 SI usuario selecciona Mostrar Calculos:
45     Mostrar resultados numricos
46
47 SI usuario selecciona Limpiar Graficos:
48     Limpiar interfaz
49
50 FIN
```

Código 2: Pseudocódigo acerca de la opción 1 (autoría propia).

El primer módulo tiene como propósito ofrecer una visión amplia del comportamiento del patrón de radiación de la antena helicoidal sin restricciones sobre el modo de operación, permitiendo explorar el espacio de parámetros de forma libre.

Los parámetros de entrada requeridos son el ángulo de paso α (entre $0°$ y $90°$), la circunferencia C en metros (mayor a cero), un factor de escala para ajustar la visibilidad del patrón en pantalla, el número de vueltas (mayor a cero) y el modo de operación m, que es un entero positivo. El papel de m es central en este módulo: a mayor valor, el patrón incorpora más lóbulos interiores y su tamaño escala proporcionalmente, lo que corresponde físicamente a modos de orden superior de la antena helicoidal.

Con estos datos se calculan los parámetros geométricos derivados conforme a las siguientes relaciones. El radio de la hélice se obtiene como:

$$R = \frac{C}{2\pi} \tag{118}$$

el diámetro como:

$$D = 2R \tag{119}$$

el espaciado entre vueltas como:

$$S = C \tan \alpha \tag{120}$$

la longitud de arco por vuelta como:

$$L = \sqrt{C^2 + S^2} \tag{121}$$

la altura total de la antena como:

$$h = \text{vueltas} \cdot S \tag{122}$$

Para la normalización se adopta la aproximación:

$$\lambda \approx m \cdot C \tag{123}$$

de donde se obtienen directamente los parámetros normalizados:

$$C_\lambda = \frac{C}{\lambda} \tag{124}$$

$$S_\lambda = \frac{S}{\lambda} \tag{125}$$

$$L_\lambda = \frac{L}{\lambda} \tag{126}$$

A continuación se evalúa la función de velocidad de fase, cuya formulación general se apoya en las funciones de Bessel de primera especie, tal como se expresa en las ecuaciones 107, 108 y 109. Se aplica una restricción explícita: si el valor calculado de la fase supera la unidad, se fija en uno, pues un valor superior implicaría propagación superlumínica, lo cual carece de sustento físico. Adicionalmente se implementa una guarda contra divisiones entre cero para garantizar la estabilidad numérica del cálculo en condiciones extremas de los parámetros.

El patrón de radiación se evalúa con la forma generalizada de la ecuación 100. La amplitud se normaliza para que la representación gráfica sea consistente independientemente de la magnitud absoluta de los valores. Se introduce un factor de escala progresiva que amplía el patrón conforme la posición recorre la longitud de la hélice, facilitando la lectura visual de la animación. El parámetro de factor de entrada permite al usuario agrandar o reducir adicionalmente el patrón si las dimensiones en pantalla no resultan adecuadas.

La geometría tridimensional de la helicoide se genera mediante una parametrización en coordenadas cartesianas y se dibuja en un sistema de ejes tridimensionales. Sobre esta geometría se superpone la animación del patrón de radiación, mostrando cómo éste evoluciona a medida que la señal recorre cada vuelta de la antena. Finalmente, la interfaz presenta los valores numéricos d todos los parámetros calculados, incluyendo radio, diámetro, espaciado, longitud de arco, altura total y parámetros normalizados.

4.5. Simulación del patrón en modo de Transmisión 1

En la Figura 29 y el Pseudocódigo 3, se observa el diagrama de flujo de la segunda opción que describe la simulación del patrón de radiación abordado con resoluciones solamente con el primer modo de transmisión, donde se introduce los parámetros $12° < \alpha < 15°$, C, factor, $n > 3$ y $.75° < C\lambda < 1.25$. Se valida la introducción, se hace el cálculo geométrico, se calcula la velocidad de fase a partir de un bessel de primera especie donde se pone la condición de que $p < 1$ y el primer modo de transmisión en la ecuación ψ. A partir de esto ya se puede hacer la ejecución de la simulación, la generación del gif, el muestreo de los cálculos obtenidos, la limpieza del gráfico.

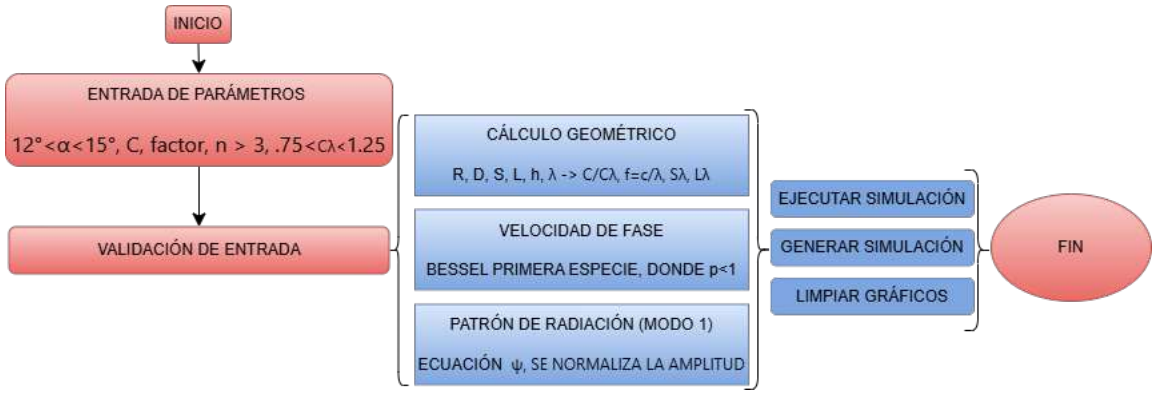


Figura 29: Diagrama de flujo acerca de la opción 2 (autoría propia).

```
INICIO

Leer parametros:
    alpha, C, factor, n, Clambda

Validar condiciones:
    12 < alpha < 15
    n >= 3
    0.75 < Clambda < 1.25

SI los parametros no son validos:
    Mostrar mensaje de error
    FINALIZAR

Calcular geometria:
    R, D, S, L, h

Calcular longitud de onda:
    lambda = C / Clambda
Calcular frecuencia:
    f = c / lambda

Calcular parametros normalizados:
    Slambda, Llambda

Evaluar velocidad de fase:
    Funciones de Bessel

SI p > 1:
    p = 1

Evaluar patron de radiacion:
    Modo de transmisin 1

Normalizar amplitud
```

```
36
37 Mostrar simulacion tridimensional
38
39 SI usuario selecciona Ejecutar Simulacion:
40     Mostrar patron de radiacion
41
42 SI usuario selecciona Generar Simulacion:
43     Crear GIF
44
45 SI usuario selecciona Limpiar Graficos:
46     Limpiar interfaz
47
48 FIN
```

Código 3: Pseudocódigo acerca de la opción 2 (autoría propia).

El segundo módulo restringe el análisis al primer modo de transmisión axial de la antena helicoidal, que es el modo de operación de mayor interés práctico, ya que produce un patrón con máximo de radiación en la dirección del eje de la hélice.

Para garantizar la operación en este modo, los parámetros de entrada se limitan a rangos específicos. El ángulo de paso α se acota entre 12° y 15°, que corresponde al intervalo donde se verifica empíricamente y teóricamente la operación axial. El número de vueltas debe ser mayor a tres, condición mínima para establecer un patrón axial estable. El parámetro C_λ se admite en el rango de 0.75 a 1.25, que define la banda de frecuencias en torno a la condición de resonancia circunferencial. El valor de la circunferencia C y el factor de escala se solicitan con las mismas restricciones que en el módulo anterior.

A diferencia del primer módulo, en este caso el patrón de radiación se evalúa mediante la ecuación 115, que incorpora la fase específica del primer modo de transmisión y garantiza la obtención del lóbulo axial característico. La normalización y la representación gráfica siguen el mismo procedimiento descrito anteriormente.

Los parámetros derivados se calculan con las mismas ecuaciones geométricas. La longitud de onda se obtiene mediante:

$$\lambda \approx \frac{C}{C_\lambda} \tag{127}$$

y la frecuencia de operación se calcula como:

$$f = \frac{c}{\lambda} \tag{128}$$

donde c es la velocidad de la luz en el vacío. Estos valores, junto con los parámetros geométricos y normalizados, se presentan en la interfaz al término de la simulación.

Se destaca que este módulo restringe los parámetros de entrada para garantizar la operación en modo axial, lo que permite validar los resultados obtenidos con la teoría presentada en capítulos anteriores y apreciar directamente el efecto de pequeñas variaciones en α y C_λ sobre la forma del patrón.

4.6. Modelo basado en circuito RLC

En la Figura 30 y el Pseudocódigo 4, se observa una parte del software que se encarga de dar la geometría necesaria en la antena para poder emitir diferentes patrones de radiación a partir de la introducción de los parámetros. Se introduce los parámetros $12° < \alpha < 15°$, V, I, carga, escala, flujo, n. Se valida la introducción, se hace el cálculo del circuito, se obtiene el rango de las impedancias para operar en el primer modo de transmisión y se da los resultados de la geometría de la antena. A partir de esto se dan resultados, se dibuja la helicoide y se limpian los gráficos.

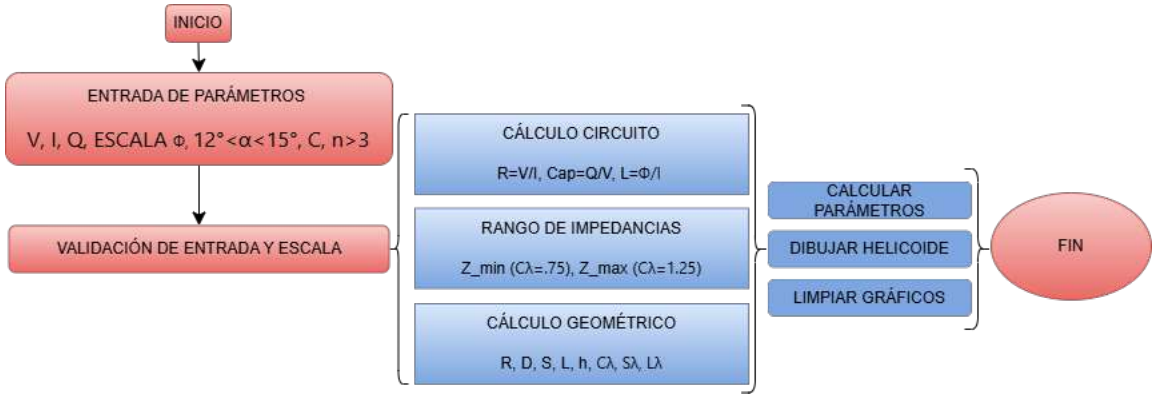


Figura 30: Diagrama de flujo acerca de la opción 3 (autoría propia).

```
INICIO

Leer parametros:
    V, I, Q, phi, alpha, C, n

Validar parametros:
    V > 0
    I > 0
    C > 0
    n > 0
    12 < alpha < 15

SI los parametros no son validos:
    Mostrar mensaje de error
    FINALIZAR

Calcular resistencia:
    R = V / I

Calcular capacitancia:
    Cap = Q / V

Calcular inductancia:
    L = phi / I
Calcular rango de impedancias:
```

```
    Zmin
    Zmax

Calcular parametros geometricos:
    R, D, S, L, h

Calcular parametros normalizados:
    Clambda, Slambda, Llambda

SI usuario selecciona Calcular Parametros:
    Mostrar resultados electricos y geometricos

SI usuario selecciona Dibujar Helicoide:
    Mostrar helicoide 3D

SI usuario selecciona Limpiar Graficos:
    Limpiar interfaz

FIN
```

Código 4: Pseudocódigo acerca de la opción 3 (autoría propia).

El tercer módulo aborda el diseño de la antena helicoidal desde una perspectiva circuital, estableciendo una analogía entre el comportamiento electromagnético de la antena y un circuito resonante de tipo RLC. Esta analogía facilita la interpretación del sistema desde un enfoque eléctrico clásico y permite estimar los valores de los componentes de un circuito equivalente que opera a la frecuencia de resonancia de la antena.

El punto de partida conceptual es que una antena tiene su máxima utilidad cuando la reactancia total del circuito equivalente es nula, condición que corresponde a la resonancia. Bajo esta premisa, los parámetros de entrada del módulo son: voltaje V, corriente I, carga del capacitor Q, una opción numérica entre uno y cuatro para seleccionar la escala del flujo magnético Φ en el inductor, el ángulo de paso α (entre $0°$ y $90°$) y el número de vueltas (mayor a cero).

La resistencia equivalente se determina mediante la ley de Ohm:

$$V = I \cdot R \tag{129}$$

La capacitancia se calcula como:

$$C_{\text{cap}} = \frac{Q}{V} \tag{130}$$

y la inductancia como:

$$L_{\text{ind}} = \frac{\Phi}{I} \tag{131}$$

La condición de reactancia nula, combinada con la aproximación:

$$C \approx \lambda \tag{132}$$

permite derivar los valores extremos de impedancia que definen el rango de operación del potenciómetro simulado:

$$Impedancia_{\text{mín}} = Q\,\Phi \left(\frac{2\pi c}{C/0.75} \right)^2 \tag{133}$$

$$Impedancia_{\text{máx}} = Q\,\Phi \left(\frac{2\pi c}{C/1.25} \right)^2 \tag{134}$$

Estos límites corresponden respectivamente a los valores de C_λ de 0.75 y 1.25, es decir, a los extremos de la banda de operación del modo axial.

El módulo implementa funciones auxiliares específicas para este cálculo. Una función selecciona la escala del flujo magnético según la opción indicada por el usuario, cubriendo rangos desde valores en el orden de los microweber hasta los miliweber. Otra función determina el orden en circunferencia m más adecuado a los parámetros dados. Dos funciones adicionales se encargan de expresar la inductancia y la capacitancia calculadas con su notación científica correcta y sus unidades correspondientes, facilitando la lectura de los resultados.

El módulo genera el gráfico tridimensional de la helicoide y presenta todos los parámetros calculados: dimensiones geométricas de la antena, valores de los componentes RLC, rango de impedancias y parámetros normalizados. Como apoyo visual en la interfaz se incorpora el diagrama esquemático del circuito RLC implementado, mostrado en la Figura 31, elaborado con la herramienta Circuit Diagram (Diagram, 2026).

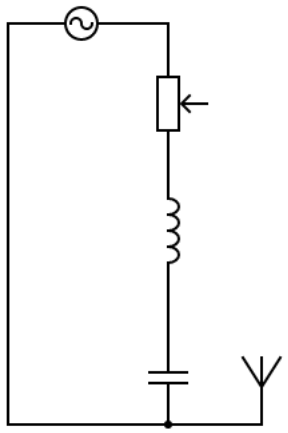


Figura 31: Circuito RLC equivalente utilizado en el módulo de diseño (Diagram, 2026).

Este modelo permite establecer una analogía entre el comportamiento electromagnético de la antena y sistemas eléctricos clásicos, facilitando su interpretación desde un enfoque circuital y abriendo la posibilidad de dimensionar físicamente los componentes de un circuito de alimentación.

4.7. Simulación acoplada: patrón de radiación y circuito RLC

En la Figura 32 y el Pseudocódigo 5, se observa el diagrama de flujo de la cuarta opción que describe una respuesta acoplada entre los resultados del circuito RLC y la simulación del patrón de radiación con el primer modo de transmisión, donde se introduce los parámetros $12° < \alpha < 15°$,V, I, factor, escala, valor de flujo, C, $n > 3$, rango del potenciómetro. Se valida la introducción, se hace el cálculo geométrico, el cálculo del circuito, el rango del potenciómetro, se calcula la velocidad de fase a partir de un bessel de primera especie donde se pone la condición de que $p < 1$ y el primer modo de transmisión m que se introduce en la ecuación ψ. A partir de esto ya se puede hacer la ejecución de la simulación, la generación del gif, el muestreo de los cálculos obtenidos, la limpieza del gráfico.

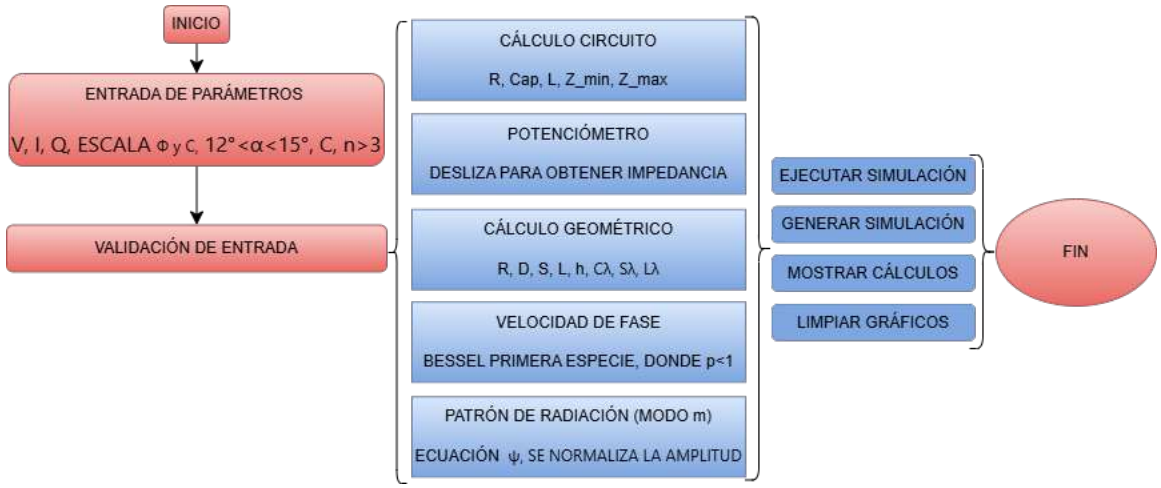


Figura 32: Diagrama de flujo acerca de la opción 4 (autoría propia).

```
INICIO

Leer parametros:
    V, I, Q, phi, alpha, C, n

Validar parametros:
    12 < alpha < 15
    n => 3

SI los parametros no son validos:
    Mostrar mensaje de error
    FINALIZAR
```

```
13
14 Calcular circuito RLC:
15     R, Cap, L
16
17 Calcular rango de impedancias:
18     Zmin, Zmax
19
20 Inicializar potencioetro
21
22 Mientras el potenciometro cambie:
23     Recalcular impedancia
24     Recalcular corriente
25     Recalcular frecuencia
26     Recalcular lambda
27
28 Calcular geometria:
29     R, D, S, L, h
30
31 Calcular velocidad de fase:
32     Funciones de Bessel
33
34 Evaluar patrn de radiacion:
35     Primer modo de transmision
36
37 Normalizar amplitud
38
39 SI usuario selecciona Ejecutar Simulacion:
40     Mostrar patron
41
42 SI usuario selecciona Generar Simulacion:
43     Crear GIF
44
45 SI usuario selecciona Mostrar Calculos:
46     Mostrar resultados electricos y geometricos
47
48 SI usuario selecciona Limpiar Graficos:
49     Limpiar interfaz
50
51 FIN
```

Código 5: Pseudocódigo acerca de la opción 4 (autoría propia).

El cuarto módulo representa la integración completa de los enfoques anteriores, combinando el modelo circuital RLC con la simulación del patrón de radiación del primer modo de transmisión. Es el módulo de mayor complejidad y el que ofrece una visión más completa de la relación entre los parámetros eléctricos y el comportamiento radiante de la antena.

Los parámetros de entrada son los mismos que en el módulo RLC, con la adición de los que garantizan la operación en el primer modo axial: el ángulo de paso α se restringe al rango entre 12° y 15°, el número de vueltas debe ser mayor a tres, y se

solicitan el valor de la circunferencia C en metros y un parámetro de orden c para seleccionar la escala de magnitud de dicha circunferencia. Esta última incorporación permite al usuario trabajar con antenas de distintas dimensiones físicas sin modificar los valores adimensionales del problema.

El elemento central de este módulo es la función de potenciómetro, que simula el ajuste continuo de la resistencia dentro del rango definido por las ecuaciones 133 y 134. A medida que el usuario desplaza el control deslizante en la interfaz, el módulo recalcula en tiempo real los valores de corriente, inductancia, frecuencia de operación, longitud de onda y lo parámetros normalizados C_λ, S_λ y L_λ. Este mecanismo ilustra cómo el ajuste de la impedancia del circuito de alimentación modifica las condiciones de operación electromagnética de la antena.

El patrón de radiación se evalúa y grafica empleando la ecuación 115, consistente con el primer modo de transmisión. La animación se genera con el mismo mecanismo de inicialización y actualización de cuadros descrito en los módulos anteriores. De esta forma, el usuario puede observar simultáneamente la geometría de la antena, el patrón de radiación animado y los valores numéricos que cambian en función de la posición del potenciómetro.

Este módulo representa la integración completa del modelo, permitiendo observa la influencia de parámetros eléctricos en el patrón de radiación y constituyendo la herramienta de análisis más rica de las cuatro desarrolladas.

4.8. Validación y alcance del modelo

Los resultados obtenidos mediante los simuladores se comparan cualitativamente con las expresiones teóricas desarrolladas en los capítulos anteriores, verificando su coherencia dentro de los rangos de operación establecidos. Esta comparación se realiza en dos niveles.

a. En este nivel se verifica que las tendencias observadas en el patrón de radiación al variar cada parámetro son consistentes con lo predicho por la teoría. Por ejemplo, al incrementar el número de vueltas manteniendo fijos los demás parámetros, el lóbulo principal del patrón en modo axial se estrecha, comportamiento que es coherente con la expresión teórica del patrón y con lo resultados reportados en la literatura de referencia. De manera análoga, las variaciones en α fuera del rango óptimo producen la degradación del lóbulo axial esperada por la teoría.

b. En el segundo nivel se comprueba que los valores numéricos calculados por el software, como la frecuencia de operación, la longitud de onda y los parámetros normalizados, corresponden correctamente a los que se obtendrían evaluando manualmente las expresiones teóricas con los mismos parámetros de entrada. Esta verificación se realizó para un conjunto representativo de configuraciones durante el proceso de desarrollo.

Es importante señalar que el modelo implementado corresponde a una aproximación teórica basada en las expresiones de MSV para la velocidad de fase y el patrón de radiación. Su validez depende de las condiciones asumidas en el análisis: hélice conductora ideal, espacio libre homogéneo, ausencia de efectos de borde y de la estructura de soporte físico de la antena. Los resultados obtenidos son por tanto una representación del comportamiento ideal del sistema y deben interpretarse como una herramienta de análisis y diseño preliminar, no como una sustitución de la simulación electromagnética de onda completa o de la medición experimental.

Dentro de estos límites, el conjunto de simuladores desarrollados cumple el objetivo de proporcionar una herramienta intuitiva y computacionalmente accesible para el estudio y la enseñanza del comportamiento radiante de las antenas helicoidales.

4.9. Consideraciones finales de implementación

La finalidad de esta sección ha sido proporcionar una descripción general y razonada del funcionamiento de los simuladores desarrollados, explicando la lógica de cada etapa y las decisiones de diseño que la sustentan. No se ha pretendido detallar exhaustivamente cada rutina de cálculo, sino ofrecer una guía clara del proceso seguido que permita a un lector familiarizado con programación científica reproducir el trabajo o extenderlo.

Algunas consideraciones prácticas merecen mención adicional. En todos los módulos se implementaron validaciones de los parámetros de entrada que informan al usuario cuando un valor introducido cae fuera del rango permitido, evitando la generación de resultados sin sentido físico. Las funciones de dibujo y animación se parametrizaron de modo que el número de cuadros y la velocidad de la animació pueden ajustarse fácilmente sin modificar la lógica del cálculo. La separación entre las funciones de cálculo, las de generación gráfica y las de construcción de la interfaz facilita el mantenimiento y la reutilización del código.

El código fuente completo de los cuatro módulos se presenta en el Anexo 1, con el propósito de permitir su reproducción, modificación y extensión en trabajos futuros. Se incluyen comentarios en línea que explican las secciones menos evidentes, con especial atención a la implementación de las funciones de Bessel, la lógica de la animación y el manejo de la interfaz gráfica.

5.. Resultados y discusión

5.1. ¿Por qué es relevante una simulación?

Existen múltiples razones que justifican la relevancia de una simulación. De los principales que se consideran es que se tiene la necesidad de poder construir un comportamiento adecuado sin el uso de un software que requiera de un gran precio.

Con esto se puede identificar la direccionalidad, la ganancia, las zonas de radiación y sus lóbulos.

5.1.1. Precios de un software en el mercado.

Se dedica una sección especialmente a los distintos precios del mercado, donde una de las estrategias más utilizadas es ofrecer mensualidades o anualidades para la obtención de licencias de los softwares más empleados en el campo de la ingeniería (Polimetro, 2025), como Ansys, SolidWorks, AutoCAD o Mathematica. Junto a estos, también se desarrollan otros programas orientados al uso, la investigación y el desarrollo en electromagnetismo.

Se consultó acerca de algunos precios en el mercado. Algunos de ellos fueron:

- AN-SOF. Es un software que se encarga del análisis ... Sus precios oscilan entre 799-1199 USD por el uso de su licencia de software, con mayor precio, mayores son los beneficios. Se puede hacer la gráfica de diagramas de Smith para las impedancias, graficar los lóbulos en coordenadas esféricas y observar el patrón de radiación en 3D de antenas con diferentes configuraciones. Esta información se obtuvo de su página web (Engineering, 2025).

- Por otro lado tenemos el Software ofrecido por REMCOM con su programa XFdtd o XGtd que es un software que utiliza el método de diferencias finitas con dominio de tiempo. Se ven análisis de dipolos, parches, celulares con sus antenas, manejo de 5G, solucionador de circuitos, editor de esquemas, predicción de ruptura dieléctrica, optimizador de elementos, simulación electromagnética, etc. Esta información fue verificada gracias a (Remcom, 2024).

- Podemos tener la opción de uso, si se conoce cómo programar con MATLAB, hacer uso de Antena Toolbox que diseña, analiza, visualiza elementos de Antenas, solucionador electromagnético, estudio de ubicaciones de Antenas, cálculos de sección transversal de radar, flujos de trabajo con IA, reconstrucción de patrones de radiación con Deep Learning, optimización de diseños de Antenas, exportación de archivos CAD, integrar arrays de Antenas, simulación de algoritmos Beamforming, etc. Verificando el precio de la licencia, se puede otorgar por anualidades o de por vida y el precio depende del tipo de plan que se desee contratar (MathWorks, 2019):

    - Standard anual: 2440 USD.

- Standar perpetuo: 6100 USD.
- Startups anual: 4090 USD.
- Académico anual: 275 USD.
- Académico mensual: 550 USD.
- Estudiantil perpetuo: 99 USD.
- Para casa perpetuo: 149 USD.

- Otra manera de hacer análisis de antenas es con el uso de ANSYS y su extensión llamada HFSS. Donde una licencia anual individual comienza por 40000 USD o por un plan empresarial alcanzar hasta 500000 USD (Lavi, 2025).

5.1.2. Los softwares de libre uso.

El propósito de esta sección es mencionar que, en el mundo tecnológico, sí existen softwares en el mercado que son de libre uso. Uno de ellos, y de los más completos, es VEMSA3D. Entonces, ¿con qué fin se realiza algo que ya existe? Este tipo de cuestionamientos resultan reduccionistas en su perspectiva, ya que el software desarrollado en este trabajo contempla cuatro casos distintos enfocados específicamente en la antena helicoidal: un análisis sin restricciones de modo de operación, la explicación del primer modo de transmisión, un modelo basado en circuitos RLC y su simulación acoplada. Por lo tanto, es claro que no se trata del mismo tipo de trabajo, ni del mismo problema, ni del mismo enfoque que los ya existentes.

A continuación, se presentan algunos softwares de libre uso disponibles en el mercado:

- 4nec2: es una herramienta utilizada para la visualización y optimización de antenas en formato 2D y 3D, que sí permite simular patrones de radiación (Voors, 2021). Proporciona el cálculo de diversos parámetros previamente mencionados, como SWR, ganancia, impedancia, entre otros.

- Antennavis: puede ejecutarse a través de OnWorks mediante el uso de comandos. Permite visualizar antenas y sus patrones desde distintos ángulos, trabajando con NEC para la parte electromagnética (OnWorks, 2026).

- openEMS: es un simulador de nivel investigativo que se utiliza mediante Python y MATLAB. Es una herramienta de código abierto que no es amigable para usuarios sin experiencia, ya que requiere conocimientos en programación (similares a los empleados en este trabajo). Es ampliamente utilizado en el área de antenas (Fedeli et al., 2019).

- VEMSA3D (Visual EM Simulator for 3D Antennas): es un software de código abierto enfocado en la visualización y enseñanza, con diseños avanzados que permiten la modificación del código. Ofrece opciones como el cambio de colores en

los patrones de radiación, así como la obtención de resultados y representaciones gráficas (Kostis et al., 2010).

Los dos primeros softwares mencionados son accesibles para cualquier usuario. Sin embargo, funcionan como una "caja negra", ya que no explican los procesos internos, aunque permiten visualizar correctamente los patrones de radiación.

Por otro lado, los dos últimos softwares requieren un mayor nivel de conocimientos en programación, uso de computadoras y fundamentos de física. De estos, VEMSA3D destaca como el más completo, como ya se ha mencionado anteriormente.

Al realizar una comparación, se observa que estos softwares están diseñados de manera general para el análisis y diseño de antenas, aprovechando su naturaleza de código abierto. Esto evidencia que existe una amplia variedad de herramientas disponibles en el mercado. No obstante, la diferencia del software desarrollado en este trabajo radica en que está orientado a un problema específico, abordado desde cuatro enfoques distintos, además de ser más interactivo y didáctico. Adicionalmente, incorpora la generación de animaciones de la emisión, a diferencia de otras herramientas.

5.1.3. La necesidad de tener nuestro propio software.

- Cabe destacar que la libertad de que al tener a la mano un Software de código abierto se permite modificar el algoritmo con la geometría y el patrón de radiación para así poder obtener la información requerida dependiendo del tipo de nueva Antena que deseen analizar los estudiantes.

- No se subestima el trabajo, ni el propósito es desvalorizar el valor intelectual, pero un Software propio reduciría los costos al evitar pagar licencias y se tendría mayor independencia frente al uso de políticas que se requieren aceptar con el uso de estos.

- Es posible hacer uso nuevamente de este Software para su optimización, integración de información sobre otros tipos de Antenas, la independencia tecnológica al no depender de empresas que podrán ya no existir o, a pesar de existir software de acceso libre, este suele funcionar como una caja negra, en la que no son claros los procesos que subyacen a los cálculos.

- Sobre todo, es una gran contribución académica para el uso de futuras investigaciones o implementaciones del programa en la carrera de ingeniería física. El valor intelectual permanece dentro de la institución.

Si se observa esta información, es claro que se puede considerar que los precios están sobrevalorados. Es razonable decir que la formación en ingeniería física proporciona las capacidades de poder desarrollar un Software gracias a la formación que la institución le imparte, con ayuda de las herramientas que se le proporcionaron, teniendo los cimientos del electromagnetismo, la electrodinámica, el análisis de radiación, el pensamiento crítico, la formación físico-matemática, el razonamiento lógico, el uso de lenguajes de programación como Python.

5.2. Sobre el software.

Se procedió a elaborar la ventana gráfica que incluyera las 4 opciones. El diseño se ve en la figura 33. Las listas de opciones son:

a. Simulación completa del patrón en una Helicoide. Esta opción se usa para la descripción del patrón de radiación de manera general, con todos los modos de transmisión posibles.

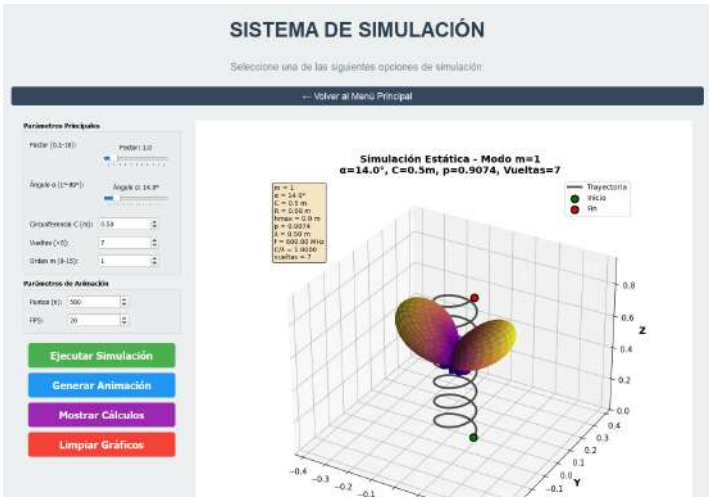
b. Simulación patrón de orden de transmisión 1. Esta interfaz describe de manera adecuada al primer modo de transmisión y sus límites físicos.

c. Cálculo de diseño de la Helicoide con circuito RLC. Se usa para diseñar un circuito muy sencillo que pueda describir los parámetros que se ocupan para que la Helicoide transmita en el primer modo y su rango de impedancias que se traten por un potenciómetro.

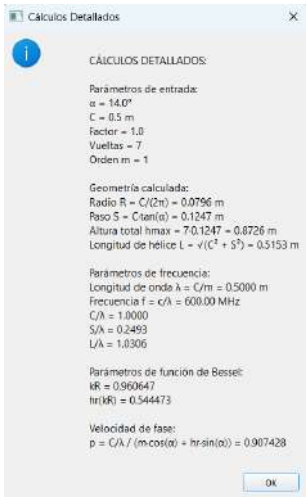
d. Simulación del patrón de orden 1 con circuito RLC. Con ayuda de la opción 3, se introducen las características y opera conforme al primer modo de transmisión en un circuito RLC.

Figura 33: Interfaz del programa con 4 opciones (autoría propia).

Una vez hecha la parte teórica para las ecuaciones y la experimental para los códigos, se diseñó la interfaz para la primera opción ya comentada. Esta opción solicita el "factor" que permite aumentar o disminuir el tamaño del patrón de radiación; se solicita un ángulo entre 0 y 90, una circunferencia, un número de vueltas mayor a 0 y el modo de transmisión. Existen varias opciones: la opción de "Ejecutar Simulación", que ejecuta y muestra la imagen de cómo se observa el patrón, como se muestra en la figura 34a; la opción de "Generar Animación", que crea el GIF de animación; la opción de "Mostrar Cálculos", que presenta los resultados de geometría, los parámetros de emisión, los parámetros de la función de Bessel y la velocidad de fase, como se muestra en la figura 34b; y la opción de "Limpiar Gráficos".



(a) Opción 1 del programa, simulación (autoría propia).



(b) Opción 1 del programa, cálculos (autoría propia).

Figura 34: Opción 1 del programa.

La siguiente interfaz que se generó fue destinada para el modo de transmisión 1 ya explicado. En esta se solicita el ángulo $12 < \alpha < 15$, la circunferencia, el factor que aumenta el tamaño del patrón, el número de vueltas con $n \geq 3$ y el valor de $.75 < C_\lambda < 1.25$ para determinar la frecuencia. Se cuenta con la opción de "Ejecutar Simulación", que ejecuta y muestra la imagen de cómo se observa el patrón; la opción de "Generar Animación", que crea el GIF; y la opción de "Limpiar Gráficos". Todo esto se puede observar en la figura 35.

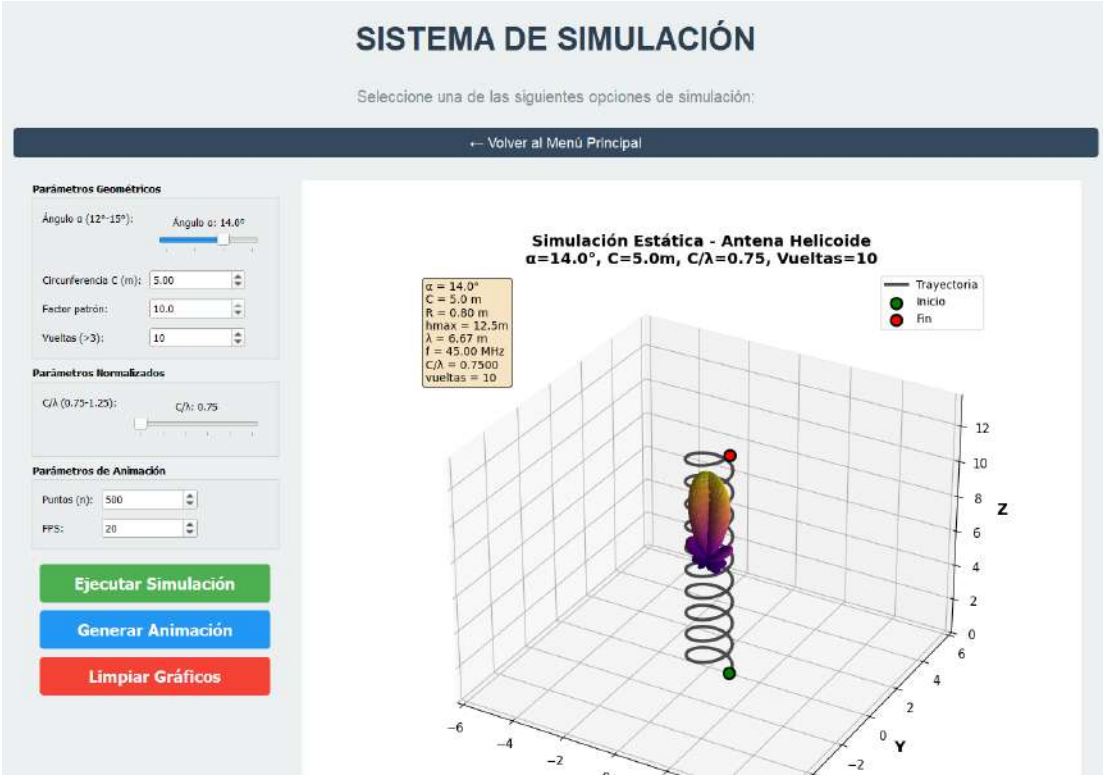
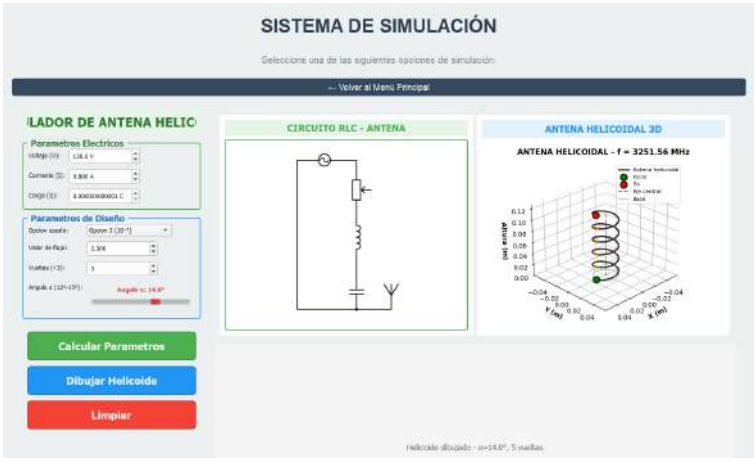


Figura 35: Opción 2 del programa, simulación (autoría propia).

Para la tercera interfaz, se incorporó la imagen de un circuito RLC con el fin de hacer más representativa la relación con el programa. Esta sección opera en el modo de transmisión 1, por lo que se solicita el ángulo $12 < \alpha < 15$, la circunferencia, el factor que aumenta el tamaño del patrón, el número de vueltas con $n \geq 3$, el voltaje del circuito, la corriente, la carga del capacitor y el flujo del inductor.

Se cuenta con la opción de "Calcular Parámetros", mostrada en la figura 36b, la cual calcula $C_\lambda = 1$, la capacitancia, la inductancia, la frecuencia, λ, la circunferencia, el radio, la altura, la longitud de vuelta, el rango de frecuencias que se pueden adaptar para $m = 1$, así como el rango de impedancias necesario para la implementación de un potenciómetro en el circuito. También se dispone de la opción de "Dibujar Helicoide", que genera la imagen como se observa en la figura 36a, y la opción de "Limpiar". Todo esto se puede observar en la figura 36.



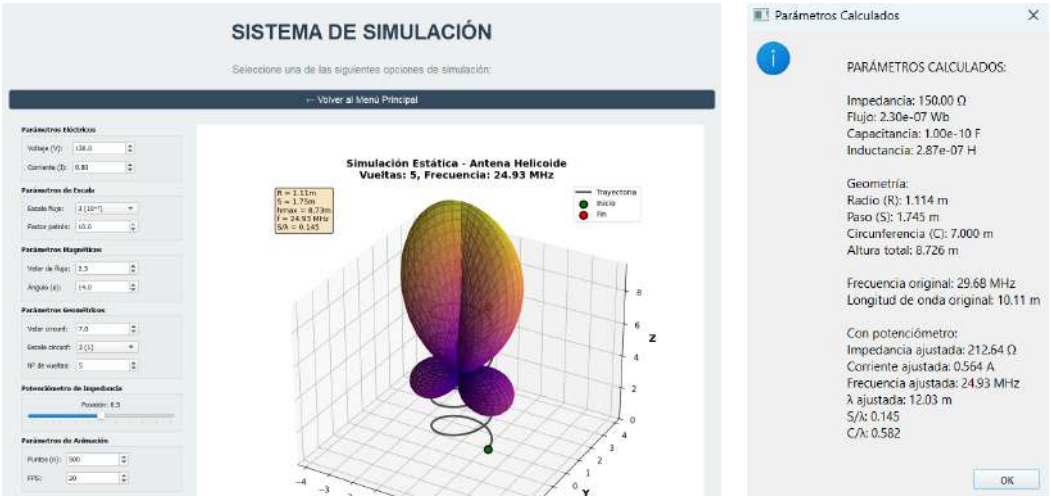
(a) Opción 3 del programa, diseño (autoría propia).

(b) Opción 3 del programa, cálculos (autoría propia).

Figura 36: Opción 3 del programa.

La opción 4 genera una interfaz que solicita los parámetros para el modo de transmisión 1. En esta se requiere el ángulo $12 < \alpha < 15$, la circunferencia, el factor que aumenta el tamaño del patrón, el número de vueltas con $n \geq 3$, el voltaje del circuito, la corriente, la carga del capacitor y el flujo del inductor; además, se introduce un potenciómetro para abarcar la impedancia correspondiente a este modo de transmisión.

Se cuenta con la opción de "Ejecutar Simulación", que ejecuta y muestra la imagen de cómo se observa el patrón, como se presenta en la figura 37a; la opción de "Generar Animación", que crea el GIF; la opción de "Mostrar Cálculos", que presenta los resultados de la impedancia, el flujo, la capacitancia, la inductancia, el radio, la longitud entre vueltas, la circunferencia, la altura, el rango del potenciómetro, la corriente ajustada a $C_\lambda = 1$, la frecuencia y λ ajustadas a $C_\lambda = 1$, así como los valores de C_λ y S_λ, como se observa en la figura 37b; y la opción de "Limpiar Gráficos".



(a) Opción 4 del programa, simulación (autoría propia).

(b) Opción 4 del programa, cálculos (autoría propia).

Figura 37: Opción 4 del programa.

5.3. Larmor.

Durante el desarrollo de este trabajo se piensa que el análisis de Larmor no funcionaba ya que era obsoleto, sin embargo al observar la Figura 21, se puede ver una región específica tomada para valores de orden de transmisión $m = 0$, donde su región va en el rango de valores de L_λ, C_λ y S_λ son menores a 0.5. Cuando esto sucede, pasa algo interesante. Si graficamos con ayuda de nuestro programa en la opción 1 lo que se observa es un comportamiento del patrón de radiación de tipo dipolo, lo que significa que Larmour sí es una aproximación de una antena Helicoidal sólo que a parámetros muy específicos como los antes mencionados.

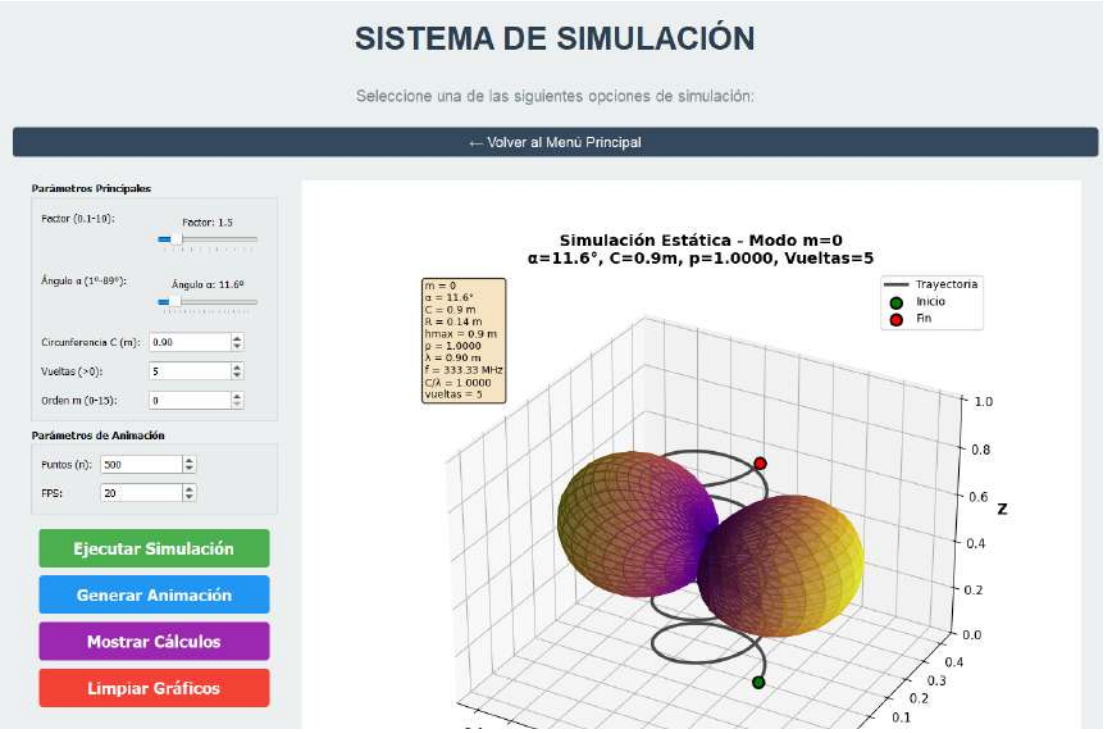
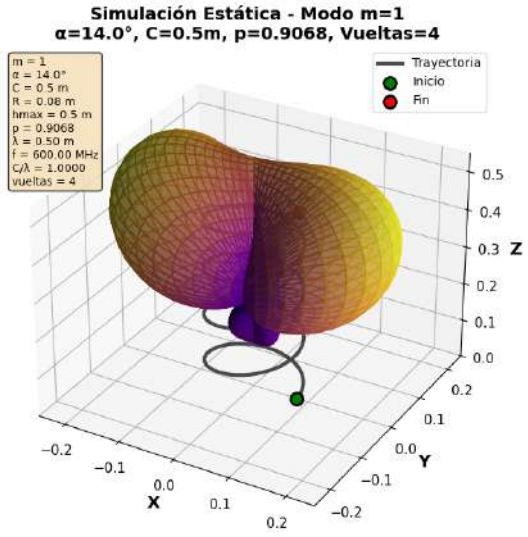


Figura 38: Larmour simulado con parámetros específicos (autoría propia).

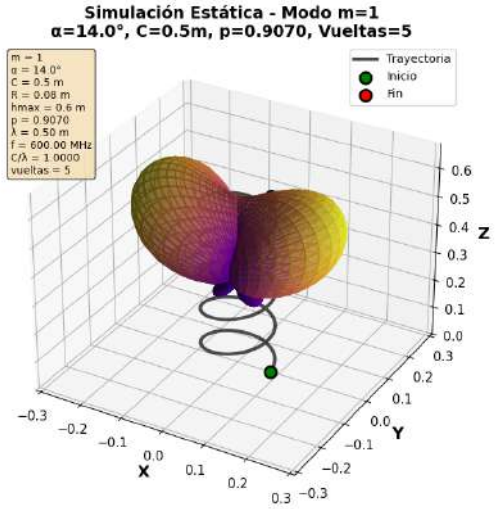
En la Figura 38, se observa el comportamiento de un dipolo. Se ha de señalar que es un aproximado, debido a que el programa de la opción 1 sólo permite un $C_\lambda \approx 1$, por lo que es un aproximado de la verdadera forma pero la naturaleza dipolar es evidente.

5.4. Validación contra la teoría.

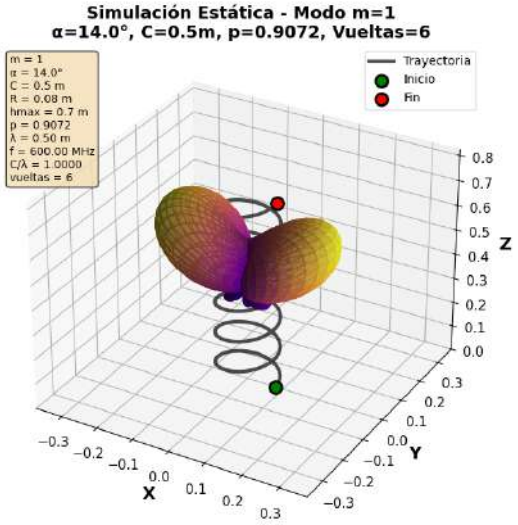
5.4.1. A mayor número de vueltas, el patrón se estrecha.



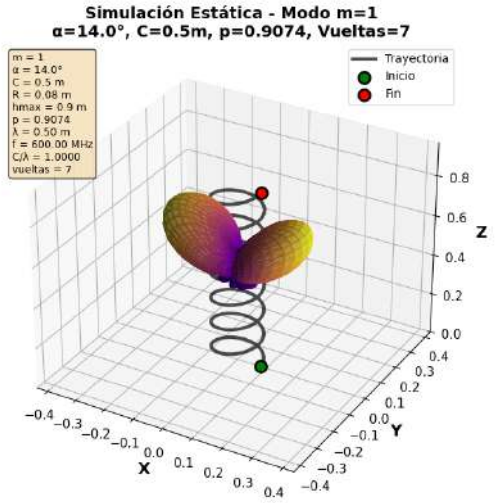
(a) Modo $m = 1$, 4 vueltas.



(b) Modo $m = 1$, 5 vueltas.



(c) Modo $m = 1$, 6 vueltas.



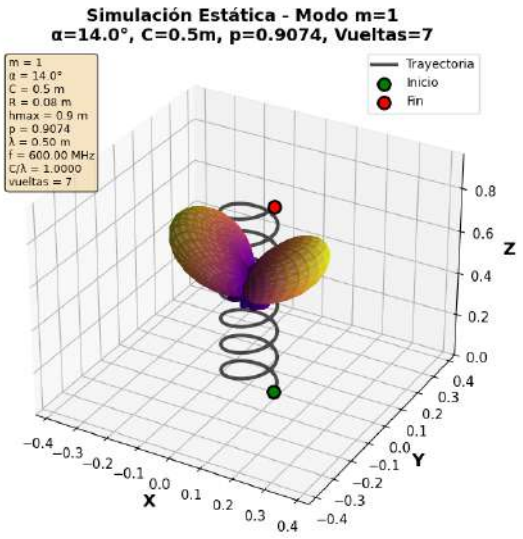
(d) Modo $m = 1$, 7 vueltas.

Figura 39: Comparación del patrón de radiación de la antena helicoidal para diferentes números de vueltas en el modo de transmisión $m = 1$ (autoría propia).

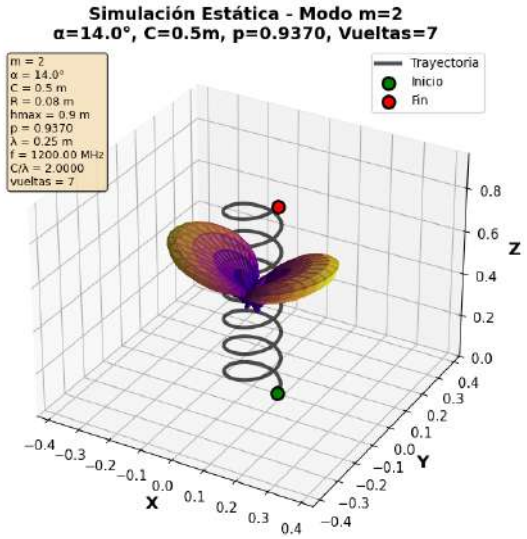
En la Figura 39 se puede observar en los resultados que el factor de paso p (recordemos que si este factor es 1, es igual a la velocidad de la luz) aumenta conforme el número de vueltas. El patrón se estrecha de manera proporcional conforme el aumento del número de vueltas.

Por otro lado, se observa que las vueltas no influyen en la longitud de onda resultante, pero sí en el incremento de lóbulos secundarios del patrón de radiación. Por lo que lo más correcto dentro de un diseño es no incrementar demasiado el número de las vueltas.

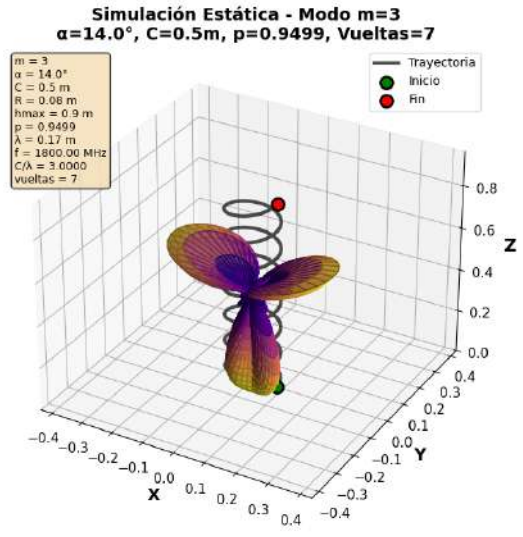
5.4.2. A mayor modo de transmisión mayor tamaño y lóbulos.



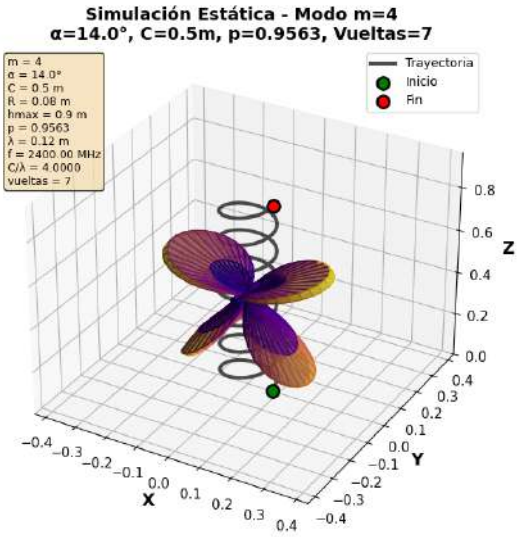
(a) Modo $m = 1$.



(b) Modo $m = 2$.



(c) Modo $m = 3$.

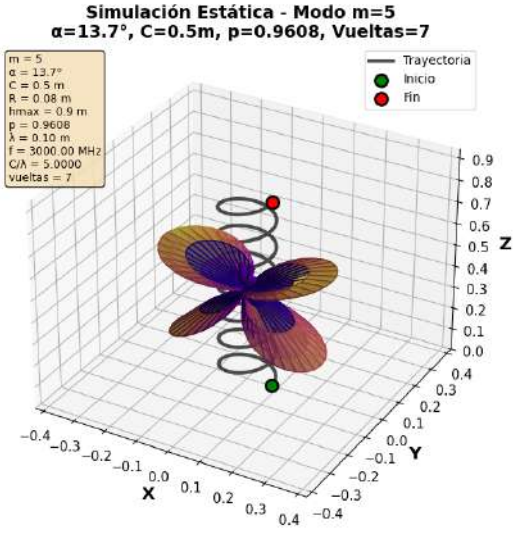


(d) Modo $m = 4$.

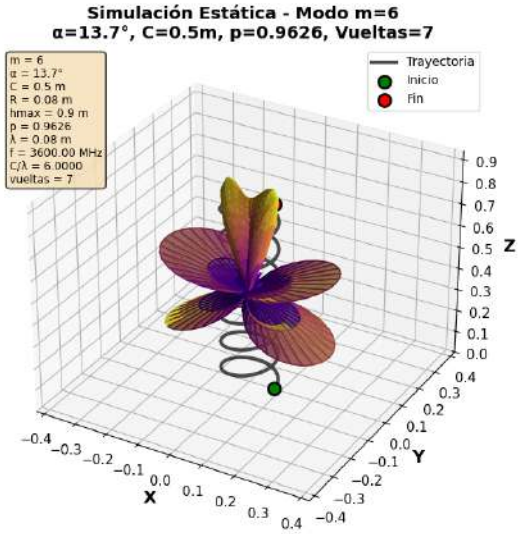
Figura 40: Comparación del patrón de radiación de la antena helicoidal con diferentes modos de transmisión parte 1 (autoría propia).

Ahora, en la Figuras 40 y 41 se optó por hacer un análisis al incremento de los

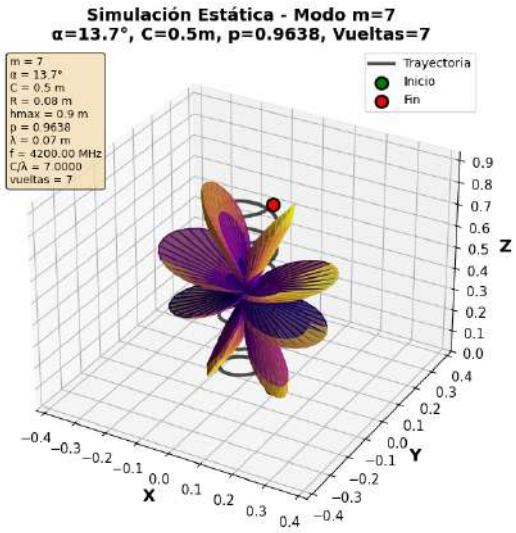
modos de transmisión 1-8, aquí se puede observar un cambio en la frecuencia de los patrones que pasaron de 600MHz a 4800MHz, lo que nos indica que es un parámetro altamente relacionado con la operación en la que trabaja la antena. También se observó que el incremento tanto de lúbulos secundarios como de lóbulos principales, de éstos últimos pasaron de 2 a 8. El factor p también se vio afectado, donde pasó de 0.9074 a 0.9648



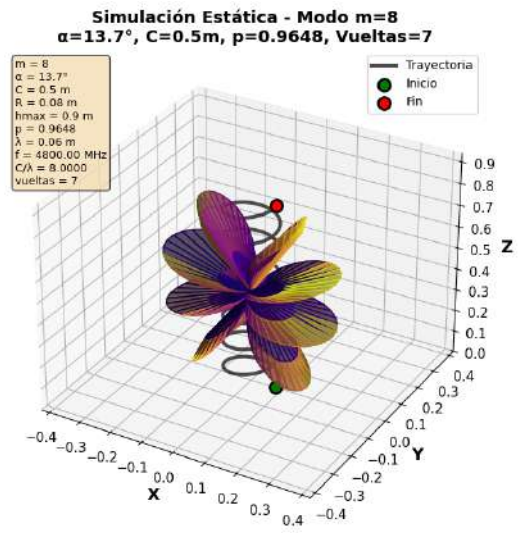
(a) Modo $m = 5$.



(b) Modo $m = 6$.



(c) Modo $m = 7$.

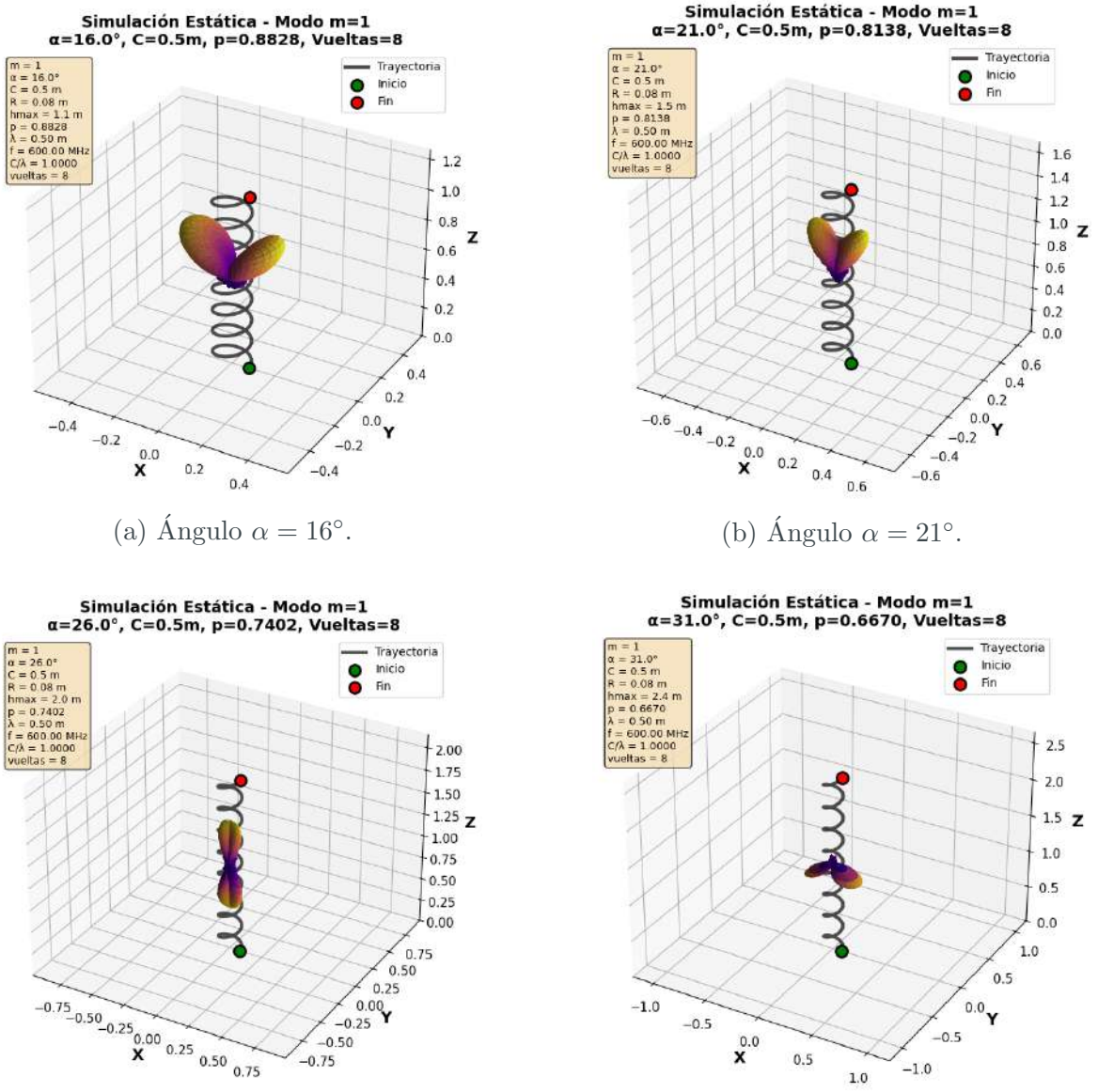


(d) Modo $m = 8$.

Figura 41: Comparación del patrón de radiación de la antena helicoidal con diferentes modos de transmisión parte 2 (autoría propia).

5.4.3. El patrón tiene deformaciones fuera de rango de alpha.

Con las variaciones que se hicieron fuera del rango donde la antena opera normalmente el ángulo $\alpha(12° - 15°)$. Se encontraron diversas deformaciones en los patrones de radiación.



(a) Ángulo $\alpha = 16°$.

(b) Ángulo $\alpha = 21°$.

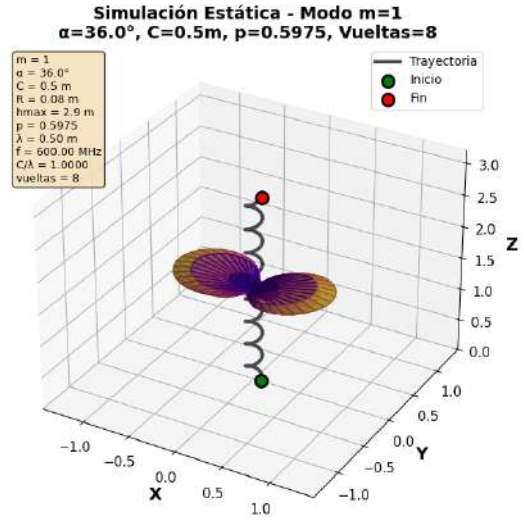
(c) Ángulo $\alpha = 26°$.

(d) Ángulo $\alpha = 31°$.

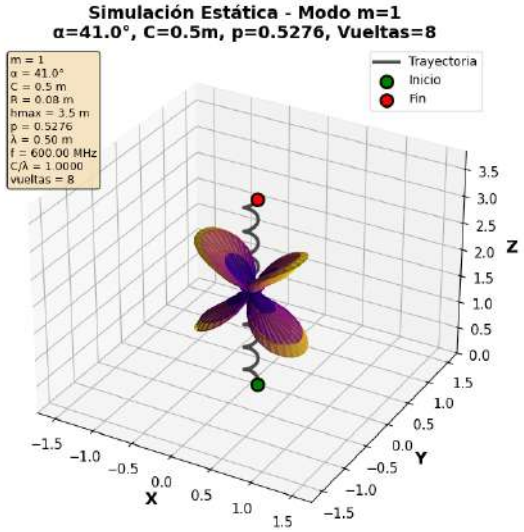
Figura 42: Comparación del patrón de radiación de la antena helicoidal variando el ángulo parte 1 (autoría propia).

Lo primero a observar en las figuras 42, 43 y 44 es que con el aumento del áangulo, el parámetro p disminuye drásticamente donde pasa de .8828 a .0411. Se pasa de

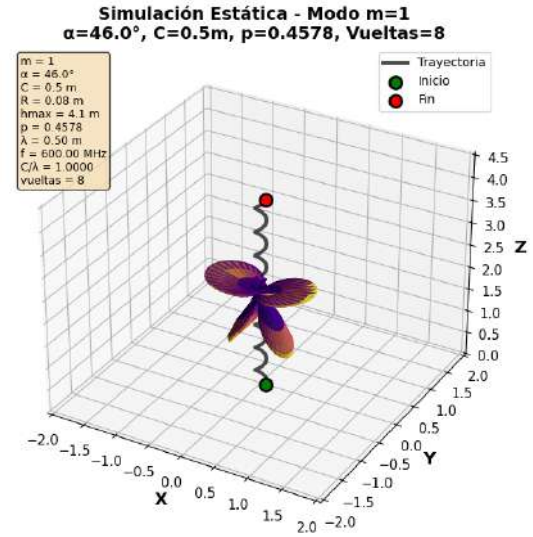
tener lóbulos bien definidos como en 42a a un lóbulo totalmente aplanado de manera horizontal en 43a e incluso tener una cantidad de 10 lóbulos en 44c.



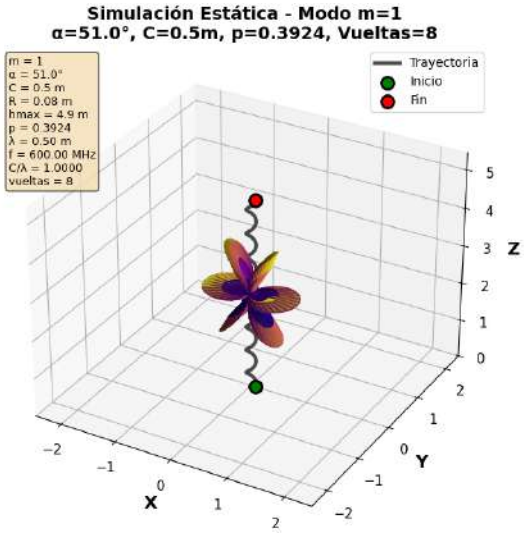
(a) Ángulo $\alpha = 36°$.



(b) Ángulo $\alpha = 41°$.



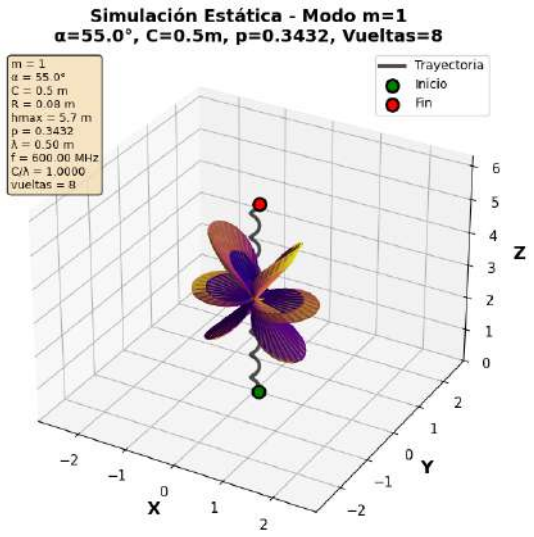
(c) Ángulo $\alpha = 46°$.



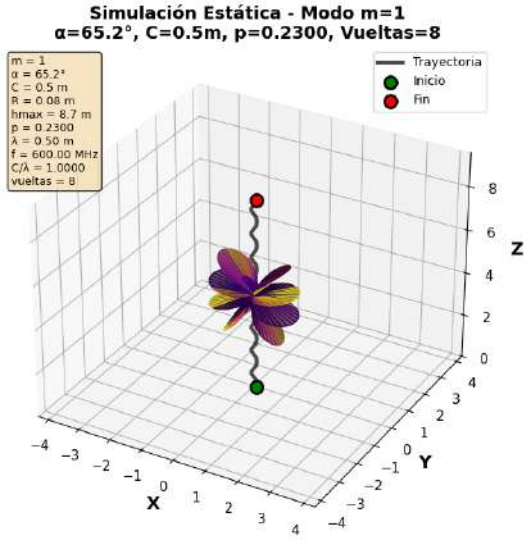
(d) Ángulo $\alpha = 51°$.

Figura 43: Comparación del patrón de radiación de la antena helicoidal variando el ángulo parte 2 (autoría propia).

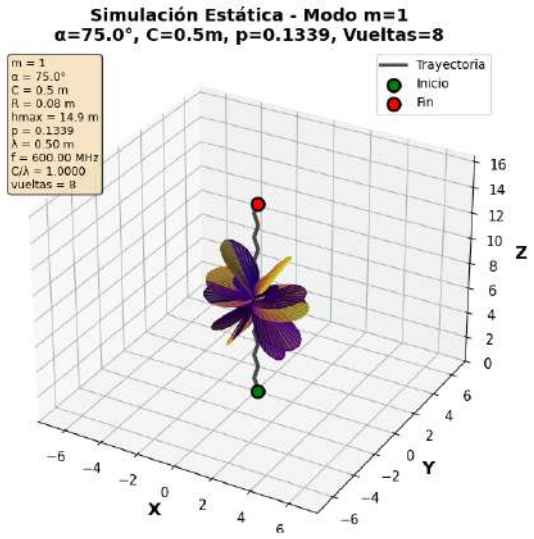
Se observa que tanto la longitud de onda como la frecuencia no se ven afectados por este parámetro, por lo que si es el caso de buscar un parámetro que afecte estos resultados, no es el correcto.



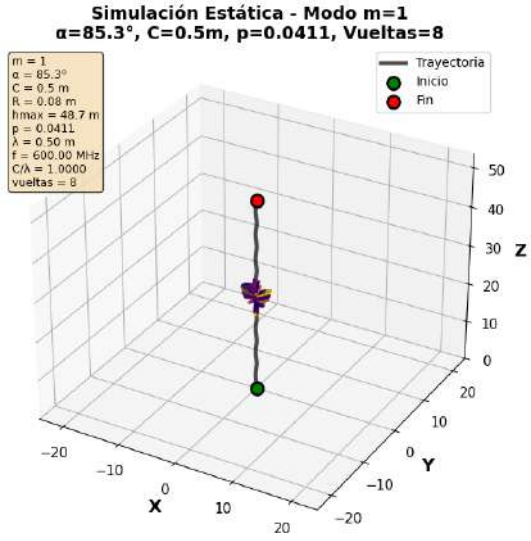
(a) Ángulo $\alpha = 55°$.



(b) Ángulo $\alpha = 65.2°$.



(c) Ángulo $\alpha = 75°$.

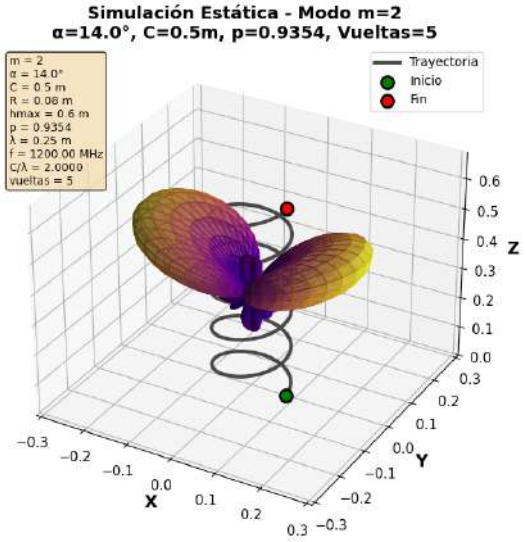


(d) Ángulo $\alpha = 85.3°$.

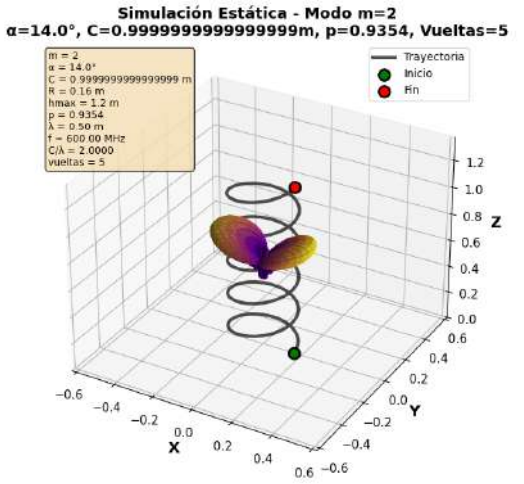
Figura 44: Comparación del patrón de radiación de la antena helicoidal variando el ángulo parte 3 (autoría propia).

Por último se ha de comentar que con los análisis visuales realizado no es recomendable elaborar el diseño de una antena con un gran grado de separación entre las espiras de la helicoide, ya que puede pasar de un estado óptimo a uno donde el estado del modo axial está fuera de lo ideal lo que implica una dispersión de lo que se busca en una antena en la transmisión total de la información.

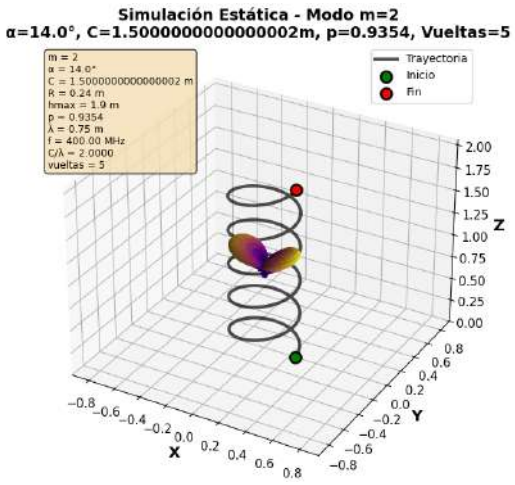
5.4.4. ¿Qué sucede al variar la circunferencia?



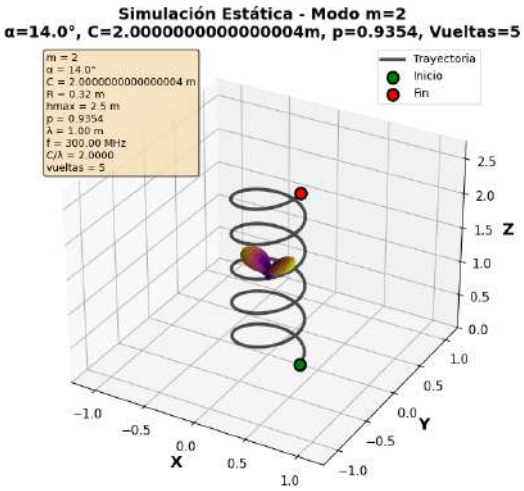
(a) Circunferencia $C = .5m$.



(b) Circunferencia $C = 1m$.



(c) Circunferencia $C = 1.5m$.

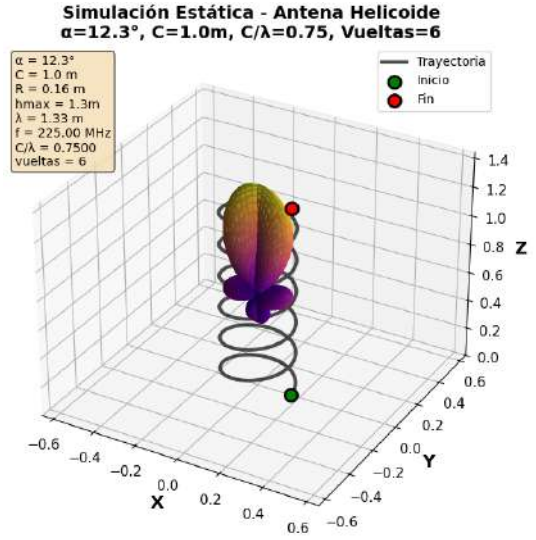


(d) Circunferencia $C = 2m$.

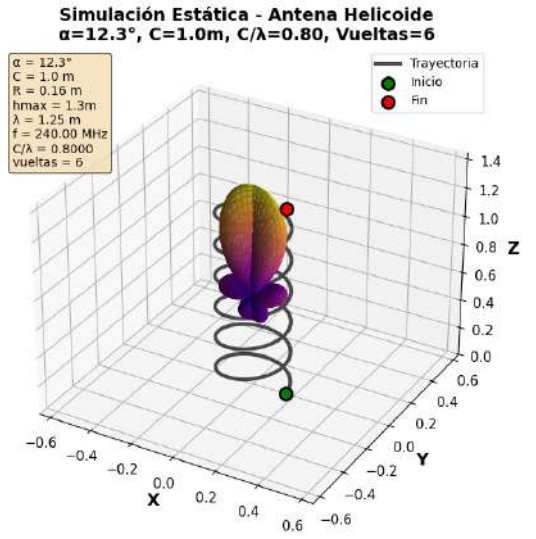
Figura 45: Comparación del patrón de radiación de la antena helicoidal variando la circunferencia (autoría propia).

Tal y como se visualiza en la Figura 45, Al hacer la variación de la circunferencia lo que varía con ésta es la frecuencia y la longitud de onda donde se pasó de 1200MHz a 300MHz. Entonces, si se busca hacer un cambio de operación en la antena puede ser viable alterar este parámetro ya que con él no se afecta la forma del patrón normalizado.

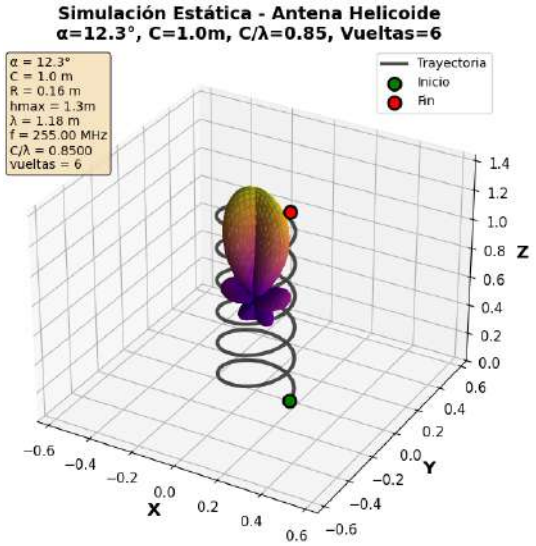
5.4.5. El primer modo de transmisión y las variaciones de C/λ.



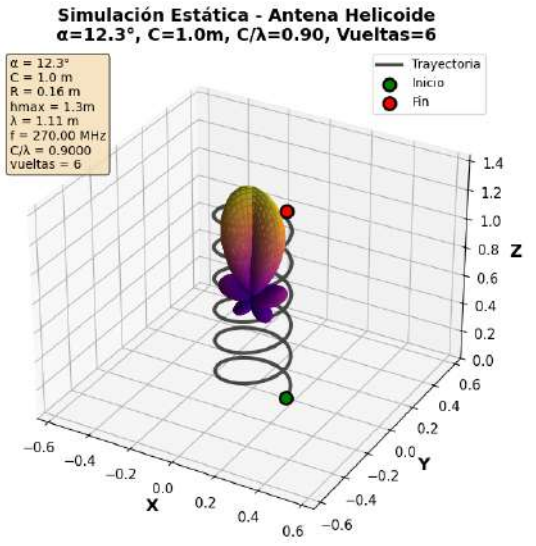
(a) $C/\lambda = .75$.



(b) $C/\lambda = .8$.



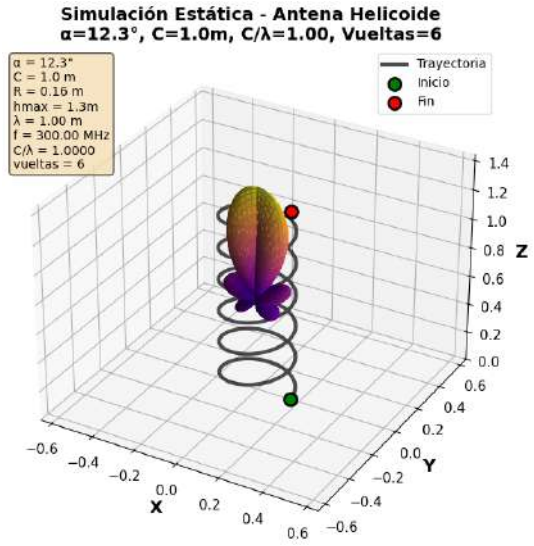
(c) $C/\lambda = .85$.



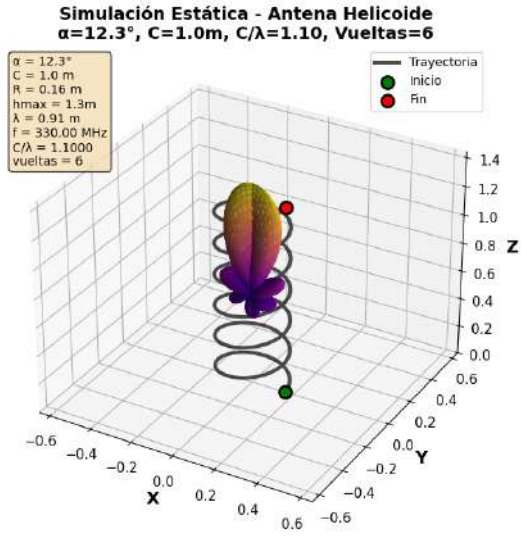
(d) $C/\lambda = .9$.

Figura 46: Comparación del patrón de radiación de la antena helicoidal variando C/λ parte 1 (autoría propia).

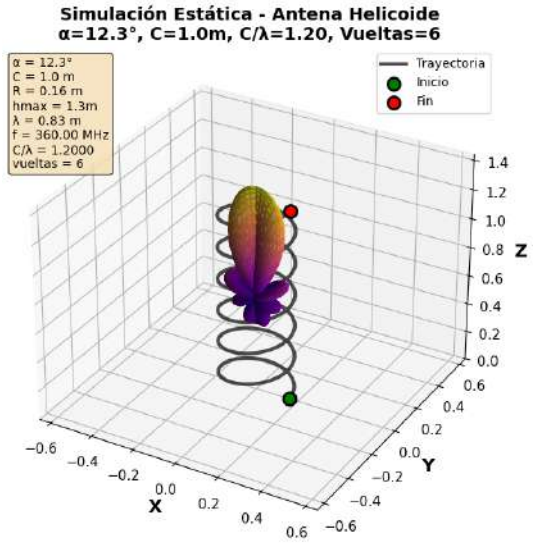
Visualmente el primer gran cambio que se observa no es del todo el cambio de la forma del patrón de radiación, pero sí el incremento de los lóbulos secundarios en la parte inferior de estos como se muestra en las Figuras 46 y 47.



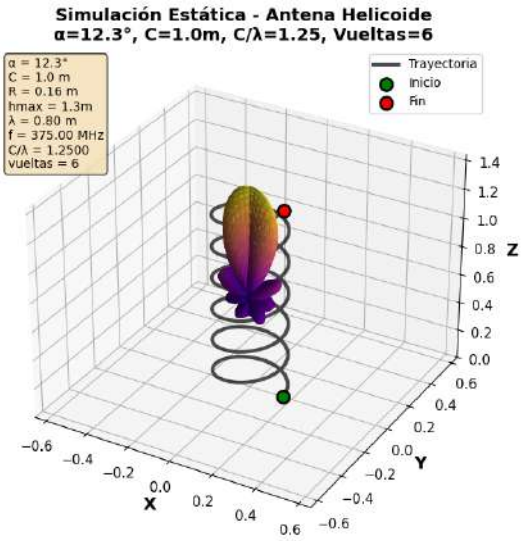
(a) $C/\lambda = 1$.



(b) $C/\lambda = 1.1$.



(c) $C/\lambda = 1.2$.

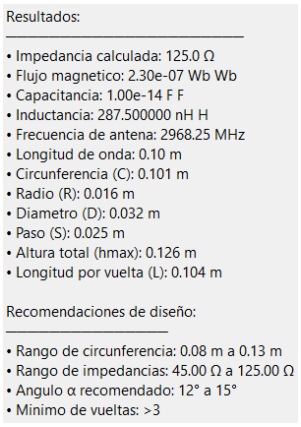


(d) $C/\lambda = 1.25$.

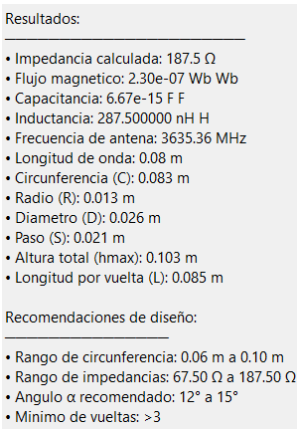
Figura 47: Comparación del patrón de radiación de la antena helicoidal variando C/λ parte 2 (autoría propia).

Lo que se quizo hacer con las Figuras 46 y 47, fue observar el comportamiento de los patrones de radiación en las condiciones óptimas de C_λ de 0.75-1.25, donde se esperaría que el comportamiento ideal se encuentra en $C_\lambda = 1$ como se ve en la Figura 47a y lo menos adecuados en los extremos de este rango como se observa en las Figuras 46a y 47d.

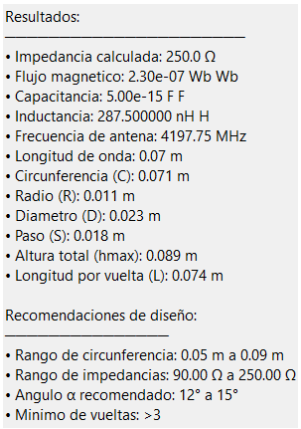
5.4.6. ¿Qué sucede al hacer cambios de voltaje?



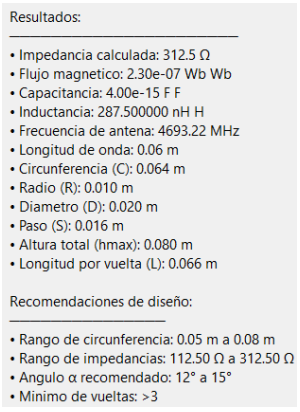
(a) $V = 100\,\mathrm{V}$



(b) $V = 150\,\mathrm{V}$



(c) $V = 200\,\mathrm{V}$



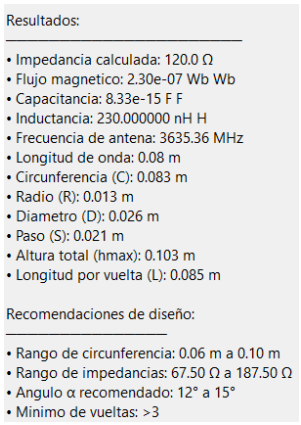
(d) $V = 250\,\mathrm{V}$

Figura 48: Resultados al variar el voltaje para una antena helicoidal con 5 vueltas, $\alpha = 14°$, $\Phi = 2.3 \times 10^{-7}\,\mathrm{Wb}$ e $I = 0.8\,\mathrm{A}$ (autoría propia).

Al analizar la Figura 48, donde se recomienda el diseño de la antena, se ve que la impedancia del circuito (de 125Ω a 300Ω) y la frecuencia de la antena (de 2968.25 MHz a 4693.22 MHz) aumentan sus valoras al incrementar V; el flujo magnético y la inductancia permanecen constantes; mientras que el diámetro, la altura, la longitud por vuelta y la capacitancia disminuyen.

Por otro lado, al observar las recomendaciones de diseño generadas por el sistema, se puede notar que el rango de circunferencias sugerido se reduce conforme aumenta el voltaje, pasando de un rango de 0.06 m a 0.12 m con V=100 V, hasta un rango de 0.05 m a 0.08 m con V=200 V. De manera análoga, el rango de impedancias recomendado se amplía con el voltaje.

5.4.7. ¿Qué sucede al cambiar la corriente?

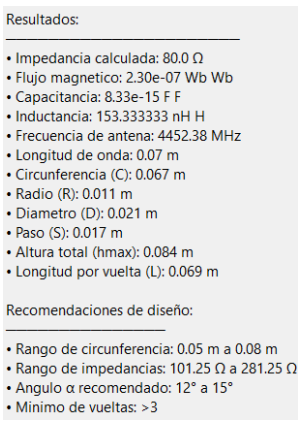

Resultados:

- Impedancia calculada: 120.0 Ω
- Flujo magnetico: 2.30e-07 Wb Wb
- Capacitancia: 8.33e-15 F F
- Inductancia: 230.000000 nH H
- Frecuencia de antena: 3635.36 MHz
- Longitud de onda: 0.08 m
- Circunferencia (C): 0.083 m
- Radio (R): 0.013 m
- Diametro (D): 0.026 m
- Paso (S): 0.021 m
- Altura total (hmax): 0.103 m
- Longitud por vuelta (L): 0.085 m

Recomendaciones de diseño:

- Rango de circunferencia: 0.06 m a 0.10 m
- Rango de impedancias: 67.50 Ω a 187.50 Ω
- Angulo α recomendado: 12° a 15°
- Minimo de vueltas: >3

(a) $I = 1.0\,\text{A}$

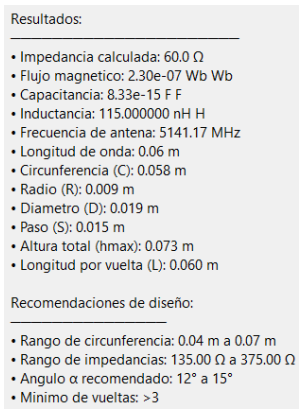

Resultados:

- Impedancia calculada: 80.0 Ω
- Flujo magnetico: 2.30e-07 Wb Wb
- Capacitancia: 8.33e-15 F F
- Inductancia: 153.333333 nH H
- Frecuencia de antena: 4452.38 MHz
- Longitud de onda: 0.07 m
- Circunferencia (C): 0.067 m
- Radio (R): 0.011 m
- Diametro (D): 0.021 m
- Paso (S): 0.017 m
- Altura total (hmax): 0.084 m
- Longitud por vuelta (L): 0.069 m

Recomendaciones de diseño:

- Rango de circunferencia: 0.05 m a 0.08 m
- Rango de impedancias: 101.25 Ω a 281.25 Ω
- Angulo α recomendado: 12° a 15°
- Minimo de vueltas: >3

(b) $I = 1.5\,\text{A}$

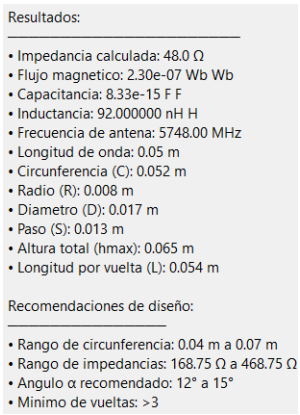

Resultados:

- Impedancia calculada: 60.0 Ω
- Flujo magnetico: 2.30e-07 Wb Wb
- Capacitancia: 8.33e-15 F F
- Inductancia: 115.000000 nH H
- Frecuencia de antena: 5141.17 MHz
- Longitud de onda: 0.06 m
- Circunferencia (C): 0.058 m
- Radio (R): 0.009 m
- Diametro (D): 0.019 m
- Paso (S): 0.015 m
- Altura total (hmax): 0.073 m
- Longitud por vuelta (L): 0.060 m

Recomendaciones de diseño:

- Rango de circunferencia: 0.04 m a 0.07 m
- Rango de impedancias: 135.00 Ω a 375.00 Ω
- Angulo α recomendado: 12° a 15°
- Minimo de vueltas: >3

(c) $I = 2.0\,\text{A}$


Resultados:

- Impedancia calculada: 48.0 Ω
- Flujo magnetico: 2.30e-07 Wb Wb
- Capacitancia: 8.33e-15 F F
- Inductancia: 92.000000 nH H
- Frecuencia de antena: 5748.00 MHz
- Longitud de onda: 0.05 m
- Circunferencia (C): 0.052 m
- Radio (R): 0.008 m
- Diametro (D): 0.017 m
- Paso (S): 0.013 m
- Altura total (hmax): 0.065 m
- Longitud por vuelta (L): 0.054 m

Recomendaciones de diseño:

- Rango de circunferencia: 0.04 m a 0.07 m
- Rango de impedancias: 168.75 Ω a 468.75 Ω
- Angulo α recomendado: 12° a 15°
- Minimo de vueltas: >3

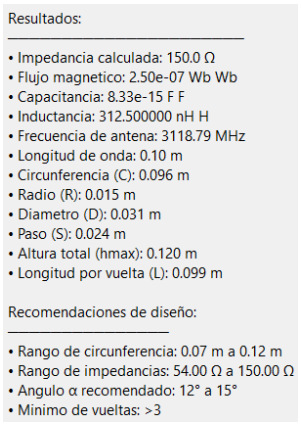
(d) $I = 2.5\,\text{A}$

Figura 49: Resultados al variar la corriente para una antena helicoidal con 5 vueltas, $\alpha = 14°$, $\Phi = 2.3 \times 10^{-7}\,\text{Wb}$ y $V = 150\,\text{V}$ (autoría propia).

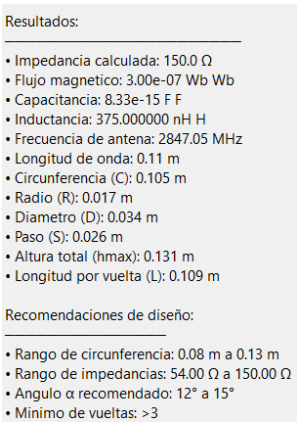
Al analizar la Figura 49, donde se presentan los resultados al realizar variaciones en la corriente eléctrica, se observa que la impedancia del circuito disminuye (de $120\,\Omega$ a $48\,\Omega$) y la frecuencia de la antena también presenta una reducción (de 8635.36 MHz a 4784.00 MHz). Por otro lado, la capacitancia incrementa su valor, mientras que el flujo magnético y la inductancia permanecen constantes. Asimismo, parámetros geométricos como la circunferencia, el radio, el diámetro, el paso, la altura total y la longitud por vuelta disminuyen a medida que se incrementa la corriente.

Al observar las recomendaciones de diseño generadas por el sistema, el rango de circunferencias sugerido se reduce, pasando de un intervalo de 0.06 m a 0.10 m con $I = 1.0\,\text{A}$, hasta un intervalo de 0.04 m a 0.07 m con $I = 2.5\,\text{A}$. De manera similar, el rango de impedancias recomendado aumenta al incrementar la corriente, pasando de un intervalo de 67.50 Ω a 187.50 Ω, hasta un intervalo de 168.75 Ω a 468.75 Ω.

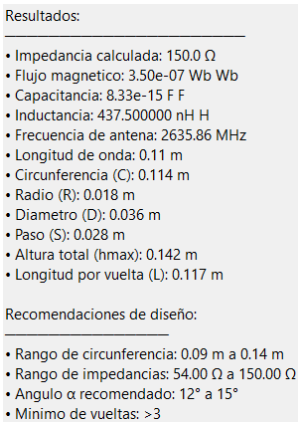
5.4.8. ¿Qué sucede al el flujo magnético?



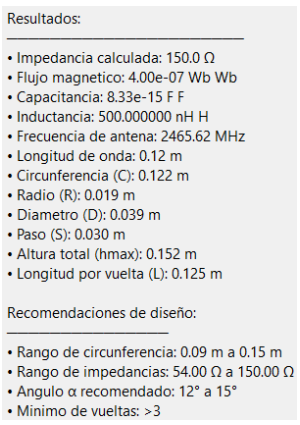
(a) $\Phi = 2.5 \times 10^{-7}\,\text{Wb}$.



(b) $\Phi = 3.0 \times 10^{-7}\,\text{Wb}$.



(c) $\Phi = 3.5 \times 10^{-7}\,\text{Wb}$.



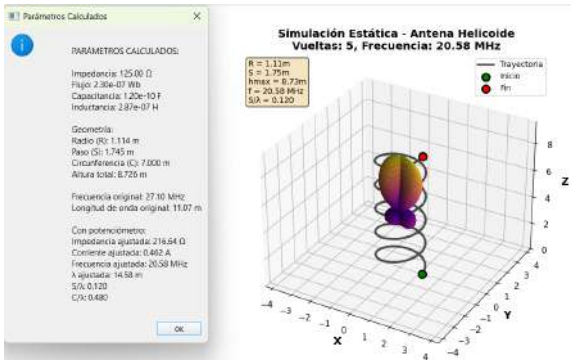
(d) $\Phi = 4.0 \times 10^{-7}\,\text{Wb}$.

Figura 50: Resultados al variar el flujo magnético para una antena helicoidal con 5 vueltas, $\alpha = 14°$, $I = 0.8\,\text{A}$ y $V = 150\,\text{V}$ (autoría propia).

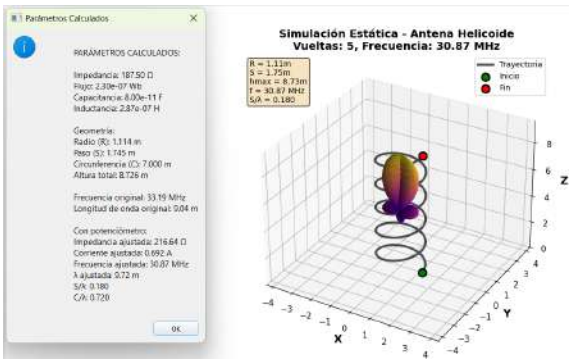
Al analizar la Figura 50, se observa que al incrementar el flujo, la capacitancia y la inductancia también aumentan, mientras que la frecuencia de la antena disminuye (de 8118.79 MHz a 2465.62 MHz). Asimismo, la longitud de onda presenta un incremento conforme a su aumento. Los parámetros geométricos aumentan con la variación de éste, mostrando un cambio significativo en la geometría recomendada de la antena.

Por otro lado, al observar las recomendaciones de diseño generadas por el sistema, se puede notar que el rango de circunferencias sugerido aumenta conforme se incrementa el flujo magnético, pasando de un intervalo de 0.07 m a 0.12 m con $\Phi = 2.5 \times 10^{-7}\,\text{Wb}$, hasta un intervalo de 0.09 m a 0.15 m con $\Phi = 4.0 \times 10^{-7}\,\text{Wb}$. En contraste, el rango de impedancias recomendado permanece constante en $54.00\,\Omega$ a $150.00\,\Omega$, indicando que este parámetro de diseño no presenta cambios dentro de las recomendaciones generadas por el sistema para las condiciones evaluadas.

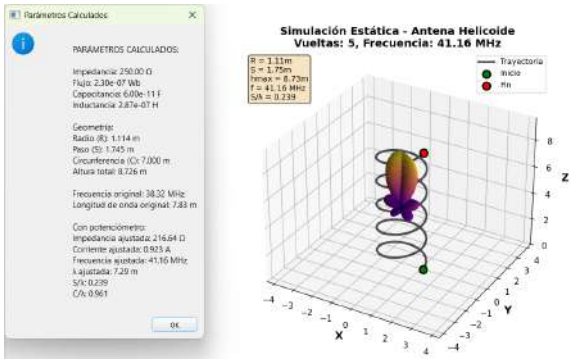
5.4.9. ¿Cómo varía el patrón si se somete a cambios de voltaje?



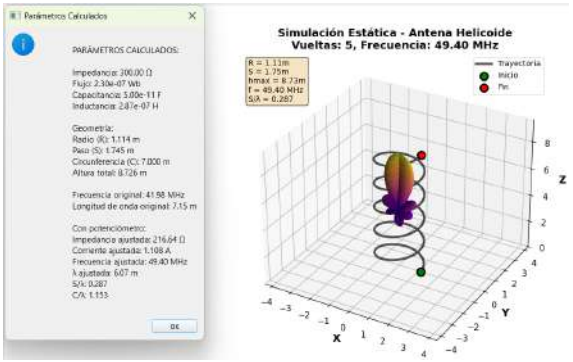
(a) 5 vueltas. $\alpha = 14°$. $\Phi = 2.3 \times 10^{-7}$ Wb. $I = 0.8$ A. $V = 100$ V.



(b) 5 vueltas. $\alpha = 14°$. $\Phi = 2.3 \times 10^{-7}$ Wb. $I = 0.8$ A. $V = 150$ V.



(c) 5 vueltas. $\alpha = 14°$. $\Phi = 2.3 \times 10^{-7}$ Wb. $I = 0.8$ A. $V = 200$ V.



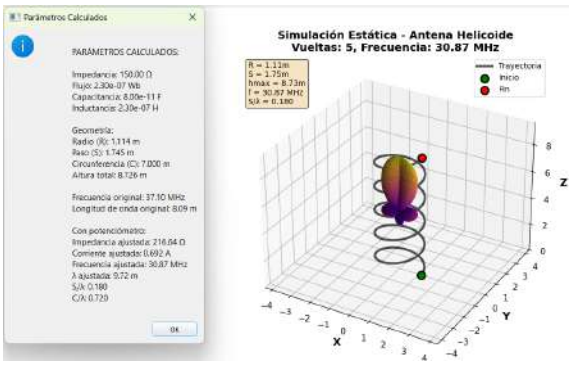
(d) 5 vueltas. $\alpha = 14°$. $\Phi = 2.3 \times 10^{-7}$ Wb. $I = 0.8$ A. $V = 240$ V.

Figura 51: Cambios del patrón de radiación con variaciones de voltaje (autoría propia).

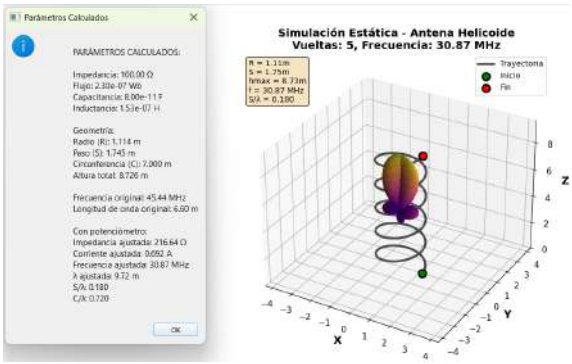
Tal y como se observa en la Figura 51, existe un aumento en la cantidad de los lóbulos secundarios, esto sugiere una redistribución de la energía radiada en distintas direcciones espaciales. La frecuencia aumenta de 20.58MHz a 49.40MHz, esto genera modificaciones en el patrón de radiación. En particular, se observa que el lóbulo principal adquiere una forma más estrecha lo que indica un incremento en la directividad del sistema.

El incremento del voltaje no solo afecta la frecuencia de operación, sino que también modifica la forma del patrón de radiación, produciendo una respuesta más direccional y compleja, con una mayor presencia de lóbulos secundarios y una concentración más pronunciada de la energía radiada en el lóbulo principal.

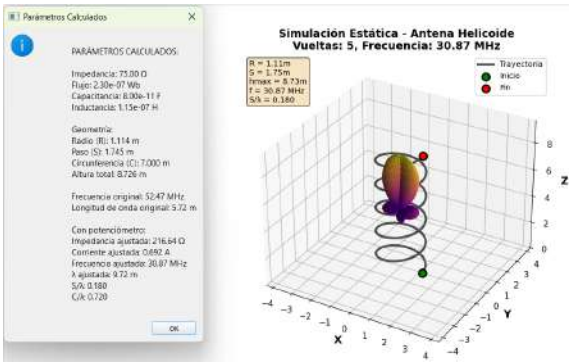
5.4.10. ¿Cómo varía el patrón si se somete a cambios de corriente?



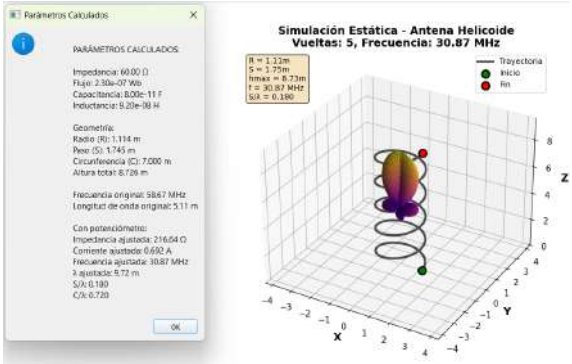
(a) 5 vueltas. $\alpha = 14°$. $\Phi = 2.3 \times 10^{-7}$ Wb. $I = 1.0$ A. $V = 150$ V.



(b) 5 vueltas. $\alpha = 14°$. $\Phi = 2.3 \times 10^{-7}$ Wb. $I = 1.5$ A. $V = 150$ V.



(c) 5 vueltas. $\alpha = 14°$. $\Phi = 2.3 \times 10^{-7}$ Wb. $I = 2.0$ A. $V = 150$ V.



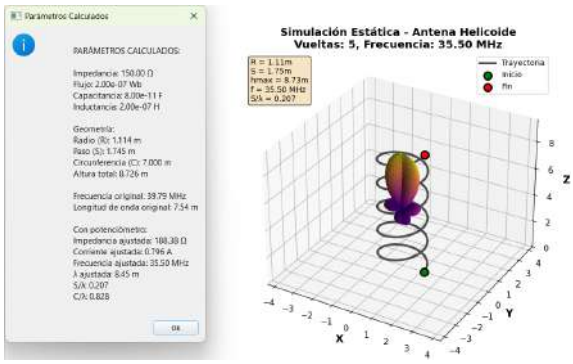
(d) 5 vueltas. $\alpha = 14°$. $\Phi = 2.3 \times 10^{-7}$ Wb. $I = 2.5$ A. $V = 150$ V.

Figura 52: Cambios del patrón de radiación con variaciones de corriente (autoría propia).

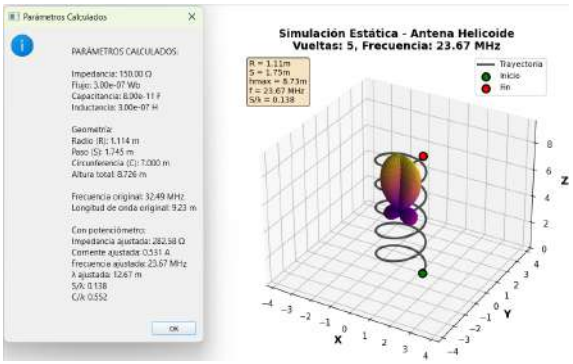
En la Figura 52, a diferencia de lo ocurrido con el voltaje, el patrón de radiación permanece prácticamente invariante al modificar la corriente. La frecuencia ajustada con el potenciómetro se mantiene en 30.87 MHz en los cuatro casos evaluados, esto implica que tanto el lóbulo principal como los lóbulos secundarios conservan su forma, orientación y amplitud relativa sin modificación apreciable. Esto sugiere que, bajo condiciones de voltaje y flujo magnético constantes, la corriente no redistribuye la energía radiada entre direcciones espaciales distintas.

Lo anterior se explica El incremento de corriente sí produce cambios eléctricos relevantes, la impedancia disminuye y la inductancia cae proporcionalmente para conservar el flujo magnético impuest. Pero estos cambios no se traducen en una modificación observable del diagrama de radiación.

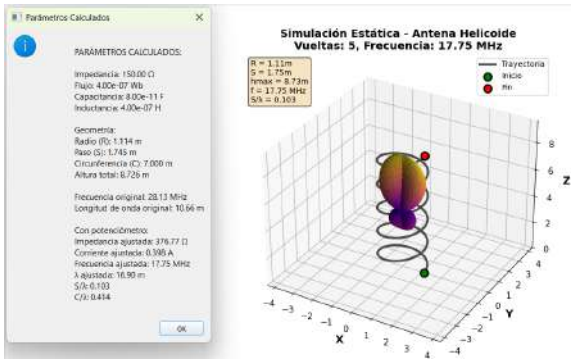
5.4.11. ¿Cómo varía el patrón si se somete a cambios de flujo?



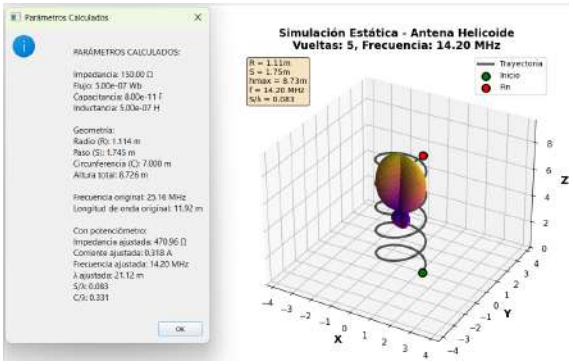
(a) 5 vueltas. $\alpha = 14°$. $\Phi = 2 \times 10^{-7}$ Wb. $I = 0.8$ A. $V = 150$ V.



(b) 5 vueltas. $\alpha = 14°$. $\Phi = 3 \times 10^{-7}$ Wb. $I = 0.8$ A. $V = 150$ V.



(c) 5 vueltas. $\alpha = 14°$. $\Phi = 4 \times 10^{-7}$ Wb. $I = 0.8$ A. $V = 150$ V.



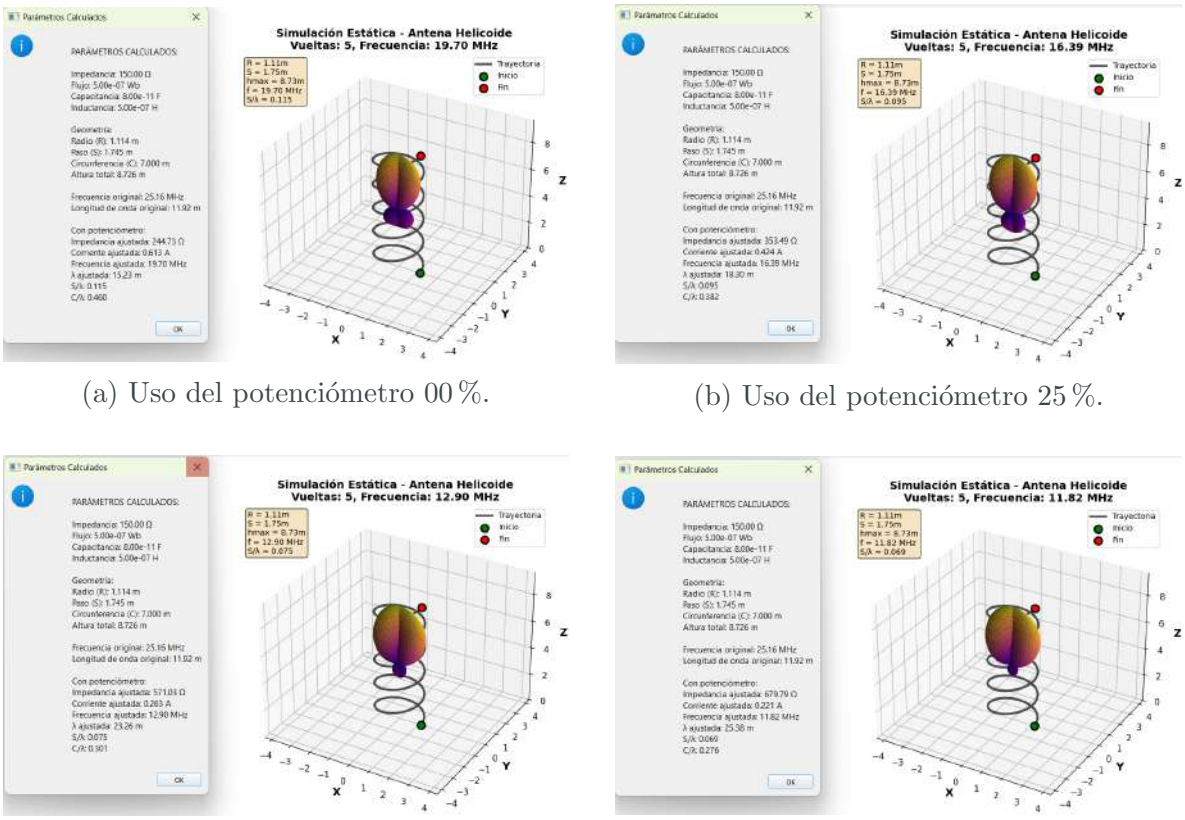
(d) 5 vueltas. $\alpha = 14°$. $\Phi = 5 \times 10^{-7}$ Wb. $I = 0.8$ A. $V = 150$ V.

Figura 53: Cambios del patrón de radiación con variaciones de flujo (autoría propia).

Para la Figura 53, el incremento del flujo magnético Φ produce el efecto opuesto al observado con el voltaje: el patrón de radiación se degrada progresivamente conforme Φ aumenta, desciende de 35.50 MHz a 14.20 MHz, lo que reduce la relación C/λ de 0.909 a 0.331, alejándola del valor óptimo unitario requerido. El lóbulo principal se ensancha y pierde concentración, mientras que los lóbulos secundarios adquieren mayor presencia relativa, indicando una redistribución de la energía radiada hacia direcciones no axiales.

El flujo magnético actúa como parámetro de control del tamaño físico de la antena, incrementa la inductancia y desplaza la frecuencia operativa fuera de la banda de diseño óptimo, lo que deteriora la directividad axial del sistema.

5.4.12. Variaciones con el potenciómetro.



(a) Uso del potenciómetro 00 %.

(b) Uso del potenciómetro 25 %.

(c) Uso del potenciómetro 75 %.

(d) Uso del potenciómetro 100 %.

Figura 54: Cambios del patrón de radiación con variaciones en el potenciómetro (autoría propia).

En el último análisis de resultados hechos para la Figura 54, el incremento del valor del potenciómetro produce una degradación progresiva y sistemática del patrón de radiación. Al aumentar el porcentaje del potenciómetro, la frecuencia desciende de 19.70 MHz a 11.82 MHz, reduciendo la relación C/λ de 0.460 a 0.279, lo que aleja al sistema de la condición óptima de modo axial. Este descenso se traduce en un ensanchamiento del lóbulo principal y una menor concentración de la energía radiada sobre el eje de la hélice, existe una pérdida progresiva de directividad conforme se incrementa la resistencia introducida por el potenciómetro.

El potenciómetro no modifica la geometría física de la antena, pero sí constituye el mecanismo de sintonía fina del sistema.

6.. Conclusiones

En este trabajo se analizó la formación del patrón de radiación de una antena helicoidal, así como su origen y condiciones de operación, a partir de fundamentos teóricos como el principio end-fire, el principio de Hansen–Woodyard, el principio de multiplicación, la geometría de la antena, la teoría de ondas electromagnéticas y los distintos modos de transmisión. Este análisis permitió comprender la complejidad del comportamiento radiativo de una antena helicoidal.

Se comprobó que el análisis basado en la formulación de Larmor permite explicar parcialmente el fenómeno de emisión de radiación electromagnética; sin embargo, para una descripción más completa del patrón de radiación es necesario recurrir a generalizaciones que consideren la geometría y el régimen de operación de la antena.

La principal aportación de este trabajo fue el desarrollo de un software capaz de simular distintos modos de transmisión, incluyendo el primer modo de operación, así como una implementación básica de un circuito RLC. Dicho software permite graficar el patrón de radiación y facilita el análisis del diseño de una antena con características adecuadas para una correcta radiación.

Es importante mencionar que, como trabajo futuro, este desarrollo puede ampliarse mediante la incorporación de nuevos parámetros, como la ganancia, la energía del sistema, su consumo, la eficiencia y la directividad, así como la inclusión de otros modelos de antenas. Esto con el fin de mejorar el análisis, ampliar sus aplicaciones y fortalecer su uso en el ámbito académico.

Finalmente, este trabajo establece una base sólida para la incorporación de parámetros adicionales como los ya mencionados, así como la posible inclusión de una formulación hamiltoniana. De este modo, se abren nuevas líneas de desarrollo y mejora, habiéndose cumplido satisfactoriamente los objetivos planteados inicialmente.

Referencias bibliográficas

Alexander, C. K., S. M. N. (2018). *Fundamentals of Electric Circuits*. McGraw-Hill, 6 edition.

ALLELCO (2025). Comprender la resistencia, inductancia y capacitancia en los circuitos eléctricos. Recuperado en enero de 2026, de `https://www.allelcoelec.es/blog/understanding-the-fundamentals-inductance-resistance, andcapacitance.html`.

Ampere, A. M. and Orsted, H. C. (s.f.). *Historia del electromagnetismo*. s.n.

Andrade, F. A. (1971). *Aceleradores de partículas*. DFN-SMF, México.

Antenna, L. (2022). Antenna gain explanation. Consultado el 1 de noviembre de 2025.

Aznar, C., Robert, J. R., Casals, J. M. R., Roca, L. J., Boris, S. B., and Bataller, M. F. (2004). *Antenas*. Universidad Politécnica de Catalunya, España.

Balanis, C. A. (2005). *Antenna Theory: Analysis and Design*. John Wiley & Sons, Hoboken, New Jersey, 3rd edition.

Belendez, A. (2016). 150 años de la publicación de la teoría electromagnética de la luz de maxwell. Recuperado en septiembre de 2025, de `https://rua.ua.es/server/api/core/bitstreams/9a9c5d4e-7fcf-4f81-aa01-50798257f595/content`.

Blattenberger, K. (2025). Dielectric constant, strength, & loss tangent. Consultado el 1 de noviembre de 2025.

Brain, M., Wilson, T. V., and Johnson, B. (2004). How wifi works. Recuperado en septiembre de 2025, de `https://www.albany.edu/~gc227838/ist523/How%20WiFi%20Works.pdf`.

Cardarelli, F. (2018). *Materials Handbook: A Concise Desktop Reference*. Springer International Publishing, Switzerland.

Carrigan, C. R., G. D. (1979). Earth's magnetic field. *Scientific American*.

Collett, E. (2005). *Field Guide to Polarization*. SPIE Press.

CONUEE (2021). Servicios energéticos, iluminación. Recuperado en septiembre de 2025, de `https://www.gob.mx/conuee/acciones-y-programas/servicios-energeticos-iluminacion`.

Cope, L. (2025). What is polarization in an antenna system? Consultado el 2 de noviembre de 2025.

Diagram, C. (2026). A free, user-friendly program for making electronic circuit diagrams. Consultado el 1 de enero de 2026.

Ellingson, S. W. (2025). *Electromagnetics II*. Virginia Tech Publishing. LibreTexts, licensed under CC BY-SA 4.0.

Engineering, G. (2025). An-sof antenna simulation software. Recuperado en septiembre de 2025, de `https://antennasimulator.com/`.

Fedeli, A., Montecucco, C., and Gragnani, G. L. (2019). Open-source software for electromagnetic scattering simulation: The case of antenna design. *Electronics*, 8(12):1506.

Fernandez, H. A. (2019). Ecuaciones de maxwell. Recuperado en septiembre de 2025.

Fontal, B., Suárez, T., Reyes, M., Bellandi, F., Contreras, R., and Romero, I. (2005). El espectro electromagnético y sus aplicaciones. *Escuela de la Ingeniería*, 1:24.

Fuentefría, A. S., Fernández, M. C., Llanes, M. V., and Peón, A. D. (2011). Evaluación de aptitud del funcionamiento de hornos de microondas. *Ingeniería Energética*, 32(3):35–44.

Fuentes, J. J. M. (2007). *Fundamentos de radiación y radiocomunicación*. Escuela Técnica Superior de Ingenieros.

GPS (2023). Space segment. Recuperado en enero de 2026, de `https://www.gps.gov/space-segment`.

Greiner, S. (2015). Antenna frequency explained. Consultado el 1 de noviembre de 2025.

Griffiths, D. J. (2013). *Introduction to Electrodynamics*. Pearson Education, Boston, MA, 4th edition.

Hardesty, G. (2024). Polarización de la antena: vertical, lineal: factor clave en la selección de una antena. Consultado el 2 de noviembre de 2025.

Hecht, E. (2017). *Optics*. Pearson, 5 edition.

Hoyt, R. C. (1958). What is radiation? *Bulletin of the Atomic Scientists*, 14(1):9–11.

Huidobro, J. M. (2013). Antenas de telecomunicaciones. *Revista Digital de Acta*, 18.

IEEE (2019). Standard for safety levels with respect to human exposure to electric, magnetic, and electromagnetic fields, 0 hz to 300 ghz. *IEEE*.

Kaplan, E. D., H. C. (2017). *Understanding GPS/GNSS: Principles and Applications*. Artech House., Estados Unidos, 3 edition.

Kostis, A. C., Yioultsis, T. V., and Zarbouti, P. E. (2010). A floss visual em simulator for 3d antennas. *arXiv preprint*.

Kraus, J. D. (1950). *Antennas*. McGraw-Hill Book Company.

Kuhnel, W. (2015). *Curves, Surfaces, Manifolds*. American Mathematical Society, USA, second edition. p. 11.

Lab, A. T. (2023). What is antenna gain? tutorial sobre dbi y db. Consultado el 1 de noviembre de 2025.

Lavi, S. (2025). Ansys hfss pricing plans vs. alternatives - why 5.4/10? Recuperado en septiembre de 2025, de `https://www.itqlick.com/ansys-hfss/pricing`.

Lee, S. (2025). Unlocking antenna efficiency secrets. Consultado el 1 de noviembre de 2025.

Martinez, J. (2018). Historia de la electricidad. Recuperado en septiembre de 2025, de `https://www.fceia.unr.edu.ar/~fisica3/historia.htm#Tesla`.

Martínez-Carpio, P. A. and Trelles, M. A. (2010). El láser y la fotónica en la cirugía plástica española e iberoamericana. *Cirugía Plástica Ibero-Latinoamericana*, 36(1):59–78.

Matan (2023). 4 tipos más comunes de aplicaciones de ondas electromagnéticas en comunicación – electricity – magnetism. Recuperado en septiembre de 2025, de https://www.electricity-magnetism.org/es/4-tipos-mas-comunes-de-aplicaciones-de-ondas-electromagneticas-en-comunicacion/.

Matan (2024). Understanding the antenna efficiency equation. Consultado el 1 de noviembre de 2025.

MathWorks (2019). Antenna toolbox – design, analyze, and visualize antenna elements and antenna arrays. Recuperado en septiembre de 2025, de `https://www.mathworks.com/products/antenna.html`.

Mayo Clinic (s.f.). Radiografía: prueba por imágenes que ayuda a realizar diagnósticos rápidamente. Recuperado en septiembre de 2025, de `https://www.mayoclinic.org/es/tests-procedures/x-ray/about/pac-20395303`.

Mitofsky, A. (2025). Absorption, spontaneous emission, stimulated emission. Recuperado en enero de 2026, de `https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Electro-Optics/Direct_Energy_(Mitofsky)/07%3A_Lamps_LEDs_and_Lasers/7.01%3A_Absorption_Spontaneous_Emission_Stimulated_Emission`.

Monachesi, E., Frenzel, A. M., Chaile, G., Agustín, C., and López, F. A. G. (2011). Conceptos generales de antenas. Recuperado en septiembre de 2025, de http://www.edutecne.utn.edu.ar/wlan_frt/antenas.pdf.

NASA (2023). What is a satellite? Recuperado en enero de 2026, de https://spaceplace.nasa.gov/satellite/en/.

of Encyclopaedia Britannica, T. I. A. (2025). Michael faraday. Recuperado en septiembre de 2025, de https://www.britannica.com/facts/Michael-Faraday.

OneSDR (2024). Antenna length vs frequency – understanding the relationship. Consultado el 1 de noviembre de 2025.

Online, M. (2023). What is antenna gain, dbi, and db? what's the difference. Consultado el 1 de noviembre de 2025.

OnWorks (2026). Antennavis. Recuperado en abril de 2026, de https://www.onworks.net/es/programs/antennavis-online.

Peña, X. P. (2025). Resonancia magnética, partes y funcionamiento contenido. Recuperado en septiembre de 2025, de https://d197for5662m48.cloudfront.net/documents/publicationstatus/240572/preprint_pdf/0852e962a78293ce0bc223a6dc6c42f9.pdf.

Physics, M. (2024). Intensidad del sonido: Medición, decibelios y formas de onda en acústica. Consultado el 1 de noviembre de 2025.

Polimetro (2025). Los mejores software para ingeniería: guía completa y actualizada. Recuperado en septiembre de 2025, de https://www.polimetro.com/los-mejores-software-para-ingenieria/.

Pozar, D. M. (2012). *Microwave Engineering*. Wiley, 4th edition.

Pérez Navarro, A. (2011). *Radiación. Radiación de un dipolo*. Universitat Oberta de Catalunya. PID_00159141.

Rajib, B. (2025). Radio frequency bands. Consultado el 1 de noviembre de 2025.

Remcom (2024). Software de simulación de antenas — diseño y modelado de antenas. Recuperado en septiembre de 2025, de https://es.remcom.com/applications/antenna-simulation-design-software.

RF, E. (2023). What is antenna beamwidth? half power beam width (hpbw) explained. Consultado el 1 de noviembre de 2025.

Riccio, G. L. (2024). *Crónica del electromagnetismo: Un viaje a través del tiempo y la ciencia*. Editorial Autores de Argentina.

Roshi, Y. (2020). Antenna impedance. Consultado el 2 de noviembre de 2025.

Saleh, B. E. A., T. M. C. (2019). *Fundamentals of Photonics*. Willey.

Side, M. T. (2021). Bandas de radiofrecuencia. Consultado el 1 de noviembre de 2025.

Tesswave (2024). What is antenna polarization and why it matters. Consultado el 2 de noviembre de 2025.

T&M Atlantic (2019). Electricity in ancient times. Recuperado en septiembre de 2025, de `https://www.tmatlantic.com/facts/index.php?ELEMENT_ID=6740`.

Valverde, R. L. (2001). *Historia del electromagnetismo*. Ediciones IES, Pablo Picasso.

Voices, E. and Voices, E. (2024). The electromagnetic spectrum: It's more than visible light. Recuperado en septiembre de 2025, de `https://earthsky.org/space/what-is-the-electromagnetic-spectrum/`.

Voors, A. (2021). Nec based antenna modeler and optimizer. Recuperado en abril de 2026, de `https://www.qsl.net/4nec2/`.

World, T. . M. (2023). Fnbw vs hpbw: Understanding the difference in antenna beamwidth. Consultado el 1 de noviembre de 2025.

Y, E. (2024). ¿qué es la mitad del ancho del haz de potencia? Consultado el 1 de noviembre de 2025.

Young, H. D. and Freedman, R. A. (2018). *Física universitaria con física moderna*. Pearson Educación, Estados Unidos, 13 edition.

Zvolensky, T. (2022). Antenna radiation efficiency. Consultado el 1 de noviembre de 2025.

7.. Anexo I.

7.1. Código principal.

```
import sys
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
from matplotlib.animation import PillowWriter
from scipy.special import jv
from PyQt5.QtWidgets import (QMainWindow, QLabel, QLineEdit, QPushButton,
                             QApplication, QVBoxLayout, QWidget, QHBoxLayout,
                             QFormLayout, QMessageBox, QGroupBox, QSpinBox,
                             QDoubleSpinBox, QCheckBox, QComboBox, QSplitter,
                             QStackedWidget, QGridLayout, QSlider, QProgressBar)
from PyQt5.QtCore import Qt, QThread, pyqtSignal
from PyQt5.QtGui import QFont, QPalette, QColor, QPixmap

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Sistema de Simulacin - Men Principal")
        self.setGeometry(100, 100, 900, 600)

        # Widget central
        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)

        # Layout principal
        self.main_layout = QVBoxLayout()
        self.central_widget.setLayout(self.main_layout)

        # Ttulo
        title = QLabel("SISTEMA DE SIMULACIN")
        title_font = QFont("Arial", 24, QFont.Bold)
        title.setFont(title_font)
        title.setAlignment(Qt.AlignCenter)
        title.setStyleSheet("color: #2c3e50; padding: 20px;")
        self.main_layout.addWidget(title)

        # Descripcin
        desc = QLabel("Seleccione una de las siguientes opciones de simulacin:")
        desc_font = QFont("Arial", 12)
        desc.setFont(desc_font)
        desc.setAlignment(Qt.AlignCenter)
        desc.setStyleSheet("color: #7f8c8d; padding: 10px;")
        self.main_layout.addWidget(desc)

        # Widget apilado para cambiar entre ventanas
        self.stacked_widget = QStackedWidget()
        self.main_layout.addWidget(self.stacked_widget)

        # Crear pginas
        self.create_menu_page()
        self.create_opcion1_page()
        self.create_opcion2_page()
        self.create_opcion3_page()
        self.create_opcion4_page()

        # Mostrar men principal primero
        self.stacked_widget.setCurrentIndex(0)

        # Estilos
        self.setStyleSheet("""
            QMainWindow {
                background-color: #ecf0f1;
            }
        """)

    def create_menu_page(self):
        """Crea la pgina del men principal con 4 opciones"""
        menu_page = QWidget()
        menu_layout = QVBoxLayout()
        menu_page.setLayout(menu_layout)

        # Grid para los botones
        button_grid = QGridLayout()
        button_grid.setSpacing(30)
        button_grid.setContentsMargins(50, 30, 50, 30)

        # Botn 1
        btn1 = QPushButton("1. Simulacin completa\ndel patrn en una Helicoide")
```

```
        btn1.setFont(QFont("Arial", 14, QFont.Bold))
        btn1.setMinimumSize(300, 100)
        btn1.setStyleSheet("""
            QPushButton {
                background-color: #3498db;
                color: white;
                border-radius: 10px;
                padding: 15px;
            }
            QPushButton:hover {
                background-color: #2980b9;
            }
        """)
        btn1.clicked.connect(lambda: self.show_opcion1())

        # Botn 2
        btn2 = QPushButton("2. Simulacin patrn\nde orden de transmisin 1")
        btn2.setFont(QFont("Arial", 14, QFont.Bold))
        btn2.setMinimumSize(300, 100)
        btn2.setStyleSheet("""
            QPushButton {
                background-color: #2ecc71;
                color: white;
                border-radius: 10px;
                padding: 15px;
            }
            QPushButton:hover {
                background-color: #27ae60;
            }
        """)
        btn2.clicked.connect(lambda: self.show_opcion2())

        # Botn 3
        btn3 = QPushButton("3. Clculo de diseo de\nla Helicoide con circuito RLC")
        btn3.setFont(QFont("Arial", 14, QFont.Bold))
        btn3.setMinimumSize(300, 100)
        btn3.setStyleSheet("""
            QPushButton {
                background-color: #e74c3c;
                color: white;
                border-radius: 10px;
                padding: 15px;
            }
            QPushButton:hover {
                background-color: #c0392b;
            }
        """)
        btn3.clicked.connect(lambda: self.show_opcion3())

        # Botn 4
        btn4 = QPushButton("4. Simulacin del patrn\nde orden 1 con circuito RLC")
        btn4.setFont(QFont("Arial", 14, QFont.Bold))
        btn4.setMinimumSize(300, 100)
        btn4.setStyleSheet("""
            QPushButton {
                background-color: #9b59b6;
                color: white;
                border-radius: 10px;
                padding: 15px;
            }
            QPushButton:hover {
                background-color: #8e44ad;
            }
        """)
        btn4.clicked.connect(lambda: self.show_opcion4())

        # Aadir botones al grid
        button_grid.addWidget(btn1, 0, 0)
        button_grid.addWidget(btn2, 0, 1)
        button_grid.addWidget(btn3, 1, 0)
        button_grid.addWidget(btn4, 1, 1)

        menu_layout.addLayout(button_grid)

        # Botn de salida
        exit_btn = QPushButton("Salir")
        exit_btn.setFont(QFont("Arial", 12))
        exit_btn.setFixedSize(150, 50)
        exit_btn.setStyleSheet("""
            QPushButton {
                background-color: #95a5a6;
                color: white;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #7f8c8d;
            }
        """)
```

```
        exit_btn.clicked.connect(self.close)
        menu_layout.addWidget(exit_btn, alignment=Qt.AlignCenter)

        # Aadir pgina al stacked widget
        self.stacked_widget.addWidget(menu_page)

    def create_opcion1_page(self):
        """Crea la pgina para la opcin 1 (v1.py)"""
        # Crear wrapper con botn de regreso
        opcion1_page = QWidget()
        opcion1_layout = QVBoxLayout()
        opcion1_page.setLayout(opcion1_layout)

        # Botn de regreso
        back_button = QPushButton(" Volver al Men Principal")
        back_button.setFont(QFont("Arial", 11))
        back_button.setStyleSheet("""
            QPushButton {
                background-color: #34495e;
                color: white;
                padding: 8px 15px;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #2c3e50;
            }
        """)
        back_button.clicked.connect(self.show_menu)
        opcion1_layout.addWidget(back_button)

        # Crear la simulacin de v1.py
        self.opcion1_widget = HelicoideTransmisionGUI()
        opcion1_layout.addWidget(self.opcion1_widget)

        self.stacked_widget.addWidget(opcion1_page)

    def create_opcion2_page(self):
        """Crea la pgina para la opcin 2 (v2.py)"""
        opcion2_page = QWidget()
        opcion2_layout = QVBoxLayout()
        opcion2_page.setLayout(opcion2_layout)

        # Botn de regreso
        back_button = QPushButton(" Volver al Men Principal")
        back_button.setFont(QFont("Arial", 11))
        back_button.setStyleSheet("""
            QPushButton {
                background-color: #34495e;
                color: white;
                padding: 8px 15px;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #2c3e50;
            }
        """)
        back_button.clicked.connect(self.show_menu)
        opcion2_layout.addWidget(back_button)

        # Crear la simulacin de v2.py
        self.opcion2_widget = HelicoideSimulatorGUI()
        opcion2_layout.addWidget(self.opcion2_widget)

        self.stacked_widget.addWidget(opcion2_page)

    def create_opcion3_page(self):
        """Crea la pgina para la opcin 3 (v3.py)"""
        opcion3_page = QWidget()
        opcion3_layout = QVBoxLayout()
        opcion3_page.setLayout(opcion3_layout)

        # Botn de regreso
        back_button = QPushButton(" Volver al Men Principal")
        back_button.setFont(QFont("Arial", 11))
        back_button.setStyleSheet("""
            QPushButton {
                background-color: #34495e;
                color: white;
                padding: 8px 15px;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #2c3e50;
            }
        """)
        back_button.clicked.connect(self.show_menu)
        opcion3_layout.addWidget(back_button)

```

```
258         # Crear la simulacin de v3.py
259         self.opcion3_widget = AntenaHelicoidalGUI()
260         opcion3_layout.addWidget(self.opcion3_widget)
261 
262         self.stacked_widget.addWidget(opcion3_page)
263 
264     def create_opcion4_page(self):
265         """Crea la pgina para la opcin 4 (v4.py)"""
266         opcion4_page = QWidget()
267         opcion4_layout = QVBoxLayout()
268         opcion4_page.setLayout(opcion4_layout)
269 
270         # Botn de regreso
271         back_button = QPushButton(" Volver al Men Principal")
272         back_button.setFont(QFont("Arial", 11))
273         back_button.setStyleSheet("""
274             QPushButton {
275                 background-color: #34495e;
276                 color: white;
277                 padding: 8px 15px;
278                 border-radius: 5px;
279             }
280             QPushButton:hover {
281                 background-color: #2c3e50;
282             }
283         """)
284         back_button.clicked.connect(self.show_menu)
285         opcion4_layout.addWidget(back_button)
286 
287         # Crear la simulacin de v4.py
288         self.opcion4_widget = AntenaHelicoideGUI()
289         opcion4_layout.addWidget(self.opcion4_widget)
290 
291         self.stacked_widget.addWidget(opcion4_page)
292 
293     def show_menu(self):
294         """Muestra el men principal"""
295         self.stacked_widget.setCurrentIndex(0)
296         self.resize(900, 600)
297 
298     def show_opcion1(self):
299         """Muestra la simulacin de la opcin 1"""
300         self.stacked_widget.setCurrentIndex(1)
301         self.resize(1400, 900)
302 
303     def show_opcion2(self):
304         """Muestra la simulacin de la opcin 2"""
305         self.stacked_widget.setCurrentIndex(2)
306         self.resize(1400, 900)
307 
308     def show_opcion3(self):
309         """Muestra la simulacin de la opcin 3"""
310         self.stacked_widget.setCurrentIndex(3)
311         self.resize(1600, 900)
312 
313     def show_opcion4(self):
314         """Muestra la simulacin de la opcin 4"""
315         self.stacked_widget.setCurrentIndex(4)
316         self.resize(1400, 900)
317 
318 class AnimationThreadV1(QThread):
319     """Hilo para generar la animacin en segundo plano"""
320     finished = pyqtSignal(str)
321     error = pyqtSignal(str)
322     progress = pyqtSignal(int)
323 
324     def __init__(self, params):
325         super().__init__()
326         self.params = params
327 
328     def run(self):
329         try:
330             # Extraer parmetros
331             alpha = self.params['alpha']
332             C = self.params['C']
333             factor = self.params['factor']
334             vueltas = self.params['vueltas']
335             m = self.params['m']
336             n = self.params['n']
337             fps = self.params['fps']
338 
339             # Clculos iniciales
340             alpha_rad = alpha * np.pi / 180
341             R = C / (2 * np.pi)
342             S = C * np.tan(alpha_rad)
343             L = np.sqrt((C**2) + (S**2))
344             hmax = vueltas * S
345             omega = 10000
```

```
            # Parmetros normalizados
            longitud_onda = C / m
            f = 3e8 / longitud_onda / 1e6  # frecuencia en megahertz
            C_lambda = C / longitud_onda
            S_lambda = S / longitud_onda
            L_lambda = L / longitud_onda

            # Funcin para calcular p
            def velocidad_p(m, C_lambda, alpha_rad, hmax, R, longitud_onda):
                hmax_R_producto = hmax * R

                # Verificar condicin para evitar nmeros complejos
                if (hmax_R_producto / longitud_onda**2)**2 >= C_lambda**2:
                    # Ajustar para evitar nmeros complejos
                    kR = 0.01  # Valor pequeo
                else:
                    kR = np.sqrt(C_lambda**2 - (hmax_R_producto / longitud_onda**2)**2)

                def hr(kR_val):
                    if kR_val <= 0:
                        return 0

                    # Usar jv (minscula) - funcin de Bessel
                    Jm = jv(m, kR_val)
                    Jmantes = jv(m-1, kR_val) if m-1 >= 0 else 0
                    Jmdespues = jv(m+1, kR_val)

                    # Evitar divisin por cero
                    if Jmantes == 0 or Jmdespues == 0:
                        return 0

                    return np.tan(alpha_rad) * (m * Jm**2) / (Jmantes * Jmdespues)

                hr_val = hr(kR)

                # Calcular p
                denominador = m * np.cos(alpha_rad) + hr_val * np.sin(alpha_rad)
                if abs(denominador) < 1e-10:
                    p = 1.0
                else:
                    p = C_lambda / denominador
                    # Limitar p entre 0 y 1 para estabilidad
                    p = max(0.01, min(0.99, p))

                return p, kR, hr_val

            # Calcular p
            p, kR, hr_val = velocidad_p(m, C_lambda, alpha_rad, hmax, R, longitud_onda)

            # Trayectoria
            theta = np.linspace(0, vueltas * 2 * np.pi, n)
            z = (S / (2 * np.pi)) * theta
            x = R * np.cos(theta)
            y = R * np.sin(theta)

            # Superficie de la helicoide
            theta_surf = np.linspace(0, vueltas * 2 * np.pi, 50)
            r_surf = np.linspace(0, R, 50)
            theta_surf, r_surf = np.meshgrid(theta_surf, r_surf)
            x_surf = r_surf * np.cos(theta_surf)
            y_surf = r_surf * np.sin(theta_surf)
            z_surf = (hmax / (vueltas * 2 * np.pi)) * theta_surf

            def power_pattern(altura_z, progress=1):
                theta_p = np.linspace(0, 2 * np.pi, 50)
                phi_p = np.linspace(0, np.pi, 50)
                theta_p, phi_p = np.meshgrid(theta_p, phi_p)

                # Usar p calculado
                p_safe = max(0.01, min(0.99, p))
                psi = 2 * np.pi * (S_lambda * np.cos(phi_p) - L_lambda / p_safe)

                E = (np.sin(np.pi/(2 * vueltas)) *
                     (np.sin(vueltas * psi/2) / np.sin(psi/2 + 1e-9)) *
                     np.cos(theta_p))

                E = np.abs(E) / np.max(np.abs(E))

                scale = 0.1 + 0.9 * progress
                Xp = factor * scale * 0.5 * E * np.sin(phi_p) * np.cos(theta_p)
                Yp = factor * scale * 0.5 * E * np.sin(phi_p) * np.sin(theta_p)
                Zp = factor * 0.5 * E * np.cos(phi_p) + altura_z

                return E, Xp, Yp, Zp

            # Configuracin de la figura
            fig = plt.figure(figsize=(12, 8))
```

```
            ax = fig.add_subplot(111, projection='3d')

            max_range = max(R, hmax/2)
            ax.set_xlim([-max_range*1, max_range*1])
            ax.set_ylim([-max_range*1, max_range*1])
            ax.set_zlim([0, hmax*1.2])

            ax.set_xlabel("X", fontsize=12)
            ax.set_ylabel("Y", fontsize=12)
            ax.set_zlabel("Z", fontsize=12)
            ax.set_title(f"Patrn de Radiacin en Helicoide\np = {p:.4f}, m = {m}", fontsize=14)

            ax.tick_params(axis='both', which='major', labelsize=12, width=2, length=6)
            ax.locator_params(axis='x', nbins=5)
            ax.locator_params(axis='y', nbins=5)
            ax.locator_params(axis='z', nbins=5)

            # Dibujar trayectoria
            filamento_gris, = ax.plot(x, y, z, color='black', linewidth=4, alpha=0.5)
            filamento_rojo, = ax.plot([], [], [], color='red', linewidth=6)

            # Inicializar partcula
            particle = ax.scatter([], [], [], c='red', s=50, zorder=10)
            U_init, Xp_init, Yp_init, Zp_init = power_pattern(z[0], progress=0)
            radiation_pattern = ax.plot_surface(
                Xp_init, Yp_init, Zp_init,
                facecolors=plt.cm.plasma(U_init),
                alpha=0.7,
                rstride=1, cstride=1,
                zorder=5
            )

            # Aadir informacin del patrn
            info_text = (f'm = {m}\n = {alpha}\nC = {C} m\nR = {R:.2f} m\n'
                         f'hmax = {hmax:.1f} m\np = {p:.4f}\n = {longitud_onda:.2f} m\n'
                         f'f = {f:.2f} MHz\nC/ = {C_lambda:.4f}\nvueltas = {vueltas}')
            ax.text2D(0.02, 0.98, info_text, transform=ax.transAxes,
                      fontsize=9, verticalalignment='top',
                      bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.8))

            def init():
                particle._offsets3d = ([], [], [])
                return particle, radiation_pattern

            def update(i):
                nonlocal radiation_pattern

                particle._offsets3d = ([x[i]], [y[i]], [z[i]])
                progress = i / (n - 1)

                U, Xp, Yp, Zp = power_pattern(z[i], progress)

                radiation_pattern.remove()
                radiation_pattern = ax.plot_surface(
                    Xp, Yp, Zp,
                    facecolors=plt.cm.plasma(U),
                    alpha=0.7,
                    rstride=1, cstride=1,
                    zorder=5
                )

                filamento_rojo.set_data(x[:i+1], y[:i+1])
                filamento_rojo.set_3d_properties(z[:i+1])

                # Emitir progreso
                self.progress.emit(int(100 * i / n))

                return particle, radiation_pattern, filamento_rojo

            # Crear animacin
            ani = animation.FuncAnimation(
                fig, update, frames=n, init_func=init,
                interval=50, blit=False, repeat=True
            )

            # Guardar animacin
            save_path = "patron_modo_transmision.gif"
            ani.save(save_path, writer=PillowWriter(fps=fps))
            plt.close(fig)

            self.finished.emit(save_path)

        except Exception as e:
            self.error.emit(str(e))

class HelicoideTransmisionGUI(QMainWindow):
    def __init__(self):
        super().__init__()
```

```
        self.setWindowTitle("Simulador de Antena Helicoide - Modo de Transmisin")
        self.setGeometry(50, 50, 1400, 900)

        # Widget central
        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)

        # Layout principal horizontal
        self.main_layout = QHBoxLayout()
        self.central_widget.setLayout(self.main_layout)

        # Frame izquierdo para controles
        self.left_frame = QWidget()
        self.left_layout = QVBoxLayout()
        self.left_frame.setLayout(self.left_layout)
        self.left_frame.setMaximumWidth(450)

        # Frame derecho para grficos
        self.right_frame = QWidget()
        self.right_layout = QVBoxLayout()
        self.right_frame.setLayout(self.right_layout)

        # Panel de control
        self.setup_control_panel()

        # rea de grficos
        self.setup_plots()

        # Aadir frames al layout principal
        self.main_layout.addWidget(self.left_frame, 1)
        self.main_layout.addWidget(self.right_frame, 3)

        # Hilo de animacin
        self.animation_thread = None

        # Barra de progreso
        self.progress_bar = QProgressBar()
        self.progress_bar.setVisible(False)
        self.right_layout.addWidget(self.progress_bar)

    def setup_control_panel(self):
        """Configura el panel de control con parmetros"""

        # Grupo de parmetros principales
        main_group = QGroupBox("Parmetros Principales")
        main_group.setStyleSheet("QGroupBox { font-weight: bold; }")
        main_layout = QFormLayout()
        main_layout.setSpacing(10)

        # Factor (0.1-10) - deslizador
        self.factor_label = QLabel("Factor: 1.0")
        self.factor_label.setAlignment(Qt.AlignCenter)

        self.factor_slider = QSlider(Qt.Horizontal)
        self.factor_slider.setRange(1, 100)  # 0.1 a 10 (multiplicado por 10)
        self.factor_slider.setValue(10)  # Valor inicial 1.0
        self.factor_slider.setTickPosition(QSlider.TicksBelow)
        self.factor_slider.setTickInterval(10)
        self.factor_slider.valueChanged.connect(self.update_factor_label)

        # ngulo alpha (1-89) - deslizador
        self.alpha_label = QLabel("ngulo : 14.0")
        self.alpha_label.setAlignment(Qt.AlignCenter)

        self.alpha_slider = QSlider(Qt.Horizontal)
        self.alpha_slider.setRange(10, 890)  # 1.0 a 89.0 (multiplicado por 10)
        self.alpha_slider.setValue(140)  # Valor inicial 14.0
        self.alpha_slider.setTickPosition(QSlider.TicksBelow)
        self.alpha_slider.setTickInterval(50)
        self.alpha_slider.valueChanged.connect(self.update_alpha_label)

        # Circunferencia C (metros)
        self.C_input = QDoubleSpinBox()
        self.C_input.setRange(0.01, 100)
        self.C_input.setValue(0.5)
        self.C_input.setDecimals(2)
        self.C_input.setSingleStep(0.1)
        self.C_input.setMaximumWidth(120)

        # Nmero de vueltas (>0)
        self.vueltas_input = QSpinBox()
        self.vueltas_input.setRange(1, 50)  # Mayor a 0
        self.vueltas_input.setValue(7)
        self.vueltas_input.setSingleStep(1)
        self.vueltas_input.setMaximumWidth(120)

        # Orden de transmisin m (0-15)
        self.m_input = QSpinBox()
```

```
610         self.m_input.setRange(0, 15)
611         self.m_input.setValue(1)
612         self.m_input.setSingleStep(1)
613         self.m_input.setMaximumWidth(120)
614 
615         # Aadir widgets al layout
616         factor_widget = QWidget()
617         factor_layout = QVBoxLayout()
618         factor_layout.addWidget(self.factor_label)
619         factor_layout.addWidget(self.factor_slider)
620         factor_widget.setLayout(factor_layout)
621 
622         alpha_widget = QWidget()
623         alpha_layout = QVBoxLayout()
624         alpha_layout.addWidget(self.alpha_label)
625         alpha_layout.addWidget(self.alpha_slider)
626         alpha_widget.setLayout(alpha_layout)
627 
628         main_layout.addRow("Factor (0.1-10):", factor_widget)
629         main_layout.addRow("ngulo  (1-89):", alpha_widget)
630         main_layout.addRow("Circunferencia C (m):", self.C_input)
631         main_layout.addRow("Vueltas (>0):", self.vueltas_input)
632         main_layout.addRow("Orden m (0-15):", self.m_input)
633         main_group.setLayout(main_layout)
634 
635         # Grupo de parmetros de animacin
636         anim_group = QGroupBox("Parmetros de Animacin")
637         anim_group.setStyleSheet("QGroupBox { font-weight: bold; }")
638         anim_layout = QFormLayout()
639         anim_layout.setSpacing(10)
640 
641         self.n_input = QSpinBox()
642         self.n_input.setRange(100, 2000)
643         self.n_input.setValue(500)
644         self.n_input.setSingleStep(100)
645         self.n_input.setMaximumWidth(120)
646 
647         self.fps_input = QSpinBox()
648         self.fps_input.setRange(1, 60)
649         self.fps_input.setValue(20)
650         self.fps_input.setMaximumWidth(120)
651 
652         anim_layout.addRow("Puntos (n):", self.n_input)
653         anim_layout.addRow("FPS:", self.fps_input)
654         anim_group.setLayout(anim_layout)
655 
656         # Frame para botones
657         button_frame = QWidget()
658         button_layout = QVBoxLayout()
659         button_layout.setSpacing(10)
660         button_frame.setLayout(button_layout)
661 
662         # Botones
663         self.simulate_btn = QPushButton("Ejecutar Simulacin")
664         self.simulate_btn.setStyleSheet("""
665             QPushButton {
666                 background-color: #4CAF50;
667                 color: white;
668                 font-weight: bold;
669                 padding: 12px;
670                 font-size: 12pt;
671                 border-radius: 5px;
672             }
673             QPushButton:hover {
674                 background-color: #45a049;
675             }
676         """)
677         self.simulate_btn.clicked.connect(self.run_simulation)
678 
679         self.animate_btn = QPushButton("Generar Animacin")
680         self.animate_btn.setStyleSheet("""
681             QPushButton {
682                 background-color: #2196F3;
683                 color: white;
684                 font-weight: bold;
685                 padding: 12px;
686                 font-size: 12pt;
687                 border-radius: 5px;
688             }
689             QPushButton:hover {
690                 background-color: #0b7dda;
691             }
692         """)
693         self.animate_btn.clicked.connect(self.generate_animation)
694 
695         self.clear_btn = QPushButton("Limpiar Grficos")
696         self.clear_btn.setStyleSheet("""
697             QPushButton {
```

```
                background-color: #f44336;
                color: white;
                font-weight: bold;
                padding: 12px;
                font-size: 12pt;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #d32f2f;
            }
        """)
        self.clear_btn.clicked.connect(self.clear_plot)

        # Botn para mostrar clculos
        self.calc_btn = QPushButton("Mostrar Clculos")
        self.calc_btn.setStyleSheet("""
            QPushButton {
                background-color: #9C27B0;
                color: white;
                font-weight: bold;
                padding: 12px;
                font-size: 12pt;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #7B1FA2;
            }
        """)
        self.calc_btn.clicked.connect(self.show_calculations)

        # Aadir botones
        button_layout.addWidget(self.simulate_btn)
        button_layout.addWidget(self.animate_btn)
        button_layout.addWidget(self.calc_btn)
        button_layout.addWidget(self.clear_btn)

        # Aadir todos los grupos al layout izquierdo
        self.left_layout.addWidget(main_group)
        self.left_layout.addWidget(anim_group)
        self.left_layout.addWidget(button_frame)
        self.left_layout.addStretch()

    def setup_plots(self):
        """Configura el rea de grficos"""
        self.figure = Figure(figsize=(10, 8), dpi=100)
        self.canvas = FigureCanvas(self.figure)
        self.ax = self.figure.add_subplot(111, projection='3d')

        # Configurar ejes
        self.ax.set_xlabel("X", fontsize=14, fontweight='bold')
        self.ax.set_ylabel("Y", fontsize=14, fontweight='bold')
        self.ax.set_zlabel("Z", fontsize=14, fontweight='bold')
        self.ax.set_title("Simulador de Antena Helicoide - Modo de Transmisin",
                          fontsize=16, fontweight='bold', pad=20)

        # Configurar ticks
        self.ax.tick_params(axis='both', which='major', labelsize=11)

        # Aadir canvas al layout derecho
        self.right_layout.addWidget(self.canvas)

    def update_factor_label(self):
        """Actualiza la etiqueta del factor"""
        value = self.factor_slider.value() / 10.0
        self.factor_label.setText(f"Factor: {value:.1f}")

    def update_alpha_label(self):
        """Actualiza la etiqueta del ngulo alpha"""
        value = self.alpha_slider.value() / 10.0
        self.alpha_label.setText(f"ngulo : {value:.1f}")

    def get_simulation_parameters(self):
        """Obtiene los parmetros de simulacin"""
        params = {
            'alpha': self.alpha_slider.value() / 10.0,  # Convertir de 1.0-89.0
            'C': self.C_input.value(),
            'factor': self.factor_slider.value() / 10.0,  # Convertir de 0.1-10.0
            'vueltas': self.vueltas_input.value(),
            'm': self.m_input.value(),
            'n': self.n_input.value(),
            'fps': self.fps_input.value()
        }
        return params

    def calculate_parameters(self, params=None):
        """Calcula los parmetros derivados"""
        if params is None:
            params = self.get_simulation_parameters()
```

```
786 |
787 |         alpha = params['alpha']
788 |         C = params['C']
789 |         vueltas = params['vueltas']
790 |         m = params['m']
791 |
792 |         # Clculos bsicos
793 |         alpha_rad = alpha * np.pi / 180
794 |         R = C / (2 * np.pi)
795 |         S = C * np.tan(alpha_rad)
796 |         L = np.sqrt((C**2) + (S**2))
797 |         hmax = vueltas * S
798 |
799 |         # Parmetros normalizados
800 |         longitud_onda = C / max(1, m)  # Evitar divisin por cero
801 |         f = 3e8 / longitud_onda / 1e6  # frecuencia en megahertz
802 |         C_lambda = C / longitud_onda
803 |         S_lambda = S / longitud_onda
804 |         L_lambda = L / longitud_onda
805 |
806 |         # Funcin para calcular p
807 |         def velocidad_p(m, C_lambda, alpha_rad, hmax, R, longitud_onda):
808 |             hmax_R_producto = hmax * R
809 |
810 |             # Verificar condicin para evitar nmeros complejos
811 |             if (hmax_R_producto / longitud_onda**2)**2 >= C_lambda**2:
812 |                 kR = 0.01  # Valor pequeo por defecto
813 |             else:
814 |                 kR = np.sqrt(C_lambda**2 - (hmax_R_producto / longitud_onda**2)**2)
815 |
816 |             def hr(kR_val):
817 |                 if kR_val <= 0:
818 |                     return 0
819 |
820 |                 # Usar jv (minscula) - funcin de Bessel
821 |                 Jm = jv(m, kR_val)
822 |                 Jmantes = jv(m-1, kR_val) if m-1 >= 0 else 0
823 |                 Jmdespues = jv(m+1, kR_val)
824 |
825 |                 # Evitar divisin por cero
826 |                 if Jmantes == 0 or Jmdespues == 0:
827 |                     return 0
828 |
829 |                 return np.tan(alpha_rad) * (m * Jm**2) / (Jmantes * Jmdespues)
830 |
831 |             hr_val = hr(kR)
832 |
833 |             # Calcular p
834 |             denominador = m * np.cos(alpha_rad) + hr_val * np.sin(alpha_rad)
835 |             if abs(denominador) < 1e-10:
836 |                 p = 1.0
837 |             else:
838 |                 p = C_lambda / denominador
839 |                 # Limitar p entre 0 y 1 para estabilidad
840 |                 p = max(0.01, min(0.99, p))
841 |
842 |             return p, kR, hr_val
843 |
844 |         # Calcular p
845 |         p, kR, hr_val = velocidad_p(m, C_lambda, alpha_rad, hmax, R, longitud_onda)
846 |
847 |         return {
848 |             'R': R, 'S': S, 'L': L, 'hmax': hmax,
849 |             'longitud_onda': longitud_onda, 'f': f,
850 |             'C_lambda': C_lambda, 'S_lambda': S_lambda, 'L_lambda': L_lambda,
851 |             'alpha_rad': alpha_rad, 'p': p, 'kR': kR, 'hr_val': hr_val
852 |         }
853 |
854 |     def power_pattern(self, altura_z, progress=1, params=None):
855 |         """Calcula el patrn de radiacin"""
856 |         if params is None:
857 |             params = self.get_simulation_parameters()
858 |
859 |         # Calcular parmetros derivados
860 |         calc_params = self.calculate_parameters(params)
861 |
862 |         factor = params['factor']
863 |         vueltas = params['vueltas']
864 |         S_lambda = calc_params['S_lambda']
865 |         L_lambda = calc_params['L_lambda']
866 |         p = calc_params['p']
867 |
868 |         # Calcular patrn
869 |         theta_p = np.linspace(0, 2 * np.pi, 50)
870 |         phi_p = np.linspace(0, np.pi, 50)
871 |         theta_p, phi_p = np.meshgrid(theta_p, phi_p)
872 |
873 |         # Usar p calculado con proteccin
```

```
        p_safe = max(0.01, min(0.99, p))
        psi = 2 * np.pi * (S_lambda * np.cos(phi_p) - L_lambda / p_safe)

        E = (np.sin(np.pi/(2 * vueltas)) *
             (np.sin(vueltas * psi/2) / np.sin(psi/2 + 1e-9)) *
             np.cos(theta_p))

        E = np.abs(E) / np.max(np.abs(E))

        scale = 0.1 + 0.9 * progress
        Xp = factor * scale * 0.5 * E * np.sin(phi_p) * np.cos(theta_p)
        Yp = factor * scale * 0.5 * E * np.sin(phi_p) * np.sin(theta_p)
        Zp = factor * 0.5 * E * np.cos(phi_p) + altura_z

        return E, Xp, Yp, Zp

    def run_simulation(self):
        """Ejecuta la simulacin esttica"""
        try:
            params = self.get_simulation_parameters()
            calc_params = self.calculate_parameters(params)

            # Extraer parmetros
            alpha = params['alpha']
            C = params['C']
            factor = params['factor']
            vueltas = params['vueltas']
            m = params['m']
            n = params['n']

            R = calc_params['R']
            S = calc_params['S']
            hmax = calc_params['hmax']
            longitud_onda = calc_params['longitud_onda']
            f = calc_params['f']
            C_lambda = calc_params['C_lambda']
            p = calc_params['p']

            # Trayectoria
            theta = np.linspace(0, vueltas * 2 * np.pi, n)
            z = (S / (2 * np.pi)) * theta
            x = R * np.cos(theta)
            y = R * np.sin(theta)

            # Superficie de la helicoide
            theta_surf = np.linspace(0, vueltas * 2 * np.pi, 50)
            r_surf = np.linspace(0, R, 50)
            theta_surf, r_surf = np.meshgrid(theta_surf, r_surf)
            x_surf = r_surf * np.cos(theta_surf)
            y_surf = r_surf * np.sin(theta_surf)
            z_surf = (hmax / (vueltas * 2 * np.pi)) * theta_surf

            self.clear_plot()

            # Dibujar trayectoria completa
            self.ax.plot(x, y, z, color='black', linewidth=3, alpha=0.7, label='Trayectoria')

            # Dibujar punto inicial y final
            self.ax.scatter(x[0], y[0], z[0], c='green', s=100, edgecolors='black', linewidth=2, label='Inicio')
            self.ax.scatter(x[-1], y[-1], z[-1], c='red', s=100, edgecolors='black', linewidth=2, label='Fin')

            # Calcular y dibujar patrn de radiacin en posicin media
            mid_idx = n // 2
            U, Xp, Yp, Zp = self.power_pattern(z[mid_idx], 0.5, params)
            self.ax.plot_surface(
                Xp, Yp, Zp,
                facecolors=plt.cm.plasma(U),
                alpha=0.6,
                rstride=1, cstride=1
            )

            # Configurar lmites
            max_range = max(R * 1.2, hmax/2)
            self.ax.set_xlim(-max_range, max_range)
            self.ax.set_ylim(-max_range, max_range)
            self.ax.set_zlim(0, hmax * 1.1)

            # Aadir informacin del patrn
            info_text = (f'm = {m}\n = {alpha}\nC = {C} m\nR = {R:.2f} m\n'
                         f'hmax = {hmax:.1f} m\np = {p:.4f}\n = {longitud_onda:.2f} m\n'
                         f'f = {f:.2f} MHz\nC/ = {C_lambda:.4f}\nvueltas = {vueltas}')
            self.ax.text2D(0.02, 0.98, info_text, transform=self.ax.transAxes,
                           fontsize=9, verticalalignment='top',
                           bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.8))

            self.ax.set_title(f"Simulacin Esttica - Modo m={m}\n"
                              f"={alpha}, C={C}m, p={p:.4f}, Vueltas={vueltas}",
                              fontsize=14, fontweight='bold')
```

```
962
963             # Aadir leyenda
964             self.ax.legend(loc='upper right', fontsize=10)
965
966             self.canvas.draw()
967
968         except Exception as e:
969             self.show_error_message(f"Error en simulacin: {str(e)}")
970
971     def show_calculations(self):
972         """Muestra los clculos detallados"""
973         try:
974             params = self.get_simulation_parameters()
975             calc_params = self.calculate_parameters(params)
976
977             alpha = params['alpha']
978             C = params['C']
979             vueltas = params['vueltas']
980             m = params['m']
981
982             R = calc_params['R']
983             S = calc_params['S']
984             hmax = calc_params['hmax']
985             longitud_onda = calc_params['longitud_onda']
986             f = calc_params['f']
987             C_lambda = calc_params['C_lambda']
988             S_lambda = calc_params['S_lambda']
989             L_lambda = calc_params['L_lambda']
990             p = calc_params['p']
991             kR = calc_params['kR']
992             hr_val = calc_params['hr_val']
993
994             # Mostrar resultados
995             result_text = f"""
996             CLCULOS DETALLADOS:
997
998             Parmetros de entrada:
999              = {alpha:.1f}
1000            C = {C} m
1001            Factor = {params['factor']:.1f}
1002            Vueltas = {vueltas}
1003            Orden m = {m}
1004
1005            Geometra calculada:
1006            Radio R = C/(2) = {R:.4f} m
1007            Paso S = Ctan() = {S:.4f} m
1008            Altura total hmax = {vueltas}{S:.4f} = {hmax:.4f} m
1009            Longitud de hlice L = (C + S) = {calc_params['L']:.4f} m
1010
1011            Parmetros de frecuencia:
1012            Longitud de onda  = C/m = {longitud_onda:.4f} m
1013            Frecuencia f = c/ = {f:.2f} MHz
1014            C/ = {C_lambda:.4f}
1015            S/ = {S_lambda:.4f}
1016            L/ = {L_lambda:.4f}
1017
1018            Parmetros de funcin de Bessel:
1019            kR = {kR:.6f}
1020            hr(kR) = {hr_val:.6f}
1021
1022            Velocidad de fase:
1023            p = C/ / (mcos() + hrsin()) = {p:.6f}
1024            """
1025
1026            QMessageBox.information(self, "Clculos Detallados", result_text)
1027
1028        except Exception as e:
1029            self.show_error_message(f"Error en clculos: {str(e)}")
1030
1031    def generate_animation(self):
1032        """Genera la animacin en un hilo separado"""
1033        try:
1034            params = self.get_simulation_parameters()
1035
1036            # Deshabilitar botones durante la animacin
1037            self.animate_btn.setEnabled(False)
1038            self.simulate_btn.setEnabled(False)
1039            self.calc_btn.setEnabled(False)
1040            self.animate_btn.setText("Generando...")
1041
1042            # Mostrar barra de progreso
1043            self.progress_bar.setVisible(True)
1044            self.progress_bar.setValue(0)
1045
1046            # Crear y ejecutar hilo de animacin
1047            self.animation_thread = AnimationThreadV1(params)
1048            self.animation_thread.finished.connect(self.animation_finished)
1049            self.animation_thread.error.connect(self.animation_error)
```

```
            self.animation_thread.progress.connect(self.update_progress)
            self.animation_thread.start()

        except Exception as e:
            self.show_error_message(f"Error al iniciar animacin: {str(e)}")

    def update_progress(self, value):
        """Actualiza la barra de progreso"""
        self.progress_bar.setValue(value)

    def animation_finished(self, save_path):
        """Se llama cuando la animacin termina exitosamente"""
        self.animate_btn.setEnabled(True)
        self.simulate_btn.setEnabled(True)
        self.calc_btn.setEnabled(True)
        self.animate_btn.setText("Generar Animacin")
        self.progress_bar.setVisible(False)

        QMessageBox.information(self, "Animacin Completada",
                                f"Animacin guardada como:\n{save_path}")

    def animation_error(self, error_msg):
        """Se llama cuando hay un error en la animacin"""
        self.animate_btn.setEnabled(True)
        self.simulate_btn.setEnabled(True)
        self.calc_btn.setEnabled(True)
        self.animate_btn.setText("Generar Animacin")
        self.progress_bar.setVisible(False)

        self.show_error_message(f"Error en animacin: {error_msg}")

    def clear_plot(self):
        """Limpia el grfico"""
        self.ax.clear()
        self.ax.set_xlabel("X", fontsize=14, fontweight='bold')
        self.ax.set_ylabel("Y", fontsize=14, fontweight='bold')
        self.ax.set_zlabel("Z", fontsize=14, fontweight='bold')
        self.ax.set_title("Simulador de Antena Helicoide - Modo de Transmisin",
                          fontsize=16, fontweight='bold', pad=20)
        self.ax.tick_params(axis='both', which='major', labelsize=11)
        self.canvas.draw()

    def show_error_message(self, message):
        """Muestra un mensaje de error"""
        QMessageBox.critical(self, "Error", message)

class AnimationThreadV2(QThread):
    """Hilo para generar la animacin en segundo plano"""
    finished = pyqtSignal(str)
    error = pyqtSignal(str)
    progress = pyqtSignal(int)

    def __init__(self, params):
        super().__init__()
        self.params = params

    def run(self):
        try:
            # Extraer parmetros
            alpha = self.params['alpha']
            C = self.params['C']
            factor = self.params['factor']
            vueltas = self.params['vueltas']
            C_lambda = self.params['C_lambda']
            n = self.params['n']
            fps = self.params['fps']

            # Clculos iniciales
            alpha_rad = alpha * np.pi / 180
            R = C / (2 * np.pi)
            D = 2 * R
            S = C * np.tan(alpha_rad)
            L = np.sqrt((C**2) + (S**2))
            hmax = vueltas * S
            omega = 10000

            # Parmetros normalizados
            longitud_onda = C / C_lambda
            f = 3e8 / longitud_onda / 1e6  # frecuencia en megahertz
            S_lambda = S / longitud_onda
            L_lambda = L / longitud_onda

            # Trayectoria
            theta = np.linspace(0, vueltas * 2 * np.pi, n)
            z = (S / (2 * np.pi)) * theta
            x = R * np.cos(theta)
            y = R * np.sin(theta)
```

```
            # Superficie de la helicoide
            theta_surf = np.linspace(0, vueltas * 2 * np.pi, 50)
            r_surf = np.linspace(0, R, 50)
            theta_surf, r_surf = np.meshgrid(theta_surf, r_surf)
            x_surf = r_surf * np.cos(theta_surf)
            y_surf = r_surf * np.sin(theta_surf)
            z_surf = (hmax / (vueltas * 2 * np.pi)) * theta_surf

            def power_pattern(altura_z, progress=1):
                theta_p = np.linspace(0, 2 * np.pi, 50)
                phi_p = np.linspace(0, np.pi, 50)
                theta_p, phi_p = np.meshgrid(theta_p, phi_p)

                psi = 2 * np.pi * (S_lambda * (1 - np.cos(phi_p)) + (1/(2*vueltas)))

                E = (np.sin(np.pi/(2 * vueltas)) *
                     (np.sin(vueltas * psi/2) / np.sin(psi/2 + 1e-9)) *
                     np.cos(theta_p))

                E = np.abs(E) / np.max(np.abs(E))

                scale = 0.1 + 0.9 * progress
                Xp = factor * scale * 0.5 * E * np.sin(phi_p) * np.cos(theta_p)
                Yp = factor * scale * 0.5 * E * np.sin(phi_p) * np.sin(theta_p)
                Zp = factor * 0.5 * E * np.cos(phi_p) + altura_z

                return E, Xp, Yp, Zp

            # Configuracin de la figura
            fig = plt.figure(figsize=(12, 8))
            ax = fig.add_subplot(111, projection='3d')

            max_range = max(R, hmax/2)
            ax.set_xlim([-max_range*1, max_range*1])
            ax.set_ylim([-max_range*1, max_range*1])
            ax.set_zlim([0, hmax*1.2])

            ax.set_xlabel("X", fontsize=12)
            ax.set_ylabel("Y", fontsize=12)
            ax.set_zlabel("Z", fontsize=12)
            ax.set_title(f"Partcula y Patrn de Radiacin en la Helicoide", fontsize=14)

            ax.tick_params(axis='both', which='major', labelsize=12, width=2, length=6)
            ax.locator_params(axis='x', nbins=5)
            ax.locator_params(axis='y', nbins=5)
            ax.locator_params(axis='z', nbins=5)

            # Dibujar trayectoria
            filamento_gris, = ax.plot(x, y, z, color='black', linewidth=4, alpha=0.5)
            filamento_rojo, = ax.plot([], [], [], color='red', linewidth=6)

            # Inicializar partcula
            particle = ax.scatter([], [], [], c='red', s=50, zorder=10)
            U_init, Xp_init, Yp_init, Zp_init = power_pattern(z[0], progress=0)
            radiation_pattern = ax.plot_surface(
                Xp_init, Yp_init, Zp_init,
                facecolors=plt.cm.plasma(U_init),
                alpha=0.7,
                rstride=1, cstride=1,
                zorder=5
            )

            # Aadir informacin del patrn
            info_text = (f' = {alpha}\nC = {C} m\nR = {R:.2f} m\nhmax = {hmax:.1f}m\n'
                         f' = {longitud_onda:.2f} m\nf = {f:.2f} MHz\n'
                         f'C/ = {C_lambda:.4f}\nvueltas = {vueltas}')
            ax.text2D(0.02, 0.98, info_text, transform=ax.transAxes,
                      fontsize=10, verticalalignment='top',
                      bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.8))

            def init():
                particle._offsets3d = ([], [], [])
                return particle, radiation_pattern

            def update(i):
                nonlocal radiation_pattern

                particle._offsets3d = ([x[i]], [y[i]], [z[i]])
                progress = i / (n - 1)

                U, Xp, Yp, Zp = power_pattern(z[i], progress)

                radiation_pattern.remove()
                radiation_pattern = ax.plot_surface(
                    Xp, Yp, Zp,
                    facecolors=plt.cm.plasma(U),
                    alpha=0.7,
                    rstride=1, cstride=1,
```

```
                    zorder=5
                )

                filamento_rojo.set_data(x[:i+1], y[:i+1])
                filamento_rojo.set_3d_properties(z[:i+1])

                # Emitir progreso
                self.progress.emit(int(100 * i / n))

                return particle, radiation_pattern, filamento_rojo

            # Crear animacin
            ani = animation.FuncAnimation(
                fig, update, frames=n, init_func=init,
                interval=50, blit=False, repeat=True
            )

            # Guardar animacin
            save_path = "patron_modotrans1.gif"
            ani.save(save_path, writer=PillowWriter(fps=fps))
            plt.close(fig)

            self.finished.emit(save_path)

        except Exception as e:
            self.error.emit(str(e))

class HelicoideSimulatorGUI(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Simulador de Antena Helicoide - Modo Transversal")
        self.setGeometry(50, 50, 1400, 900)

        # Widget central
        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)

        # Layout principal horizontal
        self.main_layout = QHBoxLayout()
        self.central_widget.setLayout(self.main_layout)

        # Frame izquierdo para controles
        self.left_frame = QWidget()
        self.left_layout = QVBoxLayout()
        self.left_frame.setLayout(self.left_layout)
        self.left_frame.setMaximumWidth(450)

        # Frame derecho para grficos
        self.right_frame = QWidget()
        self.right_layout = QVBoxLayout()
        self.right_frame.setLayout(self.right_layout)

        # Panel de control
        self.setup_control_panel()

        # rea de grficos
        self.setup_plots()

        # Aadir frames al layout principal
        self.main_layout.addWidget(self.left_frame, 1)
        self.main_layout.addWidget(self.right_frame, 3)

        # Hilo de animacin
        self.animation_thread = None

        # Barra de progreso
        self.progress_bar = QProgressBar()
        self.progress_bar.setVisible(False)
        self.right_layout.addWidget(self.progress_bar)

    def setup_control_panel(self):
        """Configura el panel de control con parmetros"""

        # Grupo de parmetros geomtricos
        geom_group = QGroupBox("Parmetros Geomtricos")
        geom_group.setStyleSheet("QGroupBox { font-weight: bold; }")
        geom_layout = QFormLayout()
        geom_layout.setSpacing(10)

        # ngulo alpha (12 < alpha < 15) - Deslizador
        self.alpha_label = QLabel("ngulo : 14.0")
        self.alpha_label.setAlignment(Qt.AlignCenter)

        self.alpha_slider = QSlider(Qt.Horizontal)
        self.alpha_slider.setRange(120, 150)  # 12.0 a 15.0 (multiplicado por 10)
        self.alpha_slider.setValue(140)  # Valor inicial 14.0
        self.alpha_slider.setTickPosition(QSlider.TicksBelow)
        self.alpha_slider.setTickInterval(10)
```

```
1314        self.alpha_slider.valueChanged.connect(self.update_alpha_label)
1315
1316        # Circunferencia C
1317        self.C_input = QDoubleSpinBox()
1318        self.C_input.setRange(0.1, 100)
1319        self.C_input.setValue(5)
1320        self.C_input.setDecimals(2)
1321        self.C_input.setSingleStep(0.5)
1322        self.C_input.setMaximumWidth(120)
1323
1324        # Factor de patrn
1325        self.factor_input = QDoubleSpinBox()
1326        self.factor_input.setRange(0.1, 10)
1327        self.factor_input.setValue(1.0)
1328        self.factor_input.setDecimals(1)
1329        self.factor_input.setSingleStep(0.1)
1330        self.factor_input.setMaximumWidth(120)
1331
1332        # Nmero de vueltas (>3)
1333        self.vueltas_input = QSpinBox()
1334        self.vueltas_input.setRange(4, 50)  # Mayor a 3
1335        self.vueltas_input.setValue(10)
1336        self.vueltas_input.setSingleStep(1)
1337        self.vueltas_input.setMaximumWidth(120)
1338
1339        # Aadir widgets al layout
1340        alpha_widget = QWidget()
1341        alpha_layout = QVBoxLayout()
1342        alpha_layout.addWidget(self.alpha_label)
1343        alpha_layout.addWidget(self.alpha_slider)
1344        alpha_widget.setLayout(alpha_layout)
1345
1346        geom_layout.addRow("ngulo  (12-15):", alpha_widget)
1347        geom_layout.addRow("Circunferencia C (m):", self.C_input)
1348        geom_layout.addRow("Factor patrn:", self.factor_input)
1349        geom_layout.addRow("Vueltas (>3):", self.vueltas_input)
1350        geom_group.setLayout(geom_layout)
1351
1352        # Grupo de parmetros normalizados
1353        norm_group = QGroupBox("Parmetros Normalizados")
1354        norm_group.setStyleSheet("QGroupBox { font-weight: bold; }")
1355        norm_layout = QFormLayout()
1356        norm_layout.setSpacing(10)
1357
1358        # C_lambda (0.75 a 1.25) - Deslizador
1359        self.C_lambda_label = QLabel("C/: 0.75")
1360        self.C_lambda_label.setAlignment(Qt.AlignCenter)
1361
1362        self.C_lambda_slider = QSlider(Qt.Horizontal)
1363        self.C_lambda_slider.setRange(75, 125)  # 0.75 a 1.25 (multiplicado por 100)
1364        self.C_lambda_slider.setValue(75)  # Valor inicial 0.75
1365        self.C_lambda_slider.setTickPosition(QSlider.TicksBelow)
1366        self.C_lambda_slider.setTickInterval(10)
1367        self.C_lambda_slider.valueChanged.connect(self.update_C_lambda_label)
1368
1369        # Aadir widget al layout
1370        C_lambda_widget = QWidget()
1371        C_lambda_layout = QVBoxLayout()
1372        C_lambda_layout.addWidget(self.C_lambda_label)
1373        C_lambda_layout.addWidget(self.C_lambda_slider)
1374        C_lambda_widget.setLayout(C_lambda_layout)
1375
1376        norm_layout.addRow("C/ (0.75-1.25):", C_lambda_widget)
1377        norm_group.setLayout(norm_layout)
1378
1379        # Grupo de parmetros de animacin
1380        anim_group = QGroupBox("Parmetros de Animacin")
1381        anim_group.setStyleSheet("QGroupBox { font-weight: bold; }")
1382        anim_layout = QFormLayout()
1383        anim_layout.setSpacing(10)
1384
1385        self.n_input = QSpinBox()
1386        self.n_input.setRange(100, 2000)
1387        self.n_input.setValue(500)
1388        self.n_input.setSingleStep(100)
1389        self.n_input.setMaximumWidth(120)
1390
1391        self.fps_input = QSpinBox()
1392        self.fps_input.setRange(1, 60)
1393        self.fps_input.setValue(20)
1394        self.fps_input.setMaximumWidth(120)
1395
1396        anim_layout.addRow("Puntos (n):", self.n_input)
1397        anim_layout.addRow("FPS:", self.fps_input)
1398        anim_group.setLayout(anim_layout)
1399
1400        # Frame para botones
1401        button_frame = QWidget()
```

```
        button_layout = QVBoxLayout()
        button_layout.setSpacing(10)
        button_frame.setLayout(button_layout)

        # Botones
        self.simulate_btn = QPushButton("Ejecutar Simulacin")
        self.simulate_btn.setStyleSheet("""
            QPushButton {
                background-color: #4CAF50;
                color: white;
                font-weight: bold;
                padding: 12px;
                font-size: 12pt;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #45a049;
            }
        """)
        self.simulate_btn.clicked.connect(self.run_simulation)

        self.animate_btn = QPushButton("Generar Animacin")
        self.animate_btn.setStyleSheet("""
            QPushButton {
                background-color: #2196F3;
                color: white;
                font-weight: bold;
                padding: 12px;
                font-size: 12pt;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #0b7dda;
            }
        """)
        self.animate_btn.clicked.connect(self.generate_animation)

        self.clear_btn = QPushButton("Limpiar Grficos")
        self.clear_btn.setStyleSheet("""
            QPushButton {
                background-color: #f44336;
                color: white;
                font-weight: bold;
                padding: 12px;
                font-size: 12pt;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #d32f2f;
            }
        """)
        self.clear_btn.clicked.connect(self.clear_plot)

        # Aadir botones
        button_layout.addWidget(self.simulate_btn)
        button_layout.addWidget(self.animate_btn)
        button_layout.addWidget(self.clear_btn)

        # Aadir todos los grupos al layout izquierdo
        self.left_layout.addWidget(geom_group)
        self.left_layout.addWidget(norm_group)
        self.left_layout.addWidget(anim_group)
        self.left_layout.addWidget(button_frame)
        self.left_layout.addStretch()

    def setup_plots(self):
        """Configura el rea de grficos"""
        self.figure = Figure(figsize=(10, 8), dpi=100)
        self.canvas = FigureCanvas(self.figure)
        self.ax = self.figure.add_subplot(111, projection='3d')

        # Configurar ejes
        self.ax.set_xlabel("X", fontsize=14, fontweight='bold')
        self.ax.set_ylabel("Y", fontsize=14, fontweight='bold')
        self.ax.set_zlabel("Z", fontsize=14, fontweight='bold')
        self.ax.set_title("Simulador de Antena Helicoide - Modo Transversal",
                          fontsize=16, fontweight='bold', pad=20)

        # Configurar ticks
        self.ax.tick_params(axis='both', which='major', labelsize=11)

        # Aadir canvas al layout derecho
        self.right_layout.addWidget(self.canvas)

    def update_alpha_label(self):
        """Actualiza la etiqueta del ngulo alpha"""
        value = self.alpha_slider.value() / 10.0
        self.alpha_label.setText(f"ngulo : {value:.1f}")
```

```
1490 |
1491 |     def update_C_lambda_label(self):
1492 |         """Actualiza la etiqueta de C_lambda"""
1493 |         value = self.C_lambda_slider.value() / 100.0
1494 |         self.C_lambda_label.setText(f"C/: {value:.2f}")
1495 |
1496 |     def get_simulation_parameters(self):
1497 |         """Obtiene los parmetros de simulacin"""
1498 |         params = {
1499 |             'alpha': self.alpha_slider.value() / 10.0,  # Convertir de 12.0-15.0
1500 |             'C': self.C_input.value(),
1501 |             'factor': self.factor_input.value(),
1502 |             'vueltas': self.vueltas_input.value(),
1503 |             'C_lambda': self.C_lambda_slider.value() / 100.0,  # Convertir de 0.75-1.25
1504 |             'n': self.n_input.value(),
1505 |             'fps': self.fps_input.value()
1506 |         }
1507 |         return params
1508 |
1509 |     def calculate_parameters(self, params=None):
1510 |         """Calcula los parmetros derivados"""
1511 |         if params is None:
1512 |             params = self.get_simulation_parameters()
1513 |
1514 |         alpha = params['alpha']
1515 |         C = params['C']
1516 |         factor = params['factor']
1517 |         vueltas = params['vueltas']
1518 |         C_lambda = params['C_lambda']
1519 |
1520 |         # Clculos
1521 |         alpha_rad = alpha * np.pi / 180
1522 |         R = C / (2 * np.pi)
1523 |         S = C * np.tan(alpha_rad)
1524 |         L = np.sqrt((C**2) + (S**2))
1525 |         hmax = vueltas * S
1526 |
1527 |         # Parmetros normalizados
1528 |         longitud_onda = C / C_lambda
1529 |         f = 3e8 / longitud_onda / 1e6  # frecuencia en megahertz
1530 |         S_lambda = S / longitud_onda
1531 |         L_lambda = L / longitud_onda
1532 |
1533 |         return {
1534 |             'R': R, 'S': S, 'L': L, 'hmax': hmax,
1535 |             'longitud_onda': longitud_onda, 'f': f,
1536 |             'S_lambda': S_lambda, 'L_lambda': L_lambda,
1537 |             'alpha_rad': alpha_rad
1538 |         }
1539 |
1540 |     def power_pattern(self, altura_z, progress=1, params=None):
1541 |         """Calcula el patrn de radiacin"""
1542 |         if params is None:
1543 |             params = self.get_simulation_parameters()
1544 |
1545 |         # Calcular parmetros derivados
1546 |         calc_params = self.calculate_parameters(params)
1547 |
1548 |         alpha = params['alpha']
1549 |         C = params['C']
1550 |         factor = params['factor']
1551 |         vueltas = params['vueltas']
1552 |         S_lambda = calc_params['S_lambda']
1553 |
1554 |         # Calcular patrn
1555 |         theta_p = np.linspace(0, 2 * np.pi, 50)
1556 |         phi_p = np.linspace(0, np.pi, 50)
1557 |         theta_p, phi_p = np.meshgrid(theta_p, phi_p)
1558 |
1559 |         psi = 2 * np.pi * (S_lambda * (1 - np.cos(phi_p)) + (1/(2*vueltas)))
1560 |
1561 |         E = (np.sin(np.pi/(2 * vueltas)) *
1562 |              (np.sin(vueltas * psi/2) / np.sin(psi/2 + 1e-9)) *
1563 |              np.cos(theta_p))
1564 |
1565 |         E = np.abs(E) / np.max(np.abs(E))
1566 |
1567 |         scale = 0.1 + 0.9 * progress
1568 |         Xp = factor * scale * 0.5 * E * np.sin(phi_p) * np.cos(theta_p)
1569 |         Yp = factor * scale * 0.5 * E * np.sin(phi_p) * np.sin(theta_p)
1570 |         Zp = factor * 0.5 * E * np.cos(phi_p) + altura_z
1571 |
1572 |         return E, Xp, Yp, Zp
1573 |
1574 |     def run_simulation(self):
1575 |         """Ejecuta la simulacin esttica"""
1576 |         try:
1577 |             params = self.get_simulation_parameters()
```

```
            calc_params = self.calculate_parameters(params)

            # Extraer parmetros
            alpha = params['alpha']
            C = params['C']
            factor = params['factor']
            vueltas = params['vueltas']
            C_lambda = params['C_lambda']
            n = params['n']

            R = calc_params['R']
            S = calc_params['S']
            hmax = calc_params['hmax']
            longitud_onda = calc_params['longitud_onda']
            f = calc_params['f']

            # Trayectoria
            theta = np.linspace(0, vueltas * 2 * np.pi, n)
            z = (S / (2 * np.pi)) * theta
            x = R * np.cos(theta)
            y = R * np.sin(theta)

            # Superficie de la helicoide
            theta_surf = np.linspace(0, vueltas * 2 * np.pi, 50)
            r_surf = np.linspace(0, R, 50)
            theta_surf, r_surf = np.meshgrid(theta_surf, r_surf)
            x_surf = r_surf * np.cos(theta_surf)
            y_surf = r_surf * np.sin(theta_surf)
            z_surf = (hmax / (vueltas * 2 * np.pi)) * theta_surf

            self.clear_plot()

            # Dibujar trayectoria completa
            self.ax.plot(x, y, z, color='black', linewidth=3, alpha=0.7, label='Trayectoria')

            # Dibujar punto inicial y final
            self.ax.scatter(x[0], y[0], z[0], c='green', s=100, edgecolors='black', linewidth=2, label='Inicio')
            self.ax.scatter(x[-1], y[-1], z[-1], c='red', s=100, edgecolors='black', linewidth=2, label='Fin')

            # Calcular y dibujar patrn de radiacin en posicin media
            mid_idx = n // 2
            U, Xp, Yp, Zp = self.power_pattern(z[mid_idx], 0.5, params)
            self.ax.plot_surface(
                Xp, Yp, Zp,
                facecolors=plt.cm.plasma(U),
                alpha=0.6,
                rstride=1, cstride=1
            )

            # Configurar lmites
            max_range = max(R * 1.2, hmax/2)
            self.ax.set_xlim(-max_range, max_range)
            self.ax.set_ylim(-max_range, max_range)
            self.ax.set_zlim(0, hmax * 1.1)

            # Aadir informacin del patrn
            info_text = (f' = {alpha}\nC = {C} m\nR = {R:.2f} m\nhmax = {hmax:.1f}m\n'
                         f' = {longitud_onda:.2f} m\nf = {f:.2f} MHz\n'
                         f'C/ = {C_lambda:.4f}\nvueltas = {vueltas}')
            self.ax.text2D(0.02, 0.98, info_text, transform=self.ax.transAxes,
                           fontsize=10, verticalalignment='top',
                           bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.8))

            self.ax.set_title(f"Simulacin Esttica - Antena Helicoide\n"
                              f"={alpha}, C={C}m, C/={C_lambda:.2f}, Vueltas={vueltas}",
                              fontsize=14, fontweight='bold')

            # Aadir leyenda
            self.ax.legend(loc='upper right', fontsize=10)

            self.canvas.draw()

        except Exception as e:
            self.show_error_message(f"Error en simulacin: {str(e)}")

    def generate_animation(self):
        """Genera la animacin en un hilo separado"""
        try:
            params = self.get_simulation_parameters()

            # Deshabilitar botones durante la animacin
            self.animate_btn.setEnabled(False)
            self.simulate_btn.setEnabled(False)
            self.animate_btn.setText("Generando...")

            # Mostrar barra de progreso
            self.progress_bar.setVisible(True)
            self.progress_bar.setValue(0)
```

```
1666 |
1667 |             # Crear y ejecutar hilo de animacin
1668 |             self.animation_thread = AnimationThreadV2(params)
1669 |             self.animation_thread.finished.connect(self.animation_finished)
1670 |             self.animation_thread.error.connect(self.animation_error)
1671 |             self.animation_thread.progress.connect(self.update_progress)
1672 |             self.animation_thread.start()
1673 |
1674 |         except Exception as e:
1675 |             self.show_error_message(f"Error al iniciar animacin: {str(e)}")
1676 |
1677 |     def update_progress(self, value):
1678 |         """Actualiza la barra de progreso"""
1679 |         self.progress_bar.setValue(value)
1680 |
1681 |     def animation_finished(self, save_path):
1682 |         """Se llama cuando la animacin termina exitosamente"""
1683 |         self.animate_btn.setEnabled(True)
1684 |         self.simulate_btn.setEnabled(True)
1685 |         self.animate_btn.setText("Generar Animacin")
1686 |         self.progress_bar.setVisible(False)
1687 |
1688 |         QMessageBox.information(self, "Animacin Completada",
1689 |                                 f"Animacin guardada como:\n{save_path}")
1690 |
1691 |     def animation_error(self, error_msg):
1692 |         """Se llama cuando hay un error en la animacin"""
1693 |         self.animate_btn.setEnabled(True)
1694 |         self.simulate_btn.setEnabled(True)
1695 |         self.animate_btn.setText("Generar Animacin")
1696 |         self.progress_bar.setVisible(False)
1697 |
1698 |         self.show_error_message(f"Error en animacin: {error_msg}")
1699 |
1700 |     def clear_plot(self):
1701 |         """Limpia el grfico"""
1702 |         self.ax.clear()
1703 |         self.ax.set_xlabel("X", fontsize=14, fontweight='bold')
1704 |         self.ax.set_ylabel("Y", fontsize=14, fontweight='bold')
1705 |         self.ax.set_zlabel("Z", fontsize=14, fontweight='bold')
1706 |         self.ax.set_title("Simulador de Antena Helicoide - Modo Transversal",
1707 |                           fontsize=16, fontweight='bold', pad=20)
1708 |         self.ax.tick_params(axis='both', which='major', labelsize=11)
1709 |         self.canvas.draw()
1710 |
1711 |     def show_error_message(self, message):
1712 |         """Muestra un mensaje de error"""
1713 |         QMessageBox.critical(self, "Error", message)
1714 |
1715 | class AntenaHelicoidalGUI(QMainWindow):
1716 |     def __init__(self):
1717 |         super().__init__()
1718 |         self.setWindowTitle("Calculador de Antena Helicoidal")
1719 |         self.setGeometry(100, 100, 1600, 900)
1720 |
1721 |         # Widget central
1722 |         self.central_widget = QWidget()
1723 |         self.setCentralWidget(self.central_widget)
1724 |
1725 |         # Layout principal horizontal
1726 |         self.main_layout = QHBoxLayout()
1727 |         self.central_widget.setLayout(self.main_layout)
1728 |
1729 |         # Panel izquierdo para controles
1730 |         self.left_panel = QWidget()
1731 |         self.left_layout = QVBoxLayout()
1732 |         self.left_panel.setLayout(self.left_layout)
1733 |         self.left_panel.setMaximumWidth(400)
1734 |
1735 |         # Panel derecho para grficos e imgenes
1736 |         self.right_panel = QWidget()
1737 |         self.right_layout = QGridLayout()
1738 |         self.right_panel.setLayout(self.right_layout)
1739 |
1740 |         # Configurar controles
1741 |         self.setup_controls()
1742 |         # Configurar rea de grficos
1743 |         self.setup_graphics()
1744 |
1745 |         # Aadir paneles al layout principal
1746 |         self.main_layout.addWidget(self.left_panel, 1)
1747 |         self.main_layout.addWidget(self.right_panel, 3)
1748 |
1749 |     def setup_controls(self):
1750 |         """Configura los controles de entrada"""
1751 |
1752 |         # Ttulo
1753 |         titulo_label = QLabel("CALCULADOR DE ANTENA HELICOIDAL")
```

```
        titulo_label.setStyleSheet("""
            QLabel {
                font-size: 16pt;
                font-weight: bold;
                color: #2E7D32;
                padding: 10px;
                background-color: #E8F5E9;
                border-radius: 5px;
            }
        """)
        titulo_label.setAlignment(Qt.AlignCenter)

        # Grupo: Parmetros elctricos
        elec_group = QGroupBox("Parametros Electricos")
        elec_group.setStyleSheet("""
            QGroupBox {
                font-weight: bold;
                font-size: 11pt;
                border: 2px solid #4CAF50;
                border-radius: 5px;
                margin-top: 10px;
            }
            QGroupBox::title {
                subcontrol-origin: margin;
                left: 10px;
                padding: 0 5px 0 5px;
                color: #2E7D32;
            }
        """)
        elec_layout = QFormLayout()
        elec_layout.setSpacing(10)

        self.voltaje_input = QDoubleSpinBox()
        self.voltaje_input.setRange(1, 1000)
        self.voltaje_input.setValue(120.0)
        self.voltaje_input.setDecimals(1)
        self.voltaje_input.setSingleStep(10)
        self.voltaje_input.setSuffix(" V")
        self.voltaje_input.setMaximumWidth(150)
        self.voltaje_input.setStyleSheet("padding: 3px;")

        self.corriente_input = QDoubleSpinBox()
        self.corriente_input.setRange(0.01, 10)
        self.corriente_input.setValue(0.8)
        self.corriente_input.setDecimals(3)
        self.corriente_input.setSingleStep(0.1)
        self.corriente_input.setSuffix(" A")
        self.corriente_input.setMaximumWidth(150)
        self.corriente_input.setStyleSheet("padding: 3px;")

        self.carga_input = QDoubleSpinBox()
        self.carga_input.setRange(1e-12, 1e-4)
        self.carga_input.setValue(1.2e-8)
        self.carga_input.setDecimals(12)
        self.carga_input.setSuffix(" C")
        self.carga_input.setMaximumWidth(150)
        self.carga_input.setStyleSheet("padding: 3px;")

        elec_layout.addRow("Voltaje (V):", self.voltaje_input)
        elec_layout.addRow("Corriente (I):", self.corriente_input)
        elec_layout.addRow("Carga (Q):", self.carga_input)
        elec_group.setLayout(elec_layout)

        # Grupo: Parmetros de diseo
        design_group = QGroupBox("Parametros de Diseo")
        design_group.setStyleSheet("""
            QGroupBox {
                font-weight: bold;
                font-size: 11pt;
                border: 2px solid #2196F3;
                border-radius: 5px;
                margin-top: 10px;
            }
            QGroupBox::title {
                subcontrol-origin: margin;
                left: 10px;
                padding: 0 5px 0 5px;
                color: #1565C0;
            }
        """)
        design_layout = QFormLayout()
        design_layout.setSpacing(10)

        self.opcion_combo = QComboBox()
        self.opcion_combo.addItems(["Opcion 1 (10)", "Opcion 2 (10)",
                                    "Opcion 3 (10)", "Opcion 4 (10)"])
        self.opcion_combo.setCurrentIndex(2)
        self.opcion_combo.setMaximumWidth(180)
```

```
        self.opcion_combo.setStyleSheet("padding: 3px;")

        self.flujo_input = QDoubleSpinBox()
        self.flujo_input.setRange(0.1, 100)
        self.flujo_input.setValue(2.3)
        self.flujo_input.setDecimals(3)
        self.flujo_input.setSingleStep(0.1)
        self.flujo_input.setMaximumWidth(150)
        self.flujo_input.setStyleSheet("padding: 3px;")

        self.vueltas_input = QSpinBox()
        self.vueltas_input.setRange(4, 100)
        self.vueltas_input.setValue(5)
        self.vueltas_input.setSingleStep(1)
        self.vueltas_input.setMaximumWidth(150)
        self.vueltas_input.setStyleSheet("padding: 3px;")

        self.alpha_label = QLabel("Angulo : 14.0")
        self.alpha_label.setAlignment(Qt.AlignCenter)
        self.alpha_label.setStyleSheet("font-weight: bold; color: #D32F2F;")

        self.alpha_slider = QSlider(Qt.Horizontal)
        self.alpha_slider.setRange(120, 150)
        self.alpha_slider.setValue(140)
        self.alpha_slider.setTickPosition(QSlider.TicksBelow)
        self.alpha_slider.setTickInterval(10)
        self.alpha_slider.setStyleSheet("""
            QSlider::groove:horizontal {
                border: 1px solid #999999;
                height: 8px;
                background: qlineargradient(x1:0, y1:0, x2:0, y2:1,
                    stop:0 #B1B1B1, stop:1 #c4c4c4);
                margin: 2px 0;
            }
            QSlider::handle:horizontal {
                background: qlineargradient(x1:0, y1:0, x2:1, y2:1,
                    stop:0 #D32F2F, stop:1 #F44336);
                border: 1px solid #5c5c5c;
                width: 18px;
                margin: -2px 0;
                border-radius: 3px;
            }
        """)
        self.alpha_slider.valueChanged.connect(self.update_alpha_label)

        alpha_widget = QWidget()
        alpha_layout = QVBoxLayout()
        alpha_layout.addWidget(self.alpha_label)
        alpha_layout.addWidget(self.alpha_slider)
        alpha_widget.setLayout(alpha_layout)

        design_layout.addRow("Opcion escala:", self.opcion_combo)
        design_layout.addRow("Valor de flujo:", self.flujo_input)
        design_layout.addRow("Vueltas (>3):", self.vueltas_input)
        design_layout.addRow("Angulo  (12-15):", alpha_widget)
        design_group.setLayout(design_layout)

        # Frame para botones
        button_frame = QWidget()
        button_layout = QVBoxLayout()
        button_layout.setSpacing(12)
        button_layout.setContentsMargins(10, 20, 10, 10)
        button_frame.setLayout(button_layout)

        # Botn "Calcular Parametros"
        self.mostrar_calc_btn = QPushButton("Calcular Parametros")
        self.mostrar_calc_btn.setStyleSheet("""
            QPushButton {
                background-color: #4CAF50;
                color: white;
                font-weight: bold;
                padding: 15px;
                font-size: 13pt;
                border-radius: 8px;
                border: 2px solid #388E3C;
            }
            QPushButton:hover {
                background-color: #45a049;
                border: 2px solid #2E7D32;
            }
            QPushButton:pressed {
                background-color: #388E3C;
            }
        """)
        self.mostrar_calc_btn.clicked.connect(self.mostrar_calculos)

        # Botn "Dibujar Helicoide"
        self.dibujar_btn = QPushButton("Dibujar Helicoide")
```

```
        self.dibujar_btn.setStyleSheet("""
            QPushButton {
                background-color: #2196F3;
                color: white;
                font-weight: bold;
                padding: 15px;
                font-size: 13pt;
                border-radius: 8px;
                border: 2px solid #1976D2;
            }
            QPushButton:hover {
                background-color: #0b7dda;
                border: 2px solid #1565C0;
            }
            QPushButton:pressed {
                background-color: #1976D2;
            }
        """)
        self.dibujar_btn.clicked.connect(self.dibujar_helicoide)

        # Botn "Limpiar"
        self.limpiar_btn = QPushButton("Limpiar")
        self.limpiar_btn.setStyleSheet("""
            QPushButton {
                background-color: #f44336;
                color: white;
                font-weight: bold;
                padding: 15px;
                font-size: 13pt;
                border-radius: 8px;
                border: 2px solid #D32F2F;
            }
            QPushButton:hover {
                background-color: #d32f2f;
                border: 2px solid #C2185B;
            }
            QPushButton:pressed {
                background-color: #D32F2F;
            }
        """)
        self.limpiar_btn.clicked.connect(self.limpiar)

        button_layout.addWidget(self.mostrar_calc_btn)
        button_layout.addWidget(self.dibujar_btn)
        button_layout.addWidget(self.limpiar_btn)
        button_layout.addStretch()

        # Aadir todos los widgets al panel izquierdo
        self.left_layout.addWidget(titulo_label)
        self.left_layout.addWidget(elec_group)
        self.left_layout.addWidget(design_group)
        self.left_layout.addWidget(button_frame)
        self.left_layout.addStretch()

    def update_alpha_label(self):
        """Actualiza la etiqueta del ngulo alpha"""
        value = self.alpha_slider.value() / 10.0
        self.alpha_label.setText(f"Angulo : {value:.1f}")

    def setup_graphics(self):
        """Configura el rea de grficos e imgenes con fondo blanco"""
        # Crear contenedor para las dos imgenes
        self.image_container = QWidget()
        self.image_layout = QHBoxLayout()
        self.image_container.setLayout(self.image_layout)

        # Panel para imagen A35 (circuito RLC) con fondo blanco
        self.a35_panel = QWidget()
        self.a35_layout = QVBoxLayout()
        self.a35_panel.setLayout(self.a35_layout)
        self.a35_panel.setStyleSheet("background-color: white;")

        a35_title = QLabel("CIRCUITO RLC - ANTENA")
        a35_title.setStyleSheet("""
            QLabel {
                font-size: 12pt;
                font-weight: bold;
                color: #4CAF50;
                padding: 5px;
                background-color: #E8F5E9;
                border-radius: 3px;
            }
        """)
        a35_title.setAlignment(Qt.AlignCenter)

        self.a35_label = QLabel()
        self.a35_label.setAlignment(Qt.AlignCenter)
        self.a35_label.setStyleSheet("""
```

```
            QLabel {
                border: 2px solid #4CAF50;
                border-radius: 5px;
                background-color: white;
                min-height: 400px;
                min-width: 500px;
            }
        """)

        # Cargamos imagen A35.png
        self.load_a35_image()

        self.a35_layout.addWidget(a35_title)
        self.a35_layout.addWidget(self.a35_label)

        # Panel para helicoide 3D con fondo blanco
        self.helicoide_panel = QWidget()
        self.helicoide_layout = QVBoxLayout()
        self.helicoide_panel.setLayout(self.helicoide_layout)
        self.helicoide_panel.setStyleSheet("background-color: white;")

        helicoide_title = QLabel("ANTENA HELICOIDAL 3D")
        helicoide_title.setStyleSheet("""
            QLabel {
                font-size: 12pt;
                font-weight: bold;
                color: #2196F3;
                padding: 5px;
                background-color: #E3F2FD;
                border-radius: 3px;
            }
        """)
        helicoide_title.setAlignment(Qt.AlignCenter)

        # Crear figura matplotlib para la helicoide con fondo blanco
        self.figure = Figure(figsize=(6, 4), dpi=100, facecolor='white')
        self.canvas = FigureCanvas(self.figure)
        self.ax = self.figure.add_subplot(111, projection='3d')

        # Configurar fondo blanco para el grfico 3D (forma correcta)
        self.ax.set_facecolor('white')

        # Configurar los planos (panes) con fondo blanco
        self.ax.xaxis.set_pane_color((1.0, 1.0, 1.0, 1.0))
        self.ax.yaxis.set_pane_color((1.0, 1.0, 1.0, 1.0))
        self.ax.zaxis.set_pane_color((1.0, 1.0, 1.0, 1.0))

        # Configurar ejes iniciales vacos
        self.ax.set_xlabel("X (m)")
        self.ax.set_ylabel("Y (m)")
        self.ax.set_zlabel("Altura (m)")
        self.ax.set_title("Helicoide - Esperando datos", fontsize=10)
        self.ax.grid(True, alpha=0.3)

        self.figure.tight_layout()

        self.helicoide_layout.addWidget(helicoide_title)
        self.helicoide_layout.addWidget(self.canvas)

        # Aadir ambos paneles al layout de imgenes (lado a lado)
        self.image_layout.addWidget(self.a35_panel, 1)
        self.image_layout.addWidget(self.helicoide_panel, 1)

        # Aadir al layout derecho
        self.right_layout.addWidget(self.image_container, 0, 0, 1, 2)

        # Etiqueta de estado
        self.status_label = QLabel("Listo para calcular")
        self.status_label.setStyleSheet("""
            QLabel {
                font-size: 10pt;
                color: #666;
                padding: 5px;
                background-color: #F5F5F5;
                border-radius: 3px;
            }
        """)
        self.status_label.setAlignment(Qt.AlignCenter)
        self.right_layout.addWidget(self.status_label, 1, 0, 1, 2)

    def load_a35_image(self):
        """Carga la imagen A35.png con fondo blanco"""
        try:
            pixmap = QPixmap("A35.png")
            if not pixmap.isNull():
                # Crear un pixmap con fondo blanco
                white_pixmap = QPixmap(pixmap.size())
                white_pixmap.fill(Qt.white)
```

```
                # Pintar la imagen original sobre el fondo blanco
                from PyQt5.QtGui import QPainter
                painter = QPainter(white_pixmap)
                painter.drawPixmap(0, 0, pixmap)
                painter.end()

                # Escalar manteniendo aspecto
                scaled_pixmap = white_pixmap.scaled(500, 400, Qt.KeepAspectRatio, Qt.SmoothTransformation)
                self.a35_label.setPixmap(scaled_pixmap)
                self.a35_label.setMinimumSize(scaled_pixmap.size())
            else:
                self.a35_label.setText("A35.png no encontrada\n(Asegurate de que este en la misma carpeta)")
                self.a35_label.setStyleSheet("""
                    QLabel {
                        border: 2px solid #f44336;
                        border-radius: 5px;
                        padding: 20px;
                        background-color: white;
                        color: #D32F2F;
                        font-weight: bold;
                        min-height: 300px;
                        min-width: 400px;
                    }
                """)
        except Exception as e:
            self.a35_label.setText(f"Error cargando A35.png:\n{str(e)}")
            self.a35_label.setStyleSheet("""
                QLabel {
                    border: 2px solid #f44336;
                    border-radius: 5px;
                    padding: 20px;
                    background-color: white;
                    color: #D32F2F;
                    font-weight: bold;
                    min-height: 300px;
                    min-width: 400px;
                }
            """)

    def formato_inductancia_detallada(self, valor):
        """Formatea la inductancia."""
        if valor >= 1:
            return f"{valor:.6f} H"
        elif valor >= 1e-3:
            return f"{valor*1e3:.6f} mH"
        elif valor >= 1e-6:
            return f"{valor*1e6:.6f} H"
        elif valor >= 1e-9:
            return f"{valor*1e9:.6f} nH"
        elif valor >= 1e-12:
            return f"{valor*1e12:.6f} pH"
        else:
            return f"{valor:.2e} H"

    def formato_capacitancia_detallada(self, valor):
        """Formatea la capacitancia."""
        if valor >= 1:
            return f"{valor:.6f} F"
        elif valor >= 1e-3:
            return f"{valor*1e3:.6f} mF"
        elif valor >= 1e-6:
            return f"{valor*1e6:.6f} F"
        elif valor >= 1e-9:
            return f"{valor*1e9:.6f} nF"
        elif valor >= 1e-12:
            return f"{valor*1e12:.6f} pF"
        else:
            return f"{valor:.2e} F"

    def calcular_parametros(self):
        """Realiza los clculos y retorna los resultados"""
        try:
            # Obtener valores de la interfaz
            V = self.voltaje_input.value()
            I = self.corriente_input.value()
            Q = self.carga_input.value()
            opcion = self.opcion_combo.currentIndex() + 1
            valordeflujo = self.flujo_input.value()
            vueltas = self.vueltas_input.value()
            alpha = self.alpha_slider.value() / 10.0

            impedancia_calculada = V / I

            # Funcin flujo segn opcin
            def flujo(opcion):
                if opcion == 1:
                    return 1e-11
```

```
                elif opcion == 2:
                    return 1e-9
                elif opcion == 3:
                    return 1e-7
                elif opcion == 4:
                    return 1e-5
                else:
                    raise ValueError("Opcion invalida")

            # Clculos
            Flujo = valordeflujo * flujo(opcion)
            alpha_rad = alpha * np.pi / 180

            capacitancia = Q / V
            inductancia = Flujo / I

            frecuencia_antena = 1 / (2 * np.pi * np.sqrt(inductancia * capacitancia))
            frecuencia_MHz = frecuencia_antena / 1e6
            longitud_onda = 3e8 / frecuencia_antena

            valordecircunferenciaminimo = longitud_onda * 0.75
            valordecircunferenciamaximo = longitud_onda * 1.25

            C = longitud_onda
            R = C / (2 * np.pi)
            D = 2 * R
            S = C * np.tan(alpha_rad)
            L = np.sqrt(C**2 + S**2)
            hmax = vueltas * S

            C_lambda = C / longitud_onda
            S_lambda = S / longitud_onda
            L_lambda = L / longitud_onda

            Impedanciaminima = ((2 * np.pi * 3e8)**2 * (Q * Flujo)) / ((C / 0.75)**2)
            Impedanciamaxima = ((2 * np.pi * 3e8)**2 * (Q * Flujo)) / ((C / 1.25)**2)

            # Formatear valores
            capacitancia_formateada = self.formato_capacitancia_detallada(capacitancia)
            inductancia_formateada = self.formato_inductancia_detallada(inductancia)
            capacitancia_exacta = f"{capacitancia:.2e} F"
            inductancia_exacta = f"{inductancia:.2e} H"
            flujo_exacto = f"{Flujo:.2e} Wb"

            return {
                'V': V, 'I': I, 'Q': Q, 'opcion': opcion,
                'valordeflujo': valordeflujo, 'vueltas': vueltas, 'alpha': alpha,
                'impedancia_calculada': impedancia_calculada,
                'Flujo': Flujo, 'alpha_rad': alpha_rad,
                'capacitancia': capacitancia, 'inductancia': inductancia,
                'frecuencia_antena': frecuencia_antena, 'frecuencia_MHz': frecuencia_MHz,
                'longitud_onda': longitud_onda,
                'valordecircunferenciaminimo': valordecircunferenciaminimo,
                'valordecircunferenciamaximo': valordecircunferenciamaximo,
                'C': C, 'R': R, 'D': D, 'S': S, 'L': L, 'hmax': hmax,
                'C_lambda': C_lambda, 'S_lambda': S_lambda, 'L_lambda': L_lambda,
                'Impedanciaminima': Impedanciaminima, 'Impedanciamaxima': Impedanciamaxima,
                'capacitancia_formateada': capacitancia_formateada,
                'inductancia_formateada': inductancia_formateada,
                'capacitancia_exacta': capacitancia_exacta,
                'inductancia_exacta': inductancia_exacta,
                'flujo_exacto': flujo_exacto
            }

        except Exception as e:
            raise ValueError(f"Error en los calculos: {str(e)}")

    def mostrar_calculos(self):
        """Muestra los clculos en un cuadro de mensaje"""
        try:
            resultados = self.calcular_parametros()

            # Crear texto de resultados
            resultados_text = f"""

            Resultados:

             Impedancia calculada: {resultados['impedancia_calculada']:.1f}
             Flujo magnetico: {resultados['flujo_exacto']} Wb
             Capacitancia: {resultados['capacitancia_formateada']} F
             Inductancia: {resultados['inductancia_formateada']} H
             Frecuencia de antena: {resultados['frecuencia_MHz']:.2f} MHz
             Longitud de onda: {resultados['longitud_onda']:.2f} m
             Circunferencia (C): {resultados['C']:.3f} m
             Radio (R): {resultados['R']:.3f} m
             Diametro (D): {resultados['D']:.3f} m
             Paso (S): {resultados['S']:.3f} m
             Altura total (hmax): {resultados['hmax']:.3f} m
```

```
            Longitud por vuelta (L): {resultados['L']:.3f} m

            Recomendaciones de diseo:

            Rango de circunferencia: {resultados['valordecircunferenciaminimo']:.2f} m a {resultados['valordecircunferenciamaximo']:.2f} m
            Rango de impedancias: {resultados['Impedanciaminima']:.2f}  a {resultados['Impedanciamaxima']:.2f}
            Angulo  recomendado: 12 a 15
            Minimo de vueltas: >3

            ESCALAS DE FLUJO POR OPCION:

            Opcion 1: Flujo base = 10
            Opcion 2: Flujo base = 10
            Opcion 3: Flujo base = 10
            Opcion 4: Flujo base = 10
            """

            self.status_label.setText(f"Calculos completados - f = {resultados['frecuencia_MHz']:.2f} MHz")
            QMessageBox.information(self, "Calculos Detallados", resultados_text)

        except Exception as e:
            self.status_label.setText("Error en calculos")
            QMessageBox.critical(self, "Error", f"Error en los calculos:\n{str(e)}")

    def dibujar_helicoide(self):
        """Dibuja la helicoide en el grfico 3D"""
        try:
            resultados = self.calcular_parametros()

            # Limpiar figura
            self.figure.clear()
            self.ax = self.figure.add_subplot(111, projection='3d')

            # Configurar fondo blanco
            self.figure.set_facecolor('white')
            self.ax.set_facecolor('white')
            self.ax.xaxis.set_pane_color((1.0, 1.0, 1.0, 1.0))
            self.ax.yaxis.set_pane_color((1.0, 1.0, 1.0, 1.0))
            self.ax.zaxis.set_pane_color((1.0, 1.0, 1.0, 1.0))

            # Extraer parmetros geomtricos
            R = resultados['R']
            S = resultados['S']
            hmax = resultados['hmax']
            vueltas = resultados['vueltas']
            alpha = resultados['alpha']
            C = resultados['C']
            f = resultados['frecuencia_MHz']

            # Generar puntos de la helicoide
            n_points = 500
            theta = np.linspace(0, vueltas * 2 * np.pi, n_points)
            z = (S / (2 * np.pi)) * theta
            x = R * np.cos(theta)
            y = R * np.sin(theta)

            # Dibujar la helicoide principal
            self.ax.plot(x, y, z, 'black', linewidth=2.5, alpha=0.8, label='Antena helicoidal')

            # Marcar puntos importantes
            self.ax.scatter(x[0], y[0], z[0], c='green', s=100, marker='o',
                            edgecolors='black', linewidth=1.5, label='Inicio', zorder=5)
            self.ax.scatter(x[-1], y[-1], z[-1], c='red', s=100, marker='o',
                            edgecolors='black', linewidth=1.5, label='Fin', zorder=5)

            # Dibujar lnea central (eje Z)
            self.ax.plot([0, 0], [0, 0], [0, hmax], 'k--', linewidth=1.5,
                         alpha=0.5, label='Eje central')

            # Dibujar crculo base
            theta_base = np.linspace(0, 2 * np.pi, 100)
            x_base = R * np.cos(theta_base)
            y_base = R * np.sin(theta_base)
            self.ax.plot(x_base, y_base, np.zeros_like(x_base), 'g-',
                         linewidth=1, alpha=0.5, label='Base')

            # Dibujar algunos puntos intermedios
            for i in range(1, vueltas):
                idx = int(i * n_points / vueltas)
                if idx < len(x):
                    self.ax.scatter(x[idx], y[idx], z[idx], c='orange', s=30,
                                    marker='o', alpha=0.6)

            # Configurar lmites
            max_range = max(R * 1.5, hmax/2)
            self.ax.set_xlim([-max_range, max_range])
            self.ax.set_ylim([-max_range, max_range])
            self.ax.set_zlim([0, hmax * 1.1])
```

```
2370 |
2371 |             # Configurar etiquetas
2372 |             self.ax.set_xlabel('X (m)', fontsize=10, fontweight='bold')
2373 |             self.ax.set_ylabel('Y (m)', fontsize=10, fontweight='bold')
2374 |             self.ax.set_zlabel('Altura (m)', fontsize=10, fontweight='bold')
2375 |
2376 |             # Ttulo con informacin
2377 |             title_text = f'ANTENA HELICOIDAL - f = {f:.2f} MHz'
2378 |             self.ax.set_title(title_text, fontsize=12, fontweight='bold', pad=12)
2379 |
2380 |             # Aadir leyenda
2381 |             self.ax.legend(loc='upper right', fontsize=8)
2382 |
2383 |             # Habilitar grid
2384 |             self.ax.grid(True, alpha=0.3)
2385 |
2386 |             # Ajustar vista para mejor perspectiva
2387 |             self.ax.view_init(elev=25, azim=45)
2388 |
2389 |             # Ajustar layout
2390 |             self.figure.tight_layout()
2391 |
2392 |             # Actualizar canvas
2393 |             self.canvas.draw()
2394 |
2395 |             self.status_label.setText(f"Helicoide dibujado - ={alpha:.1f}, {vueltas} vueltas")
2396 |
2397 |         except Exception as e:
2398 |             self.status_label.setText("Error dibujando helicoide")
2399 |             QMessageBox.critical(self, "Error", f"Error al dibujar helicoide:\n{str(e)}")
2400 |
2401 |     def limpiar(self):
2402 |         """Limpia el grfico de la helicoide"""
2403 |         self.figure.clear()
2404 |         self.ax = self.figure.add_subplot(111, projection='3d')
2405 |
2406 |         # Configurar fondo blanco
2407 |         self.figure.set_facecolor('white')
2408 |         self.ax.set_facecolor('white')
2409 |         self.ax.xaxis.set_pane_color((1.0, 1.0, 1.0, 1.0))
2410 |         self.ax.yaxis.set_pane_color((1.0, 1.0, 1.0, 1.0))
2411 |         self.ax.zaxis.set_pane_color((1.0, 1.0, 1.0, 1.0))
2412 |
2413 |         # Configurar ejes vacos
2414 |         self.ax.set_xlabel("X (m)", fontsize=10)
2415 |         self.ax.set_ylabel("Y (m)", fontsize=10)
2416 |         self.ax.set_zlabel("Altura (m)", fontsize=10)
2417 |         self.ax.set_title("Helicoide - Esperando datos", fontsize=10)
2418 |         self.ax.grid(True, alpha=0.3)
2419 |
2420 |         self.figure.tight_layout()
2421 |         self.canvas.draw()
2422 |
2423 |         self.status_label.setText("Grafico limpiado - Listo para nuevos calculos")
2424 |
2425 | class AnimationThreadV4(QThread):
2426 |     """Hilo para generar la animacin en segundo plano"""
2427 |     finished = pyqtSignal(str)
2428 |     error = pyqtSignal(str)
2429 |     progress = pyqtSignal(int)
2430 |
2431 |     def __init__(self, params):
2432 |         super().__init__()
2433 |         self.params = params
2434 |
2435 |     def run(self):
2436 |         try:
2437 |             # Extraer parmetros
2438 |             V = self.params['V']
2439 |             I = self.params['I']
2440 |             Q = 1.2e-8
2441 |             opcion = self.params['opcion']
2442 |             factor = self.params['factor']
2443 |             valordeflujo = self.params['valordeflujo']
2444 |             alpha = self.params['alpha']
2445 |             valordecircunferencia = self.params['valordecircunferencia']
2446 |             opcion_circunferencia = self.params['opcion_circunferencia']
2447 |             vueltas = self.params['vueltas']
2448 |             potenciometro_valor = self.params['potenciometro_valor']
2449 |             n = self.params['n']
2450 |             fps = self.params['fps']
2451 |
2452 |             # Clculos iniciales
2453 |             Impedancia = V / I
2454 |
2455 |             def flujo(opcion):
2456 |                 if opcion == 1:
2457 |                     return 1e-11
```

```
            elif opcion == 2:
                return 1e-9
            elif opcion == 3:
                return 1e-7
            elif opcion == 4:
                return 1e-5
            else:
                raise ValueError("Opcin invlida para el flujo magntico.")

        Flujo = valordeflujo * flujo(opcion)
        capacitancia = Q / V
        inductancia = Flujo / I

        def ordenc(opcion_circunferencia):
            if opcion_circunferencia == 1:
                return .1
            elif opcion_circunferencia == 2:
                return 1
            elif opcion_circunferencia == 3:
                return 10
            elif opcion_circunferencia == 4:
                return 100
            else:
                raise ValueError("Opcin invlida para el orden de la circunferencia.")

        C = valordecircunferencia * ordenc(opcion_circunferencia)
        R = C / (2 * np.pi)
        S = C * np.tan(alpha * np.pi / 180)
        L = np.sqrt((C**2) + (S**2))
        hmax = vueltas * S

        # Clculos de frecuencia y longitud de onda
        frecuencia_antena = 1 / (2 * np.pi * np.sqrt(inductancia * capacitancia))
        longitud_onda = 3e8 / frecuencia_antena

        # Potenciometro
        Impedanciaminima = (2*np.pi * 3e8)**2 * (Q * Flujo) / ((C/.75)**2)
        Impedanciamaxima = (2*np.pi * 3e8)**2 * (Q * Flujo) / ((C/1.25)**2)

        def potenciometro(Impedanciaminima, Impedanciamaxima, valor):
            posicion = max(0, min(1, valor))
            return Impedanciaminima + posicion * (Impedanciamaxima - Impedanciaminima)

        impedancia_ajustada = potenciometro(Impedanciaminima, Impedanciamaxima, potenciometro_valor)
        corriente_ajustada = V / impedancia_ajustada
        inductancia_ajustada = Flujo / corriente_ajustada
        frecuencia_ajustada = 1 / (2 * np.pi * np.sqrt(inductancia_ajustada * capacitancia))
        longitud_onda_ajustada = 3e8 / frecuencia_ajustada
        S_lambda_ajustada = S / longitud_onda_ajustada
        C_lambda_ajustada = C / longitud_onda_ajustada
        L_lambda_ajustada = L / longitud_onda_ajustada

        # Trayectoria
        theta = np.linspace(0, vueltas * 2 * np.pi, n)
        z = (S / (2 * np.pi)) * theta
        x = R * np.cos(theta)
        y = R * np.sin(theta)

        # Superficie de la helicoide
        theta_surf = np.linspace(0, vueltas * 2 * np.pi, 50)
        r_surf = np.linspace(0, R, 50)
        theta_surf, r_surf = np.meshgrid(theta_surf, r_surf)
        x_surf = r_surf * np.cos(theta_surf)
        y_surf = r_surf * np.sin(theta_surf)
        z_surf = (hmax / (vueltas * 2 * np.pi)) * theta_surf

        def power_pattern(altura_z, progress=1):
            theta_p = np.linspace(0, 2 * np.pi, 50)
            phi_p = np.linspace(0, np.pi, 50)
            theta_p, phi_p = np.meshgrid(theta_p, phi_p)

            psi = 2 * np.pi * (S_lambda_ajustada * (1 - np.cos(phi_p)) + (1/(2*vueltas)))

            E = (np.sin(np.pi/(2 * vueltas)) *
                 (np.sin(vueltas * psi/2) / np.sin(psi/2 + 1e-9)) *
                 np.cos(theta_p))

            E = np.abs(E) / np.max(np.abs(E))

            scale = 0.1 + 0.9 * progress
            Xp = factor * scale * E * np.sin(phi_p) * np.cos(theta_p)
            Yp = factor * scale * E * np.sin(phi_p) * np.sin(theta_p)
            Zp = factor * E * np.cos(phi_p) + altura_z

            return E, Xp, Yp, Zp

        # Configuracin de la figura
        fig = plt.figure(figsize=(12, 8))
```

```
            ax = fig.add_subplot(111, projection='3d')

            max_range = max(R, hmax/2)
            ax.set_xlim([-max_range*1, max_range*1])
            ax.set_ylim([-max_range*1, max_range*1])
            ax.set_zlim([0, hmax*1.2])

            ax.set_xlabel("X", fontsize=12)
            ax.set_ylabel("Y", fontsize=12)
            ax.set_zlabel("Z", fontsize=12)
            ax.set_title(f"Antena Helicoide - {vueltas} vueltas\nFrecuencia: {frecuencia_ajustada/1e6:.2f} MHz", fontsize=14)

            # Dibujar trayectoria
            filamento_gris, = ax.plot(x, y, z, color='black', linewidth=2, alpha=0.7)
            filamento_rojo, = ax.plot([], [], [], color='red', linewidth=4)

            # Inicializar partcula
            particle = ax.scatter([], [], [], c='red', s=80, edgecolors='black', linewidth=2, zorder=10)
            U_init, Xp_init, Yp_init, Zp_init = power_pattern(z[0], progress=0.1)
            radiation_pattern = ax.plot_surface(
                Xp_init, Yp_init, Zp_init,
                facecolors=plt.cm.plasma(U_init),
                alpha=0.6,
                rstride=1, cstride=1,
                zorder=5
            )

            def init():
                particle._offsets3d = ([], [], [])
                return particle, radiation_pattern

            def update(i):
                nonlocal radiation_pattern

                particle._offsets3d = ([x[i]], [y[i]], [z[i]])
                progress = 0.1 + 0.9 * (i / (n - 1))

                U, Xp, Yp, Zp = power_pattern(z[i], progress)

                radiation_pattern.remove()
                radiation_pattern = ax.plot_surface(
                    Xp, Yp, Zp,
                    facecolors=plt.cm.plasma(U),
                    alpha=0.6,
                    rstride=1, cstride=1,
                    zorder=5
                )

                filamento_rojo.set_data(x[:i+1], y[:i+1])
                filamento_rojo.set_3d_properties(z[:i+1])

                # Emitir progreso
                self.progress.emit(int(100 * i / n))

                return particle, radiation_pattern, filamento_rojo

            # Crear animacin
            ani = animation.FuncAnimation(
                fig, update, frames=n, init_func=init,
                interval=50, blit=False, repeat=True
            )

            # Guardar animacin
            save_path = "4-patron_antena_helicoide.gif"
            ani.save(save_path, writer=PillowWriter(fps=fps))
            plt.tight_layout()
            plt.close(fig)

            self.finished.emit(save_path)

        except Exception as e:
            self.error.emit(str(e))

class AntenaHelicoideGUI(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Simulador de Antena Helicoide RLC")
        self.setGeometry(50, 50, 1400, 900)

        # Widget central
        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)

        # Layout principal horizontal
        self.main_layout = QHBoxLayout()
        self.central_widget.setLayout(self.main_layout)

        # Frame izquierdo para controles
```

```
2634            self.left_frame = QWidget()
2635            self.left_layout = QVBoxLayout()
2636            self.left_frame.setLayout(self.left_layout)
2637            self.left_frame.setMaximumWidth(450)
2638
2639            # Frame derecho para grficos
2640            self.right_frame = QWidget()
2641            self.right_layout = QVBoxLayout()
2642            self.right_frame.setLayout(self.right_layout)
2643
2644            # Panel de control
2645            self.setup_control_panel()
2646
2647            # rea de grficos
2648            self.setup_plots()
2649
2650            # Aadir frames al layout principal
2651            self.main_layout.addWidget(self.left_frame, 1)
2652            self.main_layout.addWidget(self.right_frame, 3)
2653
2654            # Hilo de animacin
2655            self.animation_thread = None
2656
2657            # Barra de progreso
2658            self.progress_bar = QProgressBar()
2659            self.progress_bar.setVisible(False)
2660            self.right_layout.addWidget(self.progress_bar)
2661
2662        def setup_control_panel(self):
2663            """Configura el panel de control con parmetros"""
2664
2665            # Grupo de parmetros elctricos
2666            elec_group = QGroupBox("Parmetros Elctricos")
2667            elec_group.setStyleSheet("QGroupBox { font-weight: bold; }")
2668            elec_layout = QFormLayout()
2669            elec_layout.setSpacing(10)
2670
2671            self.V_input = QDoubleSpinBox()
2672            self.V_input.setRange(1, 240)
2673            self.V_input.setValue(120)
2674            self.V_input.setDecimals(1)
2675            self.V_input.setSingleStep(10)
2676            self.V_input.setMaximumWidth(120)
2677
2678            self.I_input = QDoubleSpinBox()
2679            self.I_input.setRange(0.1, 10)
2680            self.I_input.setValue(0.8)
2681            self.I_input.setDecimals(2)
2682            self.I_input.setSingleStep(0.1)
2683            self.I_input.setMaximumWidth(120)
2684
2685            elec_layout.addRow("Voltaje (V):", self.V_input)
2686            elec_layout.addRow("Corriente (I):", self.I_input)
2687            elec_group.setLayout(elec_layout)
2688
2689            # Grupo de parmetros de escala
2690            escala_group = QGroupBox("Parmetros de Escala")
2691            escala_group.setStyleSheet("QGroupBox { font-weight: bold; }")
2692            escala_layout = QFormLayout()
2693            escala_layout.setSpacing(10)
2694
2695            self.opcion_combo = QComboBox()
2696            self.opcion_combo.addItems(["1 (10)", "2 (10)", "3 (10)", "4 (10)"])
2697            self.opcion_combo.setCurrentIndex(2)
2698            self.opcion_combo.setMaximumWidth(120)
2699
2700            self.factor_input = QDoubleSpinBox()
2701            self.factor_input.setRange(0.1, 10)
2702            self.factor_input.setValue(1.0)
2703            self.factor_input.setDecimals(1)
2704            self.factor_input.setSingleStep(0.1)
2705            self.factor_input.setMaximumWidth(120)
2706
2707            escala_layout.addRow("Escala flujo:", self.opcion_combo)
2708            escala_layout.addRow("Factor patrn:", self.factor_input)
2709            escala_group.setLayout(escala_layout)
2710
2711            # Grupo de parmetros magnticos
2712            mag_group = QGroupBox("Parmetros Magnticos")
2713            mag_group.setStyleSheet("QGroupBox { font-weight: bold; }")
2714            mag_layout = QFormLayout()
2715            mag_layout.setSpacing(10)
2716
2717            self.valordeflujo_input = QDoubleSpinBox()
2718            self.valordeflujo_input.setRange(0.1, 10)
2719            self.valordeflujo_input.setValue(2.3)
2720            self.valordeflujo_input.setDecimals(1)
2721            self.valordeflujo_input.setSingleStep(0.1)
```

```
2722            self.valordeflujo_input.setMaximumWidth(120)
2723
2724            self.alpha_input = QDoubleSpinBox()
2725            self.alpha_input.setRange(12, 15)
2726            self.alpha_input.setValue(14)
2727            self.alpha_input.setDecimals(1)
2728            self.alpha_input.setSingleStep(0.5)
2729            self.alpha_input.setMaximumWidth(120)
2730
2731            mag_layout.addRow("Valor de flujo:", self.valordeflujo_input)
2732            mag_layout.addRow("ngulo ():", self.alpha_input)
2733            mag_group.setLayout(mag_layout)
2734
2735            # Grupo de parmetros geomtricos
2736            geom_group = QGroupBox("Parmetros Geomtricos")
2737            geom_group.setStyleSheet("QGroupBox { font-weight: bold; }")
2738            geom_layout = QFormLayout()
2739            geom_layout.setSpacing(10)
2740
2741            self.valordecircunferencia_input = QDoubleSpinBox()
2742            self.valordecircunferencia_input.setRange(0.1, 20)
2743            self.valordecircunferencia_input.setValue(7)
2744            self.valordecircunferencia_input.setDecimals(1)
2745            self.valordecircunferencia_input.setSingleStep(0.5)
2746            self.valordecircunferencia_input.setMaximumWidth(120)
2747
2748            self.opcion_circunferencia_combo = QComboBox()
2749            self.opcion_circunferencia_combo.addItems(["1 (0.1)", "2 (1)", "3 (10)", "4 (100)"])
2750            self.opcion_circunferencia_combo.setCurrentIndex(1)
2751            self.opcion_circunferencia_combo.setMaximumWidth(120)
2752
2753            self.vueltas_input = QSpinBox()
2754            self.vueltas_input.setRange(3, 20)
2755            self.vueltas_input.setValue(5)
2756            self.vueltas_input.setSingleStep(1)
2757            self.vueltas_input.setMaximumWidth(120)
2758
2759            geom_layout.addRow("Valor circunf:", self.valordecircunferencia_input)
2760            geom_layout.addRow("Escala circunf:", self.opcion_circunferencia_combo)
2761            geom_layout.addRow("N de vueltas:", self.vueltas_input)
2762            geom_group.setLayout(geom_layout)
2763
2764            # Grupo del potencimetro
2765            pot_group = QGroupBox("Potencimetro de Impedancia")
2766            pot_group.setStyleSheet("QGroupBox { font-weight: bold; }")
2767            pot_layout = QVBoxLayout()
2768
2769            self.pot_label = QLabel("Posicin: 0.5")
2770            self.pot_label.setAlignment(Qt.AlignCenter)
2771
2772            self.pot_slider = QSlider(Qt.Horizontal)
2773            self.pot_slider.setRange(0, 100)
2774            self.pot_slider.setValue(50)
2775            self.pot_slider.setTickPosition(QSlider.TicksBelow)
2776            self.pot_slider.setTickInterval(10)
2777            self.pot_slider.valueChanged.connect(self.update_pot_label)
2778
2779            pot_layout.addWidget(self.pot_label)
2780            pot_layout.addWidget(self.pot_slider)
2781            pot_group.setLayout(pot_layout)
2782
2783            # Grupo de parmetros de animacin
2784            anim_group = QGroupBox("Parmetros de Animacin")
2785            anim_group.setStyleSheet("QGroupBox { font-weight: bold; }")
2786            anim_layout = QFormLayout()
2787            anim_layout.setSpacing(10)
2788
2789            self.n_input = QSpinBox()
2790            self.n_input.setRange(100, 2000)
2791            self.n_input.setValue(500)
2792            self.n_input.setSingleStep(100)
2793            self.n_input.setMaximumWidth(120)
2794
2795            self.fps_input = QSpinBox()
2796            self.fps_input.setRange(1, 60)
2797            self.fps_input.setValue(20)
2798            self.fps_input.setMaximumWidth(120)
2799
2800            anim_layout.addRow("Puntos (n):", self.n_input)
2801            anim_layout.addRow("FPS:", self.fps_input)
2802            anim_group.setLayout(anim_layout)
2803
2804            # Frame para botones
2805            button_frame = QWidget()
2806            button_layout = QVBoxLayout()
2807            button_layout.setSpacing(10)
2808            button_frame.setLayout(button_layout)
2809
```

```
        # Botones - MODIFICADO: Agregado botn de simulacin
        self.calc_btn = QPushButton("Calcular Parmetros")
        self.calc_btn.setStyleSheet("""
            QPushButton {
                background-color: #4CAF50;
                color: white;
                font-weight: bold;
                padding: 10px;
                font-size: 11pt;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #45a049;
            }
        """)
        self.calc_btn.clicked.connect(self.calculate_parameters)

        self.simulate_btn = QPushButton("Ejecutar Simulacin")
        self.simulate_btn.setStyleSheet("""
            QPushButton {
                background-color: #FF9800;
                color: white;
                font-weight: bold;
                padding: 10px;
                font-size: 11pt;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #F57C00;
            }
        """)
        self.simulate_btn.clicked.connect(self.run_simulation)

        self.animate_btn = QPushButton("Generar Animacin")
        self.animate_btn.setStyleSheet("""
            QPushButton {
                background-color: #2196F3;
                color: white;
                font-weight: bold;
                padding: 10px;
                font-size: 11pt;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #0b7dda;
            }
        """)
        self.animate_btn.clicked.connect(self.generate_animation)

        self.clear_btn = QPushButton("Limpiar Grficos")
        self.clear_btn.setStyleSheet("""
            QPushButton {
                background-color: #f44336;
                color: white;
                font-weight: bold;
                padding: 10px;
                font-size: 11pt;
                border-radius: 5px;
            }
            QPushButton:hover {
                background-color: #d32f2f;
            }
        """)
        self.clear_btn.clicked.connect(self.clear_plot)

        # Aadir botones en orden lgico
        button_layout.addWidget(self.calc_btn)
        button_layout.addWidget(self.simulate_btn)
        button_layout.addWidget(self.animate_btn)
        button_layout.addWidget(self.clear_btn)

        # Aadir todos los grupos al layout izquierdo
        self.left_layout.addWidget(elec_group)
        self.left_layout.addWidget(escala_group)
        self.left_layout.addWidget(mag_group)
        self.left_layout.addWidget(geom_group)
        self.left_layout.addWidget(pot_group)
        self.left_layout.addWidget(anim_group)
        self.left_layout.addWidget(button_frame)
        self.left_layout.addStretch()

    def setup_plots(self):
        """Configura el rea de grficos"""
        self.figure = Figure(figsize=(10, 8), dpi=100)
        self.canvas = FigureCanvas(self.figure)
        self.ax = self.figure.add_subplot(111, projection='3d')

        # Configurar ejes
```

```
        self.ax.set_xlabel("X", fontsize=14, fontweight='bold')
        self.ax.set_ylabel("Y", fontsize=14, fontweight='bold')
        self.ax.set_zlabel("Z", fontsize=14, fontweight='bold')
        self.ax.set_title("Simulador de Antena Helicoide RLC", fontsize=16, fontweight='bold', pad=20)

        # Configurar ticks
        self.ax.tick_params(axis='both', which='major', labelsize=11)

        # Aadir canvas al layout derecho
        self.right_layout.addWidget(self.canvas)

    def update_pot_label(self):
        """Actualiza la etiqueta del potencimetro"""
        value = self.pot_slider.value() / 100.0
        self.pot_label.setText(f"Posicin: {value:.2f}")

    def get_simulation_parameters(self):
        """Obtiene los parmetros de simulacin"""
        params = {
            'V': self.V_input.value(),
            'I': self.I_input.value(),
            'opcion': self.opcion_combo.currentIndex() + 1,
            'factor': self.factor_input.value(),
            'valordeflujo': self.valordeflujo_input.value(),
            'alpha': self.alpha_input.value(),
            'valordecircunferencia': self.valordecircunferencia_input.value(),
            'opcion_circunferencia': self.opcion_circunferencia_combo.currentIndex() + 1,
            'vueltas': self.vueltas_input.value(),
            'potenciometro_valor': self.pot_slider.value() / 100.0,
            'n': self.n_input.value(),
            'fps': self.fps_input.value(),
            'Q': 1.2e-8  # Valor fijo
        }
        return params

    def power_pattern(self, altura_z, progress=1, params=None):
        """Calcula el patrn de radiacin"""
        if params is None:
            params = self.get_simulation_parameters()

        # Obtener parmetros necesarios
        Q = params['Q']
        V = params['V']
        I = params['I']
        opcion = params['opcion']
        valordeflujo = params['valordeflujo']
        alpha = params['alpha']
        valordecircunferencia = params['valordecircunferencia']
        opcion_circunferencia = params['opcion_circunferencia']
        vueltas = params['vueltas']
        potenciometro_valor = params['potenciometro_valor']
        factor = params['factor']

        # Funciones auxiliares
        def flujo(opcion):
            if opcion == 1:
                return 1e-11
            elif opcion == 2:
                return 1e-9
            elif opcion == 3:
                return 1e-7
            elif opcion == 4:
                return 1e-5

        def ordenc(opcion_circunferencia):
            if opcion_circunferencia == 1:
                return .1
            elif opcion_circunferencia == 2:
                return 1
            elif opcion_circunferencia == 3:
                return 10
            elif opcion_circunferencia == 4:
                return 100

        # Clculos
        Flujo = valordeflujo * flujo(opcion)
        capacitancia = Q / V
        inductancia = Flujo / I
        C = valordecircunferencia * ordenc(opcion_circunferencia)
        R = C / (2 * np.pi)
        S = C * np.tan(alpha * np.pi / 180)

        # Potencimetro
        Impedanciaminima = (2*np.pi * 3e8)**2 * (Q * Flujo) / ((C/.75)**2)
        Impedanciamaxima = (2*np.pi * 3e8)**2 * (Q * Flujo) / ((C/1.25)**2)
        impedancia_ajustada = Impedanciaminima + potenciometro_valor * (Impedanciamaxima - Impedanciaminima)
        corriente_ajustada = V / impedancia_ajustada
        inductancia_ajustada = Flujo / corriente_ajustada
```

```
        frecuencia_ajustada = 1 / (2 * np.pi * np.sqrt(inductancia_ajustada * capacitancia))
        longitud_onda_ajustada = 3e8 / frecuencia_ajustada
        S_lambda_ajustada = S / longitud_onda_ajustada

        # Calcular patrn
        theta_p = np.linspace(0, 2 * np.pi, 50)
        phi_p = np.linspace(0, np.pi, 50)
        theta_p, phi_p = np.meshgrid(theta_p, phi_p)

        psi = 2 * np.pi * (S_lambda_ajustada * (1 - np.cos(phi_p)) + (1/(2*vueltas)))

        E = (np.sin(np.pi/(2 * vueltas)) *
             (np.sin(vueltas * psi/2) / np.sin(psi/2 + 1e-9)) *
             np.cos(theta_p))

        E = np.abs(E) / np.max(np.abs(E))

        scale = 0.1 + 0.9 * progress
        Xp = factor * scale * E * np.sin(phi_p) * np.cos(theta_p)
        Yp = factor * scale * E * np.sin(phi_p) * np.sin(theta_p)
        Zp = factor * E * np.cos(phi_p) + altura_z

        return E, Xp, Yp, Zp

    def run_simulation(self):
        """Ejecuta la simulacin esttica (similar a ventana.py)"""
        try:
            params = self.get_simulation_parameters()

            # Obtener parmetros necesarios
            Q = params['Q']
            V = params['V']
            I = params['I']
            opcion = params['opcion']
            valordeflujo = params['valordeflujo']
            alpha = params['alpha']
            valordecircunferencia = params['valordecircunferencia']
            opcion_circunferencia = params['opcion_circunferencia']
            vueltas = params['vueltas']
            potenciometro_valor = params['potenciometro_valor']
            n = params['n']
            factor = params['factor']

            # Funciones auxiliares
            def flujo(opcion):
                if opcion == 1:
                    return 1e-11
                elif opcion == 2:
                    return 1e-9
                elif opcion == 3:
                    return 1e-7
                elif opcion == 4:
                    return 1e-5

            def ordenc(opcion_circunferencia):
                if opcion_circunferencia == 1:
                    return .1
                elif opcion_circunferencia == 2:
                    return 1
                elif opcion_circunferencia == 3:
                    return 10
                elif opcion_circunferencia == 4:
                    return 100

            # Clculos
            Flujo = valordeflujo * flujo(opcion)
            capacitancia = Q / V
            inductancia = Flujo / I
            C = valordecircunferencia * ordenc(opcion_circunferencia)
            R = C / (2 * np.pi)
            S = C * np.tan(alpha * np.pi / 180)
            L = np.sqrt((C**2) + (S**2))
            hmax = vueltas * S

            # Potencimetro
            Impedanciaminima = (2*np.pi * 3e8)**2 * (Q * Flujo) / ((C/.75)**2)
            Impedanciamaxima = (2*np.pi * 3e8)**2 * (Q * Flujo) / ((C/1.25)**2)
            impedancia_ajustada = Impedanciaminima + potenciometro_valor * (Impedanciamaxima - Impedanciaminima)
            corriente_ajustada = V / impedancia_ajustada
            inductancia_ajustada = Flujo / corriente_ajustada
            frecuencia_ajustada = 1 / (2 * np.pi * np.sqrt(inductancia_ajustada * capacitancia))
            longitud_onda_ajustada = 3e8 / frecuencia_ajustada
            S_lambda_ajustada = S / longitud_onda_ajustada

            # Trayectoria
            theta = np.linspace(0, vueltas * 2 * np.pi, n)
            z = (S / (2 * np.pi)) * theta
            x = R * np.cos(theta)
```

```
            y = R * np.sin(theta)

            # Superficie de la helicoide
            theta_surf = np.linspace(0, vueltas * 2 * np.pi, 50)
            r_surf = np.linspace(0, R, 50)
            theta_surf, r_surf = np.meshgrid(theta_surf, r_surf)
            x_surf = r_surf * np.cos(theta_surf)
            y_surf = r_surf * np.sin(theta_surf)
            z_surf = (hmax / (vueltas * 2 * np.pi)) * theta_surf

            self.clear_plot()

            self.ax.plot(x, y, z, color='black', linewidth=3, alpha=0.7, label='Trayectoria')

            # Dibujar punto inicial y final
            self.ax.scatter(x[0], y[0], z[0], c='green', s=100, edgecolors='black', linewidth=2, label='Inicio')
            self.ax.scatter(x[-1], y[-1], z[-1], c='red', s=100, edgecolors='black', linewidth=2, label='Fin')

            # Calcular y dibujar patrn de radiacin en posicin media
            mid_idx = n // 2
            U, Xp, Yp, Zp = self.power_pattern(z[mid_idx], 0.5, params)
            self.ax.plot_surface(
                Xp, Yp, Zp,
                facecolors=plt.cm.plasma(U),
                alpha=0.6,
                rstride=1, cstride=1
            )

            # Configurar lmites
            max_range = max(R * 1.2, hmax/2)
            self.ax.set_xlim(-max_range, max_range)
            self.ax.set_ylim(-max_range, max_range)
            self.ax.set_zlim(0, hmax * 1.1)

            # Actualizar ttulo
            self.ax.set_title(f"Simulacin Esttica - Antena Helicoide\nVueltas: {vueltas}, Frecuencia: {frecuencia_ajustada/1e6:.2f} MHz",
                              fontsize=14, fontweight='bold')

            # Aadir leyenda
            self.ax.legend(loc='upper right', fontsize=10)

            # Mostrar parmetros en el grfico
            info_text = f"R = {R:.2f}m\nS = {S:.2f}m\nhmax = {hmax:.2f}m\nf = {frecuencia_ajustada/1e6:.2f} MHz\nS/ = {S_lambda_ajustada:.3
        f}"
            self.ax.text2D(0.02, 0.98, info_text, transform=self.ax.transAxes,
                           fontsize=10, verticalalignment='top',
                           bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.8))

            self.canvas.draw()

        except Exception as e:
            self.show_error_message(f"Error en simulacin: {str(e)}")

    def calculate_parameters(self):
        """Calcula y muestra los parmetros de la antena"""
        try:
            params = self.get_simulation_parameters()

            # Extraer parmetros
            V = params['V']
            I = params['I']
            opcion = params['opcion']
            valordeflujo = params['valordeflujo']
            alpha = params['alpha']
            valordecircunferencia = params['valordecircunferencia']
            opcion_circunferencia = params['opcion_circunferencia']
            vueltas = params['vueltas']
            Q = params['Q']

            # Funciones auxiliares
            def flujo(opcion):
                if opcion == 1:
                    return 1e-11
                elif opcion == 2:
                    return 1e-9
                elif opcion == 3:
                    return 1e-7
                elif opcion == 4:
                    return 1e-5

            def ordenc(opcion_circunferencia):
                if opcion_circunferencia == 1:
                    return .1
                elif opcion_circunferencia == 2:
                    return 1
                elif opcion_circunferencia == 3:
                    return 10
                elif opcion_circunferencia == 4:
```

```
                    return 100

            # Clculos
            Impedancia = V / I
            Flujo = valordeflujo * flujo(opcion)
            capacitancia = Q / V
            inductancia = Flujo / I
            C = valordecircunferencia * ordenc(opcion_circunferencia)
            R = C / (2 * np.pi)
            S = C * np.tan(alpha * np.pi / 180)
            L = np.sqrt((C**2) + (S**2))
            hmax = vueltas * S

            # Frecuencia y longitud de onda
            frecuencia_antena = 1 / (2 * np.pi * np.sqrt(inductancia * capacitancia))
            longitud_onda = 3e8 / frecuencia_antena

            # Potencimetro
            Impedanciaminima = (2*np.pi * 3e8)**2 * (Q * Flujo) / ((C/.75)**2)
            Impedanciamaxima = (2*np.pi * 3e8)**2 * (Q * Flujo) / ((C/1.25)**2)
            impedancia_ajustada = Impedanciaminima + params['potenciometro_valor'] * (Impedanciamaxima - Impedanciaminima)
            corriente_ajustada = V / impedancia_ajustada

            # Parmetros ajustados
            inductancia_ajustada = Flujo / corriente_ajustada
            frecuencia_ajustada = 1 / (2 * np.pi * np.sqrt(inductancia_ajustada * capacitancia))
            longitud_onda_ajustada = 3e8 / frecuencia_ajustada
            S_lambda_ajustada = S / longitud_onda_ajustada
            C_lambda_ajustada = C / longitud_onda_ajustada

            # Mostrar resultados
            result_text = f"""
            PARMETROS CALCULADOS:

            Impedancia: {Impedancia:.2f}
            Flujo: {Flujo:.2e} Wb
            Capacitancia: {capacitancia:.2e} F
            Inductancia: {inductancia:.2e} H

            Geometra:
            Radio (R): {R:.3f} m
            Paso (S): {S:.3f} m
            Circunferencia (C): {C:.3f} m
            Altura total: {hmax:.3f} m

            Frecuencia original: {frecuencia_antena/1e6:.2f} MHz
            Longitud de onda original: {longitud_onda:.2f} m

            Con potencimetro:
            Impedancia ajustada: {impedancia_ajustada:.2f}
            Corriente ajustada: {corriente_ajustada:.3f} A
            Frecuencia ajustada: {frecuencia_ajustada/1e6:.2f} MHz
             ajustada: {longitud_onda_ajustada:.2f} m
            S/: {S_lambda_ajustada:.3f}
            C/: {C_lambda_ajustada:.3f}
            """

            QMessageBox.information(self, "Parmetros Calculados", result_text)

            # Actualizar grfico con geometra bsica
            self.clear_plot()

            # Trayectoria bsica
            theta = np.linspace(0, vueltas * 2 * np.pi, 200)
            z = (S / (2 * np.pi)) * theta
            x = R * np.cos(theta)
            y = R * np.sin(theta)

            # Dibujar helicoide
            self.ax.plot(x, y, z, color='black', linewidth=2, alpha=0.7, label='Helicoide')

            # Configurar lmites
            max_range = max(R * 1.5, hmax/2)
            self.ax.set_xlim([-max_range, max_range])
            self.ax.set_ylim([-max_range, max_range])
            self.ax.set_zlim([0, hmax * 1.2])

            self.ax.set_title(f"Antena Helicoide - {vueltas} vueltas\nRadio: {R:.2f}m, Altura: {hmax:.2f}m",
                              fontsize=14, fontweight='bold')
            self.ax.legend()
            self.canvas.draw()

        except Exception as e:
            self.show_error_message(f"Error en clculo: {str(e)}")

    def generate_animation(self):
        """Genera la animacin en un hilo separado"""
        try:
```

```
            params = self.get_simulation_parameters()

            # Deshabilitar botones durante la animacin
            self.animate_btn.setEnabled(False)
            self.calc_btn.setEnabled(False)
            self.simulate_btn.setEnabled(False)
            self.animate_btn.setText("Generando...")

            # Mostrar barra de progreso
            self.progress_bar.setVisible(True)
            self.progress_bar.setValue(0)

            # Crear y ejecutar hilo de animacin
            self.animation_thread = AnimationThreadV4(params)
            self.animation_thread.finished.connect(self.animation_finished)
            self.animation_thread.error.connect(self.animation_error)
            self.animation_thread.progress.connect(self.update_progress)
            self.animation_thread.start()

        except Exception as e:
            self.show_error_message(f"Error al iniciar animacin: {str(e)}")

    def update_progress(self, value):
        """Actualiza la barra de progreso"""
        self.progress_bar.setValue(value)

    def animation_finished(self, save_path):
        """Se llama cuando la animacin termina exitosamente"""
        self.animate_btn.setEnabled(True)
        self.calc_btn.setEnabled(True)
        self.simulate_btn.setEnabled(True)
        self.animate_btn.setText("Generar Animacin")
        self.progress_bar.setVisible(False)

        QMessageBox.information(self, "Animacin Completada",
                                f"Animacin guardada como:\n{save_path}")

    def animation_error(self, error_msg):
        """Se llama cuando hay un error en la animacin"""
        self.animate_btn.setEnabled(True)
        self.calc_btn.setEnabled(True)
        self.simulate_btn.setEnabled(True)
        self.animate_btn.setText("Generar Animacin")
        self.progress_bar.setVisible(False)

        self.show_error_message(f"Error en animacin: {error_msg}")

    def clear_plot(self):
        """Limpia el grfico"""
        self.ax.clear()
        self.ax.set_xlabel("X", fontsize=14, fontweight='bold')
        self.ax.set_ylabel("Y", fontsize=14, fontweight='bold')
        self.ax.set_zlabel("Z", fontsize=14, fontweight='bold')
        self.ax.set_title("Simulador de Antena Helicoide RLC", fontsize=16, fontweight='bold', pad=20)
        self.ax.tick_params(axis='both', which='major', labelsize=11)
        self.canvas.draw()

    def show_error_message(self, message):
        """Muestra un mensaje de error"""
        QMessageBox.critical(self, "Error", message)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    app.setStyle('Fusion')
    main_window = MainWindow()
    main_window.show()

    sys.exit(app.exec_())
```