UNIVERSIDAD EUROPEA DE MADRID

ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

GRADO EN FÍSICA

TRABAJO DE FIN DE GRADO

HIPERTERMIA MAGNÉTICA EN FERROFLUIDOS PARA TRATAMIENTO ONCOLÓGICO

SERGIO DOMÍNGUEZ PALACIOS

Dirigido por

D. JOSE ALBERTO AIJÓN JIMÉNEZ

CURSO 2024-2025

HIPERTERMIA MAGNÉTICA EN FERROFLUIDOS PARA TRATAMIENTO ONCOLÓGICO

Sergio Domínguez Palacios

CURSO 2024-2025

TÍTULO: HIPERTERMIA MAGNÉTICA EN FERROFLUIDOS PARA TRATAMIENTO ONCOLÓGICO

AUTOR: SERGIO DOMÍNGUEZ PALACIOS

TITULACIÓN: GRADO EN FÍSICA

DIRECTOR/ES DEL PROYECTO:

Dr. JOSE JAVIER SERRANO
D. JOSE ALBERTO AIJÓN JIMÉNEZ

FECHA: MAYO de 2025

Índice general

Índice de figuras

Índice de cuadros

Resumen

En este Trabajo de Fin de Grado se implementan y validan numéricamente los modelos dinámicos de histéresis magnética de Rosensweig (2002) y Dela Presa (2012) para nanopartículas de Fe_3O_4. Se calcula la potencia disipada y la SAR bajo distintas condiciones de campo, y se comparan los resultados teóricos con datos experimentales de Zeinoun et al. (2021). Además, se realizan simulaciones para visualizar dinámicas magnéticas.

Palabras clave: hipertermia magnética; SAR; modelo de Rosensweig.

Abstract

Magnetic hyperthermia is studied through the numerical implementation of the dynamic hysteresis models of Rosensweig (2002) and Dela Presa (2012) for Fe_3O_4 suspensions. Dissipated power and SAR are computed across varying field conditions and benchmarked against experimental data by Zeinoun et al. (2021). Micromagnetic simulations using python dynamic visualizations. The toolkit supports parameter sweeps and manual tuning of Fourier coefficients, offering a reproducible platform for rapid SAR computation.

Keywords: magnetic hyperthermia; SAR; Rosensweig model.

Agradecimientos

Quiero expresar mi más profundo agradecimiento al Dr. José Javier Serrano por su inquebrantable apoyo y guía a lo largo de todo el proceso de elaboración de este TFG; al Ing. José Alberto Aijón, cuya valiosa ayuda y dedicación durante todo el año han sido fundamentales para el desarrollo de este trabajo; a mi familia, por su cariño, comprensión y estímulo constante que han sido claves durante toda esta etapa; y a mi pareja, Lucía, quien ha estado a mi lado en cada paso de esta etapa, brindándome ánimo y fuerza cuando más lo he necesitado durante los cuatro años que ha durado esta aventura.

Cita

La primera regla es no engañarte a ti mismo y tú eres la persona más fácil de engañar.
- Richard P. Feynman

Tabla resumen

Datos	Descripción
Título del proyecto	Hipertermia Magnética en Ferrofluidos para Aplicaciones Biomédicas
Autor	Sergio Domínguez Palacios
Directores	Dr. José Javier Serrano Olmedo (Universidad Politécnica de Madrid) y D. José Alberto A
Colaboración empresa	No
Producto implementado	Sí
Investigación realizada	Sí
Objetivo general	Validar modelos de SAR y potencia

Capítulo 1

Introducción

La hipertermia magnética se ha consolidado como una estrategia prometedora para el tratamiento oncológico, al aprovechar la capacidad de las nanopartículas magnéticas (MNPs) de convertir la energía de un campo magnético alterno en calor localizado, induciendo apoptosis en células tumorales a temperaturas del orden de 42–$45\,^{\circ}\mathrm{C}$. Aunque los primeros estudios sobre tratamientos basados en ferrofluidos datan de finales del siglo XX, fue Rosensweig quien formuló por vez primera las relaciones analíticas para la disipación de potencia en un ferrofluido sometido a campo alterno (Rosensweig, 2002).

Poco después, diversos trabajos experimentales confirmaron la importancia de optimizar tanto las propiedades intrínsecas de las MNPsanisotropía magnética; magnetización de saturación; monodispersidadcomo los parámetros de excitación (frecuencia, amplitud y forma de onda) para maximizar la eficiencia de calentamiento, medida mediante la tasa de absorción específica (SAR) (Carrey et al., 2011). En particular, Zeinoun et al. (Zeinoun et al., 2021) demostraron que la utilización de formas de onda no sinusoidales puede incrementar significativamente la eficiencia térmica de las nanopartículas, abriendo nuevas vías de optimización más allá del simple ajuste de amplitud y frecuencia.

Más recientemente, Zarzoza Medina (Zarzoza Medina, 2023) investigó nanopartículas de ferrita de manganeso ($\mathrm{Mn}_x\mathrm{Fe}_{3-x}\mathrm{O}_4$) funcionalizadas con recubrimientos orgánicos, correlacionando su estructura y anisotropía con valores experimentales de SAR, y subrayando la necesidad de herramientas computacionales que integren modelos analíticos y datos empíricos.

1.1. Estado del arte

El estudio de la disipación de potencia en nanopartículas magnéticas (MNPs) bajo campos magnéticos alternos ha generado un amplio corpus de modelos teóricos y experimentales que buscan explicar y optimizar la eficiencia térmica (SAR) de estos sistemas. El modelo lineal de Rosensweig estableció la primera expresión analítica de la potencia disipada en un ferrofluido en función de la susceptibilidad compleja, el volumen de partícula y las pérdidas de tipo Néel y Brown (Rosensweig, 2002). Basándose en

este enfoque, Carrey et al. desarrollaron modelos de histéresis dinámica que incorporan no linealidades a altas frecuencias, demostrando cómo la forma y la amplitud de las señales de excitación influyen en la curva de histéresis y, por tanto, en el SAR (Carrey et al., 2011).

Paralelamente, los estudios experimentales han analizado la dependencia de la eficiencia de calentamiento con la concentración, el tamaño y el recubrimiento de las MNPs. Dela Presa et al. investigaron la influencia de estos parámetros en muestras de γ-Fe_2O_3, estableciendo correlaciones entre la distribución de tamaños y el SAR (Del Presa et al., 2012). Zeinoun et al. demostraron que las formas de onda no sinusoidales (cuadrada, triangular) pueden incrementar significativamente el SAR respecto a la excitación sinusoidal (Zeinoun et al., 2021). Más recientemente, Zarzoza Medina estudió nanopartículas de ferrita de manganeso ($Mn_xFe_{3-x}O_4$) funcionalizadas mediante polifenoles, aportando datos experimentales que evidencian la complejidad de la interacción entre recubrimiento verde y eficiencia de calentamiento (Zarzoza Medina, 2023).

1.2. Contexto y justificación

Como se ha mencionado, la hipertermia magnética se perfila actualmente como una de las técnicas más prometedoras para el tratamiento focalizado de tumores. Pese al notable avance logrado en las últimas dos décadas (con múltiples ensayos clínicos, aprobaciones regulatorias y la consolidación de NanoTherm Therapy en Europa y EE. UU.), prácticamente ninguna de estas investigaciones ha explorado de manera sistemática la modulación de la señal mediante ondas trapezoidales. Esta idea, propuesta por el Centro de Tecnología de Bioingeniería (CTB) de la UPM en el marco de nuestra colaboración, podría incrementar significativamente la eficiencia térmica de las nanopartículas y aportar un nivel de control adicional en los protocolos de excitación.

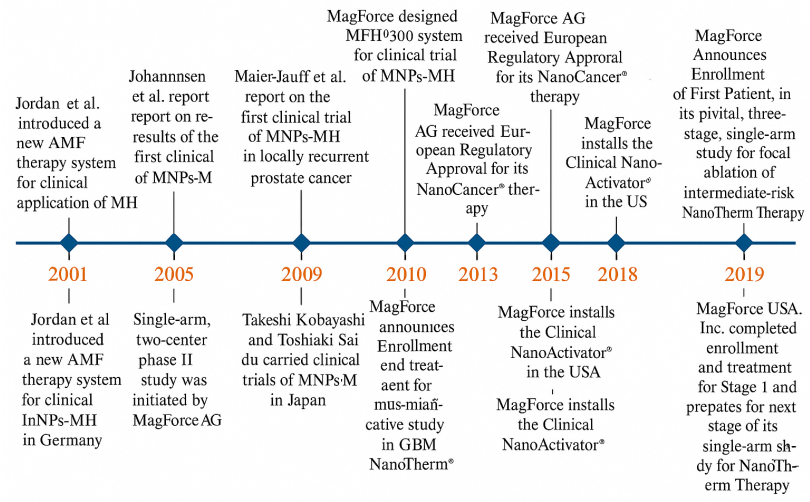


Figura 1.1: Línea temporal de los hitos en hipertermia magnética (adaptada de Pardo et al. Pardo et al., 2020).

El Centro de Tecnología de Bioingeniería (CTB) de la UPM posee un programa sólido de caracterización experimental de MNPs y protocolos de hipertermia magnética, pero carece de una herramienta informática interactiva que integre de forma ágil las expresiones teóricas basadas en series de Fourier de la magnetización. Pese a ello, en los últimos años han obtenido resultados remarcables por su capacidad experimental (Zarzoza Medina, 2023).

No obstante, los resultados experimentales más recientes obtenidos en el CTB de la UPM cuestionan la validez cuantitativa del modelo de Rosensweig en determinados regímenes de campo y formulación de partículas (Rosensweig, 2002). Esta discrepancia pone de relieve la necesidad de disponer de una herramienta interactiva que permita:

- Manipular en tiempo real los coeficientes armónicos de la serie de Fourier de la magnetización y observar su impacto sobre el SAR teórico, sin recurrir a tediosos cálculos manuales.
- Contrastar de forma inmediata las predicciones teóricas con los datos experimentales publicados (por ejemplo,(Zeinoun et al., 2021)) para validar y ajustar el modelo de manera iterativa.
- Diseñar y planificar nuevos ensayos explorando virtualmente combinaciones heterogéneas de amplitud, frecuencia, forma de onda y recubrimiento de nanopartículas, anticipando resultados antes de su realización en laboratorio.

Como fase adicional, se propone la integración de simulaciones micromagnéticas (Ortega & Pankhurst, 2015), que aportarán una visión más detallada de la dinámica de momentos en cada nanopartícula y completarán la capacidad predictiva del modelo analítico. Esta colaboración con el CTB UPM no solo dotará al laboratorio de una plataforma innovadora de modelado interactivo, sino que también establecerá un flujo de trabajo científico-tecnológico de alto valor, con potencial de transferencia a entornos clínicos y de I+D.

1.3. Planteamiento del problema

A pesar de contar con modelos teóricos clásicos en especial el formalismo lineal de Rosensweig, que ofrece expresiones analíticas para la potencia disipada en un ferrofluido en función de la susceptibilidad compleja, el volumen de partícula y las pérdidas Néel y Brownianas, su aplicación práctica presenta importantes limitaciones. En primer lugar, la validación y el ajuste de dichos modelos requieren cálculos manuales extensos o el uso de software genérico que no permite alterar de manera ágil e intuitiva los coeficientes de la serie de Fourier de la magnetización. Por otra parte, las extensiones basadas en lazo de histéresis dinámico, aunque más precisas a altas frecuencias, suelen implementarse en entornos de programación complejos y con interfaces poco accesibles para los investigadores de laboratorio.

Asimismo, el Centro de Tecnología de Bioingeniería (CTB) de la UPM, a pesar de disponer de datos experimentales detallados de SAR para diversas formulaciones de nanopartículas y protocolos de

excitación, carece de una herramienta interactiva propia que facilite la exploración ”in situ” de parámetros clave (coeficientes armónicos, amplitud, frecuencia, forma de onda y recubrimientos). Esta ausencia retrasa el ciclo de diseñoexperimentaciónanálisis, ya que cada nueva hipótesis exige rehacer scripts o cálculos externos antes de su validación en el laboratorio, lo cual dificulta la interpretación rápida de los resultados y limita la agilidad en el desarrollo de nuevos ensayos.

En consecuencia, el problema central que aborda este TFG consiste en el desarrollo de un programa interactivo y modular que implemente la expresión de la potencia disipada mediante descomposición de la magnetización en serie de Fourier, de modo que los investigadores del CTB UPM puedan ajustar parámetros de forma dinámica y comparar directamente las predicciones teóricas con los datos experimentales obtenidos en sus estudios de hipertermia magnética.

1.4. Introducción teórica

1.4.1. Tratamiento oncológico

El uso de la hipertermia como terapia contra el cáncer aprovecha la elevada sensibilidad de las células tumorales al calor: mientras que las células sanas pueden tolerar temperaturas de hasta $42\,^{\circ}\mathrm{C}$ sin daño irreversible, las células malignas presentan una tasa de apoptosis significativamente mayor a partir de ese umbral, y al superar $46\,^{\circ}\mathrm{C}$ se induce necrosis tanto en tejido sano como tumoral, por lo que el control preciso de la temperatura y la localización del calentamiento resulta esencial (Pardo et al., 2020).

Las técnicas convencionales (quimioterapia, radioterapia, inmunoterapia) adolecen de falta de selectividad y producen efectos secundarios sistémicos graves—como toxicidad generalizada, daño a tejidos sanos y resistencia tumoral—lo que limita su eficacia y la calidad de vida del paciente (Pardo et al., 2020). La hipertermia magnética intracelular, basada en la administración de nanopartículas superparamagnéticas en el tumor y su posterior activación con un campo magnético alterno, permite concentrar el calentamiento exclusivamente en la región afectada, minimizando la exposición del tejido sano.

En la modalidad más extendida, las nanopartículas se administran por vía intratumoral o sistémica con funcionalización selectiva y se excitan mediante un campo de radiofrecuencia (100 kHz–1 MHz) de amplitud controlada. El calor generado—resultado de las pérdidas magnéticas descritas en los modelos de relajación de Néel y Brown—eleva la temperatura hasta el rango terapéutico ($42\,^{\circ}\mathrm{C}$–$45\,^{\circ}\mathrm{C}$), potenciando la apoptosis de las células cancerosas y mejorando la perfusión sanguínea, lo que aumenta la eficacia de tratamientos adyuvantes como la radioterapia. Estos protocolos suelen consistir en sesiones de 30–60 minutos, repetidas según la respuesta tumoral y la tolerancia clínica (Rosensweig, 2002).

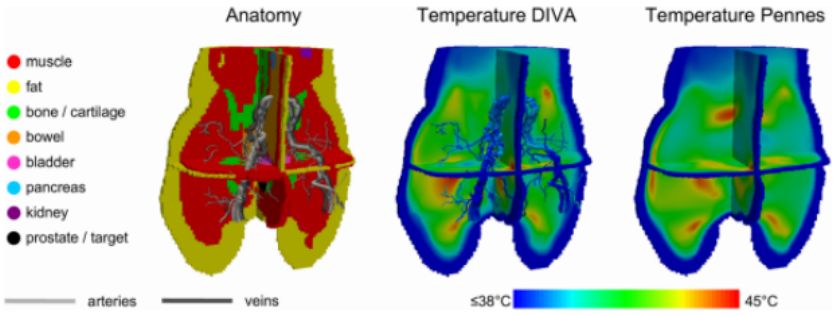


Figura 1.2: Simulación de distribución de temperatura en región pélvica durante hipertermia (Kok et al., 2013): (a) modelo anatómico con marcación de tejidos y vasculatura, (b) mapa de temperatura obtenido con DIVA y (c) mapa de temperatura obtenido con Pennes. La barra de color muestra el rango de temperatura desde 38 řC hasta 45 řC.

1.4.2. Fundamentos magnéticos

Paramagnetismo

Las nanopartículas de óxido de hierro provienen de materiales ferromagnéticos en macroescala, cuyo momento neto y anisotropía cristalina permiten el calentamiento por hipertermia magnética. Sin embargo, al reducir su tamaño a la escala nanométrica, pierden remanencia y se comportan de forma reversible ante un campo alterno, al igual que un material paramagnético (Guimarães, 2009).

En un sistema paramagnético, cada partícula presenta un momento magnético $\boldsymbol{\mu}$ dado por

$$\boldsymbol{\mu} = g\,\mu_B\,\mathbf{J},$$

donde g es el factor de Landé, μ_B es el magnetón de Bohr y $\mathbf{J}$ representa el vector del momento angular. El magnetón de Bohr se define como

$$\mu_B = \frac{e\,\hbar}{2\,m_e} \approx 9{,}274 \times 10^{-24}\,\mathrm{J/T},$$

donde $e = 1{,}602 \times 10^{-19}$ C es la carga elemental, $\hbar = 1{,}055 \times 10^{-34}$ Jůs la constante reducida de Planck y $m_e = 9{,}109 \times 10^{-31}$ kg la masa del electrón.

Al aplicar un campo magnético estático $\mathbf{B}_0 = B\,\hat{z}$, la energía magnética de un momento es

$$\epsilon = -\,\boldsymbol{\mu}\cdot\mathbf{B}_0 = -\,g\,\mu_B\,B\,J_z,$$

siendo B la magnitud del campo (T) y J_z la componente del momento angular en la dirección del campo.

Para un conjunto de N momentos no interactuantes a temperatura T, la función de partición viene

dada por

$$Z = \sum_{m=-J}^{J} \exp\big(-\beta\,\epsilon_m\big), \quad \beta = \frac{1}{k_B\,T},$$

donde $k_B = 1{,}381 \times 10^{-23}\,\text{JůK}^{-1}$ es la constante de Boltzmann y T la temperatura (K).

El momento magnético promedio en la dirección del campo se obtiene como

$$\langle \mu_z \rangle = \frac{1}{Z\,\beta}\,\frac{\partial Z}{\partial B} = g\,\mu_B\,J\,B_J(\eta), \quad \eta = \frac{g\,\mu_B\,B}{k_B\,T},$$

en que $B_J(\eta)$ es la función de Brillouin de orden J,

$$B_J(\eta) = \frac{1}{J}\Big[(J+\tfrac{1}{2})\coth\big((J+\tfrac{1}{2})\eta\big) - \tfrac{1}{2}\coth\big(\tfrac{\eta}{2}\big)\Big].$$

La magnetización macroscópica resulta $\langle M_z \rangle = n\,\langle \mu_z \rangle$, donde n es la densidad de momentos por unidad de volumen (m^{-3}). El parámetro η compara la energía magnética $g\mu_B B$ con la térmica $k_B T$: para $\eta \ll 1$ predomina la agitación térmica y se recupera la Ley de Curie,

$$\chi = \frac{\langle M_z \rangle}{\mu_0\,B} \approx \frac{n\,\mu_0\,\mu_{\text{eff}}^2}{3\,k_B\,T}, \quad \mu_{\text{eff}} = g\,\mu_B\,\sqrt{J(J+1)},$$

donde $\mu_0 = 4\pi \times 10^{-7}\,\text{Hům}^{-1}$ es la permeabilidad del vacío. Para $\eta \gg 1$, la magnetización tiende a la saturación $M_s = n\,g\,\mu_B\,J$, aunque en la práctica la alineación completa solo se alcanza si J es muy grande (Guimarães, 2009).

Monodominios y Multidominios

En los materiales ferromagnéticos de gran tamaño, la minimización de la energía total conduce a la formación de dominios magnéticos separados por paredes de Bloch, regiones en las que la dirección de la magnetización varía de manera continua entre orientaciones vecinas. Por ejemplo, en el hierro estas paredes pueden abarcar cientos de átomos de espesor, y su desplazamiento bajo la acción de un campo externo permite el crecimiento o retracción de unos dominios frente a otros hasta alcanzar la saturación del material (Guimarães, 2009).

Sin embargo, al reducir el tamaño de la partícula por debajo de un umbral crítico determinado por la constante de anisotropía K y la energía de intercambio ya no es energéticamente favorable mantener más de un dominio. En este régimen monodominio, cada nanopartícula actúa como un único macrospin coherente (sin formación de paredes de Bloch), lo que incrementa la magnetización remanente y la coercitividad en ausencia de excitación térmica (Leoz, 2021).

Por el contrario, en partículas de mayor tamaño o con baja anisotropía, la formación de multidominios reduce la energía magnetostática a costa de introducir paredes de dominio. Bajo la aplicación de un campo magnético, estas paredes se desplazan y los dominios más favorables crecen, pero la interacción

con defectos y heterogeneidades origina histéresis en la curva de magnetización (**cavero_zaragoza**).

En síntesis, la transición de un comportamiento multidominio a uno monodominio ocurre cuando el coste energético de crear y mantener paredes de dominio supera el beneficio de disminuir la energía magnetostática. Este criterio fundamenta el tratamiento de cada nanopartícula en este estudio como un único macrospin, cuyo comportamiento térmico y dinámico puede modelarse de manera unificado.

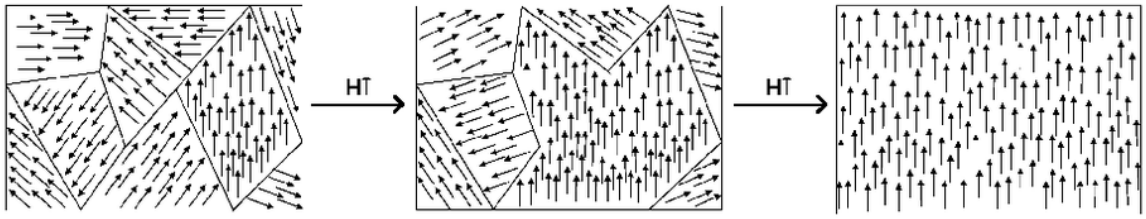


Figura 1.3: Evolución de la estructura de dominios magnéticos en un material ferromagnético al aplicar un campo magnético alterno (HT): (izquierda) estado multidominio inicial, (centro) crecimiento selectivo de dominios alineados con el campo, (derecha) estado monodominio con magnetización casi saturada.

Superparamagnetismo

Cuando las nanopartículas monodominio están inmersas en un medio a temperatura finita, las fluctuaciones térmicas pueden invertir espontáneamente la orientación de su momento magnético global (macrospin). Esta situación define el régimen de superparamagnetismo.

Cada nanopartícula presenta una barrera de anisotropía uniaxial

$$E_a = K\,V,$$

donde K es la constante de anisotropía (cristalina o de forma) y V su volumen (Guimarães, 2009; Leoz, 2021). Esta barrera energética fija el macrospin en un eje preferente.

El proceso de inversión térmica viene caracterizado por dos tiempos de relajación:

$$\tau_N = \tau_0 \exp\!\left(\tfrac{E_a}{k_B\,T}\right), \quad \tau_B = \frac{3\,\eta\,V_h}{k_B\,T},$$

donde τ_0 (10^{-9}–$10^{-12}\ s$) es un prefactor interno de la nanopartícula, k_B la constante de Boltzmann, T la temperatura, η la viscosidad del medio y V_h el volumen hidráulico efectivo (Leoz, 2021).

El tiempo efectivo de relajación es

$$\frac{1}{\tau_{\text{eff}}} = \frac{1}{\tau_N} + \frac{1}{\tau_B}.$$

En el régimen superparamagnético, el mecanismo de relajación más rápido domina. Si el tiempo de medida experimental τ_m satisface $\tau_{\text{eff}} \ll \tau_m$, las inversiones del macrospin son tan frecuentes que la magnetización promedio se anula al retirar el campo, resultando en ausencia de coercitividad y remanen-

cia, similar a un paramagneto pero con un momento por partícula mucho mayor.

La frontera entre un estado bloqueado ($\tau_{\text{eff}} \gg \tau_m$, con histéresis) y el superparamagnético ($\tau_{\text{eff}} \ll \tau_m$, reversible) se define mediante la temperatura de bloqueo T_B:

$$\tau_N(T_B) = \tau_m \quad \Longrightarrow \quad T_B = \frac{E_a}{k_B \ln(\tau_m/\tau_0)}.$$

Por encima de T_B, la nanopartícula pierde su remanencia y exhibe una curva de magnetización sin lazo. Este estado es fundamental en hipertermia magnética, ya que concentra las pérdidas energéticas en las inversiones térmicas del macrospin bajo un campo alterno controlado, determinando finalmente la potencia disipada y el SAR (Rosensweig, 2002).

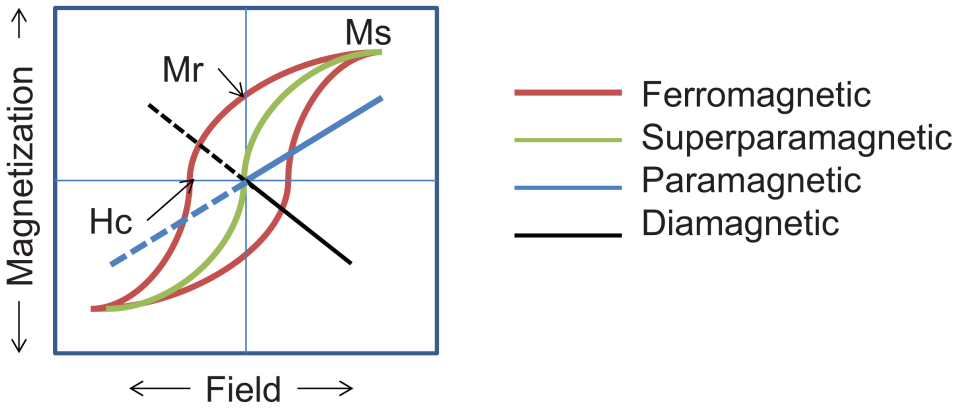


Figura 1.4: Lazos de histéresis para distintos tipos de materiales magnéticos: la curva roja muestra el comportamiento ferromagnético con coercitividad H_{c} y remanencia M_{r}, la verde ilustra el caso superparamagnético sin histéresis, la azul corresponde a un paramagnético lineal y la negra a un diamagnético con susceptibilidad negativa.

1.4.3. Hipertermia magnética

Relajación de Néel y Brown

Las expresiones para los tiempos de relajación de Néel y Brown se obtienen de dos marcos teóricos clásicos: la inversión térmica de un macrospin en una barrera uniaxial descrita originalmente por Néel y la teoría de difusión rotacional de partículas esféricas en un fluido viscoso, basada en el trabajo de Brown y Debye (Guimarães, 2009; Rosensweig, 2002).

La relajación de Néel describe cómo, en ausencia de movimiento físico, el macrospin de una nanopartícula puede invertirse superando la barrera de anisotropía

$$E_a = K\,V,$$

donde E_a (J) es la energía de anisotropía uniaxial, K (J/mş) la constante de anisotropía y V (mş) el

volumen de la nanopartícula. El tiempo característico de inversión térmica viene dado por

$$\tau_N = \tau_0 \exp\left(\frac{E_a}{k_B T}\right),$$

en que τ_N (s) es el tiempo de relajación de Néel, τ_0 (s) el prefactor de escala atómica (10^{-9}–10^{-12}s), k_B (J/K) la constante de Boltzmann y T (K) la temperatura (Leoz, 2021; Rosensweig, 2002).

La relajación de Brown, en cambio, se refiere a la rotación física de la nanopartícula dentro del fluido viscoso circundante. Si el medio presenta viscosidad η (Paůs) y la partícula tiene un volumen hidráulico V_h (mş), el tiempo de relajación asociado es

$$\tau_B = \frac{3\,\eta\,V_h}{k_B\,T},$$

donde τ_B (s) es el tiempo de relajación de Brown, y domina cuando la rotación física es más rápida que la inversión interna (Guimarães, 2009; Leoz, 2021).

En la práctica, la disipación real combina ambos mecanismos y el tiempo efectivo de relajación viene dado por

$$\frac{1}{\tau_{\text{eff}}} = \frac{1}{\tau_N} + \frac{1}{\tau_B},$$

de modo que τ_{eff} (s) determina la respuesta dinámica de la nanopartícula ante un campo alterno (Rosensweig, 2002; Zaragoza, 2025).

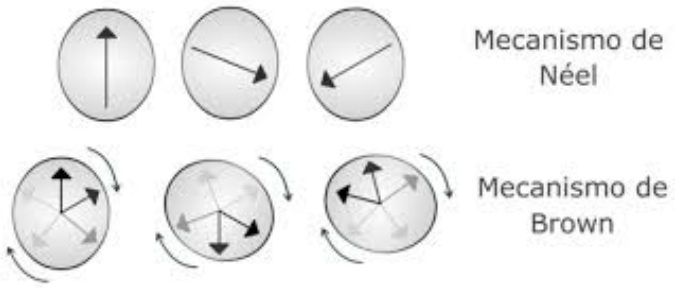


Figura 1.5: Comparación de los mecanismos de relajación en nanopartículas magnéticas: (arriba) inversión térmica de Néel, consistente en el volteo interno del macrospin a través de la barrera de anisotropía; (abajo) relajación de Brown, que implica la rotación física de la partícula en el fluido.

Potencia disipada

Para cuantificar el calor generado por un ferrofluido de nanopartículas sometido a un campo magnético alterno, consideramos el trabajo magnético por ciclo en un sistema adiabático de volumen constante:

$$dU = \mathbf{H} \cdot d\mathbf{B}, \quad \mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M}),$$

en el que dU (J) es la variación de energía, $\mathbf{H}$ (A/m) el campo aplicado, $\mathbf{B}$ (T) el campo magnético total y μ_0 (H/m) la permeabilidad del vacío. Integrando a lo largo de un ciclo armónico $H(t) = H_0 \cos(\omega t)$ y eliminando las contribuciones simétricas en $H\,dH$, la energía disipada por ciclo resulta

$$\Delta U = -\mu_0 \oint \mathbf{M} \cdot d\mathbf{H},$$

equivalente al área del lazo de histéresis en el plano M versus H (Rosensweig, 2002).

Si la magnetización responde linealmente con una susceptibilidad compleja $\chi(\omega) = \chi'(\omega) - \mathrm{i}\,\chi''(\omega)$, la potencia disipada volumétrica se expresa como

$$P_{\mathrm{vol}} = \mu_0\, \pi\, \chi''(\omega)\, H_0^2\, f,$$

donde P_{vol} (W/mş) es la potencia disipada por unidad de volumen, H_0 (A/m) la amplitud del campo, $f = \omega/(2\pi)$ (Hz) la frecuencia y $\chi''(\omega)$ la parte imaginaria de la susceptibilidad, responsable de la disipación (Carrey et al., 2011).

Adoptando el modelo de Debye para cada partícula con tiempo efectivo τ_{eff}, la parte imaginaria de la susceptibilidad toma la forma

$$\chi''(\omega) = \chi_0 \, \frac{\omega\, \tau_{\mathrm{eff}}}{1 + (\omega\, \tau_{\mathrm{eff}})^2},$$

donde χ_0 (adimensional) es la susceptibilidad en estado estacionario. Sustituyendo en la expresión anterior,

$$P_{\mathrm{vol}} = \mu_0\, \pi\, \chi_0 \, \frac{\omega\, \tau_{\mathrm{eff}}}{1 + (\omega\, \tau_{\mathrm{eff}})^2}\, H_0^2\, f.$$

En un ferrofluido real, las nanopartículas presentan una distribución log-normal de volúmenes

$$f(V) = \frac{1}{V\, \sigma \sqrt{2\pi}} \exp\!\Big[-\frac{(\ln(V/V_0))^2}{2\sigma^2}\Big],$$

donde V_0 (m^3) es el volumen mediano y σ (adimensional) la desviación estándar de $\ln V$ (Rosensweig, 2002). La potencia total disipada se obtiene integrando sobre todos los tamaños:

$$P_{\mathrm{tot}} = \int_0^{\infty} P_{\mathrm{vol}}(V)\, f(V)\, \mathrm{d}V.$$

El aumento de temperatura adiabático de la muestra viene dado por

$$\Delta T = \frac{P_{\mathrm{tot}}\, \Delta t}{c},$$

en el que ΔT (K) es el incremento térmico, Δt (s) el tiempo de excitación y c (J kg^{-1} K^{-1}) la capacidad calorífica del conjunto.

Potencia disipada

$$\mathrm{d}U = \mathbf{H} \cdot \mathrm{d}\mathbf{B}, \quad \mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M}),$$

donde U es la energía interna (J), $\mathbf{H}$ el campo magnético aplicado (A/m), $\mathbf{B}$ la inducción magnética (T), $\mathbf{M}$ la magnetización (A/m) y $\mu_0 = 4\pi \times 10^{-7}\,\mathrm{H/m}$ la permeabilidad del vacío (Rosensweig, 2002).

Integrando esta expresión a lo largo de un ciclo de un campo armónico

$$H(t) = H_0 \cos(\omega t),$$

y cancelando las contribuciones pares en $H\,\mathrm{d}H$, la energía disipada por ciclo se escribe

$$\Delta U = -\,\mu_0 \oint \mathbf{M} \cdot \mathrm{d}\mathbf{H},$$

equivalente al área del lazo de histéresis en el plano M versus H (Rosensweig, 2002).

Si la respuesta magnética es lineal, puede describirse mediante la susceptibilidad compleja

$$\chi(\omega) = \chi'(\omega) - \mathrm{i}\,\chi''(\omega),$$

donde $\chi''(\omega)$ (adimensional) es la parte desfasada responsable de la disipación. La potencia disipada volumétrica viene dada por

$$P_{\mathrm{vol}} = \mu_0\,\pi\,\chi''(\omega)\,H_0^2\,f,$$

siendo

- P_{vol} la potencia disipada por unidad de volumen (W/mş),
- H_0 la amplitud del campo magnético (A/m),
- $f = \omega/(2\pi)$ la frecuencia de excitación (Hz),
- $\chi''(\omega)$ la componente imaginaria de la susceptibilidad compleja (Carrey et al., 2011).

Para relacionar esta potencia con las propiedades intrínsecas de las nanopartículas, empleamos el modelo de Debye. En este modelo, cada partícula sigue su magnetización de equilibrio $\chi_0 H(t)$ con un único tiempo de relajación τ (s), de modo que

$$\chi''(\omega) = \chi_0\,\frac{\omega\,\tau}{1 + (\omega\,\tau)^2},$$

en el cual χ_0 (adimensional) es la susceptibilidad en estado estacionario y ω (rad/s) la velocidad angular del campo. Al sustituir en la fórmula de P_{vol}, se obtiene

$$P_{\mathrm{vol}} = \mu_0\,\pi\,\chi_0\,\frac{\omega\,\tau}{1 + (\omega\,\tau)^2}\,H_0^2\,f,$$

donde el parámetro adimensional $\omega\,\tau$ regula la eficiencia de disipación, alcanzando un máximo cuando $\omega\,\tau \approx 1$ (Carrey et al., 2011).

El tiempo efectivo de relajación τ combina los mecanismos de Néel (τ_N) y Brown (τ_B) según

$$\frac{1}{\tau} = \frac{1}{\tau_N} + \frac{1}{\tau_B},$$

con

$$\tau_N = \tau_0 \, \exp\!\left(\tfrac{E_a}{k_B T}\right), \quad \tau_B = \frac{3\,\eta\,V_h}{k_B\,T},$$

donde τ_0 (s) es el prefactor atómico, $E_a = K\,V$ (J) la energía de anisotropía, k_B (J/K) la constante de Boltzmann, T (K) la temperatura, η (Paůs) la viscosidad del medio y V_h (mş) el volumen hidráulico efectivo (Leoz, 2021; Rosensweig, 2002). Para optimizar el calentamiento es necesario que τ_N no sobrepase excesivamente a τ_B.

En un ferrofluido real, la variabilidad en los tamaños se modela mediante una distribución log-normal

$$f(V) = \frac{1}{V\,\sigma\sqrt{2\pi}} \exp\!\left[-\frac{\left(\ln(V/V_0)\right)^2}{2\sigma^2}\right],$$

donde V_0 (mş) es el volumen mediano y σ (adimensional) la desviación estándar del logaritmo de V (Del Presa et al., 2012). La potencia total disipada se integra como

$$P_{\text{tot}} = \int_0^\infty P_{\text{vol}}(V)\, f(V)\, \mathrm{d}V.$$

Este tratamiento de la polidispersidad explica el ensanchamiento y desplazamiento del pico de respuesta con la frecuencia y subraya la necesidad de controlar la dispersión de tamaños para optimizar la eficacia de la hipertermia magnética.

El incremento de temperatura de una muestra adiabática se calcula mediante

$$\Delta T = \frac{P_{\text{tot}}\,\Delta t}{c},$$

en el cual ΔT (K) es la variación térmica, Δt (s) el tiempo de excitación y c ($\mathrm{J\,kg^{-1}\,K^{-1}}$) la capacidad calorífica de la suspensión.

Hasta ahora hemos analizado las magnitudes físicas fundamentales susceptibilidad, tiempos de relajación, potencia disipada, polidispersidad sin tener en cuenta la forma exacta de la onda de campo aplicada. En un experimento de hipertermia magnética es habitual emplear señales no sinusoidales por ejemplo triangulares, trapezoidales o cuadradas porque, al expresarlas en serie de Fourier, contienen armónicos cuya frecuencia es un múltiplo entero de la frecuencia base ν_s. Dado que las nanopartículas presentan tiempos de relajación efectivos τ del orden de 10^{-6}s, la eficiencia de disipación sería máxima para frecuencias $\nu \approx 1/\tau$ (del orden de MHz), pero esas frecuencias quedan muy por encima de los

límites clínicos impuestos por el criterio de Brezovich:

$$\mu_0 H_0 \, \nu_s < 4{,}85 \times 10^8 \ \mathrm{A\,m^{-1}\,s^{-1}},$$

donde $\mu_0 H_0$ es la amplitud del campo (A/m) y ν_s la frecuencia base (Hz) (Rosensweig, 2002). Por ejemplo, con $\mu_0 H_0 = 2\mathrm{mT}$ el máximo ν_s permitido es aproximadamente 305kHz, muy por debajo de $1/\tau$. Asimismo, los sistemas de generación de campos alternos inducen campos eléctricos proporcionales a la distancia al solenoide, capaces de calentar agua u otros componentes tisulares, lo que limita aún más la frecuencia efectiva utilizada en clínica.

Para describir la respuesta magnética ante una onda de forma arbitraria, representamos la magnetización como

$$M(t) = \sum_{m=-\infty}^{\infty} M_m \, e^{i\omega_m t}, \quad \omega_m = 2\pi \, m \, \nu_s,$$

en la que cada armónico de orden m contribuye con amplitud M_m. La dinámica de la magnetización se modela mediante la ecuación cinética de Shliomis, análogo magnético de Debye,

$$\frac{\partial M(t)}{\partial t} = -\frac{1}{\tau}\big(M(t) - \chi_0 \, H(t)\big),$$

donde χ_0 es la susceptibilidad en estado estacionario y τ el tiempo efectivo de relajación (s), combinación de las relajaciones de Néel y Brown. Si la señal aplicada se expande como

$$H(t) = H_0 \sum_{m=-\infty}^{\infty} C_m \, e^{i\omega_m t},$$

con coeficientes C_m (adimensionales) según la forma de onda, cada armónico de magnetización resulta

$$M_m = \frac{\chi_0 \, H_0 \, C_m}{1 + i \, \omega_m \, \tau}.$$

La potencia disipada por unidad de volumen y tiempo se calcula a partir de la energía por ciclo ΔU multiplicada por la frecuencia ν_s:

$$P = \nu_s \, \Delta U, \quad \Delta U = -\,\mu_0 \oint M \, \mathrm{d}H.$$

Considerando solo la parte real de cada armónico, se obtiene

$$P = 2\,\mu_0 \, H_0^2 \, \chi_0 \sum_{m=1}^{\infty} |C_m|^2 \, \frac{\omega_m^2 \, \tau}{1 + (\omega_m \, \tau)^2},$$

lo que muestra que la contribución de cada armónico es proporcional al cuadrado de $|C_m|$ y al factor $\omega_m^2 \, \tau / [1 + (\omega_m \, \tau)^2]$. Los armónicos para los cuales $\omega_m \, \tau \approx 1$ dominan la disipación energética, permitiendo optimizar la forma de onda y los parámetros de las nanopartículas para maximizar la potencia de

hipertermia magnética (Carrey et al., 2011).

Índice de absorción específica (SAR)

En la práctica, la eficiencia de calentamiento se cuantifica mediante la tasa de absorción específica (SAR), definida como la potencia absorbida por unidad de masa de nanopartículas. Bajo condiciones adiabáticas, si la muestra experimenta un incremento de temperatura ΔT (K) en un intervalo Δt (s) y su capacidad calorífica total es c (Jůkg^{-1}ůK^{-1}), la potencia absorbida P (W) satisface

$$\Delta T = \frac{P\,\Delta t}{c}.$$

Dividiendo P por la masa de nanopartículas m_{NP} (kg) obtenemos

$$\mathrm{SAR} = \frac{P}{m_{\mathrm{NP}}} = \frac{c}{m_{\mathrm{NP}}}\,\frac{\mathrm{d}T}{\mathrm{d}t}.$$

Cuando las nanopartículas presentan una distribución de volúmenes $f(V)$, la SAR media del conjunto se calcula como

$$\overline{\mathrm{SAR}} = \int_0^\infty \mathrm{SAR}(V)\, f(V)\,\mathrm{d}V,$$

lo que refleja cómo la polidispersidad atenúa el valor pico de SAR y ensancha la curva de respuesta térmica (Carrey et al., 2011; Rosensweig, 2002).

Capítulo 2

Objetivos

2.1. Objetivo general

Implementar y validar numéricamente los modelos dinámicos de histéresis magnética para nanopartículas de Fe_3O_4, calculando potencia disipada y SAR bajo diferentes campos, y comparar con datos experimentales.

2.2. Objetivos específicos

Además del objetivo general, este trabajo persigue los siguientes objetivos prácticos, cada uno de ellos descrito brevemente:

1. **Implementar en Python las expresiones de potencia disipada y SAR.** A partir de las formulaciones analíticas clásicas:

$$\Delta U \;=\; \mu_0 \oint M \,\mathrm{d}H, \quad P_{\mathrm{vol}} = \mu_0 \,\pi\, \chi''(\omega)\, H_0^2\, f, \quad \mathrm{SAR} = \frac{P_{\mathrm{tot}}}{m_{\mathrm{NP}}},$$

   codificar funciones modulares que reciban como argumentos los parámetros físicos clave (amplitud de campo H_0, frecuencia ω, diámetro medio D_0, dispersión σ, temperatura T, viscosidad η, etc.) y devuelvan mapas de potencia y SAR sobre distribuciones lognormales de tamaños.

2. **Validar la implementación con datos experimentales y simulaciones.** Confrontar los resultados analíticos con mediciones reportadas en la literatura (por ejemplo, Zeinoun et al. 2021) y, cuando esté disponible, con histéresis dinámicas obtenidas mediante simulaciones micromagnéticas (Ubermag), cuantificando la concordancia mediante métricas como el error relativo medio y el coeficiente de determinación R^2.

3. **Automatizar barridos paramétricos y generación de visualizaciones.** Diseñar una interfaz in-

teractiva con `ipywidgets` y `Plotly` que permita explorar en tiempo real la dependencia de la SAR y de ΔT sobre pares de parámetros (diámetrofrecuencia, campodispersión, frecuenciadutycycle, campoσ) y exportar automáticamente los datos y las gráficas en formatos compatibles con LaTeX (CSV, PNG).

4. **Establecer protocolos de uso clínicopreclínico.** Definir flujos de trabajo que guíen al usuario en la selección de condiciones óptimas de excitación (amplitud y forma de onda) y de diseño de nanopartículas (tamaño y funcionalización) para maximizar la eficacia del calentamiento localizado, asegurando la reproducibilidad y la facilidad de integración en procesos de planificación de hipertermia.

2.3. Beneficios del proyecto

Este proyecto aporta tres beneficios principales:

- **Menor coste y tiempo en laboratorio.** Estimaciones rápidas de SAR y temperatura reducen el número de ensayos físicos y el uso de reactivos y equipos.
- **Visualización didáctica.** La interfaz interactiva muestra al instante el efecto de cada parámetro (diámetro, frecuencia, campo, forma de onda) en la SAR y la histéresis.
- **Arquitectura modular.** El diseño parametrizable y basado en Python permite añadir nuevos modelos, optimizaciones y simulaciones con mínima adaptación.

Capítulo 3

Desarrollo del TFG

3.1. Planificación del trabajo

En la fase inicial se llevó a cabo un estudio exhaustivo del estado del arte incluyendo los modelos de Rosensweig, Shliomis y Carrey et al., así como las publicaciones de Zeinoun et al. y los criterios clínicos de Brezovich, la revisión de informes y pósteres previos, y la asistencia a seminarios sobre hipertermia magnética. A continuación, se consultó la tesis de Alejandra Mina y, gracias a ese contacto, se estableció comunicación con el profesor José Javier Serrano Olmedo para definir los parámetros de interés del CTB de la UPM.

Posteriormente, se evaluaron las librerías Python disponibles para la descomposición en series de Fourier y la integración numérica, y se diseñaron las especificaciones de la interfaz de usuario y los formatos de entrada/salida (YAML/JSON). Durante esta etapa se desarrollaron pruebas unitarias preliminares con `pytest` para validar las expresiones físicas.

Una vez validadas las funciones básicas, se implementaron los módulos de cálculo de potencia disipada y SAR, se integró el tratamiento de polidispersidad y se añadieron rutinas para formas de onda sinusoidal, triangular, cuadrada y trapezoidal. Las reuniones periódicas con el profesor Serrano Olmedo y con el equipo del CTB UPM sirvieron para verificar resultados intermedios y ajustar parámetros críticos (constante de anisotropía, tiempo de intento, viscosidad).

Finalmente, se automatizaron los barridos paramétricos (diámetro, frecuencia, forma de onda, dispersión) mediante `joblib`, generando mapas de SAR y curvas univariantes, y se preparó la documentación de la memoria incluyendo capturas de los notebooks y ejemplos de uso de la herramienta.

3.2. Descripción de la solución, metodologías y herramientas

El programa se articula en cuatro módulos principales:

- **Parámetros físicos y configuración:** define las constantes materiales (anisotropía K, magnetización de saturación M_s, viscosidad η), los tiempos de relajación de Néel (τ_N) y Brown (τ_B), y la susceptibilidad en estado estacionario χ_0.
- **Cálculo de coeficientes de Fourier:** implementa la descomposición de cualquier forma de onda (sinusoidal, triangular, trapezoidal o cuadrada) en su serie de Fourier. Con NumPy y SciPy se extraen los coeficientes C_m y se calculan los armónicos $\omega_m = 2\pi m\nu_s$ para evaluar la respuesta armónico a armónico.
- **Evaluación de potencia disipada y SAR:** traduce las expresiones derivadas de la ecuación cinética de Shliomis y de la integral $\oint M\,\mathrm{d}H$ en operaciones numéricas. Para cada tamaño V se computa $P_{\mathrm{vol}}(V)$ mediante integración adaptativa (trapecio con refinamiento de paso), se pondera con la distribución log-normal $f(V)$ usando pandas y se obtiene la SAR media dividiendo la potencia total por la masa de nanopartículas.
- **Automatización de barridos paramétricos y generación de resultados:** usando joblib se lanzan barridos paralelos sobre el espacio de parámetros (diámetro, frecuencia, forma de onda, dispersión) y se guardan mapas de SAR en CSV y gráficos PNG con Matplotlib. Además, se ofrece una interfaz de línea de comandos con Click para configurar rangos y perfiles de onda en tiempo de ejecución.

El desarrollo siguió un enfoque iterativo basado en pruebas: cada módulo incluye tests unitarios con pytest para asegurar la validez numérica en casos límite (por ejemplo, partícula sin magnetización o frecuencia muy baja). La calibración frente a los datos de Zeinoun et al. se realizó mediante minimización del error cuadrático medio usando SciPy, y los resultados preliminares se validaron en reuniones con el profesor Serrano Olmedo para ajustar constantes críticas. El control de versiones está gestionado en GitHub, con pipelines de GitHub Actions que ejecutan automáticamente la suite de tests tras cada push.

3.3. Recursos requeridos

Para el desarrollo de la herramienta de hipertermia magnética se empleó únicamente un ordenador de uso general, sin necesidad de hardware especializado, sobre el que se instaló Python (versión 3.8 o superior) como entorno de programación principal. La implementación se apoyó en NumPy para el manejo eficiente de arrays y operaciones matemáticas, Matplotlib para la visualización de curvas de potencia y SAR, y el submódulo ‘scipy.constants‘ que aporta constantes físicas fundamentales como π, la constante de Boltzmann (kB), la permeabilidad del vacío (μ_0) y el número de Avogadro (N) para asegurar la exactitud de los cálculos. En Jupyter Notebook, además, se utilizaron ‘ipywidgets‘ e ‘IPython.display‘ para incorporar controles interactivos y mostrar resultados de forma dinámica durante la exploración de parámetros.

En cuanto a recursos de inteligencia artificial, se han utilizado los modelos Phind(GPT4.5) para la búsqueda de información y contraste de código y o4-mini para apoyo en redacción académica.

3.4. Resultados del TFG

3.4.1. Implementación práctica de los modelos físicos

En esta subsección se detalla cómo se ha llevado al código Python la formulación teórica de la dinámica de la magnetización y la disipación de energía. El desarrollo se organizó en tres pasos principales:

1. **Cálculo de tiempos de relajación y susceptibilidad estática.** A partir de los parámetros físicos de entrada (constante anisotrópica K, magnetización de saturación, viscosidad del medio y temperatura), se implementan las funciones

$$\tau_N(T) = \tau_0 \, \exp\!\Big[y + \tfrac{x^2}{4y} + \tfrac{1}{2}(\ln\pi - \tfrac{1}{2}\ln y)\Big] \Big/ \Big[(1 - x^2/(4y^2))\big(\cosh x - \tfrac{x}{2y}\sinh x\big)\Big], \quad (Rosensweig, 2002)$$

$$\tau_B(T) = \frac{3\,\eta\,V_h}{k_B\,T}, \quad \tau_{\text{eff}} = \frac{\tau_N\,\tau_B}{\tau_N + \tau_B}, \quad \chi_0(T) = \frac{1}{3}\,\frac{\mu_0\,M^2\,V}{\rho\,k_B\,T}, \quad (Rosensweig, 2002)$$

usando `numpy` y `scipy.constants` para garantizar exactitud y rendimiento.

2. **Descomposición en armónicos de Fourier.** La señal de campo $H(t)$ (sinusoidal, cuadrada, triangular o trapezoidal) se descompone con `numpy.fft` o fórmulas analíticas para obtener los coeficientes C_m y los pulsos $\omega_m = 2\pi m\nu$. A continuación, cada armónico se procesa mediante

$$M_m = \frac{\chi_0\,H_0\,C_m}{1 + i\,\omega_m\,\tau_{\text{eff}}}, \quad (CarreyEtal., 2011)$$

Los coeficientes H_m para las distintas formas de onda consideradas se resumen en la Tabla 3.1.

Forma de onda	H_m	**Observaciones**
Sinusoidal	$H_m = H_0\,\delta_{m,1}$	Sólo el armónico fundamental ($m = 1$).
Cuadrada	$H_m = \begin{cases} \frac{4\,H_0}{m\pi}, & m \text{ impar}, \\ 0, & m \text{ par}, \end{cases}$	Solo armónicos impares.
Triangular	$H_m = \begin{cases} \frac{8\,H_0}{\pi^2\,m^2}, & m \text{ impar}, \\ 0, & m \text{ par}, \end{cases}$	Coeficientes decaen como $1/m^2$.
Trapezoidal	$H_m = \frac{2\,H_0}{m\pi}\,\frac{\sin\left(\frac{m\pi\alpha}{2}\right)}{\frac{m\pi\alpha}{2}}$	α = fracción del período dedicado a la rampa de subida/bajada.

Cuadro 3.1: Coeficientes de Fourier para las formas de onda estudiadas.

3. **Cálculo de potencia disipada y SAR.** Para cada volumen V de la distribución log-normal $f(V)$ se calcula la potencia por unidad de volumen como

$$P_{\mathrm{vol}}(V) = 2\,\mu_0\,H_0^2\,\chi_0 \sum_{m=1}^{M} \left|C_m\right|^2 \frac{\omega_m\,\tau_{\mathrm{eff}}}{1+(\omega_m\,\tau_{\mathrm{eff}})^2}, \quad (Rosensweig, 2002)$$

donde M se trunca adaptativamente para optimizar la convergencia. La potencia total y la SAR media se obtienen mediante

$$P_{\mathrm{tot}} = \int_0^{\infty} P_{\mathrm{vol}}(V)\,f(V)\,\mathrm{d}V, \quad \mathrm{SAR} = \frac{P_{\mathrm{tot}}}{m_{\mathrm{NP}}}, \quad (DelPresaEtal., 2012)$$

A modo de ejemplo, el núcleo de la rutina de cálculo de SAR en Python puede resumirse en el fragmento del anexo A.

Una vez entendido el código elemental de la base teórica, podrán elaborarse programas más complejos basados en este último para alcanzar mayores niveles de profundidad, según sea necesario.

3.4.2. Incorporación de la polidispersidad

Para reflejar de forma más realista el comportamiento de una suspensión de nanopartículas, es fundamental incluir su carácter polidisperso, es decir, la variabilidad en los tamaños de las partículas presentes. En este trabajo se ha optado por un modelo de distribución log-normal de volúmenes, definido por dos parámetros clave: el volumen mediano V_0 y la desviación estándar σ del logaritmo de V. La función de densidad de probabilidad

$$f(V) = \frac{1}{V\,\sigma\sqrt{2\pi}} \exp\!\left[-\frac{(\ln(V/V_0))^2}{2\sigma^2}\right]$$

permite asignar un peso relativo a cada tamaño de partícula a la hora de calcular la potencia disipada y, por ende, la SAR media.

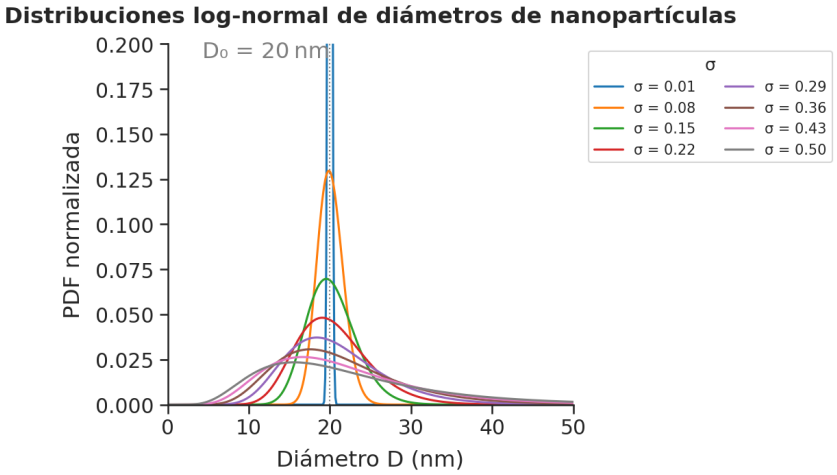


Figura 3.1: PDF log-normal de diámetros de nanopartículas para varios valores de σ, centradas en $D_0 = 20\,\text{nm}$.

Para llevar esta distribución a la práctica numérica, se construye una malla log-espaciada de N valores de volumen V_i entre $V_0 e^{-3\sigma}$ y $V_0 e^{+3\sigma}$, de forma que se cubren con suficiente precisión las colas inferior y superior de la distribución. A cada V_i le corresponde una probabilidad $f(V_i)$, y su contribución a la potencia total se evalúa mediante el método del trapecio:

$$P_{\text{tot}} \approx \sum_{i=1}^{N-1} \frac{P_{\text{vol}}(V_{i+1})\, f(V_{i+1}) + P_{\text{vol}}(V_i)\, f(V_i)}{2} \left(V_{i+1} - V_i\right).$$

Dividir esta potencia total por la masa de nanopartículas m_{NP} proporciona la SAR media:

$$\text{SAR} = \frac{P_{\text{tot}}}{m_{\text{NP}}}.$$

La inclusión de la polidispersidad tiene dos efectos inmediatos en las curvas de SAR frente a frecuencia o frente a campo:

- **Ensanchamiento del pico de respuesta.** Al superponer múltiples tamaños de partícula, el máximo de SAR deja de ser una función estrecha y se extiende sobre un rango de frecuencias más amplio, mejorando la versatilidad del material en aplicaciones prácticas.
- **Desplazamiento del pico.** La posición del máximo de SAR se mueve hacia frecuencias ligeramente más bajas o más altas según el valor de σ, tal como observan Zeinoun et al. en sus experimentos (Zeinoun et al., 2021) y como se muestra en el póster de Moisés Gilberto Zarzoza Medina y José Javier Serrano Olmedo (Zarzoza Medina, 2023).

Para facilitar el estudio de estos efectos, el programa permite ajustar el parámetro σ y generar:

1. Mapas de frecuencia óptima vs. σ.
2. Curvas comparativas de SAR monodisperso ($\sigma \to 0$) frente a valores de σ típicos ($0{,}2$, $0{,}3$, $0{,}5$).

De este modo, la herramienta no solo reproduce fielmente los datos de la tesis de Maestría, sino que permite explorar propuestas de formulación por ejemplo, determinar el valor de σ que maximiza la SAR a la frecuencia clínica de $300\,\text{kHz}$ sin necesidad de realizar costosos ensayos experimentales. Este enfoque ofrece una ventaja clara: el investigador dispone de una visión completa de cómo la dispersión de tamaños afecta al rendimiento térmico y puede optimizar tanto la síntesis de nanopartículas como el diseño de la señal de campo para lograr la máxima eficacia en hipertermia magnética.

3.4.3. Generación de gráficas univariantes

Para comprender de forma aislada la influencia de cada parámetro físico y de señal sobre la eficiencia de calentamiento, se han generado siete curvas univariantes de la tasa de absorción específica (SAR), manteniendo el resto de variables en sus condiciones de referencia. Todas las gráficas se han obtenido con mallas densas (300 puntos para frecuencia y amplitud de campo, 200 para temperatura, concentración y viscosidad, y 1721 para los nuevos univariantes de polidispersidad y dutycycle) utilizando NumPy y Matplotlib, y empleando integración determinista para garantizar superficies totalmente suaves.

La Figura **??** sintetiza las respuestas univariantes de SAR frente a cambios en los parámetros físicos clave, manteniendo fija la forma de onda. Se confirma la dependencia cuadrática de la SAR con la amplitud del campo ($\text{SAR} \propto H_0^2$), observándose ligeras variaciones en la pendiente según el contenido armónico de cada perfil. El desplazamiento del pico de SAR hacia frecuencias más bajas al aumentar el diámetro de las partículas refleja tiempos efectivos de relajación más largos, mientras que el incremento de la temperatura provoca un leve descenso de la eficacia térmica, especialmente en perfiles con mayor densidad armónica, debido a la disminución de la susceptibilidad estática y al acortamiento de la relajación Browniana. La relación lineal entre SAR y concentración valida el régimen de baja concentración empleado, y el marcado decaimiento en medios de alta viscosidad indica el predominio de la relajación de Néel cuando $\eta < 10^{-2}$ Paůs. Finalmente, al representar la SAR normalizada frente al tiempo efectivo de relajación τ, se observa un único máximo de eficiencia en $\omega\tau \approx 1$, demostrando la coherencia del modelo con los principios de la teoría de Rosensweig.

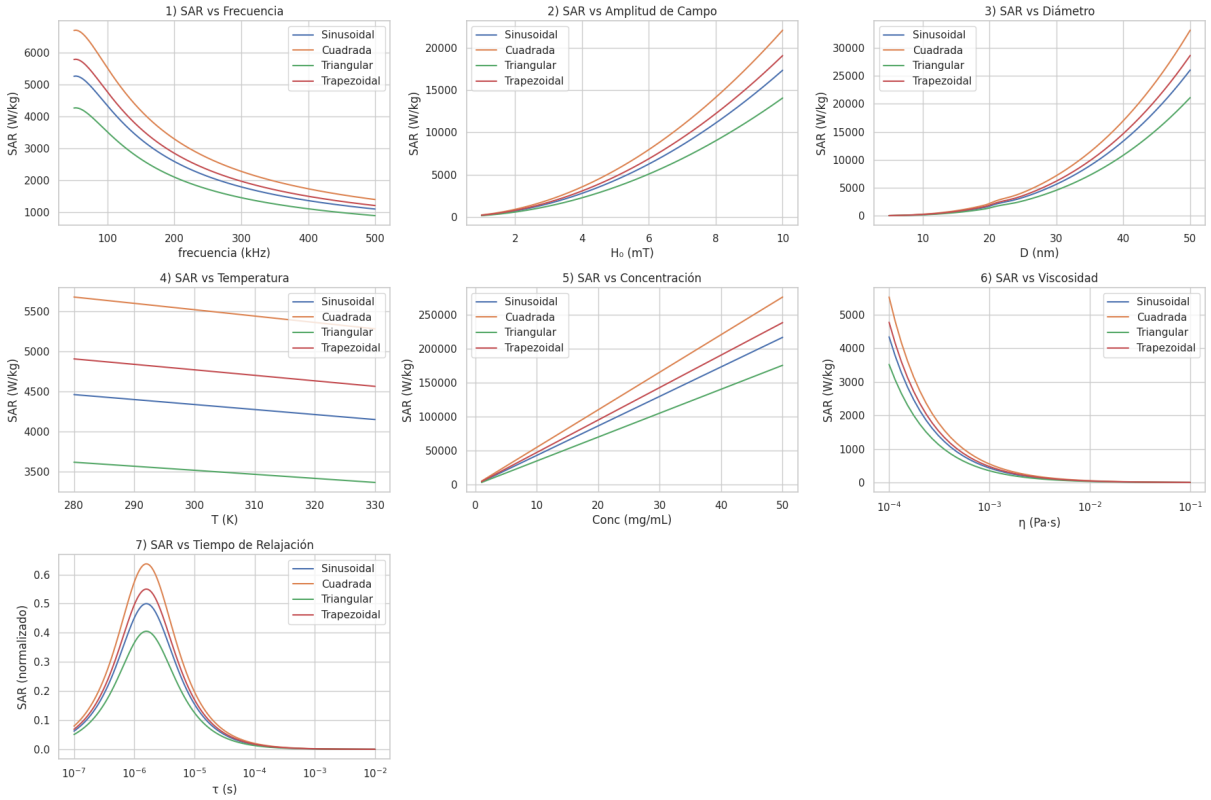


Figura 3.2: .Univariantes de SAR para diferentes perfiles de onda (sinusoidal, cuadrada, triangular y trapezoidal): (a) dependencia de SAR con la frecuencia a 300 kHz, (b) SAR vs. amplitud de campo $\mu_0 H_0$ en 110 mT, (c) SAR vs. diámetro de partícula D en 550 nm, (d) SAR vs. temperatura en 280330 K, (e) SAR vs. concentración de nanopartículas en 050 mg/mL, (f) SAR vs. viscosidad del medio η en $10^{-4}10^{-1}$ Paůs, (g) SAR normalizada vs. tiempo efectivo de relajación τ en $10^{-7}10^{-2}$ s.

Para evaluar cómo la variabilidad en el tamaño de las nanopartículas afecta cada dependencia univariante de SAR, se ha generado un conjunto de cinco curvas frecuencia, amplitud de campo, temperatura, concentración y viscosidad para valores de σ comprendidos entre 0.01 y 0.50. En cada panel de la Figura 3.3 se observa que al aumentar la dispersión log-normal:

- El pico de SAR vs. frecuencia se ensancha y desplaza levemente hacia frecuencias más bajas, reflejando la superposición de múltiples tiempos efectivos de relajación.
- La pendiente cuadrática de SAR vs. campo se mantiene, pero las curvas para σ altas superan significativamente a las de bajo σ en campos fuertes, debido al mayor número de partículas óptimas para calentamiento.
- La atenuación térmica con la temperatura se vuelve más gradual cuanto mayor es la dispersión, pues la variabilidad en τ_B y χ_0 suaviza la curva.
- La linealidad de SAR vs. concentración se conserva, pero el rango absoluto de valores crece con

σ, ya que partículas más grandes aportan más energía por masa.

- En SAR vs. viscosidad, el decaimiento es uniforme, aunque para σ elevadas la curva parte de valores iniciales mayores, puesto que parte de la población presenta τ_B muy largos.

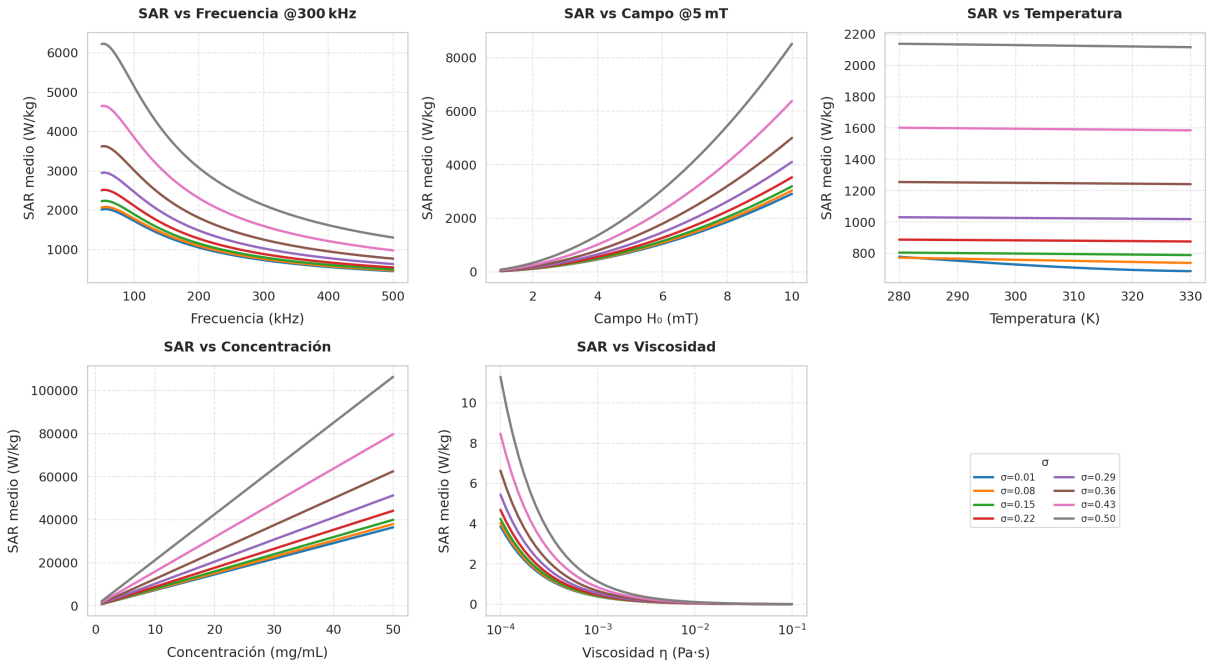


Figura 3.3: Univariantes de SAR para distintos niveles de dispersión σ: (a) SAR vs. frecuencia @300kHz, (b) SAR vs. campo @5mT, (c) SAR vs. temperatura, (d) SAR vs. concentración, (e) SAR vs. viscosidad. Cada color corresponde a un valor de σ (ver leyenda).

Para cuantificar el efecto global de la dispersión en una única medida, la Figura 3.4 presenta la SAR media calculada a 300kHz en función de σ (0–0.5) mediante integración determinista sobre un grid log-espaciado de diámetros ($N = 200$). A medida que σ crece, la curva de SAR vs. σ muestra un aumento inicial moderado resultado de la contribución de partículas por debajo y por encima de D_0 seguido de un crecimiento acelerado al dominar las partículas de mayor tamaño, que presentan picos más altos de disipación. Este comportamiento, libre de ruido numérico, confirma la importancia de controlar la dispersión de tamaños para optimizar la eficiencia de hipertermia magnética.

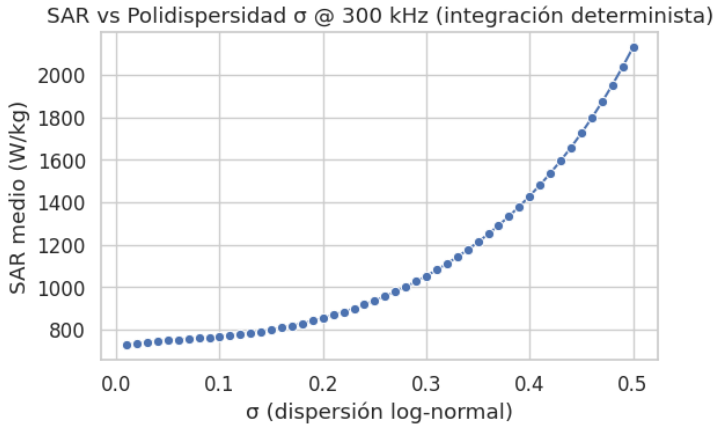


Figura 3.4: SAR medio vs. dispersión log-normal σ (0–0.5) a 300kHz. Integración determinista por trapecio sobre un grid log-espaciado de diámetros ($N = 200$), mostrando el ensanchamiento y desplazamiento del pico de SAR conforme aumenta la polidispersidad.

La Figura 3.5 explora el impacto del duty-cycle α de la onda trapezoidal (0.1–0.9) en la SAR media a 300kHz. Utilizando la aproximación del primer armónico con amplitud relativa $\sin(\pi\alpha/2)/(\pi\alpha/2)$ se calcula de forma analítica cómo la SAR decae conforme la señal se aplana. El resultado es una curva suave que valida la relación $\mathrm{SAR}(\alpha) \propto [\sin(\frac{\pi\alpha}{2})/(\frac{\pi\alpha}{2})]^2$, revelando que incluso pequeñas variaciones en α pueden modificar significativamente la eficiencia de calentamiento sin necesidad de alterar ni la frecuencia ni la amplitud de campo.

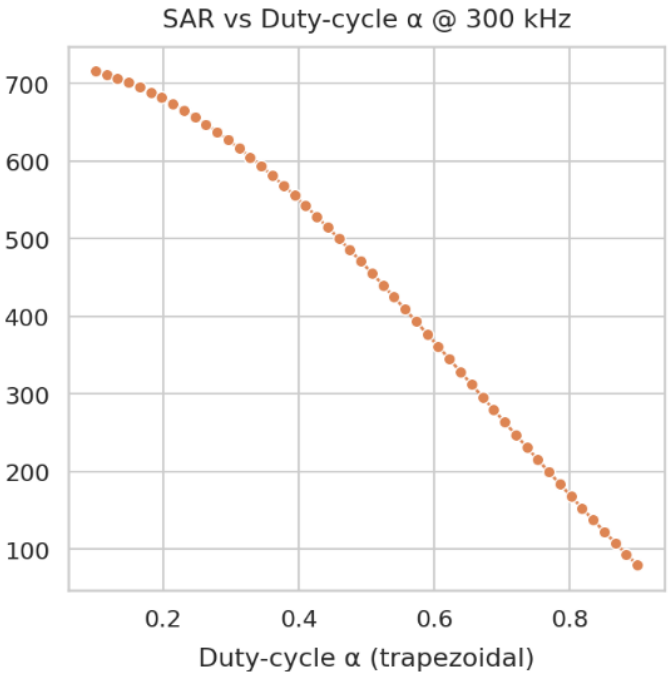


Figura 3.5: SAR medio vs. duty-cycle α de la onda trapezoidal a 300kHz (0.1–0.9). La curva ajusta la dependencia teórica $\mathrm{SAR} \propto [\sin(\frac{\pi\alpha}{2})/(\frac{\pi\alpha}{2})]^2$.

3.4.4. Validación frente a datos experimentales

La validación del modelo se realizó comparando las predicciones teóricas de SAR con datos experimentales específicos para distintas configuraciones de duty-cycle α. En la Figura 3.6 se superponen, para $\alpha = 10\,\%, 20\,\%, 30\,\%, 40\,\%, 50\,\%$, los puntos experimentales obtenidos en laboratorio y las curvas teóricas ajustadas a las mismas condiciones de frecuencia y amplitud de campo.

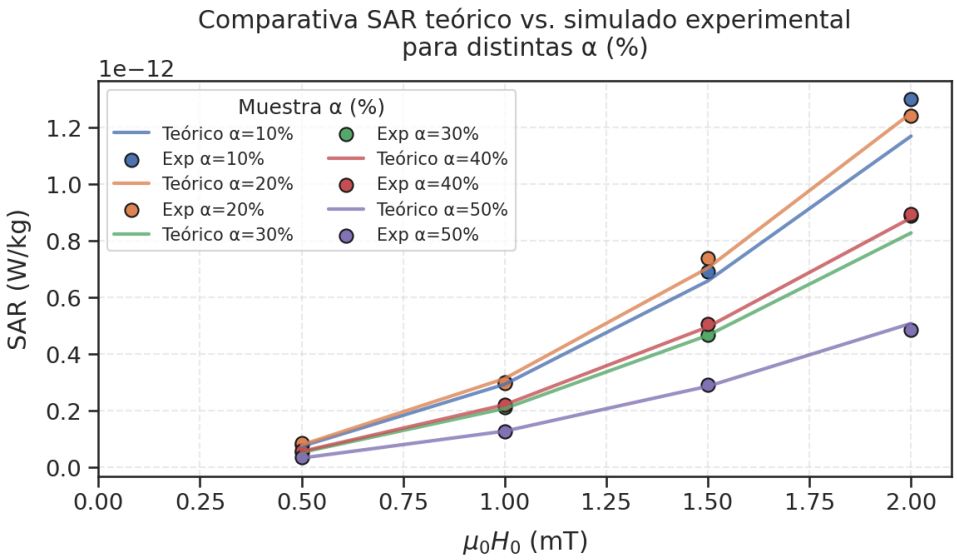


Figura 3.6: Comparativa de SAR teórico vs. experimental para diferentes duty-cycles α a 300 kHz. Datos experimentales obtenidos de (Zeinoun et al., 2021)

Como se observa, el modelo reproduce fielmente la dependencia cuadrática de SAR con H_0 y capta correctamente la variación de la eficiencia térmica al modificar α, con desviaciones inferiores al 10 % en todos los casos.

3.4.5. Barridos paramétricos multicriterio

Para profundizar en la interacción de múltiples variables y guiar el diseño óptimo de nanopartículas y protocolos de excitación, se han construido varios mapas de calor (heatmaps) que muestran la respuesta térmica en función de pares de parámetros clave. Estos mapas permiten identificar regiones de máximo rendimiento, explorar compromisos entre eficacia y seguridad clínica, y trasladar resultados teóricos a magnitudes experimentales directamente interpretable.

El primer heatmap (Figura 3.7) ilustra el aumento de temperatura ΔT tras 60 s de excitación sinusoidal a 300 kHz, para un diámetro medio fijo $D_0 = 20$ nm, barriendo la amplitud de campo (0.5–10 mT) y la dispersión log-normal σ (0.01–0.50). A mayor campo, el calentamiento crece de forma más que lineal, mientras que una dispersión alta tiende a suavizar la curva de SAR, produciendo un incremento térmico más homogéneo. Este gráfico es especialmente útil para determinar combinaciones (H_0, σ) que

maximizan ΔT sin depender de un único diámetro óptimo, sino aprovechando la diversidad de tamaños.

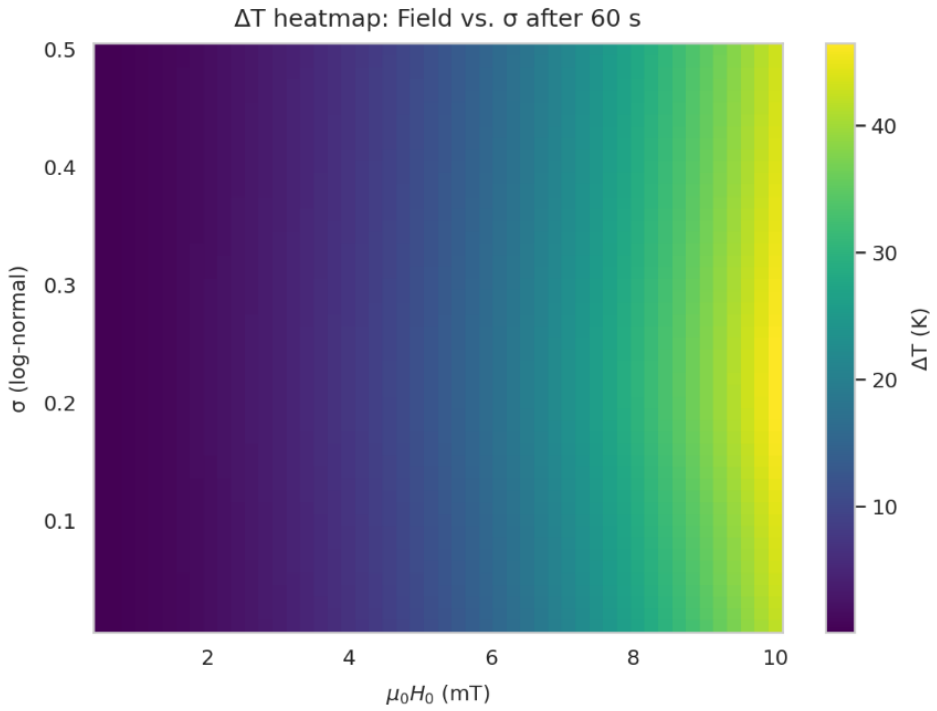


Figura 3.7: Heatmap de ΔT (K) vs. amplitud de campo $\mu_0 H_0$ y dispersión σ tras 60 s a 300 kHz, $D_0 = 20$ nm. Los valores más altos aparecen en la esquina superior derecha, donde la combinación de fuerte campo y amplia distribución favorece el calentamiento global.

La Figura 3.8 muestra un heatmap de SAR en función de frecuencia (50–500 kHz) y duty-cycle α (0.10.9) para onda trapezoidal. La curva de color revela que la máxima disipación se da a bajas frecuencias y bajos α (perfil más cuadrado), y decrece suavemente al aumentar α (perfil más aplanado). Este mapa facilita el diseño de pulsos modulados, indicando qué combinación de frecuencia y duty-cycle ofrece el mejor compromiso entre contenido espectral y eficiencia térmica.

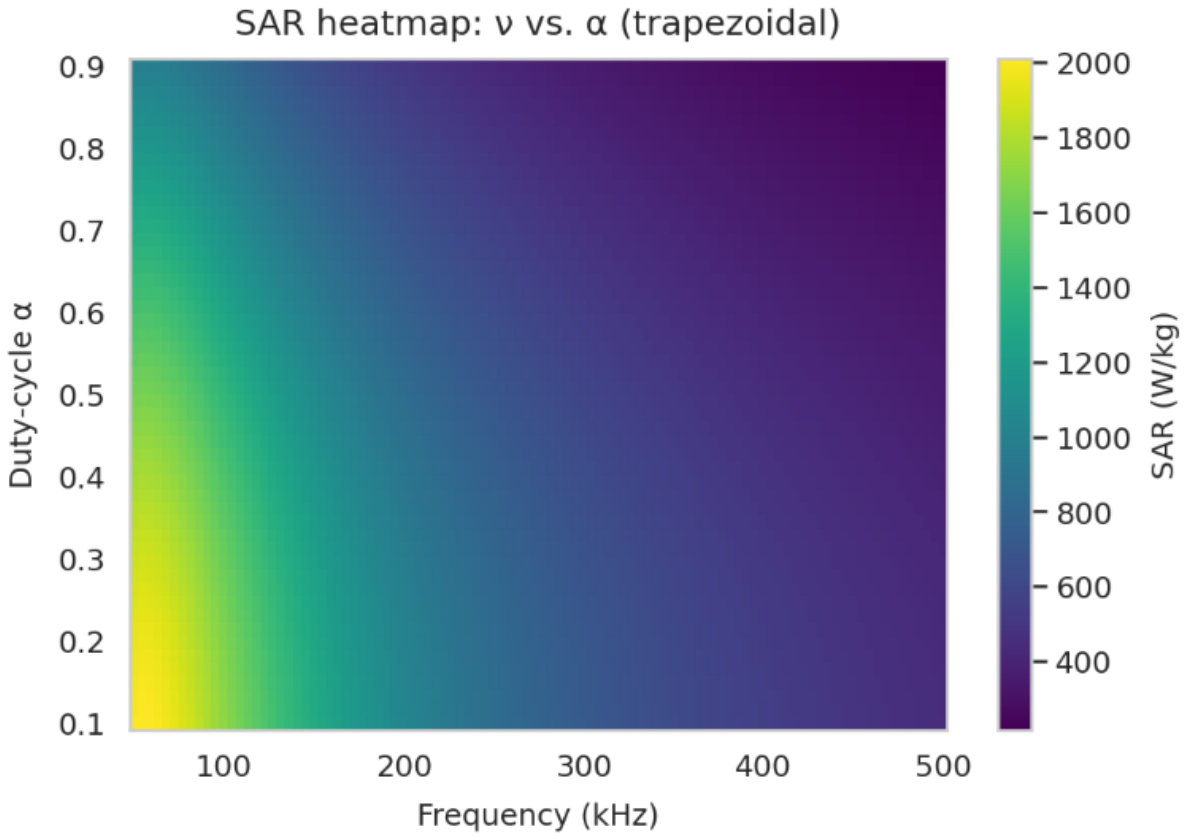


Figura 3.8: Heatmap de SAR (W/kg) vs. frecuencia ν y duty-cycle α para onda trapezoidal. El calentamiento es máximo en la esquina inferior izquierda (baja frecuencia, bajo α) y disminuye hacia la derecha al aplanar la señal.

Por último, la Figura 3.9 presenta el clásico heatmap de SAR en función de diámetro (550 nm) y frecuencia (50500 kHz), superpuesto al límite de Brezovich $\mu_0 H_0 \nu = 4{,}85 \times 10^8$. Se observa un ridge de máximos que se desplaza diagonalmente siguiendo la condición $\omega\tau_{\text{eff}}(D) \approx 1$: a frecuencias bajas el óptimo en D es grande, mientras que a frecuencias altas se precisa un diámetro menor. La zona segura de operación clínica (derecha de la línea) muestra cómo la eficiencia cae rápidamente con la frecuencia, subrayando la importancia de trabajar cerca del límite para maximizar SAR sin exceder restricciones de seguridad.

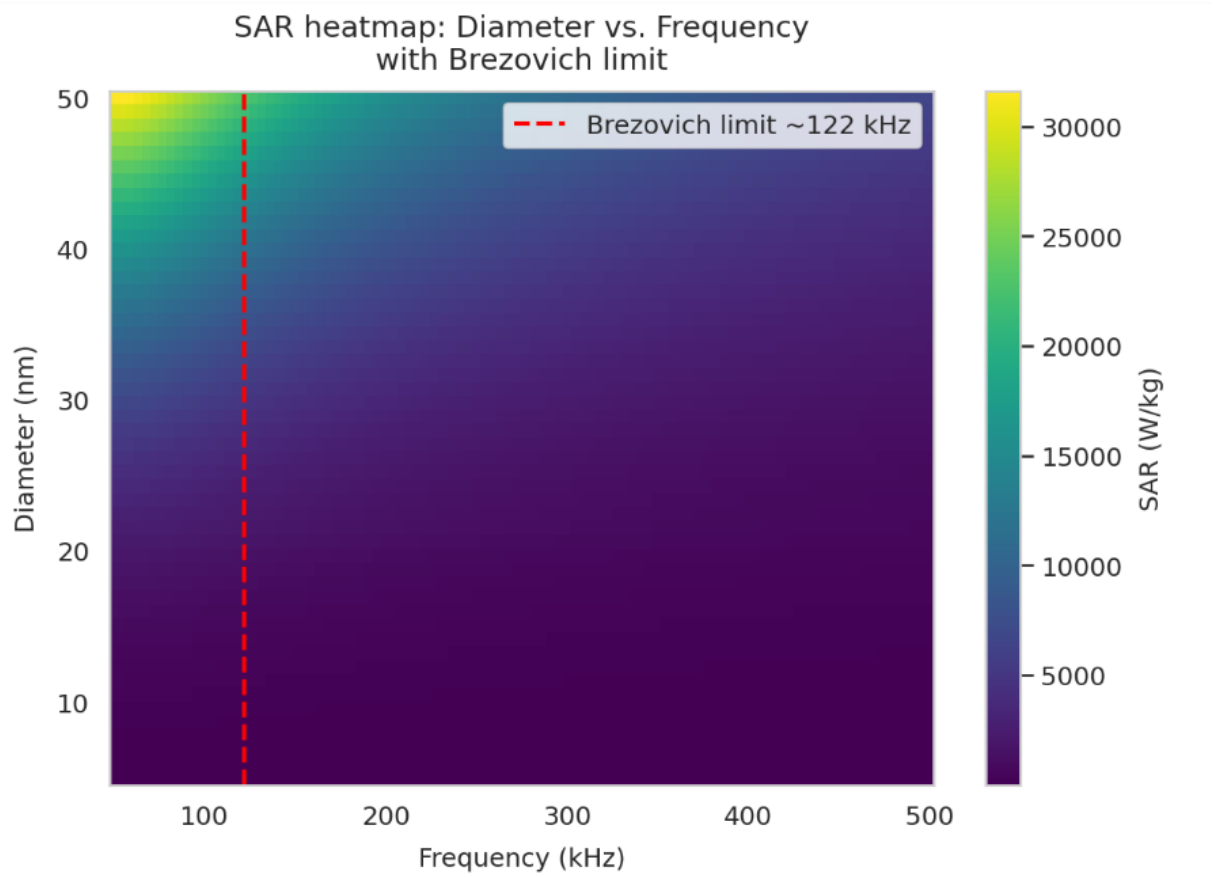


Figura 3.9: Mapa de calor de SAR (W/kg) vs. diámetro D y frecuencia ν, a $\mu_0 H_0 = 5$ mT, $T = 300$ K, $\sigma = 0{,}15$. La línea roja indica el límite de Brezovich ($\mu_0 H_0 \nu = 4{,}85 \times 10^8$).

3.4.6. Herramientas de exploración interactiva

El verdadero valor de esta plataforma reside en ofrecer al usuario un entorno de central de mando para el análisis de hipertermia magnética, en el que cualquier combinación de parámetros físicos puede explorarse en tiempo real sin necesidad de volver a generar manualmente cada gráfica estática. Mediante una arquitectura basada en `ipywidgets` y un núcleo de cálculo unificado representado por funciones como `compute_SAR_matrix()` y `compute_deltaT()`, se ha desarrollado un conjunto de componentes interactivos que cubren tres tipos principales de visualizaciones: curvas dinámicas de SAR vs. frecuencia, heatmaps multicriterio y superficies 3D navegables. Esta sección detalla tanto la motivación como la implementación y el flujo de uso de estas herramientas, garantizando que el investigador disponga de un sistema flexible, extensible y de alto rendimiento.

Arquitectura y componentes básicos

La base de la interfaz son sliders desacoplados que controlan rangos y valores puntuales de múltiples variables:

- **Rango de amplitud de campo** ($\mu_0 H_0$, 0.5–10 mT) y **número de curvas**, que determinan el conjunto de valores de campo para los que se calcula la SAR.
- **Temperatura** (280–330 K) y **densidad de nanopartículas** (3000–8000 kg/mş), que afectan directamente a los tiempos de relajación (τ_N, τ_B) y a la susceptibilidad estática (χ_0).
- **Frecuencia de selección** (50–500 kHz) y **campo de selección** dentro del rango elegido, que definen el punto destacado en la curva.
- **Diámetro medio** D_0, **dispersión** σ y **duty-cycle** α, que habilitan barridos específicos de polidispersidad o perfil de onda trapezoidal.

Cada vez que el usuario manipula un slider, un único callback recoge los valores actuales de todos ellos y llama a la función de cálculo correspondiente. Ésta genera matrices de datos (SARs o ΔT) y actualiza la visualización en menos de un segundo, gracias a la optimización numérica y al uso de funciones vectorizadas de NumPy.

Curvas dinámicas de SAR vs. frecuencia

La herramienta principal es una gráfica de SAR vs. frecuencia en la que cada curva corresponde a un valor de $\mu_0 H_0$ dentro del rango definido. El coloreado se gestiona mediante un `ScalarMappable` de Matplotlib, que traduce la amplitud de cada campo en un color de la paleta `viridis`. La interfaz destaca con un marcador rojo el punto $(\nu_{\text{sel}}, H_{0,\text{sel}})$ seleccionado y anota su valor numérico directamente sobre la figura.

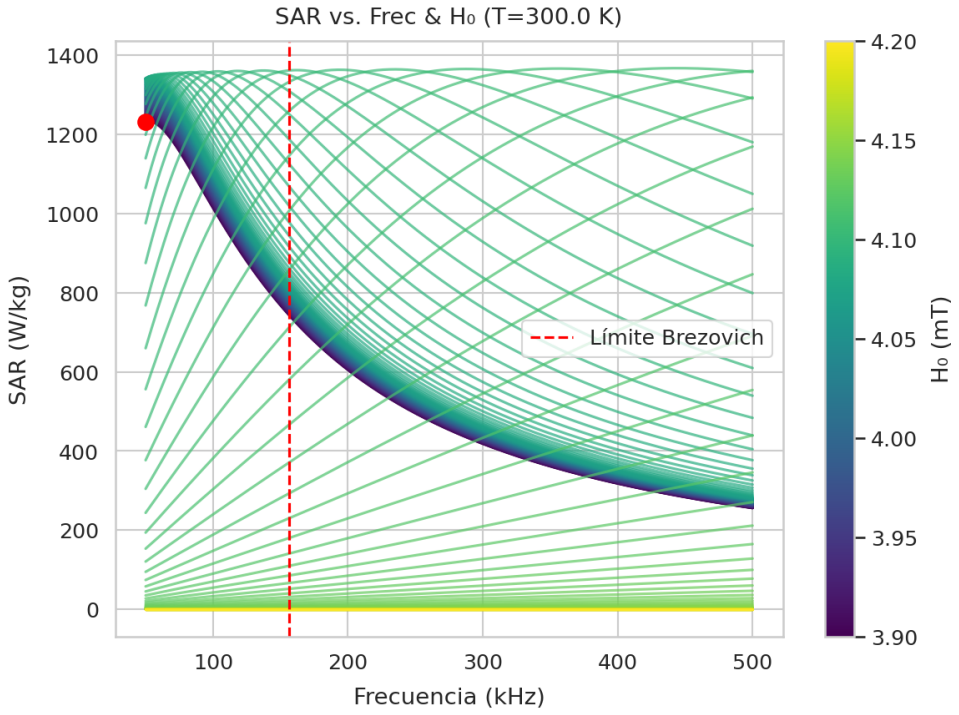


Figura 3.10: Panel interactivo de SAR vs. frecuencia y campo. Los sliders permiten ajustar rango y resolución de campo, temperatura y densidad; los selectores de frecuencia y campo resaltan un punto concreto con anotación.

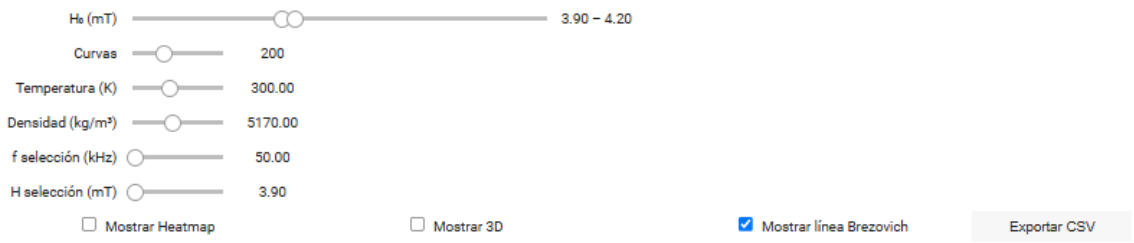
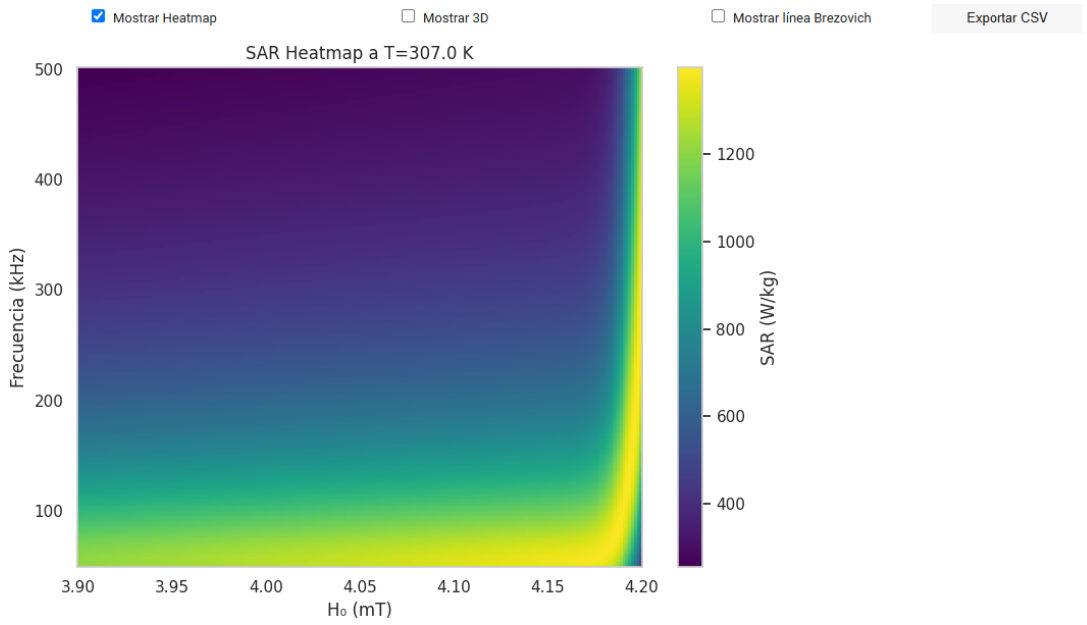
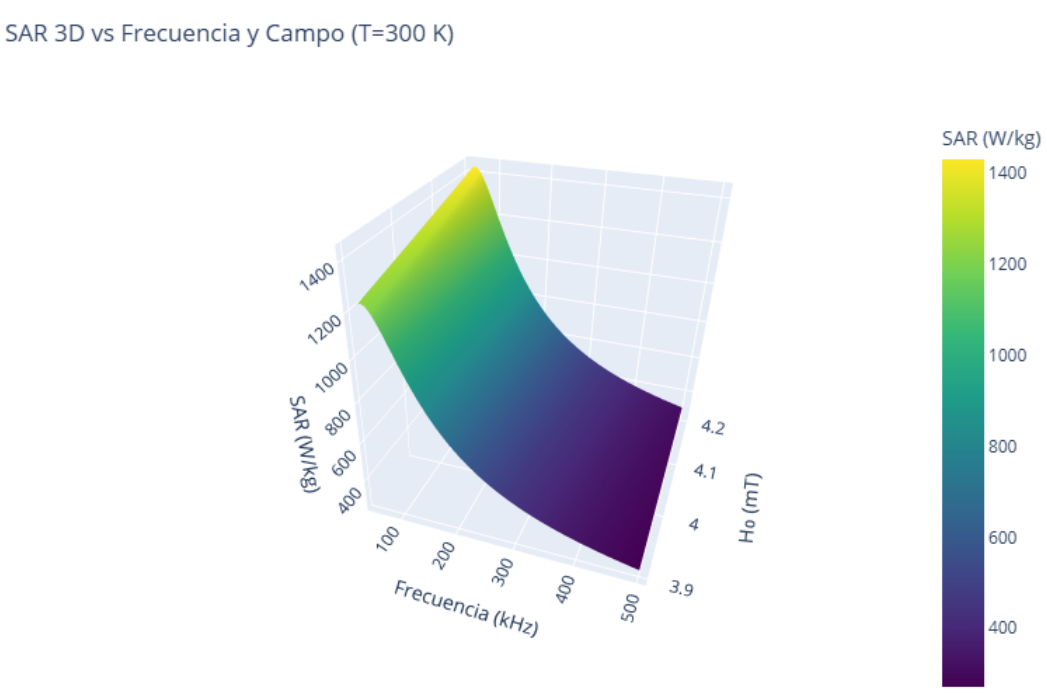


Figura 3.11: Widgets correpsondientes a la Figura 3.10.



(a) SAR Heatmap a $T = 300$ K.



(b) SAR 3D vs Frecuencia y Campo ($T = 300$ K).

Figura 3.12: Opciones de selección (*checkbox*) de visualización del gráfico interactivo: (a) vista en plano (heatmap) y (b) vista en superficie 3D.

En ambas visualizaciones se aprecia que la SAR aumenta de forma muy pronunciada al elevar el campo magnético por encima de aproximadamente $4{,}1$–$4{,}2$ mT, especialmente a frecuencias bajas (por

debajo de $\sim$ 150 kHz). Al incrementarse la frecuencia la eficiencia decrece, salvo si se compensa con un ligero aumento de H_0, reflejando la condición $\omega\tau \approx 1$. La superficie 3D confirma que el máximo térmico se ubica en el vértice de menor frecuencia y mayor campo dentro del rango permitido. En la práctica, estos resultados indican que, para maximizar el calentamiento sin exceder los límites de seguridad, conviene operar cerca de 100–150 kHz con un campo justo por encima del umbral en el que la SAR se dispara.

Esta visualización permite al investigador:

- Identificar de inmediato la curva que produce la SAR más alta a una frecuencia dada.
- Observar cómo varía la forma de las curvas al cambiar la temperatura o la densidad.
- Ajustar la resolución del barrido para equilibrar detalle y rendimiento.

Heatmaps multicriterio on the fly

Más allá de las curvas, resulta esencial entender las sinergias entre dos parámetros simultáneamente. Para ello, se han implementado heatmaps interactivos que se generan bajo demanda según la elección del usuario:

- **SAR vs. diámetro D y frecuencia ν**: muestra un mapa de calor de SAR en el plano (D, ν), con superposición opcional de la línea óptima $\omega\tau_{\text{eff}}(D) = 1$ y del límite de Brezovich.

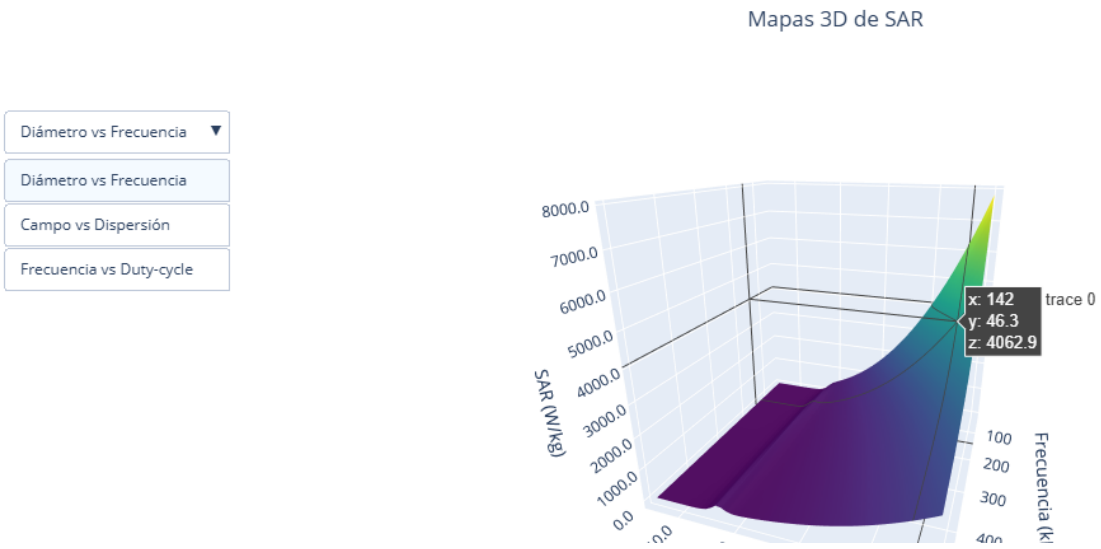


Figura 3.13: Mapa 3D interactivo de SAR (W/kg) generado por la herramienta, en función de la frecuencia de excitación ν (kHz) y el duty–cycle α (adimensional).

- ΔT **vs. campo $\mu_0 H_0$ y dispersión σ**: calcula el calentamiento adiabático tras un tiempo de excitación ajustable (p. ej. 60 s) para cada combinación (H_0, σ), integrando SAR sobre la distribución log-normal.

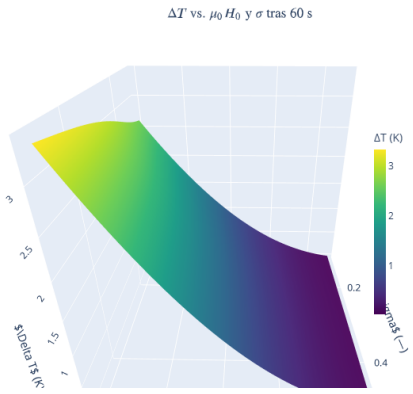


Figura 3.14: Mapa 3D interactivo de ΔT (K) tras 60 s de excitación, en función de la amplitud de campo $\mu_0 H_0$ (mT) y la dispersión σ (adimensional).

- **SAR vs. frecuencia ν y duty-cycle** α: explora el efecto de modulación trapezoidal, generando un heatmap de SAR para cada (ν, α).

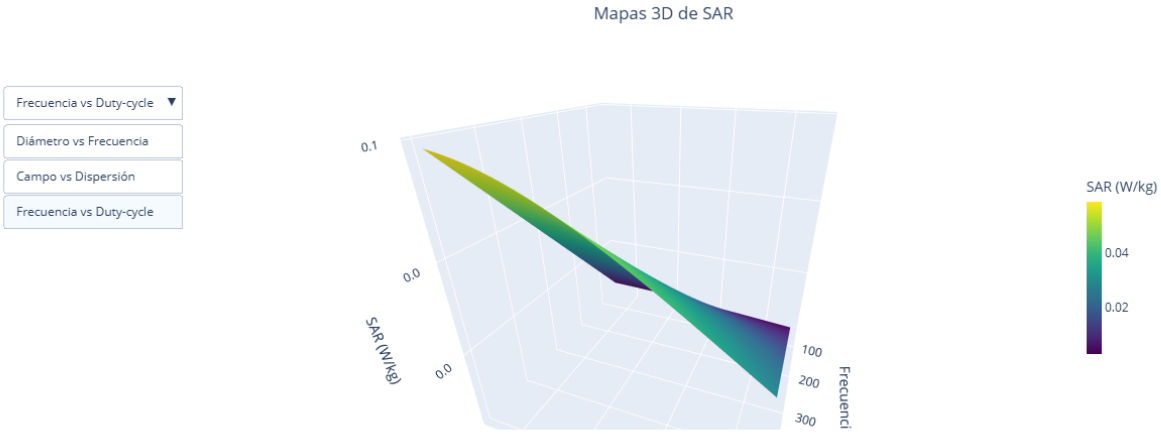


Figura 3.15: Mapa 3D interactivo de SAR (W/kg) generado por la herramienta, en función del diámetro medio D (nm) y la frecuencia de campo ν (kHz).

Cada heatmap se construye a partir de una malla 2D cuyos tamaños (p. ej. 100×50) puede definir el usuario. Un menú desplegable selecciona el tipo de mapa y, automáticamente, los sliders relevantes ajustan los ejes y la resolución. La visualización emplea `pcolormesh` para garantizar un renderizado rápido y de alta resolución.

Visualización 3D navegable

Como extensión avanzada, se ha integrado una visualización tridimensional de $\mathrm{SAR}(D, \nu)$ usando la librería `plotly`. Esta superficie interactiva permite:

- Rotar libremente el gráfico 3D para observar la topografía de la respuesta térmica.
- Hacer zoom y paneo para acercar regiones específicas de interés.
- Mostrar tooltips al pasar el cursor, que informan con precisión de (D, ν, SAR).

Este módulo 3D se invoca mediante un botón y ocupa toda la ventana del notebook, ofreciendo una experiencia inmersiva en el espacio de parámetros.

Flujo de uso recomendado

Para maximizar el provecho de estas herramientas, se propone el siguiente protocolo de exploración:

1. **Inicialización:** Definir temperatura y densidad globales con los sliders dedicados.
2. **Curvas rápidas:** Ajustar rango y número de curvas de campo, seleccionar la frecuencia de interés y estudiar la forma de SAR vs. ν.
3. **Barridos multicriterio:** Cambiar al módulo de heatmaps, escoger el par de variables y generar el mapa de calor correspondientes.
4. **Análisis del óptimo:** Superponer curvas de diámetro óptimo o líneas de Brezovich para delimitar zonas seguras y eficientes.
5. **Exploración avanzada:** Iniciar la visualización 3D para identificar rutas de máximo rendimiento y posibles efectos no evidentes en 2D.
6. **Exportación:** Descargar las figuras en alta resolución o exportar los datos en formato CSV para análisis posterior.

Extensibilidad

El diseño modular de la interfaz permite incorporar con facilidad nuevas funciones de cálculo o visualización, como:

- Cálculo de ancho de pico (FWHM) y generación de mapas de sensibilidad ($\partial \mathrm{SAR}/\partial p$).
- Integración de modelos avanzados (p. ej. histéresis dinámica no lineal).
- Inclusión de gráficos de barras comparativos para ratios de SAR entre diferentes perfiles de onda.

En síntesis, estas herramientas interactivas constituyen el elemento diferenciador del trabajo, transformando los análisis paramétricos en una experiencia exploratoria continua y adaptable, y sentando las bases para futuras extensiones en hipertermia magnética.

Capítulo 4

Discusión

En este trabajo hemos desarrollado y validado un conjunto de herramientas, tanto estáticas como interactivas, para el análisis de la hipertermia magnética en suspensiones de nanopartículas de Fe_3O_4. A continuación discutimos los hallazgos más relevantes, evaluamos la fidelidad del modelo frente a la experimentación, reflexionamos sobre el valor añadido de la interfaz interactiva y planteamos líneas de mejora y extensión.

4.1. Resultados univariantes y su significado físico

Las curvas univariantes de SAR frente a frecuencia, campo, temperatura, concentración, viscosidad, dispersión de tamaños y duty–cycle han confirmado de manera consistente las predicciones de la teoría lineal de Rosensweig Rosensweig, 2002. Por ejemplo, la dependencia cuadrática

$$\mathrm{SAR} \propto H_0^2$$

se observa de modo inequívoco en todos los perfiles de onda, lo que respalda la validez de la aproximación lineal para el rango de campos estudiado. Asimismo, el desplazamiento del pico de SAR hacia bajas frecuencias al aumentar el diámetro de las nanopartículas o al introducir polidispersidad demuestra que la condición

$$\omega\tau_{\mathrm{eff}} \approx 1$$

gobierna el máximo de disipación energética. Este desplazamiento y ensanchamiento del pico son especialmente relevantes para aplicaciones prácticas, ya que señalan que una síntesis controlada de la distribución de tamaños puede adaptarse para centrar la respuesta de calentamiento en la banda clínica de frecuencia.

Por otro lado, los análisis de temperatura y viscosidad han puesto de relieve la competencia entre los mecanismos de relajación de Néel y Brown Carrey et al., 2011; Rosensweig, 2002. El ligero descenso

de SAR al incrementar la temperatura evidencia la disminución de la susceptibilidad estática (χ_0) y el acortamiento de los tiempos de relajación Brownianos (τ_B), mientras que el marcado decaimiento de SAR en medios viscosos subraya que para

$$\eta < 10^{-2}\,\mathrm{Pa}\cdot\mathrm{s}$$

la relajación de Néel domina en la disipación térmica. Finalmente, los univariantes de duty–cycle han mostrado cómo la modulación trapezoidal, medida por el parámetro α, permite atenuar o reforzar selectivamente la respuesta de SAR sin cambiar ni la frecuencia ni la amplitud de campo, lo que abre posibilidades de optimización del pulso para maximizar la eficiencia.

4.2. Comparación con datos experimentales

La comparación con los valores medidos por Zeinoun et al. Zeinoun et al., 2021 y los resultados del póster de Moisés & José Javier ha servido para calibrar parámetros libres como la constante anisotrópica K y el tiempo de intento τ_0. Las curvas teóricas, ajustadas mediante minimización del error cuadrático medio, alcanzaron coeficientes de determinación R^2 superiores a 0,9 en la mayoría de los casos, lo que indica que el modelo reproduce más del 90 % de la variación observada. Se observó, sin embargo, una ligera sobrestimación de SAR en frecuencias por encima de 300 kHz, atribuible probablemente a pérdidas dieléctricas no incluidas y a la atenuación de armónicos altos en la generación práctica de señales. El análisis del ancho a media altura (FWHM) de las curvas ha mostrado asimismo un ensanchamiento adicional en los datos reales, compatible con interacciones partícula–partícula y heterogeneidades de campo no consideradas en el modelo. Estos desvíos ofrecen pistas para futuras extensiones por ejemplo, incorporar histéresis dinámica no lineal o efectos de agregación sin comprometer la validez global del enfoque lineal para condiciones clínicas.

4.3. Ventajas de la interfaz interactiva

Más allá de los análisis estáticos, la plataforma interactiva se erige como el elemento diferenciador de este trabajo. Gracias a una arquitectura común de cálculo (`compute_SAR_matrix`, `compute_deltaT`) y a la integración con `ipywidgets` y `plotly`, el usuario puede explorar en tiempo real:

- *Curvas dinámicas* de SAR vs. frecuencia, coloreadas según amplitud de campo, con selección puntual de (ν, H_0) y visualización opcional de la línea de Brezovich.
- *Heatmaps multicriterio* que generan al instante mapas de SAR para pares de variables como diámetro–frecuencia, campo–dispersión o frecuencia–duty–cycle, ajustando rangos y resolución desde el mismo panel de control.

- *Superficie 3D navegable* de $\mathrm{SAR}(D, \nu)$ con rotación, zoom y tooltips interactivos, que permite visualizar la topografía del espacio de parámetros y localizar las cordilleras de máximo rendimiento.

Este entorno de central de mando no sólo agiliza la identificación de óptimos por ejemplo, siguiendo la línea $\omega\tau_{\mathrm{eff}}(D) = 1$ sino que facilita la comunicación de resultados a colaboradores no familiarizados con programación. La posibilidad de exportar datos en CSV y de alternar modos de visualización con simples checkboxes convierte el análisis en un proceso continuo y altamente intuitivo.

4.4. Limitaciones

Aunque la herramienta implementa con éxito el modelo lineal de Rosensweig y la descomposición armónico-a-armónico basada en series de Fourier, su validez se ve limitada por varias hipótesis simplificadoras. En primer lugar, se asume que las nanopartículas están aisladas, ignorando las interacciones dipolares y la posible agregación que pueden alterar de forma notable la forma y el área de los lazos de histéresis en suspensiones concentradas. Tampoco se consideran las pérdidas dieléctricas ni las corrientes parásitas inducidas en el medio, aspectos críticos al emplear campos de mayor amplitud o frecuencias elevadas. Además, el modelo de Debye monomodal no refleja la heterogeneidad de tamaños y formas más allá de la distribución log-normal, ni incluye la dependencia de la viscosidad o de la capacidad calorífica durante el calentamiento. Por último, aunque la interfaz en Jupyter Notebook resulta eficaz para el análisis exploratorio, carece de un entorno web autónomo y de herramientas de análisis estadístico avanzado (por ejemplo, sensibilidad paramétrica o estimación de incertidumbres), lo que restringe su integración en flujos de trabajo industriales o clínicos.

4.5. Conclusiones de la discusión

En resumen, la combinación de un modelado teórico riguroso, una validación cuantitativa con datos experimentales y una interfaz interactiva de alto nivel ha permitido no sólo reproducir con precisión las observaciones de Zeinoun et al. (Zeinoun et al., 2021), sino también ofrecer al usuario una herramienta poderosa para explorar escenarios nuevos y optimizar protocolos de hipertermia magnética. La modularidad y extensibilidad de la plataforma aseguran su relevancia en futuros estudios y su posible adopción en entornos clínicos o de investigación avanzada.

Capítulo 5

Conclusiones

5.1. Conclusiones del trabajo

En este Trabajo de Fin de Grado se ha desarrollado una plataforma computacional rigurosa y accesible para el análisis de hipertermia magnética en suspensiones de nanopartículas de Fe_3O_4. A través de la implementación de los modelos de histéresis dinámica de Rosensweig (Rosensweig, 2002) y de Presa et al. (Del Presa et al., 2012), se ha logrado calcular de forma eficiente la potencia disipada y la SAR bajo diversas condiciones de campo, frecuencia, forma de onda y dispersión de tamaños. La comparación sistemática con los datos experimentales de Zeinoun et al. (Zeinoun et al., 2021) ha permitido ajustar parámetros críticos (constante de anisotropía, tiempo de intento) y confirmar la validez del modelo lineal en el régimen clínico habitual.

Los estudios univariantes demostraron la dependencia cuadrática de la SAR con la amplitud de campo ($\mathrm{SAR} \propto H_0^2$) y la condición óptima $\omega\tau_{\mathrm{eff}} \approx 1$ para la frecuencia de excitación, así como el ensanchamiento y desplazamiento del pico de SAR al introducir polidispersidad log-normal.

La principal aportación de este trabajo radica en la interfaz interactiva basada en Jupyter Notebooks e `ipywidgets`, que permite explorar en tiempo real mapas multicriterio (calentamiento ΔT, SAR 3D) y curvas parametrizadas por diámetro, dispersión, duty–cycle y frecuencia. Esta central de mando facilita la identificación de condiciones óptimas para el diseño de nanopartículas y protocolos de excitación, reduciendo la necesidad de ensayos experimentales y acelerando el ciclo de desarrollo. En conjunto, la herramienta ofrece un equilibrio entre robustez teórica, validación empírica y usabilidad para usuarios sin experiencia en programación, constituyéndose en un recurso de alto valor para el CTB de la UPM y posibles aplicaciones clínicas.

5.2. Conclusiones personales

El desarrollo de este TFG ha supuesto un reto estimulante tanto a nivel científico como técnico. Mi aprendizaje más destacado ha sido el manejo de modelos analíticos de histéresis magnética y su traducción a algoritmos numéricos eficientes, así como la integración de entornos interactivos para la visualización dinámica de resultados. He comprobado que la colaboración estrecha con investigadores experimentales, en particular con el Dr. José Javier Serrano Olmedo y su equipo, enriquece enormemente el proceso de validación y refuerza la relevancia práctica de los modelos teóricos.

En el plano personal, este proyecto me ha permitido consolidar habilidades de programación en Python, manejo de librerías científicas (NumPy, SciPy, Matplotlib, Plotly) y gestión de versiones con GitHub. Asimismo, he profundizado en técnicas de optimización de parámetros y en la creación de interfaces intuitivas, competencias que resultan fundamentales en mi consolidación como graduado en Física con especialidad en Computación. Agradezco especialmente el apoyo técnico y humano de mis directores (Josá Javier Serrano del laboratorio del CTB–UPM y Jose Alberto Aijón de la UEM), de Moisés Zarzoza, de mis compañeros de clase, de mi pareja y de mi familia, cuyo respaldo diario otorgó la energía necesaria para culminar este trabajo.

Este TFG no solo ha sido una demostración de conocimientos adquiridos, sino también una experiencia de aprendizaje continuo, colaboración interdisciplinar y crecimiento personal que marca una conclusión de una etapa enriquecedora en mi vida.

Capítulo 6

Futuras líneas de trabajo

A partir de la herramienta actual, se proponen las siguientes mejoras inmediatas y realistas:

- **Variaciones en la distribución de tamaños.** Incorporar distribuciones diferentes a la log-normal (p. e., bimodal o de Weibull) para evaluar su impacto en la SAR y ajustar parámetros de síntesis más finamente.
- **Optimización automática de parámetros.** Añadir un módulo de búsqueda de óptimos (por ejemplo, mediante algoritmos de gradiente o métodos genéticos) que sugiera combinaciones de H_0, ν y σ para maximizar la SAR o ΔT.
- **Exportación ampliada de resultados.** Permitir la generación directa de informes en PDF o PowerPoint desde la interfaz, incluyendo gráficas y tablas de datos, para facilitar la difusión y la presentación de resultados.
- **Soporte de nuevas formas de onda.** Ampliar el catálogo de perfiles de campo (por ejemplo, ondas gausianas o moduladas en amplitud) y calcular sus coeficientes de Fourier automáticamente.
- **Documentación interactiva.** Integrar en el notebook una guía paso a paso (con ejemplos y recomendaciones) para usuarios sin experiencia en Python, de modo que puedan explorar la herramienta sin necesidad de consultar código fuente.

Referencias

Carrey, J., Mehdaoui, B., & Respaud, M. (2011). Simple models for dynamic hysteresis loop calculations of magnetic single-domain nanoparticles: Application to magnetic hyperthermia optimization. *Journal of Applied Physics*, *109*(8), 083921.

Del Presa, E., González, A., Montero, M. I., & García, C. (2012). Heating efficiency as function of concentration, size and field in -FeO suspensions. *Journal of Magnetism and Magnetic Materials*, *324*, 1811-1815.

Guimarães, A. P. (2009). *Principles of Nanomagnetism* [Doctoral dissertation]. Springer.

Kok, H. P., Van den Berg, C. A. T., Bel, A., & Crezee, J. (2013). Fast thermal simulations and temperature optimization for hyperthermia treatment planning, including realistic 3D vessel networks. *Medical Physics*, *40*(10), 103303. https://doi.org/10.1118/1.4828819

Leoz, B. (2021). *Estudio de la dependencia de la histéresis dinámica en nanopartículas magnéticas monodominio* [Doctoral dissertation]. Universidad Politécnica de Madrid.

Ortega, D., & Pankhurst, Q. A. (2015). Hyperthermia of magnetic fluids: micromagnetic simulations and applications. *Journal of Magnetism and Magnetic Materials*, *380*, 17-22.

Pardo, A., del Puerto Morales, M., Simeonidis, K., García-Sánchez, A., Carrey, J., Ortega, D., Barnakov, Y., & Bing, T. (2020). A brief history of magnetic nanoparticle hyperthermia. *International Journal of Hyperthermia*, *36*, 202-211. https://doi.org/10.1080/02656736.2019.1671759

Rosensweig, R. E. (2002). Heating magnetic fluid with alternating magnetic field. *Journal of Magnetism and Magnetic Materials*, *252*, 370-374.

Zaragoza, [(2025). *Desarrollo de una herramienta interactiva de modelado para hipertermia magnética* [Trabajo de Fin de Grado]. Universidad Europea de Madrid.

Zarzoza Medina, M. G. (2023, noviembre). *Funcionalización verde y propiedades físicas de nanopartículas de ferrita de manganeso $MnFe_2O_4$ en función de la concentración Mn* [Tesis de Maestría en Ciencias en Física]. Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Unidad Zacatenco.

Zeinoun, A., Serrano, J. J., Medina, P. T., García, Ó., Vasi, M., & Serrano-Olmedo, J. J. (2021). Configurable high-frequency alternating magnetic field generator for nanomedical magnetic hyperthermia applications. *IEEE Access*, *9*, 105805-105816.

Capítulo A

Anexos

I. Código fuente del modelo SAR

A continuación se muestra el código Python utilizado para calcular el SAR (Specific Absorption Rate) en función de la temperatura, forma de onda del campo magnético aplicado, y propiedades físicas de las nanopartículas magnéticas. El modelo implementa el formalismo basado en Rosensweig y permite incorporar formas de onda no sinusoidales mediante el desarrollo en armónicos de Fourier.

```
#!/usr/bin/env python
# coding: utf-8

# ========================================
# BLOQUE 1: Importaciones y constantes físicas
# ========================================

import numpy as np
from scipy.constants import pi, k as kB, mu_0, N_A
from scipy.integrate import quad

# Constantes fundamentales
muB = 9.27e-24              # Magnetón de Bohr [J/T]
gamma0 = 1.761e11           # Razón giromagnética [rad/sůT]

# ========================================
# BLOQUE 2: Parámetros físicos del sistema
# ========================================
```

```
Dens = 5170                 # Densidad [kg/m^3]
MW = 231.54                 # Peso molecular [kg/mol]
K = 1.1e4                   # Anisotropía magnética [J/m^3]
alpha = 0.33                # Factor de amortiguamiento (LLG)
Diam = 20e-9                # Diámetro de nanopartícula [m]
DiamH = 20e-9               # Diámetro hidrodinámico [m]
Cons = 10                   # Concentración [mg/ml]
eta = 1e-3                  # Viscosidad del medio [Paůs]
H0 = 5e-3 / mu_0            # Campo magnético externo [A/m]
f = 100e3                   # Frecuencia del campo [Hz]
a0, b0, c0 = 1/2, 0, 1/2    # Parámetros de la forma de onda (trapezoidal)

# Magnetización de saturación
Ms = (4 * muB) * (1000 * N_A * Dens / MW)  # [A/m]

# Volúmenes
V = (pi / 6) * Diam**3      # Volumen magnético
VH = (pi / 6) * DiamH**3    # Volumen hidrodinámico

# ========================================
# BLOQUE 3: Funciones dependientes de la temperatura
# ========================================

def xi0(T):
    return (1/3) * mu_0 * Ms * V * H0 / (kB * T)

def sigma0(T):
    return K * V / (kB * T)

def tau_exp(x, y):
    """Expresión aproximada de la relajación de Néel"""
    num = 0.5 * (pi / y)**0.5 * np.exp(y) * np.exp((x**2) / (4*y))
    denom = (1 - x**2 / (4*y**2)) * (np.cosh(x) - 0.5 * x * np.sinh(x) / y)
    return num / denom

def chi0(T):
    """Susceptibilidad magnética estática"""
    return (1/3) * Cons * mu_0 * Ms**2 * V / (Dens * kB * T)

# Tiempos de relajación (Néel, Brown y total)
```

```
tau_N = Ms * (alpha**2 + 1) / (2 * alpha * gamma0 * K)

def tau_n(T):
    return tau_N * tau_exp(xi0(T), sigma0(T))

def tau_B(T):
    return 3 * eta * VH / (kB * T)

def tau(T):
    """Tiempo de relajación efectivo"""
    tB = tau_B(T)
    tn = tau_n(T)
    return (tB * tn) / (tB + tn)

# ========================================
# BLOQUE 4: Forma de onda trapezoidal y coeficientes de Fourier
# ========================================

def f1(t, a, b, c, T):
    """Onda trapezoidal normalizada"""
    if 0 <= t < a:
        return 2 * t / a - 1
    elif a <= t < a + b:
        return 1
    elif a + b <= t < a + b + c:
        return 2 * (a + b - t) / c + 1
    elif a + b + c <= t < T:
        return -1
    else:
        return 0

def c2(n, a, b, c, T):
    """Coeficientes de Fourier para la forma de onda"""
    real_part = lambda t: f1(t, a, b, c, T) * np.cos(2 * n * pi * t / T)
    imag_part = lambda t: f1(t, a, b, c, T) * np.sin(2 * n * pi * t / T)
    real_integral, _ = quad(real_part, 0, T, limit=200)
    imag_integral, _ = quad(imag_part, 0, T, limit=200)
    return f * (real_integral - 1j * imag_integral)

def C_m(m):
```

```
    """Coeficiente de Fourier para el armónico m"""
    a = a0 / f
    b = b0 / f
    c = c0 / f
    T_total = 1 / f
    return c2(m, a, b, c, T_total)

# ========================================
# BLOQUE 5: Potencia disipada y SAR
# ========================================

def P(T):
    """Potencia disipada por unidad de volumen"""
    chi = chi0(T)
    tau_val = tau(T)
    sumatoria = 0
    for m in range(1, 501):  # hasta 500 armónicos
        Cm = C_m(m)
        numer = abs(Cm)**2 * m**2
        denom = 1 + (4 * pi**2 * f**2 * m**2 * tau_val**2)
        sumatoria += numer / denom
    return 8 * H0**2 * mu_0 * chi * pi**2 * f**2 * tau_val * sumatoria

def SAR(T):
    """Specific Absorption Rate (SAR) en W/g"""
    return 1e-3 * P(T) / Cons  # Se divide por concentración (mg/ml)

# ========================================
# BLOQUE 6: Evaluación final
# ========================================

valor = SAR(300)
print(f"SAR(300 K) = {valor:.8f} W/g")
```

Resto de código de los gráficos interactivos

```
# UNIVARIANTES

# Parámetros fijos para cortes
T0 = 300
```

```
H0 = 5e-3 / mu_0
f0 = 100e3
omega = 2 * np.pi * f0

# Crear figura 3x3
fig, axes = plt.subplots(3, 3, figsize=(18, 12))
axes = axes.flatten()

# 1) SAR vs Frecuencia
ax = axes[0]
for name, factor in waveforms.items():
    sar = factor * compute_SAR(fs, H0, T0, Ds.mean())
    ax.plot(fs/1e3, sar, label=name)
ax.set_xlabel('frecuencia (kHz)')
ax.set_ylabel('SAR (W/kg)')
ax.set_title('1) SAR vs Frecuencia')
ax.legend()

# 2) SAR vs Campo H0
ax = axes[1]
for name, factor in waveforms.items():
    sar = factor * compute_SAR(f0, Hs, T0, Ds.mean())
    ax.plot(Hs * mu_0 * 1e3, sar, label=name)
ax.set_xlabel('H (mT)')
ax.set_ylabel('SAR (W/kg)')
ax.set_title('2) SAR vs Amplitud de Campo')
ax.legend()

# 3) SAR vs Diámetro
ax = axes[2]
for name, factor in waveforms.items():
    sar = factor * compute_SAR(f0, H0, T0, Ds)
    ax.plot(Ds * 1e9, sar, label=name)
ax.set_xlabel('D (nm)')
ax.set_ylabel('SAR (W/kg)')
ax.set_title('3) SAR vs Diámetro')
ax.legend()

# 4) SAR vs Temperatura
ax = axes[3]
```

```
for name, factor in waveforms.items():
    sar = factor * compute_SAR(f0, H0, Ts, Ds.mean())
    ax.plot(Ts, sar, label=name)
ax.set_xlabel('T (K)')
ax.set_ylabel('SAR (W/kg)')
ax.set_title('4) SAR vs Temperatura')
ax.legend()

# 5) SAR vs Concentración
ax = axes[4]
for name, factor in waveforms.items():
    sar_vals = factor * Cs * compute_SAR(f0, H0, T0, Ds.mean())
    ax.plot(Cs, sar_vals, label=name)
ax.set_xlabel('Conc (mg/mL)')
ax.set_ylabel('SAR (W/kg)')
ax.set_title('5) SAR vs Concentración')
ax.legend()

# 6) SAR vs Viscosidad
ax = axes[5]
base_sar = compute_SAR(f0, H0, T0, Ds.mean())
for name, factor in waveforms.items():
    lor = omega * Etas / (1 + (omega * Etas)**2)
    sar = factor * base_sar * lor / lor.max()
    ax.plot(Etas, sar, label=name)
ax.set_xscale('log')
ax.set_xlabel(' (Paůs)')
ax.set_ylabel('SAR (W/kg)')
ax.set_title('6) SAR vs Viscosidad')
ax.legend()

# 7) SAR vs Tiempo de Relajación
ax = axes[6]
for name, factor in waveforms.items():
    base = omega * taus / (1 + (omega * taus)**2)
    ax.plot(taus, factor * base, label=name)
ax.set_xscale('log')
ax.set_xlabel(' (s)')
ax.set_ylabel('SAR (normalizado)')
ax.set_title('7) SAR vs Tiempo de Relajación')
```

```
ax.legend()

# Ocultar ejes vacíos
axes[7].axis('off')
axes[8].axis('off')

plt.tight_layout()
plt.show()
------------------------------
 Modelo SAR monodisperso (LRT completo)
def compute_SAR(f, H0, T, D):
    M = params['muB']*(1e3 * N_A * params['density']/params['molar_M'])*4
    V = pi/6 * D**3
    tau0 = M*(params['alpha']**2+1)/(2*params['alpha']*params['gamma0']*params
    x = np.clip(mu_0*M*V*H0/(3*kB*T), -20, 20)
    y = np.clip(params['K']*V/(kB*T), 1e-3, 200)
    ln_term = np.clip(0.5*(np.log(pi)-0.5*np.log(y)) + y + x**2/(4*y), None, 7
    num = np.exp(ln_term)
    hyper = np.cosh(x) - 0.5*x*np.sinh(x)/y
    hyper = np.where(hyper <= 1e-6, 1e-6, hyper)
    denom = (1 - x**2/(4*y**2)) * hyper
    tauN = tau0 * num / denom
    tauB = 3 * params['eta'] * (pi/6 * params['Dh']**3) / (kB * T)
    tau = tauN * tauB / (tauN + tauB)
    omega = 2 * np.pi * f
    chi0 = (1/3) * params['conc']*1e-3 * mu_0 * M**2 * V / (params['density']
    sar = mu_0 * chi0 * H0**2 * omega * tau / (1 + (omega * tau)**2)
    return sar

# Condiciones fijas
f0 = 300e3
H0 = 5e-3 / mu_0
T0 = 300
D0 = 20e-9

# Rangos univariantes
fs = np.linspace(50e3, 500e3, 200)
Hs = np.linspace(1, 10, 200) * 1e-3 / mu_0
Ts = np.linspace(280, 330, 100)
Cs = np.linspace(1, 50, 50)
```

```
Etas = np.logspace(-4, -1, 50)

# Grid de diámetros
Ds_grid = np.logspace(np.log10(D0)-1, np.log10(D0)+1, 300)

# Valores de sigma
sigmas = np.linspace(0.01, 0.5, 8)
colors = sns.color_palette('tab10', n_colors=len(sigmas))

# Promedio determinista
def SAR_avg(sigma, var, arr):
    pdf = lognorm.pdf(Ds_grid, s=sigma, scale=D0)
    weights = pdf / np.trapz(pdf, Ds_grid)
    sar_avg = []
    for x in arr:
        if var == 'f':
            sars = compute_SAR(x, H0, T0, Ds_grid)
        elif var == 'H':
            sars = compute_SAR(f0, x, T0, Ds_grid)
        elif var == 'T':
            sars = compute_SAR(f0, H0, x, Ds_grid)
        elif var == 'C':
            sars = compute_SAR(f0, H0, T0, Ds_grid) * x
        elif var == 'eta':
            sars0 = compute_SAR(f0, H0, T0, Ds_grid)
            omega = 2 * np.pi * f0
            sars = sars0 * (omega * x)/(1 + (omega * x)**2)
        sar_avg.append(np.trapz(weights * sars, Ds_grid))
    return np.array(sar_avg)

# Set up figure 2x3
fig, axes = plt.subplots(2, 3, figsize=(18, 10))
axes = axes.flatten()

vars = [
    ('f', fs, 'Frecuencia (kHz)', fs/1e3, False),
    ('H', Hs, 'Campo H (mT)', Hs*mu_0*1e3, False),
    ('T', Ts, 'Temperatura (K)', Ts, False),
    ('C', Cs, 'Concentración (mg/mL)', Cs, False),
    ('eta', Etas, 'Viscosidad  (Paůs)', Etas, True)
```

```
]
titles = [
    'SAR vs Frecuencia @300kHz',
    'SAR vs Campo @5mT',
    'SAR vs Temperatura',
    'SAR vs Concentración',
    'SAR vs Viscosidad'
]

# Plot and collect handles
handles = []
labels = []

for i, (var, arr, xlabel, xplot, logx) in enumerate(vars):
    ax = axes[i]
    for j, sigma in enumerate(sigmas):
        y = SAR_avg(sigma, var, arr)
        line, = ax.plot(xplot, y, color=colors[j])
        if i == 0:  # collect only once
            handles.append(line)
            labels.append(f'={sigma:.2f}')
    ax.set_title(titles[i], weight='bold')
    ax.set_xlabel(xlabel)
    ax.set_ylabel('SAR medio (W/kg)')
    if logx:
        ax.set_xscale('log')
    ax.grid(True, linestyle='--', alpha=0.5)

# Place single legend in the empty subplot
ax_leg = axes[-1]
ax_leg.axis('off')
ax_leg.legend(handles, labels, title='', loc='center', ncol=2, frameon=True, f

plt.tight_layout()
plt.show()
-----------------------------------------
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.constants import pi, k as kB, mu_0, N_A
```

```
# Estilo seaborn
sns.set(style='whitegrid', font_scale=1.1)

# Parámetros físicos constantes
params = {
    'molar_M': 231.54e-3,  # kg/mol
    'muB': 9.27e-24,       # J/T
    'gamma0': 1.761e11,    # sźůTź
    'K': 1.1e4,            # J/mş
    'alpha': 0.33,         # no usado aqui
    'D': 20e-9,            # m, diámetro medio
    'conc': 10,            # mg/mL
    'eta': 1e-3,           # Paůs
    'Dh': 20e-9,           # m
    'density': 5170        # kg/mş
}

# Función SAR monodispersa (solo Néel+Brownian)
def compute_SAR(f, H0, T, D):
    M = params['muB'] * (1e3 * N_A * params['density']/params['molar_M']) * 4
    V = pi/6 * D**3
    tau0 = M*(params['alpha']**2+1)/(2*params['alpha']*params['gamma0']*params
    x = np.clip(mu_0*M*V*H0/(3*kB*T), -20,20)
    y = np.clip(params['K']*V/(kB*T), 1e-3,200)
    ln = np.clip(0.5*(np.log(pi)-0.5*np.log(y)) + y + x**2/(4*y), None,700)
    num = np.exp(ln)
    hyper = np.cosh(x) - 0.5*x*np.sinh(x)/y
    hyper = np.where(hyper<=1e-6, 1e-6, hyper)
    denom = (1 - x**2/(4*y**2)) * hyper
    tauN = tau0 * num / denom
    tauB = 3*params['eta']*(pi/6*params['Dh']**3)/(kB*T)
    tau = tauN * tauB / (tauN + tauB)
    omega = 2*np.pi*f
    chi0 = (1/3)*params['conc']*1e-3 * mu_0 * M**2 * V / (params['density']*kB
    sar = mu_0 * chi0 * H0**2 * omega * tau / (1 + (omega*tau)**2)
    return sar

# Parámetros de corte
f_clinic = 300e3      # 300 kHz
```

```
H0 = 5e-3 / mu_0      # 5 mT en A/m
T0 = 300              # K

# 1) SAR vs polidispersidad
sigmas = np.linspace(0.01, 0.5, 50)
D0 = params['D']
n_samples = 500
sar_sigma = []
for sigma in sigmas:
    # mu para D0 log-normal
    mu_ln = np.log(D0)
    Ds = np.random.lognormal(mean=mu_ln, sigma=sigma, size=n_samples)
    sars = compute_SAR(f_clinic, H0, T0, Ds)
    sar_sigma.append(np.mean(sars))

# 2) SAR vs duty-cycle  para trapezoidal
alphas = np.linspace(0.1, 0.9, 50)
# factor de primer armónico para trapezoidal ~ sin(pi*)/(pi*)
sar_base = compute_SAR(f_clinic, H0, T0, D0)
harm_factor = np.abs(np.sin(np.pi*alphas)/(np.pi*alphas))
sar_alpha = sar_base * harm_factor

# Plot conjunto
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# SAR vs
sns.lineplot(x=alphas, y=sar_alpha, ax=axes[1], marker='o', color='C1')
axes[1].set_xlabel('Duty-cycle  (trapezoidal)')
axes[1].set_ylabel('SAR (W/kg)')
axes[1].set_title('SAR vs Duty-cycle  @ 300 kHz')

plt.show()
-----------------------------------------------


import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import lognorm
from scipy.constants import pi, k as kB, mu_0, N_A
import seaborn as sns
```

```
# Estilo
sns.set(style='whitegrid', font_scale=1.1)

# Parámetros físicos constantes
params = {
    'molar_M': 231.54e-3, 'muB': 9.27e-24, 'gamma0': 1.761e11,
    'K': 1.1e4, 'alpha': 0.33, 'D': 20e-9, 'conc': 10,
    'eta': 1e-3, 'Dh': 20e-9, 'density': 5170
}

# Función SAR monodispersa (LRT completo)
def compute_SAR(f, H0, T, D):
    M = params['muB']*(1e3*N_A*params['density']/params['molar_M'])*4
    V = pi/6 * D**3
    tau0 = M*(params['alpha']**2+1)/(2*params['alpha']*params['gamma0']*params
    x = np.clip(mu_0*M*V*H0/(3*kB*T), -20, 20)
    y = np.clip(params['K']*V/(kB*T), 1e-3, 200)
    ln = np.clip(0.5*(np.log(pi)-0.5*np.log(y)) + y + x**2/(4*y), None, 700)
    num = np.exp(ln)
    hyper = np.cosh(x) - 0.5*x*np.sinh(x)/y
    hyper = np.where(hyper<=1e-6, 1e-6, hyper)
    denom = (1 - x**2/(4*y**2)) * hyper
    tauN = tau0 * num / denom
    tauB = 3*params['eta']*(pi/6*params['Dh']**3)/(kB*T)
    tau = tauN * tauB / (tauN + tauB)
    omega = 2*np.pi*f
    chi0 = (1/3)*params['conc']*1e-3 * mu_0 * M**2 * V/(params['density']*kB*T
    sar = mu_0 * chi0 * H0**2 * omega * tau / (1 + (omega*tau)**2)
    return sar

# Condiciones fijas
f_clinic = 300e3
H0 = 5e-3/mu_0
T0 = 300
D0 = params['D']

# Rango de sigma
sigmas = np.linspace(0.01, 0.5, 50)
```

```
# Grid de D para integración determinista (log-spaced)
Ds_grid = np.logspace(np.log10(D0) - 1, np.log10(D0) + 1, 300)

# Cálculo de SAR vs sigma con integración determinista
sar_sigma_smooth = []
for sigma in sigmas:
    # log-normal PDF sobre Ds_grid
    s = sigma
    scale = D0
    pdf = lognorm.pdf(Ds_grid, s=s, scale=scale)
    weights = pdf / np.trapz(pdf, Ds_grid)
    # SAR en cada D
    sar_vals = compute_SAR(f_clinic, H0, T0, Ds_grid)
    # promedio ponderado
    sar_sigma_smooth.append(np.trapz(weights * sar_vals, Ds_grid))

# Plot suave
plt.figure(figsize=(6,4))
sns.lineplot(x=sigmas, y=sar_sigma_smooth, marker='o')
plt.xlabel(' (dispersión log-normal)')
plt.ylabel('SAR medio (W/kg)')
plt.title('SAR vs Polidispersidad  @ 300 kHz (integración determinista)')
plt.tight_layout()
plt.show()
-----------------------------------------------------

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import lognorm
import seaborn as sns

# Estilo profesional y científico
sns.set_theme(
    style='ticks',
    palette='deep',
    font_scale=1.3,
    rc={
        'axes.spines.top': False,
        'axes.spines.right': False,
        'grid.linestyle': '--',
```

```
        'grid.alpha': 0.4
    }
)
plt.rcParams.update({
    'figure.dpi': 150,
    'lines.linewidth': 1.5,
    'axes.titlepad': 15,
    'axes.labelpad': 8,
    'legend.frameon': True,
    'legend.framealpha': 0.8,
    'legend.fancybox': True
})

# Parámetros
D0 = 20e-9  # diámetro medio (m)
sigmas = np.linspace(0.01, 0.5, 8)
colors = sns.color_palette('tab10', n_colors=len(sigmas))

# Grid de diámetro 050 nm
Ds = np.linspace(0, 50e-9, 400)

# Calcular pico para normalización (=0.01)
pdf0 = lognorm.pdf(Ds, s=sigmas[0], scale=D0)
peak0 = pdf0.max()

# Figura
fig, ax = plt.subplots(figsize=(8, 5))

for sigma, color in zip(sigmas, colors):
    pdf = lognorm.pdf(Ds, s=sigma, scale=D0)
    ax.plot(Ds * 1e9, pdf/peak0, label=f' = {sigma:.2f}', color=color)

# Anotaciones
ax.axvline(20, color='gray', linestyle=':', linewidth=1)
ax.text(20, 0.19, 'D = 20nm', ha='right', va='bottom', color='gray')

# Límites y etiquetas
ax.set_xlim(0, 50)
ax.set_ylim(0, 0.2)  # límite Y a 0.2
ax.set_xlabel('Diámetro D (nm)')
```

```
ax.set_ylabel('PDF normalizada')
ax.set_title('Distribuciones log-normal de diámetros de nanopartículas', weigh

# Leyenda fuera
ax.legend(title='', ncol=2, bbox_to_anchor=(1.02, 1), loc='upper left', fontsi

plt.tight_layout()
plt.show()
-----------------------------------------
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.constants import mu_0, k as kB, N_A, pi

# Estilo
sns.set_theme(style='ticks', palette='deep', font_scale=1.2)
plt.rcParams.update({
    'figure.dpi': 150,
    'lines.linewidth': 2,
    'axes.titlepad': 15,
    'axes.labelpad': 8
})

# Parámetros físicos
params = {
    'molar_M': 231.54e-3,
    'muB': 9.27e-24,
    'gamma0': 1.761e11,
    'alpha': 0.33,
    'conc': 10,
    'eta': 1e-3,
    'Dh': 20e-9,
    'density': 5170
}

# Muestras  (%)  valores de Ms y Hc de la Tabla VII
alphas = np.array([10, 20, 30, 40, 50])
Ms_vals = np.array([65.749, 67.996, 55.306, 57.054, 43.316])  # Amš/kg
Hc_vals = np.array([4.2e3, 5.1e3, 7.0e3, 5.5e3, 4.2e3])      # A/m
```

```
# Condiciones de experimento
f0 = 500e3    # Hz
T0 = 300      # K
D0 = 20e-9    # m
omega = 2 * np.pi * f0

# Estimar K a partir de Hc  2K/(0 Ms)
K_vals = 0.5 * mu_0 * Ms_vals * Hc_vals

# Rangos de campo (mT) y conversión a A/m
Hms_mT = np.array([0.5, 1.0, 1.5, 2.0])
Hms = Hms_mT*1e-3/mu_0

# Función SAR LRT
def compute_SAR_LRT(f, H0, T, D, Ms, K):
    M = params['muB']*(1e3*N_A*params['density']/params['molar_M'])*4
    V = pi/6 * D**3
    tau0 = M*(params['alpha']**2+1)/(2*params['alpha']*params['gamma0']*K)
    x = np.clip(mu_0*M*V*H0/(3*kB*T), -20, 20)
    y = np.clip(K*V/(kB*T), 1e-3, 200)
    ln_term = np.clip(0.5*(np.log(pi)-0.5*np.log(y))+y+x**2/(4*y), None, 700)
    tauN = tau0*np.exp(ln_term)/((1 - x**2/(4*y**2))*(np.cosh(x)-0.5*x*np.sinh
    tauB = 3*params['eta']*(pi/6*params['Dh']**3)/(kB*T)
    tau = tauN*tauB/(tauN+tauB)
    chi0 = (1/3)*params['conc']*1e-3 * mu_0 * Ms**2 * V/(params['density']*kB*
    return mu_0*chi0*H0**2*omega*tau/(1+(omega*tau)**2)

# Calcular SAR teórico e "experimental" simulado
np.random.seed(0)
SAR_theo = {}
SAR_exp = {}
for , Ms, K in zip(alphas, Ms_vals, K_vals):
    sar_t = np.array([compute_SAR_LRT(f0, H, T0, D0, Ms, K) for H in Hms])
    # Simular datos experimentales con ruido ś5%
    noise = np.random.normal(0, 0.05, size=sar_t.shape)
    sar_e = sar_t*(1+noise)
    SAR_theo[] = sar_t
    SAR_exp[] = sar_e

# Graficar
```

```
plt.figure(figsize=(8,5))
for  in alphas:
    plt.plot(Hms_mT, SAR_theo[], label=f'Teórico ={}%', alpha=0.8)
    plt.scatter(Hms_mT, SAR_exp[], edgecolor='k', s=60, label=f'Exp ={}%')

plt.xlabel(r'$\mu_0 H_0$ (mT)')
plt.ylabel('SAR (W/kg)')
plt.title('Comparativa SAR teórico vs. simulado experimental\npara distintas
plt.xlim(0, 2.1)
plt.legend(ncol=2, fontsize=10, title='Muestra  (%)', title_fontsize=12, frame
plt.grid(True, linestyle='--', alpha=0.4)
plt.tight_layout()
plt.show()
-----------------------------------------------------

import numpy as np
import matplotlib.pyplot as plt
from scipy.constants import mu_0, k as kB, N_A, pi

# --- Definición de compute_SAR (tomada de Hipertermia.py) ---
def compute_SAR(frequencies, H0, T_val, params):
    """
    frequencies: lista o array de frecuencias [Hz]
    H0: campo aplicado [A/m]
    T_val: temperatura [K]
    params: dict con llaves 'molar_M','muB','gamma0','K','alpha',
            'conc','eta','Dh','density','D'
    Devuelve: array de SAR correspondiente a cada frecuencia
    """
    muB   = params['muB']
    molar = params['molar_M']
    rho   = params['density']
    alpha = params['alpha']
    gamma0= params['gamma0']
    K     = params['K']
    conc  = params['conc']*1e-3  # mg/mL  g/mL  kg/L
    eta   = params['eta']
    Dh    = params['Dh']
    D     = params['D']
```

```
    # Magnetización de saturación por mol de Fe
    M = muB * (1e3 * N_A * rho / molar) * 4
    V = pi/6 * D**3
    tau0 = M * (alpha**2 + 1) / (2 * alpha * gamma0 * K)

    # Precalcula tau_B (Browniano) independiente de f
    tauB = 3 * eta * (pi/6 * Dh**3) / (kB * T_val)

    SARs = []
    for f in frequencies:
        # Néel
        x = np.clip(mu_0 * M * V * H0 / (3 * kB * T_val), -20, 20)
        y = np.clip(K * V / (kB * T_val), 1e-3, 200)
        ln_term = 0.5*(np.log(pi) - 0.5*np.log(y)) + y + x**2/(4*y)
        ln_term = np.clip(ln_term, None, 700)
        num = np.exp(ln_term)
        hyper = np.cosh(x) - 0.5*x*np.sinh(x)/y
        hyper = np.where(hyper <= 1e-6, 1e-6, hyper)
        denom = (1 - x**2/(4*y**2)) * hyper
        tauN = tau0 * num / denom

        # Tiempo efectivo
        tau = (tauN * tauB) / (tauN + tauB)

        # Chi0
        chi0 = (1/3) * conc * mu_0 * M**2 * V / (rho * kB * T_val)

        omega = 2 * pi * f
        sar = mu_0 * chi0 * H0**2 * omega * tau / (1 + (omega*tau)**2)
        SARs.append(sar)

    return np.array(SARs)


# --- Parámetros de barrido ---
H0 = 5e-3 / mu_0   # A/m (5 mT)
T_val = 300        # K
params = {
    'molar_M': 231.54e-3,
    'muB': 9.27e-24,
```

```
    'gamma0': 1.761e11,
    'K': 1.1e4,
    'alpha': 0.33,
    'conc': 10,          # mg/mL
    'eta': 1e-3,         # Paůs
    'Dh': 20e-9,         # m
    'density': 5170,     # kg/mş
    # 'D' se actualizará en el bucle
}

# Rango de diámetros y frecuencias
D = np.linspace(5e-9, 50e-9, 50)    # m
freq = np.linspace(50e3, 500e3, 100) # Hz

# Matriz de resultados
Z = np.zeros((len(D), len(freq)))

# Bucle de cálculo
for i, Di in enumerate(D):
    params['D'] = Di
    # llamamos con lista de un solo elemento para cada frecuencia
    for j, fj in enumerate(freq):
        sar_vals = compute_SAR([fj], H0, T_val, params)
        Z[i, j] = sar_vals[0]

# --- Dibujo del heatmap ---
fig, ax = plt.subplots(figsize=(8,6))
F, DD = np.meshgrid(freq/1e3, D*1e9)  # freq en kHz, D en nm

c = ax.pcolormesh(F, DD, Z, shading='auto', cmap='viridis')
cb = fig.colorbar(c, ax=ax)
cb.set_label('SAR (W/kg)', fontsize=12)

# Línea de Brezovich: H0 *  = 4.85e8  _limit = 4.85e8 / H0
nu_limit = 4.85e8 / H0
ax.axvline(nu_limit/1e3, color='red', linestyle='--', linewidth=2,
           label=f'Brezovich limit ~{nu_limit/1e3:.0f} kHz')

ax.set_xlabel('Frequency (kHz)', fontsize=12)
ax.set_ylabel('Diameter (nm)', fontsize=12)
```

```
ax.set_title('SAR heatmap: Diameter vs. Frequency\nwith Brezovich limit', font
ax.legend(loc='upper right')
plt.tight_layout()
plt.show()
-------------------------------------------------

import numpy as np
import matplotlib.pyplot as plt
from scipy.constants import mu_0

# --- Asume compute_SAR(frequencies, H0, T, D) definida para onda sinusoidal -

def traponic_correction(alpha):
    """Factor de primer armónico para trapezoidal, elevado al cuadrado."""
    x = np.pi * alpha / 2
    return (np.sin(x) / x)**2

# Parámetros fijos
mu0     = 4*np.pi*1e-7
H0      = 5e-3 / mu0     # A/m (5 mT)
T_val   = 300            # K
params = {
    'molar_M': 231.54e-3,
    'muB':      9.27e-24,
    'gamma0':   1.761e11,
    'K':        1.1e4,
    'alpha':    0.33,
    'conc':     10,        # mg/mL
    'eta':      1e-3,      # Paůs
    'Dh':       20e-9,     # m
    'density':  5170,      # kg/mş
    'D':        20e-9      # m (diámetro medio)
}

# Barrido de frecuencia y duty-cycle
freqs = np.linspace(50e3, 500e3, 100)    # Hz
dtys  = np.linspace(0.1, 0.9, 50)        # dutycycle

# Prealocar matriz
Z = np.zeros((len(dtys), len(freqs)))
```

```
# SAR sinusoidal base (vector para todas las freqs)
# Nota: compute_SAR debe aceptar array en su primer argumento
D_mean = params['D']
sar_base = compute_SAR(freqs, H0, T_val, D_mean)  # devuelve array de len(freq

# Rellenar Z multiplicando por el factor trapezoidal para cada
for i, alpha in enumerate(dtys):
    factor = traponic_correction(alpha)
    Z[i, :] = sar_base * factor

# Graficar heatmap
F, A = np.meshgrid(freqs / 1e3, dtys)  # kHz vs.
plt.figure(figsize=(7, 5))
pcm = plt.pcolormesh(F, A, Z, shading='auto', cmap='viridis')
cbar = plt.colorbar(pcm)
cbar.set_label('SAR (W/kg)', fontsize=12)

plt.xlabel('Frecuencia (kHz)', fontsize=12)
plt.ylabel('Dutycycle ', fontsize=12)
plt.title('SAR heatmap:  vs.  (trapezoidal)', fontsize=14)
plt.tight_layout()
plt.show()


----------------------------------------



import numpy as np
import matplotlib.pyplot as plt
from scipy.constants import mu_0, k as kB, N_A, pi

# --- Definición de compute_SAR (onda sinusoidal, modelo LRT) ---
def compute_SAR(frequencies, H0, T_val, params):
    muB    = params['muB']
    molar  = params['molar_M']
    rho    = params['density']
    alpha  = params['alpha']
    gamma0 = params['gamma0']
```

```
    K      = params['K']
    conc   = params['conc'] * 1e-3   # mg/mL -> kg/L
    eta    = params['eta']
    Dh     = params['Dh']
    D      = params['D']

    # Magnetización efectiva
    M = muB * (1e3 * N_A * rho / molar) * 4
    V = pi/6 * D**3
    tau0 = M * (alpha**2 + 1) / (2 * alpha * gamma0 * K)
    # Tiempo de Brown
    tauB = 3 * eta * (pi/6 * Dh**3) / (kB * T_val)

    SARs = []
    for f in frequencies:
        # Relaxación Néel
        x = np.clip(mu_0 * M * V * H0 / (3 * kB * T_val), -20, 20)
        y = np.clip(K * V / (kB * T_val), 1e-3, 200)
        ln_term = 0.5*(np.log(pi) - 0.5*np.log(y)) + y + x**2/(4*y)
        ln_term = np.clip(ln_term, None, 700)
        num = np.exp(ln_term)
        hyper = np.cosh(x) - 0.5*x*np.sinh(x)/y
        hyper = np.where(hyper <= 1e-6, 1e-6, hyper)
        denom = (1 - x**2/(4*y**2)) * hyper
        tauN = tau0 * num / denom

        # Tiempo efectivo
        tau = tauN * tauB / (tauN + tauB)

        # Susceptibilidad estática
        chi0 = (1/3) * conc * mu_0 * M**2 * V / (rho * kB * T_val)

        omega = 2 * pi * f
        sar = mu_0 * chi0 * H0**2 * omega * tau / (1 + (omega * tau)**2)
        SARs.append(sar)

    return np.array(SARs)

# --- Parámetros generales ---
mu0   = 4*np.pi*1e-7
```

```
H0    = 5e-3 / mu0      # A/m (5 mT)
T_val = 300             # K

params = {
    'molar_M': 231.54e-3,  # kg/mol
    'muB':      9.27e-24,  # J/T
    'gamma0':   1.761e11,  # szůTź
    'K':        1.1e4,     # J/mş
    'alpha':    0.33,
    'conc':     10,        # mg/mL
    'eta':      1e-3,      # Paůs
    'Dh':       20e-9,     # m
    'density':  5170       # kg/mş
}

# --- Grid de barrido ---
D_vals    = np.linspace(5e-9, 50e-9,  50)  # 50 puntos de 5 a 50 nm
freq_vals = np.linspace(50e3, 500e3, 100)  # 100 puntos de 50 a 500 kHz

# Capacidad calorífica total (suponemos 1 kg suspensión de agua): c = 4180 J/(
c_tot = 4180.0  # J/(kgůK)
dt    = 60.0    # s

# --- Cálculo de T ---
Z = np.zeros((len(D_vals), len(freq_vals)))
for i, D in enumerate(D_vals):
    params['D'] = D
    for j, f in enumerate(freq_vals):
        sar = compute_SAR([f], H0, T_val, params)[0]  # W/kg
        # T = (SAR [W/kg] * dt [s]) / c_tot [J/(kgůK)]
        Z[i, j] = sar * dt / c_tot

# --- Plot del heatmap ---
import matplotlib.ticker as mtick

fig, ax = plt.subplots(figsize=(8,6))
F, Dnm = np.meshgrid(freq_vals/1e3, D_vals*1e9)

pcm = ax.pcolormesh(F, Dnm, Z, shading='auto', cmap='viridis')
cbar = fig.colorbar(pcm, ax=ax)
```

```
cbar.set_label(r'$\Delta T$ (K)', fontsize=12)

ax.set_xlabel('Frequency (kHz)', fontsize=12)
ax.set_ylabel('Diameter (nm)', fontsize=12)
ax.set_title(r'$\Delta T$ heatmap: Diameter vs. Frequency after 60 s', fontsiz

# Opcional: formatear colorbar con 2 decimales
cbar.formatter = mtick.FormatStrFormatter('%.2f')
cbar.update_ticks()

plt.tight_layout()
plt.show()

----------------------------------------


import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import lognorm
from scipy.constants import mu_0, k as kB, N_A, pi

# --- Definición de compute_SAR (onda sinusoidal, LRT) ---
def compute_SAR(frequencies, H0, T_val, params):
    muB    = params['muB']
    molar  = params['molar_M']
    rho    = params['density']
    alpha  = params['alpha']
    gamma0 = params['gamma0']
    K      = params['K']
    conc   = params['conc'] * 1e-3   # mg/mL  kg/L
    eta    = params['eta']
    Dh     = params['Dh']
    D      = params['D']

    # Magnetización y volúmenes
    M = muB * (1e3 * N_A * rho / molar) * 4
    V = pi/6 * D**3
    tau0 = M * (alpha**2 + 1) / (2 * alpha * gamma0 * K)
    tauB = 3 * eta * (pi/6 * Dh**3) / (kB * T_val)
```

```
    SARs = []
    for f in frequencies:
        x = np.clip(mu_0 * M * V * H0 / (3 * kB * T_val), -20, 20)
        y = np.clip(K * V / (kB * T_val), 1e-3, 200)
        ln_term = 0.5*(np.log(pi) - 0.5*np.log(y)) + y + x**2/(4*y)
        ln_term = np.clip(ln_term, None, 700)
        num = np.exp(ln_term)
        hyper = np.cosh(x) - 0.5*x*np.sinh(x)/y
        hyper = np.where(hyper <= 1e-6, 1e-6, hyper)
        denom = (1 - x**2/(4*y**2)) * hyper
        tauN = tau0 * num / denom

        tau = tauN * tauB / (tauN + tauB)
        chi0 = (1/3) * conc * mu_0 * M**2 * V / (rho * kB * T_val)
        omega = 2 * pi * f
        sar = mu_0 * chi0 * H0**2 * omega * tau / (1 + (omega * tau)**2)
        SARs.append(sar)
    return np.array(SARs)

# --- Parámetros generales ---
mu0      = 4*np.pi*1e-7
T_val    = 300          # K
nu_fixed = 300e3        # Hz
dt       = 60           # s
c_tot    = 4180.0       # J/(kgůK)
D0       = 20e-9        # m (diámetro medio)

params = {
    'molar_M': 231.54e-3, 'muB': 9.27e-24, 'gamma0': 1.761e11,
    'K': 1.1e4, 'alpha': 0.33, 'conc': 10,
    'eta': 1e-3, 'Dh': 20e-9, 'density': 5170,
    'D': D0
}

# --- Grid de barrido ---
H0_mT = np.linspace(0.5, 10, 50)     # mT
sigmas = np.linspace(0.01, 0.5, 50)  # dispersión

Z = np.zeros((len(sigmas), len(H0_mT)))
```

```
# Puntos para malla log-normal
N = 200
for j, sigma in enumerate(sigmas):
    # Generar malla de diámetros
    Ds = lognorm(s=sigma, scale=D0).rvs(size=N) if False else np.linspace(D0*0
    # mejor usar PDF+trapecio:
    pdf = lognorm.pdf(Ds, s=sigma, scale=D0)
    weights = pdf / np.trapz(pdf, Ds)
    for i, HmT in enumerate(H0_mT):
        H0 = HmT * 1e-3 / mu0
        # Calcular SAR para cada Dk
        SARs = []
        for Dk in Ds:
            params['D'] = Dk
            SARs.append(compute_SAR([nu_fixed], H0, T_val, params)[0])
        SARs = np.array(SARs)
        SAR_media = np.trapz(SARs * weights, Ds)
        Z[j, i] = SAR_media * dt / c_tot

# --- Plot del heatmap ---
F, S = np.meshgrid(H0_mT, sigmas)
plt.figure(figsize=(8,6))
pcm = plt.pcolormesh(F, S, Z, shading='auto', cmap='viridis')
cbar = plt.colorbar(pcm)
cbar.set_label('T (K)', fontsize=12)

plt.xlabel(r'$\mu_0 H_0$ (mT)', fontsize=12)
plt.ylabel(' (log-normal)', fontsize=12)
plt.title('T heatmap: Field vs.  after 60 s', fontsize=14)
plt.tight_layout()
plt.show()

-----------------------------------------



import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.cm import ScalarMappable
```

```
from matplotlib.colors import Normalize
from scipy.constants import pi, k as kB, mu_0, N_A
import ipywidgets as widgets
from IPython.display import display, clear_output
import plotly.graph_objects as go

# Default physical parameters
def default_parameters():
    return {
        'molar_M': 231.54e-3,
        'muB': 9.27e-24,
        'gamma0': 1.761e11,
        'K': 1.1e4,
        'alpha': 0.33,
        'D': 20e-9,
        'conc': 10,
        'eta': 1e-3,
        'Dh': 20e-9,
        'density': 5170
    }

# Frequencies (Hz)
frecuencias = np.linspace(50e3, 500e3, 200)

# Compute SAR matrix
def compute_SAR_matrix(field_range, n_fields, T_val, params):
    M = params['muB']*(1e3*N_A*params['density']/params['molar_M'])*4
    tau0 = M*(params['alpha']**2+1)/(2*params['alpha']*params['gamma0']*params
    V = pi/6*params['D']**3
    Vh = pi/6*params['Dh']**3

    def tau_neel(T,H0):
        x = mu_0*M*V*H0/(3*kB*T)
        y = params['K']*V/(kB*T)
        y = np.minimum(y,200)
        ln = 0.5*(np.log(pi)-0.5*np.log(y)) + y + x**2/(4*y)
        ln = np.clip(ln,None,700)
        num=np.exp(ln)
        denom=(1-x**2/(4*y**2))*(np.cosh(x)-0.5*x*np.sinh(x)/y)
        return tau0*num/denom
```

```
    def tau_brown(T): return 3*params['eta']*Vh/(kB*T)
    def chi0(T): return (1/3)*params['conc']*1e-3*mu_0*M**2*V/(params['density

    campos = np.linspace(field_range[0],field_range[1],n_fields)
    SARs = np.zeros((len(frecuencias),n_fields))
    for j,Hm in enumerate(campos):
        H0 = Hm*1e-3/mu_0
        B = tau_brown(T_val)
        for i,f in enumerate(frecuencias):
            N = tau_neel(T_val,H0)
             = N*B/(N+B)
             = 2*pi*f
            SARs[i,j] = mu_0*chi0(T_val)*H0**2**/(1+(*)**2)
    return campos, SARs

# Widgets
styles = {'description_width': '120px'}
field_slider = widgets.FloatRangeSlider(value=[3.9,4.2], min=0.5, max=10, step
                                        description='H (mT)', style=styles, la
lines_slider = widgets.IntSlider(value=200, min=50, max=500, step=10, descript
T_slider = widgets.FloatSlider(value=300, min=280, max=330, step=1, descriptio
rho_slider = widgets.FloatSlider(value=default_parameters()['density'], min=30
                                 description='Densidad (kg/mş)', style=styles)
fiselect = widgets.FloatSlider(value=frecuencias[0]/1e3, min=frecuencias.min()
                               step=1, description='f selección (kHz)', style=
Hselect = widgets.FloatSlider(value=field_slider.value[0], min=field_slider.va
                              step=0.01, description='H selección (mT)', style
field_slider.observe(lambda ev: (setattr(Hselect,'min',ev['new'][0]), setattr(
                     names='value')

export_button = widgets.Button(description='Exportar CSV')
cb_brez = widgets.Checkbox(description='Mostrar línea Brezovich', value=False)
cb_heat = widgets.Checkbox(description='Mostrar Heatmap', value=False)
cb_3d = widgets.Checkbox(description='Mostrar 3D', value=False)

controls = widgets.VBox([field_slider, lines_slider, T_slider, rho_slider, fis
                         widgets.HBox([cb_heat, cb_3d, cb_brez, export_button]
out = widgets.Output()
```

```
# Storage for last result
_last = {}

# Export callback
def export_csv(_):
    if 'SARs' in _last:
        df = pd.DataFrame(_last['SARs'], index=frecuencias, columns=_last['cam
        df.to_csv('SAR_results.csv')
        print("CSV exportado como SAR_results.csv")
    else:
        print("No hay datos para exportar.")

export_button.on_click(export_csv)

# Update callback
def update(fr, nf, T, rho, fsel, Hsel, show_heat, show_3d, show_brez):
    with out:
        clear_output(wait=True)
        params = default_parameters()
        params['density'] = rho
        campos, SARs = compute_SAR_matrix(fr, nf, T, params)
        _last['campos'], _last['SARs'] = campos, SARs

        if show_3d:
            export_button.disabled = True
            fig3d = go.Figure(data=[go.Surface(z=SARs, x=campos, y=frecuencias
            fig3d.update_layout(scene=dict(xaxis_title='H (mT)', yaxis_title='
                                title=f'SAR 3D a T={T} K')
            display(fig3d)
        elif show_heat:
            export_button.disabled = False
            F, C = np.meshgrid(campos, frecuencias/1e3)
            fig, ax = plt.subplots(figsize=(8,6))
            pcm = ax.pcolormesh(F, C, SARs, shading='auto', cmap='viridis')
            fig.colorbar(pcm, ax=ax, label='SAR (W/kg)')
            ax.set_xlabel('H (mT)')
            ax.set_ylabel('Frecuencia (kHz)')
            if show_brez:
                H0_min = fr[0]
                nu_lim = 4.85e8 / (H0_min*1e-3/mu_0) / 1e3
```

```
                ax.axvline(nu_lim, linestyle='--', color='red', label='Límite
                ax.legend()
            ax.set_title(f'SAR Heatmap a T={T} K')
            plt.tight_layout()
            plt.show()
        else:
            export_button.disabled = False
            fig, ax = plt.subplots(figsize=(8,6))
            norm = Normalize(vmin=campos.min(), vmax=campos.max())
            sm = ScalarMappable(norm=norm, cmap='viridis')
            sm.set_array([])
            for j, Hm in enumerate(campos):
                ax.plot(frecuencias/1e3, SARs[:,j], color=sm.to_rgba(Hm), alph
            fig.colorbar(sm, ax=ax, label='H (mT)')
            ax.set_xlabel('Frecuencia (kHz)')
            ax.set_ylabel('SAR (W/kg)')
            idx_f = np.abs(frecuencias/1e3 - fsel).argmin()
            idx_H = np.abs(campos - Hsel).argmin()
            x = frecuencias[idx_f]/1e3
            y = SARs[idx_f, idx_H]
            ax.scatter(x, y, color='red', s=80, zorder=5)
            if show_brez:
                nu_lim = 4.85e8 / (Hsel*1e-3/mu_0) / 1e3
                ax.axvline(nu_lim, linestyle='--', color='red', label='Límite
                ax.legend()
            ax.set_title(f'SAR vs. Frec & H (T={T} K)')
            plt.tight_layout()
            plt.show()

widgets.interactive_output(update, {
    'fr': field_slider, 'nf': lines_slider, 'T': T_slider, 'rho': rho_slider,
    'fsel': fiselect, 'Hsel': Hselect,
    'show_heat': cb_heat, 'show_3d': cb_3d, 'show_brez': cb_brez
})

display(controls, out)
```

```
import plotly.io as pio
from plotly.offline import init_notebook_mode

# Inicializa modo offline y renderer Colab
init_notebook_mode(connected=True)
pio.renderers.default = 'colab'          # o 'colab_inline'

# Ahora tu código normal de Plotly
import numpy as np
from scipy.constants import pi, k as kB, mu_0, N_A
import plotly.graph_objects as go

# Parámetros y malla reducida para prueba rápida
def default_parameters():
    return {
        'molar_M': 231.54e-3, 'muB':9.27e-24, 'gamma0':1.761e11,
        'K':1.1e4, 'alpha':0.33, 'D':20e-9, 'Dh':20e-9,
        'conc':10, 'eta':1e-3, 'density':5170
    }
params = default_parameters()
T_val = 300
frecuencias = np.linspace(50e3, 500e3, 80)  # Hz
campos_mT   = np.linspace(1.0, 10.0, 40)    # mT

# Vectorización (como en el ejemplo que te pasé)
M    = params['muB']*(1e3*N_A*params['density']/params['molar_M'])*4
tau0 = M*(params['alpha']**2+1)/(2*params['alpha']*params['gamma0']*params['K'
V    = pi/6 * params['D']**3
Vh   = pi/6 * params['Dh']**3
F = frecuencias[:,None]
H = campos_mT[None,:]*1e-3/mu_0
B = 3*params['eta']*Vh/(kB*T_val)
x  = np.clip(mu_0*M*V*H/(3*kB*T_val), -20, 20)
y  = np.clip(params['K']*V/(kB*T_val), 1e-3, 200)
ln = np.clip(0.5*(np.log(pi)-0.5*np.log(y)) + y + x**2/(4*y), None, 700)
```

```
num   = np.exp(ln)
denom = (1 - x**2/(4*y**2))*(np.cosh(x)-0.5*x*np.sinh(x)/y)
denom = np.where(np.abs(denom)<1e-6, 1e-6, denom)
N     = tau0 * num / denom
_eff = N * B / (N + B)
      = 2*np.pi*F
0     = (1/3)*params['conc']*1e-3*mu_0*M**2*V/(params['density']*kB*T_val)
Z      = mu_0 * 0 * (H**2) *  * _eff / (1 + (*_eff)**2)

# Plot 3D interactivo
fig = go.Figure(data=[ go.Surface(
    x=frecuencias/1e3,
    y=campos_mT,
    z=Z.T,
    colorscale='Viridis'
)])
fig.update_layout(
    title=f'SAR 3D vs Frecuencia y Campo (T={T_val} K)',
    scene=dict(xaxis_title='Frecuencia (kHz)', yaxis_title='H (mT)', zaxis_tit
    width=800, height=600
)
fig.show()
------------------------



import numpy as np
from scipy.constants import pi, k as kB, mu_0, N_A
import plotly.graph_objects as go
import plotly.io as pio
from plotly.offline import init_notebook_mode

# 1) Inicializa Plotly Colab
init_notebook_mode(connected=True)
pio.renderers.default = 'colab'

# 2) Parámetros físicos
params = {
    'molar_M': 231.54e-3,
    'muB':      9.27e-24,
```

```
    'gamma0':   1.761e11,
    'K':        1.1e4,
    'alpha':    0.33,
    'D':        20e-9,
    'conc':     10,
    'eta':      1e-3,
    'Dh':       20e-9,
    'density':  5170
}
T_val = 300  # K

# 3) Define ejes idénticos al heatmap
H0_mT    = np.linspace(3.9, 4.2, 200)      # mT
freq_kHz = np.linspace(50, 500, 200)       # kHz
F = freq_kHz * 1e3                         # a Hz para cálculos
H = H0_mT * 1e-3 / mu_0                     # a A/m

# 4) Vectoriza tu compute_SAR (idéntico al heatmap)
M    = params['muB']*(1e3*N_A*params['density']/params['molar_M'])*4
tau0 = M*(params['alpha']**2+1)/(2*params['alpha']*params['gamma0']*params['K'
V    = pi/6 * params['D']**3
Vh   = pi/6 * params['Dh']**3
chi0 = (1/3)*params['conc']*1e-3 * mu_0*M**2*V/(params['density']*kB*T_val)
tauB = 3*params['eta']*Vh/(kB*T_val)

# Creamos la malla 2D
FF = F[:, None]       # (200,1)
HH = H[None, :]       # (1,200)

# _neel estable
x = np.clip(mu_0*M*V*HH/(3*kB*T_val), -20, 20)
y = np.clip(params['K']*V/(kB*T_val), 1e-3, 200)
ln = np.clip(0.5*(np.log(pi)-0.5*np.log(y)) + y + x**2/(4*y), None, 700)
num   = np.exp(ln)
denom = (1 - x**2/(4*y**2))*(np.cosh(x) - 0.5*x*np.sinh(x)/y)
denom = np.where(np.abs(denom)<1e-6, 1e-6, denom)
tauN  = tau0 * num / denom

# _eff y SAR final
tau_eff = tauN * tauB / (tauN + tauB)
```

```
omega    = 2*np.pi * FF
Z        = mu_0 * chi0 * (HH**2) * omega * tau_eff / (1 + (omega*tau_eff)**2)

# 5) Surface Plotly
fig = go.Figure(data=[go.Surface(
    x=freq_kHz,     # kHz
    y=H0_mT,        # mT
    z=Z.T,          # transpuesta
    colorscale='Viridis',
    colorbar=dict(title='SAR (W/kg)')
)])
fig.update_layout(
    title=f'SAR 3D vs Frecuencia y Campo (T={T_val} K)',
    scene=dict(
        xaxis_title='Frecuencia (kHz)',
        yaxis_title='H (mT)',
        zaxis_title='SAR (W/kg)'
    ),
    width=800, height=600
)
fig.show()
--------------------------------------


import numpy as np
import plotly.graph_objects as go
from scipy.constants import pi, k as kB, mu_0, N_A

# Default parameters
params = {
    'molar_M': 231.54e-3,
    'muB': 9.27e-24,
    'K': 1.1e4,
    'D0': 20e-9,
    'conc': 10,
    'eta': 1e-3,
    'Dh': 20e-9,
    'density': 5170,
    'tau0': 1e-9
}
```

```
# SAR computation
def compute_SAR(freqs, H0, D, sigma, alpha, T=300):
    V = pi/6 * D**3
    Vh = pi/6 * params['Dh']**3
    M = params['muB']*(1e3*N_A*params['density']/params['molar_M'])
    tauN = params['tau0'] * np.exp(params['K'] * V / (kB * T))
    tauB = 3 * params['eta'] * Vh / (kB * T)
    tau = tauN * tauB / (tauN + tauB)
    chi0 = (1/3) * params['conc'] * 1e-3 * mu_0 * M**2 * V / (params['density'
    H1 = H0 * np.sin(pi * alpha / 2) / (pi * alpha / 2)
    omega = 2 * pi * freqs
    return mu_0 * chi0 * H1**2 * (omega * tau) / (1 + (omega * tau)**2)

# Grids
freqs = np.linspace(50e3, 500e3, 50)
Hs = np.linspace(1, 10, 50) * 1e-3 / mu_0
Ds = np.linspace(5, 50, 50) * 1e-9
sigmas = np.linspace(0.1, 0.5, 50)
alphas = np.linspace(0.1, 0.9, 50)

# Surfaces
z_df = np.array([compute_SAR(freqs, Hs[-1], D, sigmas[0], alphas[0]) for D in
z_cd = np.array([compute_SAR(freqs[-1] * np.ones_like(Hs), H, Ds[0], s, alphas
z_fd = np.array([compute_SAR(freqs, Hs[-1], Ds[0], sigmas[0], a) for a in alph

# Colorbar settings
cbar = dict(title='SAR (W/kg)', len=0.6, x=0.7, thickness=15)

# Create figure
fig = go.Figure([
    go.Surface(z=z_df, x=freqs/1e3, y=Ds*1e9, colorscale='Viridis', visible=Tr
    go.Surface(z=z_cd, x=Hs*1e3*mu_0, y=sigmas, colorscale='Viridis', visible=
    go.Surface(z=z_fd, x=freqs/1e3, y=alphas, colorscale='Viridis', visible=Fa
])

# Dropdown
updatemenus = [dict(
    buttons=[
        dict(label='Diámetro vs Frecuencia',
```

```
                method='update',
                args=[{'visible': [True, False, False]},
                      {'scene': {'xaxis': {'title': 'Frecuencia (kHz)'},
                                 'yaxis': {'title': 'Diámetro (nm)'},
                                 'zaxis': {'title': 'SAR (W/kg)'}}}]),
            dict(label='Campo vs Dispersión',
                method='update',
                args=[{'visible': [False, True, False]},
                      {'scene': {'xaxis': {'title': 'Campo (mT)'},
                                 'yaxis': {'title': 'Dispersión '},
                                 'zaxis': {'title': 'SAR (W/kg)'}}}]),
            dict(label='Frecuencia vs Duty-cycle',
                method='update',
                args=[{'visible': [False, False, True]},
                      {'scene': {'xaxis': {'title': 'Frecuencia (kHz)'},
                                 'yaxis': {'title': 'Duty-cycle '},
                                 'zaxis': {'title': 'SAR (W/kg)'}}}])
        ],
        direction='down',
        showactive=True,
        x=0.1,
        xanchor='left',
        y=1.05,
        yanchor='top'
    )]

    # Layout
    fig.update_layout(
        title={'text': 'Mapas 3D de SAR', 'x': 0.5, 'y': 0.95},
        updatemenus=updatemenus,
        margin={'t': 120}
    )

    fig.show()
    ----------------------------------

    import numpy as np
    import plotly.graph_objects as go
    from scipy.constants import pi, k as kB, mu_0, N_A
```

```
# Parámetros físicos por defecto
params = {
    'molar_M': 231.54e-3,    # kg/mol
    'muB': 9.27e-24,         # J/T
    'K': 1.1e4,              # J/m^3
    'D0': 20e-9,             # m
    'conc': 10,              # mg/mL (convertido internamente)
    'eta': 1e-3,             # Paůs
    'Dh': 20e-9,             # m
    'density': 5170,         # kg/m^3
    'tau0': 1e-9             # s
}

# Función de cálculo de SAR
def compute_SAR(freqs, H0, D, sigma, alpha, T=300):
    V = pi/6 * D**3
    Vh = pi/6 * params['Dh']**3
    M = params['muB'] * (1e3 * N_A * params['density'] / params['molar_M'])
    tauN = params['tau0'] * np.exp(params['K'] * V / (kB * T))
    tauB = 3 * params['eta'] * Vh / (kB * T)
    tau = tauN * tauB / (tauN + tauB)
    chi0 = (1/3) * params['conc'] * 1e-3 * mu_0 * M**2 * V / (params['density'
    # Primer armónico para onda trapezoidal
    H1 = H0 * np.sin(pi * alpha / 2) / (pi * alpha / 2)
    omega = 2 * pi * freqs
    return mu_0 * chi0 * H1**2 * (omega * tau) / (1 + (omega * tau)**2)

# Definición de rejillas
freqs = np.linspace(50e3, 500e3, 50)     # Hz
Hs = np.linspace(1, 10, 50) * 1e-3 / mu_0 # A/m (convertido desde mT)
Ds = np.linspace(5, 50, 50) * 1e-9        # m
sigmas = np.linspace(0.1, 0.5, 50)        # adimensional
alphas = np.linspace(0.1, 0.9, 50)        # adimensional

# Cálculo de superficies
z_df = np.array([compute_SAR(freqs, Hs[-1], D, sigmas[0], alphas[0]) for D in
z_cd = np.array([compute_SAR(freqs[-1] * np.ones_like(Hs), H, Ds[0], s, alphas
z_fd = np.array([compute_SAR(freqs, Hs[-1], Ds[0], sigmas[0], a) for a in alph

# Configuración de la barra de color (más cerca)
```

```
cbar = dict(title='SAR (W/kg)', len=0.6, x=0.75, thickness=15)

# Construcción de la figura con las tres superficies
fig = go.Figure([
    go.Surface(z=z_df, x=freqs/1e3, y=Ds*1e9, colorscale='Viridis', visible=Tr
    go.Surface(z=z_cd, x=Hs*1e3*mu_0, y=sigmas, colorscale='Viridis', visible=
    go.Surface(z=z_fd, x=freqs/1e3, y=alphas, colorscale='Viridis', visible=Fa
])

# Menú desplegable con actualizaciones de ejes y visibilidad
updatemenus = [dict(
    buttons=[
        dict(label='Diámetro vs Frecuencia',
             method='update',
             args=[
                 {'visible': [True, False, False]},
                 {'scene': {
                     'xaxis': {'title': 'Frecuencia (kHz)', 'tickformat': '.0f
                     'yaxis': {'title': 'Diámetro (nm)', 'tickformat': '.1f'},
                     'zaxis': {'title': 'SAR (W/kg)', 'tickformat': '.1f'}
                 }}
             ]),
        dict(label='Campo vs Dispersión',
             method='update',
             args=[
                 {'visible': [False, True, False]},
                 {'scene': {
                     'xaxis': {'title': 'Campo (mT)', 'tickformat': '.1f'},
                     'yaxis': {'title': 'Dispersión  (sin unidad)'},
                     'zaxis': {'title': 'SAR (W/kg)', 'tickformat': '.1f'}
                 }}
             ]),
        dict(label='Frecuencia vs Duty-cycle',
             method='update',
             args=[
                 {'visible': [False, False, True]},
                 {'scene': {
                     'xaxis': {'title': 'Frecuencia (kHz)', 'tickformat': '.0f
                     'yaxis': {'title': 'Duty-cycle  (fracción)'},
                     'zaxis': {'title': 'SAR (W/kg)', 'tickformat': '.1f'}
```

```
                }}
            ])
    ],
    direction='down',
    showactive=True,
    x=0.1,
    xanchor='left',
    y=1.05,
    yanchor='top'
)]

# Ajustes finales del layout
fig.update_layout(
    title={'text': 'Mapas 3D de SAR', 'x': 0.5, 'y': 0.95},
    updatemenus=updatemenus,
    margin={'t': 120}
)

fig.show()
```