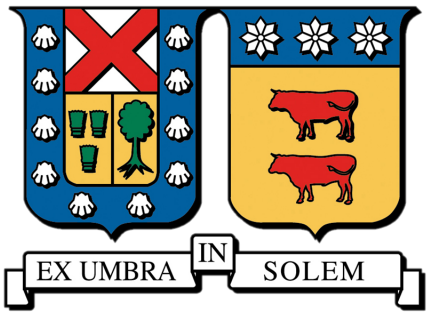
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INGENIERÍA MECÁNICA

VALPARAÍSO - CHILE



ESTUDIO DE LAS PROPIEDADES DE MICROHILOS FERROMAGNÉTICOS SOMETIDOS A RADIACIÓN DE MICROONDAS, Y LA VALIDACIÓN DE UNA HERRAMIENTA COMPUTACIONAL PARA SU SIMULACIÓN.

FELIPE ROSSI KARELOVIC

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO MENCIÓN ENERGÍA

PROFESOR GUÍA: CHRISTOPHER COOPER VILLAGRÁN

PROFESOR CORREFERENTE: ALEJANDRO PACHECO SANJUAN

MARZO - 2017

Agradecimientos

A mi familia por su apoyo constante e incondicional en cada paso que he dado en mi vida y en los duros años de carrera, dándome todas las herramientas necesarias para éstos. A mi profesor Christopher Cooper y a mi compañero y amigo Milan Ungerer, por su cooperación incansable y apoyo en la realización del presente trabajo y vida universitaria.

Resumen

Los microhilos ferromagnéticos tienen propiedades que varían en función de la tensión mecánica, por lo que se han presentado como una herramienta promisoria para el desarrollo de materiales inteligentes, capaces de entregar información del estado de tensiones internas. Gracias a esta variación de sus propiedades, es posible medir su cambio con algún campo externo, y asi establecer una relación directa. Sin embargo, las propiedades de permitividad y permeabilidad de los hilos están lejos de ser comprendidas a cabalidad, dificultando el uso de herramientas computacionales para su simulación. Sus valores dependen altamente de varios factores que se deben tener en cuenta y considerar, o en su defecto, establecer si pueden despreciarse.

El objetivo del presente trabajo es utilizar simulaciones numéricas para determinar las propiedades antes mencionadas (permeabilidad y permitividad) de los microhilos ferromagnéticos para luego ser comparados con los resultados experimentales, para luego validar su efectividad.

Para su realización, se deberán estudiar las propiedades electromagnéticas de los microhilos en función de diferentes variables, como lo son frecuencia, tensión, campo externo, GMI, etc. Se implementará un código de resolución de acuerdo al método de elementos de borde con una modelación de un problema de dispersión de ondas electromagnéticas con las ecuaciones de Maxwell y Helmholtz. Se diseñarán mallas para el trabajo de los microhilos y su simulación y se utilizarán en el código para generar un protocolo numérico-experimental para determinar la permitividad y permeabilidad de los microhilos ferromagnéticos. Finalmente, se compararán los resultados con otros modelos y datos experimentales, dando cuenta de la validez de los modelos computacionales y las limitaciones presentes.

Palabras claves: microhilos ferromagnéticos, propiedades microondas, permeabilidad, permitividad, BEM, BEM++, dispersión electromagnética.

Abstract

Ferromagnetic microwires have properties that change due to mechanical stress, so they have been presented as a promising tool for the development of inteligent materials capable of providing information about the condition of internal stress. Thanks to this variation of those properties, it is posible to measure how it changes because of an external field, and thus, establish a direct relation. However, microwire properties like permeability and permittivity are far from being fully understood, making it difficult to use computational tools for its simulation. These values depend strongly of several factors that must be taken into account, or in some cases, decide if they can be neglected.

The objective of this work is to use numerical simulations to determine the properties mentioned before (permeability and permittivity) of the ferromagnetic microwires, so they can be compared with the experimental results to validate their effectiveness.

For its realization, the electromagnetic properties of the microwires must be studied according to several variables, such as frecuency, stress, external fields, GMI, etc. A computational code will be implemented using the Boundary Elements Method with the devolopment of a model containing an electromagnetic scattering problem with Maxwell and Helmholtz equations. Meshes will be designed for the microwires and its simulation, and will be used in the code to generate a numerical-experimental protocole to determine the permeabilidad and permittivity of the ferromagnetic microwires. Finally, the obtained results will be compared with other models and experimental data, establishing the validity of the computational methods and their limitations.

Keywords: ferromagnetic wires, microwave properties, permeability, permittivity, BEM, BEM++, electromagnetic scattering.

Glosario

- **Radiación electromagnética:** tipo de campo electromagnético variable, es decir, una combinación de campos eléctricos y magnéticos oscilantes, que se propagan a través del espacio transportando energía de un lugar a otro.
- **Dispersión:** es el fenómeno por el cual un conjunto de partículas que se mueve en una dirección determinada rebota sucesivamente con las partículas del medio por el que se mueve hasta perder una dirección privilegiada de movimiento.
- **Ferromagnético:** compuestos de hierro y sus aleaciones con cobalto, tungsteno, níquel, aluminio y otros metales, son los materiales magnéticos más comunes.
- **Dieléctrico:** corresponde a un material con una baja conductividad eléctrica, o en términos más simples, un aislante.
- **Conductor:** material con una alta conductividad eléctrica o térmica.
- **Microhilo:** hilo con diámetro de algunos micrones.
- **Microondas:** ondas electromagnéticas; generalmente de entre 300 [MHz] y 30 [GHz] de frecuencia.
- **Anisotropía:** es la propiedad general de materia según las cualidades como: elasticidad, temperatura, conductividad, etc. Cuando se habla de anisotropía magnética, es la dependencia direccional con respecto a propiedades magnéticas.
- **Impedancia:** es la resistencia de un circuito o material al flujo de una corriente eléctrica alterna.
- **Impedancia magnética:** aparición de impedancia en un material debido al efecto causado por un campo magnético.
- **Impedancia magnética gigante:** se define como una larga variación en la impedancia magnética que ocurre en un conductor suave que conduce una corriente alterna cuando este es sometido a un campo magnético externo.
- **Magnetosctricción:** propiedad de los materiales magnéticos que hace que estos cambien de forma al encontrarse en presencia de un campo magnético.
- **Polarización:** es una propiedad de las ondas que pueden oscilar con más de una orientación. En una onda electromagnética, tanto el campo eléctrico y el campo magnético son oscilantes, pero en diferentes direcciones; ambas perpendiculares entre si y perpendiculares a la dirección de propagación de la onda.
- **Efecto pelicular o ”skin effect”:** efecto que se da cuando la densidad de corriente no es la misma en todo el conductor y se observa que hay una mayor densidad de corriente en la superficie que en el centro, cuando se tiene corriente alterna.

- **Frecuencia angular:** se refiere a la frecuencia del movimiento circular expresada en proporción del cambio de ángulo.
- **Compósito:** se refiere a dos o más materiales en conjunto, para el cual las características cambian y debe tratarse como un todo.
- **BEM:** método de elementos de borde (del inglés, Boundary Elements Method).
- **BEM++:** libreria de uso libre codificada con lenguaje C++, pero que permite el uso de distintos lenguajes, entre ellos Python.
- **Python:** lenguaje de programación que facilita una sintaxis para un código legible.
- **SLPO:** Single Layer Potential Operator (Operador potencial de capa simple)
- **DLPO:** Double Layer Potential Operator (Operador potencial de doble capa)
- **HYP:** Hypersingular Operator (Operador Hipersingular)
- **ADJ:** Adjoint Operator (Operador Adjunto)
- **DLADJ:** Adjoint Double Layer Boundary Operator (Operador adjunto de borde de doble capa)

Índice

1. Introducción

Recientes avances tecnológicos e industriales en lo que concierne a sensores y dispositivos magnéticos, requieren materiales con propiedades magnéticas excepcionales. Como un fuerte candidato, los microhilos ferromagnéticos amorfos han sido estudio durante décadas debido a su propiedades de sensibilidad. Su respuesta electromagnética puede ser medida convenientemente a través del diseño de su composición química y tamaño, sobre todo en un espectro de microondas, alterándose gracias a estímulos externos, e.g. campos magnéticos y estrés mecánico. Estos efectos, proveen variadas aplicaciones, tales como absorción de microondas, monitoreo de salud estructural (debido a que las propiedades ferromagnéticas de los hilos se ven directamente afectadas con un esfuerzo aplicado), entre otras.

En general se les clasifica en microhilos basados en Fe y en Co, siendo estos dos los componentes que dan mejores propiedades electromagnéticas con las microondas incidentes. Se han hechos numerosos esfuerzos en fabricar hilos con componentes químicos diferentes para mejorar sus propiedades, pero probó ser muy costoso.

Si bien los hilos proveen una buena sensibilidad magnética, cabe destacar que existe un problema en cuanto a su uso, debido a que el diámetro de los hilos es de unas decenas de micras, lo que implica que éstos posean una naturaleza frágil respecto a fracturas, limitando sus aplicaciones. Por lo que una de las incógnitas de las investigaciones actuales consiste en averiguar cómo fabricar microhilos así de pequeños, sin alterar sus propiedades magnéticas.

Recientemente, se ha estado utilizando una técnica que consiste en posicionar los microhilos en un compuesto basado en polímeros, el cual a la vez permite mejorar el rendimiento de los hilos al aumentar su número y preservarlos de mejor manera en una matriz. Este método logra solucionar de buena forma el problema anteriormente mencionado.

Aún así, el conocimiento actual respecto de los microhilos consiste únicamente en estudios empíricos, con ciertas propiedades y ciertos valores. Es por eso que se genera una limitante en cuanto a qué sucede cuando se van variando factores del experimento, como lo son permeabilidad y permitividad del hilo, dimensión del hilo, y sobre todo, la frecuencia de incidencia del campo magnético.

El objetivo del presente trabajo, es generar una herramienta analítica con aproximaciones matemáticas en un código computacional, el cual permita realizar pruebas con numerosos hilos en un compuesto, y así poder estudiar la variación del campo electromagnético en sus alrededores, confirmando así estudios empíricos respecto a sus propiedades. Para esto es necesario generar un código capaz de solucionar estas aproximaciones, para luego llevarlo a su uso con las geometrías necesarias, y determinar una manera sencilla de validar los experimentos que se han realizado a lo largo del tiempo, generando una herramienta de fácil uso y rápida respuesta. De esta manera se podrá determinar como se ve afectado el campo electromagnético por los hilos, teniendo una amplia gama de valores para utilizar.

2. Planteamiento del problema

Como se mencionó anteriormente, el uso de microhilos para la medición de la variación de campos electromagnéticos es de gran utilidad para el presente ingenieril, permitiendo realizar numerosas aplicaciones, entre las cuales se destaca la medición de esfuerzos mecánicos. Teniendo esto en cuenta, se pretende estudiar un compósito que contenga numerosos hilos para su estudio y análisis. El campo electromagnético se hará pasar a través de este compósito de hilos como se muestra en la figura (1).

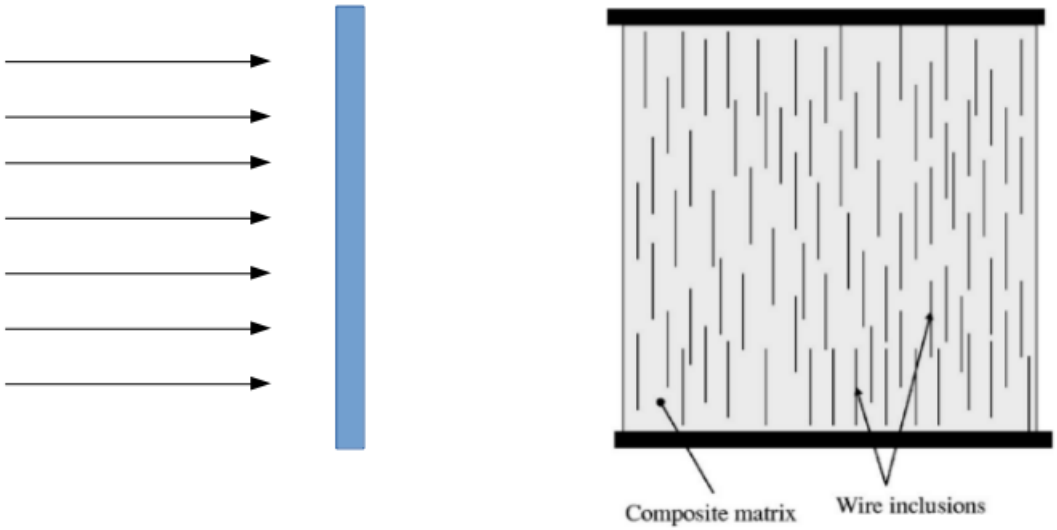


Figura 1: Compósito de hilos.

Una vez que el campo atraviese el compósito, se verá afectado por los microhilos que están en él, y al ubicar un receptor a una distancia deseada posterior al compósito, se podrá medir el valor del campo electromagnético y ver la variación de este con cada variación del compósito, ya sea tanto en cantidad de hilos como en su composición. Como se verá más adelante, las configuraciones de hilos irán variando, al igual que su número, siempre y cuando se mantenga una proporción de hilos en el compósito determinada, que en este caso corresponde al 0,0022 % y 0,01 % en volumen. Ya que el objetivo es medir el campo electromagnético, se desea que el cambio sea el mayor para tener resultados claros.

Si bien lo mencionado anteriormente no representa mayor problema, este se genera en lo que respecta a las propiedades magnéticas de los microhilos y la capacidad computacional con la que se cuenta. Esto sucede porque la permitividad y permeabilidad, dos variables necesarias para la simulación, dependen fuertemente del material que compone al hilo, de la frecuencia a la que se está trabajando el campo electromagnético, de la dimensión del hilo, de estímulos externos, entre otras. Por lo tanto, es de suma dificultad poseer estos valores para el caso que se quiera estudiar. Se deben conocer estos valores tanto para los hilos como para el material del compósito. Es por eso que se requiere diseñar una herramienta que sea capaz de validar los valores que se han obtenido empíricamente, para poder así extrapolar a otras configuraciones y frecuencias, y así finalmente

obtener el valor de estas propiedades para su uso.

2.1. Microhilos

Los microhilos que se utilizarán y estudiarán, corresponden a un núcleo conformado por el material amorfo y magnético, principalmente compuesto de CoFeCrSiB, del cual un 70-80 % corresponde a FeCo, y una cobertura de vidrio, la cual no tiene gran influencia sobre el efecto generado en el campo electromagnético, pero si en la protección de los hilos en cuanto a corrosión.

Estos hilos se preparan bajo un método de solidificación rápida, en un proceso rápido que permite la fabricación de 400 [m/min] de hilo de hasta 15 micras de diámetro.

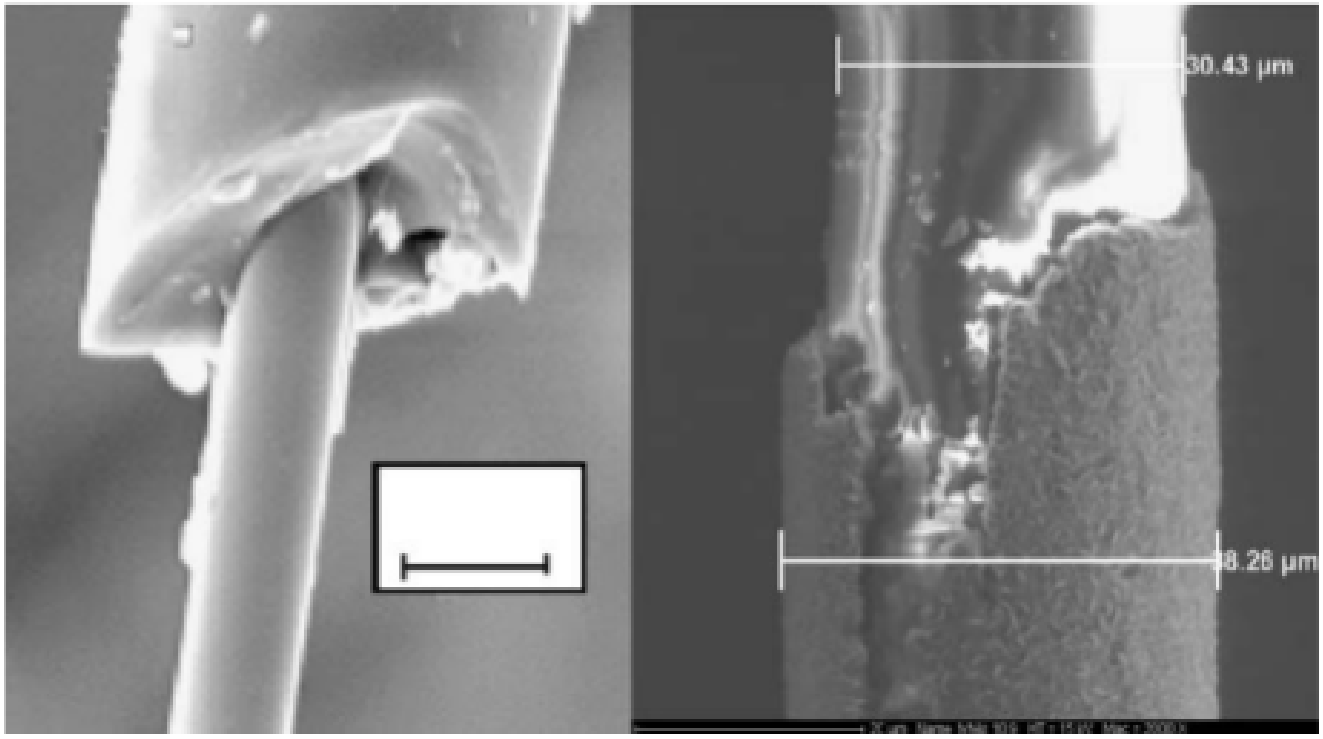


Figura 2: Microhilo cubierto de pyrex.

Como se mencionó anteriormente, estos hilos poseen cualidades únicas en cuanto a sus propiedades ferromagnéticas. La anisotropía magnética determina la respuesta general de cualquier material magnético por excitación de un campo electromagnético, particularmente en altas frecuencias. La rápida solidificación de los microhilos no solo determina su forma cilindrica, si no que tambien su naturaleza amorfa. En consecuencia, la anisotropía magnética de los hilos tiene una contribución uniaxial, la cual depende fuertemente de su magnetostricción, que tambien se ve determinada en su proceso de fabricación.

La forma longitudinal de los microhilos determina además sus propiedades, las que se diferencian en gran cantidad de materiales amorfos con forma de lazos o películas. Es por esto, que su orientación respecto a la llegada del campo magnético es primordial.

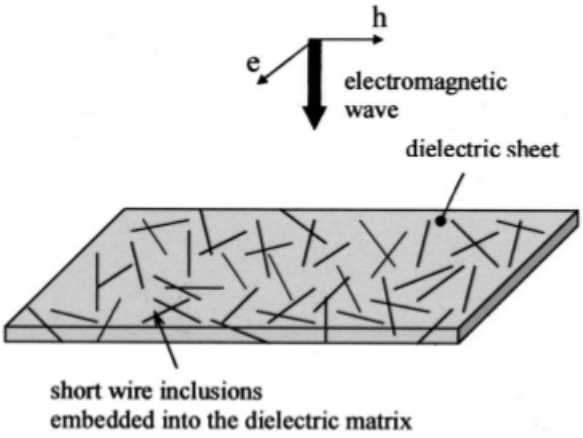


Figura 3: Compósito sometido a una onda electromagnética.

2.2. Permeabilidad y Permitividad

Los microhilos contienen importantes propiedades que determinan como se ve afectado el campo electromagnético irradiado sobre ellos. Las dos principales corresponden a la permeabilidad y permitividad magnética.

La primera indica la medida de la habilidad de un material para soportar la formación de un campo magnético sobre sí mismo, i.e. es el grado de magnetización que un material obtiene en respuesta a la aplicación de un campo magnético. Se representa con la letra griega μ (mu).

Se utiliza generalmente como permeabilidad relativa:

$$\mu_r = \frac{\mu}{\mu_0}, \tag{2.1}$$

donde $\mu_0 = 4\pi \times 10^7 [NA^{-2}]$ es la permeabilidad del vacío.

La segunda se refiere a la medida de la resistencia que es encontrada cuando se forma un campo eléctrico en un medio, i.e. la permitividad es una medida de como un campo eléctrico afecta, y es afectado por, un medio dieléctrico. Más flujo eléctrico existe en un medio de baja permitividad debido a efectos de polarización. Esta derechamente relacionada con la susceptibilidad eléctrica, que indica que tan fácil se polariza un objeto en respuesta a un campo eléctrico. Se representa con la letra griega ε.

Al igual que la permeabilidad, es común utilizarla como permitividad relativa:

$$\varepsilon_r = \frac{\varepsilon}{\varepsilon_0}, \tag{2.2}$$

donde $\varepsilon_0 = 8,8541878176 \times 10^{-12} [F/m]$ corresponde a la permitividad del vacío.

2.3. Impedancia Magnética

En el rango de frecuencias altas (hasta algunos GHz), el campo magnético que llega al hilo debe ser manejado en conjunto con el campo eléctrico, como se muestra en la Tabla 4.

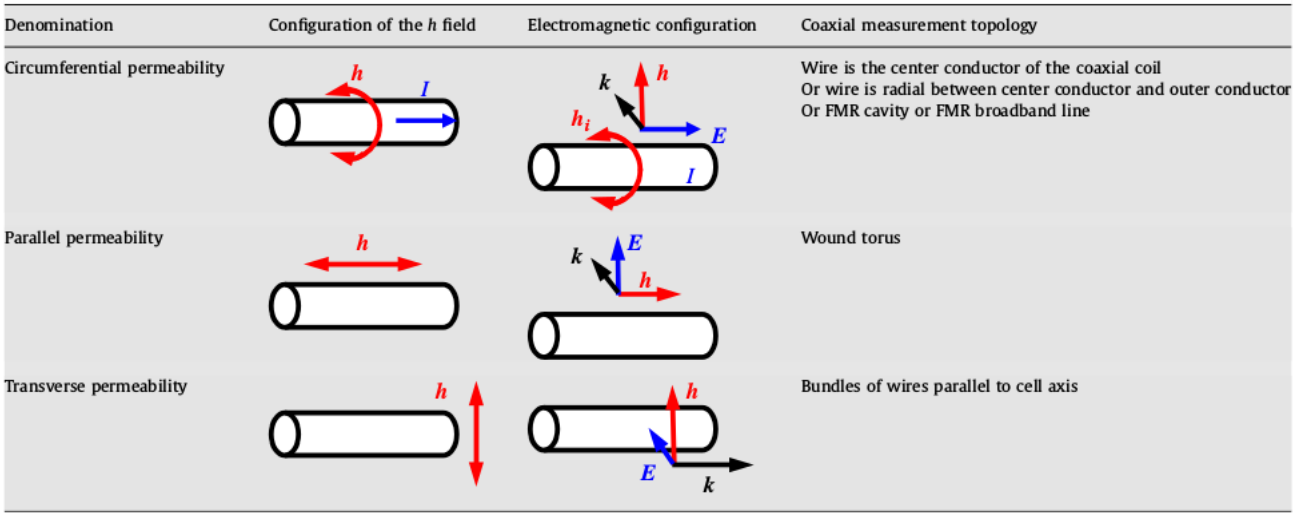


Figura 4: Configuraciones del campo electromagnético en el rango de microondas.

Al tener distintas configuraciones de incidencia de campo y orientación de hilos, se genera un efecto importante llamado Impedancia Magnética Gigante o GMI. La GMI se define como una larga variación en la impedancia magnética que ocurre en un conductor suave que conduce una corriente alterna cuando este es sometido a un campo magnético externo. Esta corriente que se genera produce un campo magnético circular en la superficie del microhilo, lo que modifica la corriente eléctrica, superponiéndose. Este fenómeno es causado por la dependencia de la permeabilidad del material con el campo magnético, la cual causa variaciones en la profundidad de penetración de las corrientes que fluyen por el objeto. La superposición concuerda con la solución a la ecuación de Maxwell para la densidad de corriente, que se explica en secciones posteriores. Este efecto se conoce también como “skin effect”.

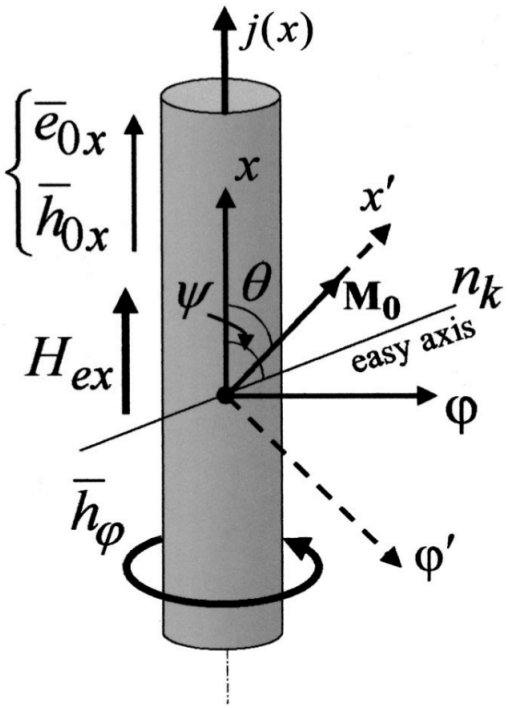


Figura 5: Esquema de estímulos externos sobre un microhilo y campo magnético circular generado por impedancia magnética.

Estos efectos deben ser tomados en consideración cuando se estudian las propiedades fundamentales de estos hilos, que para este caso en particular corresponden a la permeabilidad y permitividad del hilo, debido a que son capaces de modificarlas de manera abrupta.

2.4. Propiedades del compósito

Una vez que se integran los microhilos al compósito, ya no se pueden tratar sus componentes por separado. Tanto el material del compósito como los hilos tienen sus propiedades, por lo que es necesario compactarlas en una sola geometría que represente de buena forma sus partes por separado. Es por eso que al estudiar un compósito se deja de hablar de permeabilidad y permitividad del o los hilos y se comienza a hablar de "permeabilidad y permitividad efectiva", la cual congrega todas las propiedades en una generalización.

Los compósitos compuestos de microhilos pueden y deben ser tratados como un medio continuo y luego ser caracterizados según la permeabilidad efectiva. Si el "skin effect" es fuerte, la permitividad efectiva, comúnmente denominada ε_{ef}, es determinada por la geometría del hilo, la concentración de hilos y la permitividad del material que "almacena" a los hilos. Se ha demostrado que la cantidad necesaria de microhilos depende de la impedancia, la que involucra la permeabilidad μ del hilo y su estructura magnética. En compósitos que contengan microhilos ferromagnéticos con impedancias magnéticas en frecuencias de microondas, la permitividad efectiva ε_{ef} podría depender de un campo magnético estático de acuerdo a su dependencia con la impedancia en la superficie de los microhilos. Por lo tanto, ε_{ef} dependería de un estrés externo o esfuerzo.

Además, otra importante propiedad a considerar es la permeabilidad efectiva del compósito, comúnmente denomidad μ_{ef}.

Estas propiedades mencionadas no son constantes, y como se dijo anteiormente, dependen de numerosos factores. Para el cálculo de la variación del campo electromagnético aplicado en el compósito de microhilos, se necesitan las permeabilidades y permitividades básicas y efectivas, por lo que es de suma importancia contar con una herramienta para su cálculo. Es por eso que se intentará establecer un código computacional, como se mencionó antes, para lograr simular estos valores para distintas frecuencias y condiciones.

3. Software utilizado

Para el modelamiento del problema se requieren distintos software. Es importante mencionar además que estos fueron utilizados en el OS Linux 16.04, por lo que no se asegura compatibilidad con distintos sistemas operativos:

- *Anaconda v4.3.1:* este software corresponde a un paquete de distribución, el cual contiene herramientas para procesamiento de datos de larga escala, en lenguaje Python y R, capaz de proveer análisis predictivo y computamiento científico, simplificando el manejo de paquetes. Utiliza una versión de hojas de trabajo para Python v2.7.

- *BEM++ v3.0.3:* corresponde a una biblioteca de fuente abierta de elementos de borde de Galerkin, capaz de resolver problemas de borde con Laplace, Helmholtz y Maxwell. En las secciones posteriores se explicará en mayor detalle.

- *Freecad v0.15:* este corresponde a una herramienta muy eficaz para la creación de geometrías a utilizar en los problemas. Permite crear la forma requerida, para luego crear una malla con herramientas matemáticas.

- *gmsh v2.16.0:* este simple software permite revisar las mallas de las geometrías y cambiarlas a numerosos formatos para su posterior uso.

- *MeshLab v1.3.3:* este software provee un sistema de visualización, procesamiento y edición de mallas triangulares de geometrías. Contiene numerosas herramientas para la visualización de las mallas en conjunto, limpieza, reparación, inspección, etc.

Es importane mencionar que el computador a ocupar posee 32 Gb de Memoria RAM y 24 núcleos.

4. Modelación de las geometrías

Como se mencionó anteriormente, la modelación de estos hilos se hace mediante el software FreeCad, el cual permite hacer las distintas geometrías y generar una malla de ésta, a su vez suavizándola. El mallado se realiza con la función de Mefisto del software, y la suavización a través de una metodología de aproximación de Laplace.

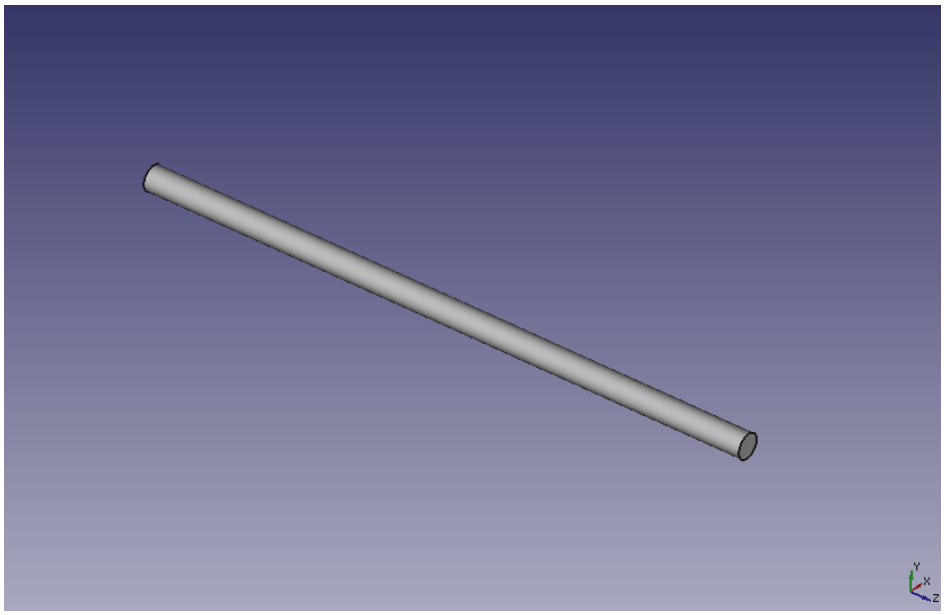


Figura 6: Geometría utilizada.

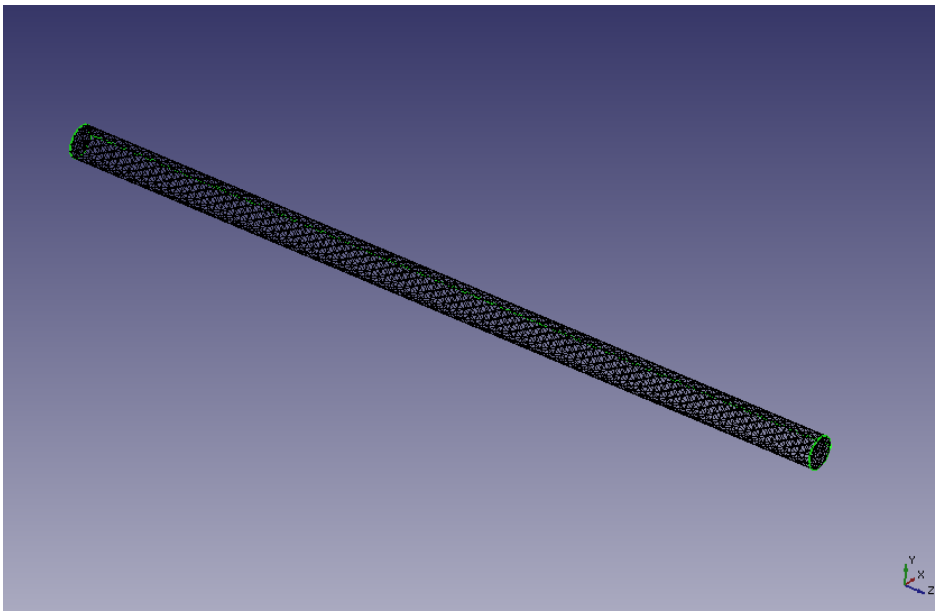


Figura 7: Generación de una malla en el hilo.

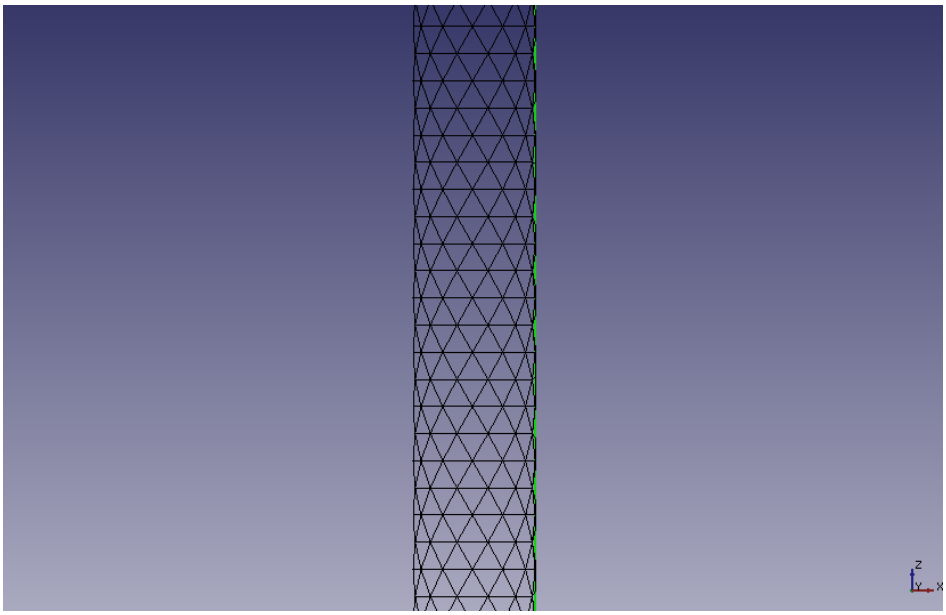


Figura 8: Malla del hilo.

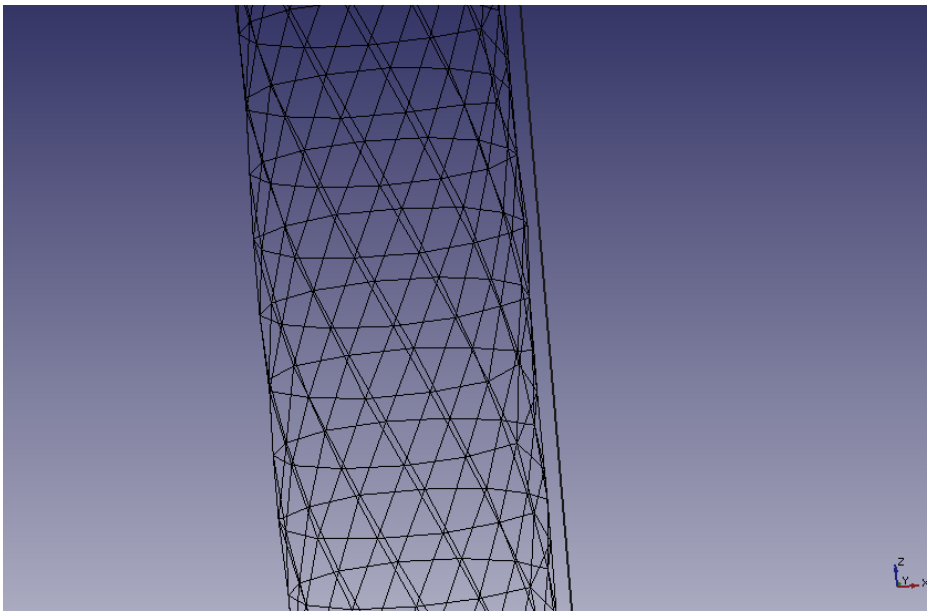


Figura 9: Malla del hilo suavizada y libre de errores.

Una vez obtenida la malla, es necesario ajustar el formato a los soportados por la biblioteca de BEM++, que en este caso corresponde a ".msh". Para esto, se utiliza el software GMSH, que permite utilizar el archivo ".stl" de FreeCad y exportar la malla al formato deseado.

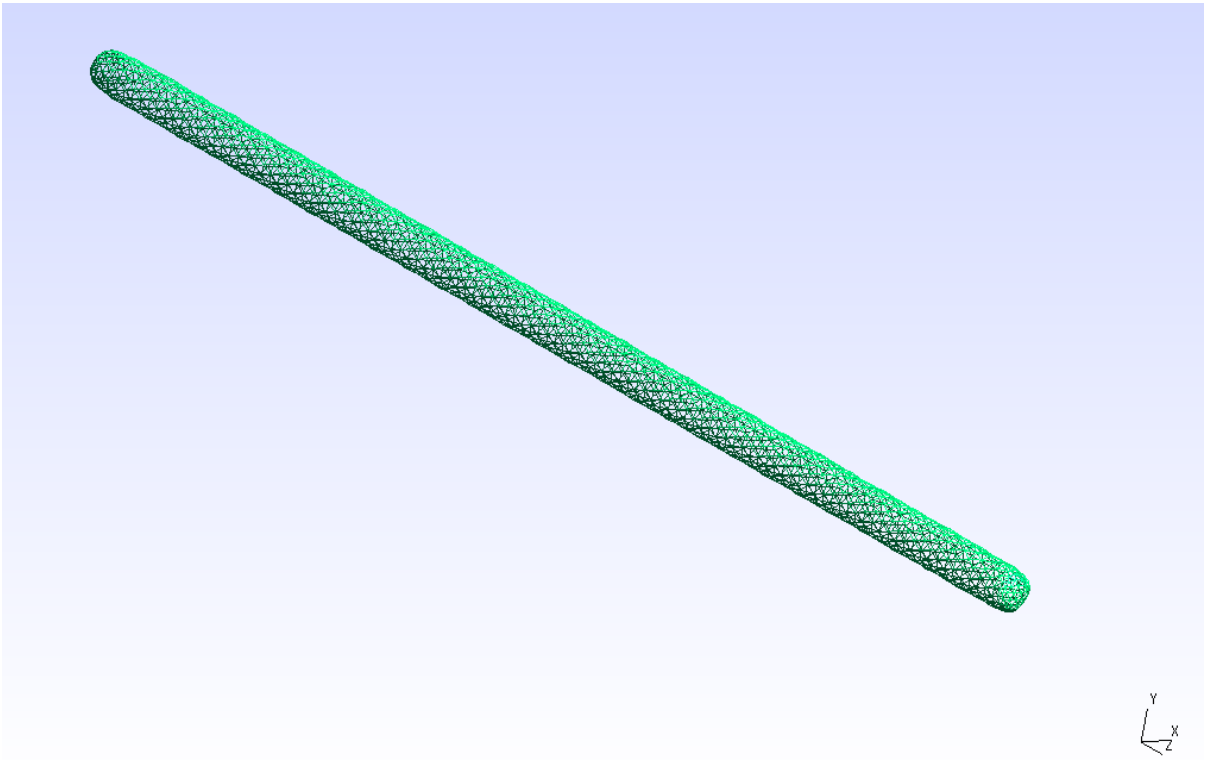


Figura 10: Malla del hilo en software GMSH.

Además de modelar los hilos requeridos para el análisis, se debe modelar la matriz donde estarán contenidos los hilos, que si bien se desea que no afecte de gran manera al campo, es necesaria para la programación del problema.

Esta matriz se crea de la misma manera que los hilos, simplemente cambiando la geometría deseada.

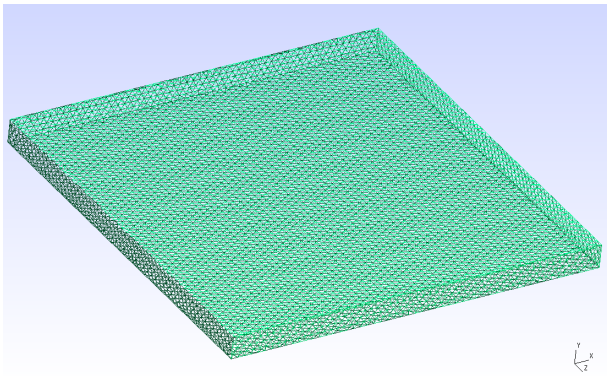


Figura 11: Malla de la matriz que contendrá el compuesto de hilos.

Una vez modeladas ambas geometrías, es necesario revisar su orientación, ubicación, y distintos factores que son importantes a la hora de evaluar los códigos, como por ejemplo, la orientación de

las normales de los elementos, que se desea que estén orientadas hacia el exterior del objeto, y también la duplicación de caras y vértices.

Para comprobar lo anterior se utilizar el software MeshLab, el cual permite corregir estos errores, en caso de existir, y además permite juntar las geometrías para comprobar su ubicación y orientación en el espacio.

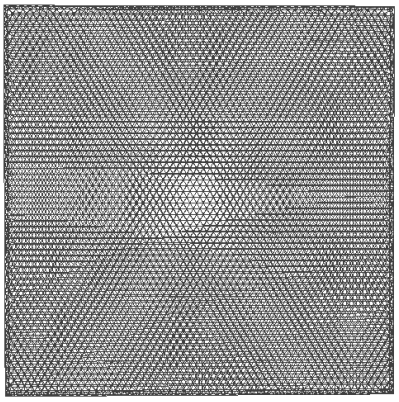


Figura 12: Malla de la matriz con 6 hilos.

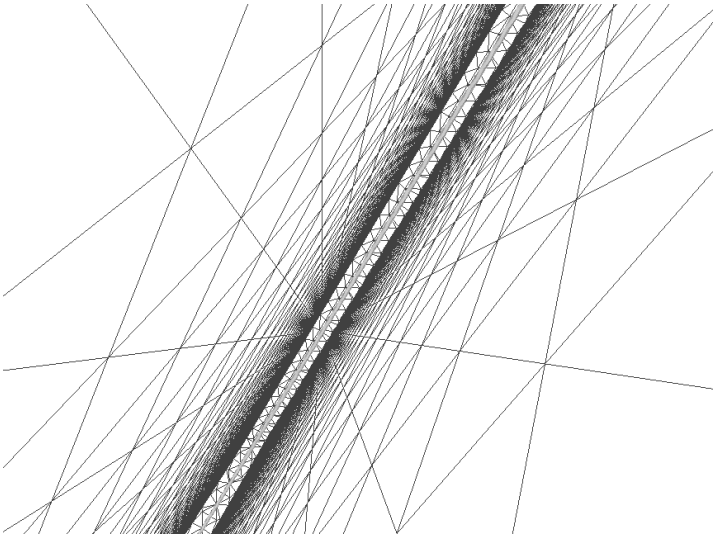


Figura 13: Malla de matriz desde el interior, donde se puede ver un hilo.

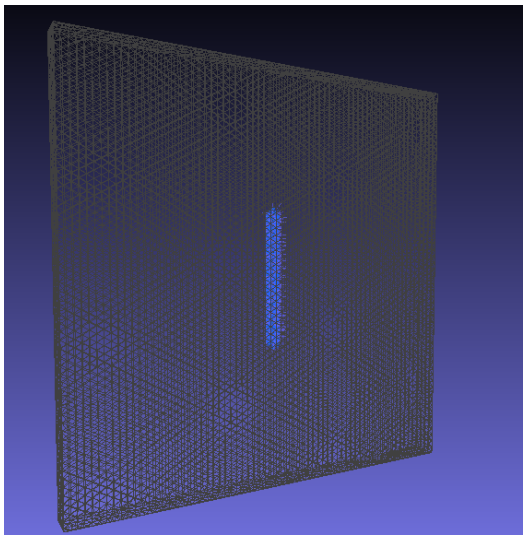


Figura 14: Malla de la matriz con un hilo, y las normales del hilo.

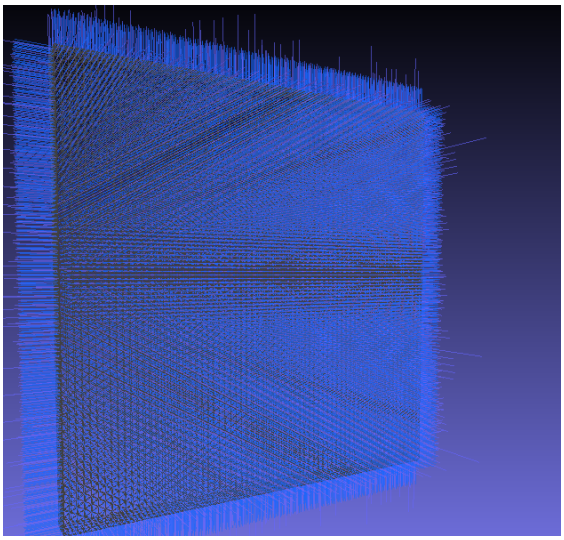


Figura 15: Malla de la matriz con sus normales hacia afuera.

Una vez posicionados, se pueden utilizar las geometrías para el código y realizar las pruebas requeridas. Es importante mencionar que cada cambio en la geometría utilizada, implica repetir todo el proceso anterior en lo que es creación de geometría y malla y de revisión de factores que podrían influir.

5. Ecuación integral de borde

Las ecuaciones de Maxwell son una herramienta fundamental para desarrollar el problema presentado debido al uso de campos electromagnéticos en las geometrías utilizadas. Es por esto que se

debe conocer su funcionamiento y además el uso para este problema en particular. Las ecuaciones de Maxwell se presentan a continuación, donde **E** es el campo eléctrico y **B** es el campo magnético:

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0} \tag{5.1}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{5.2}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{5.3}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}. \tag{5.4}$$

La ecuación 5.1 corresponde a la "Ley de Gauss", tambien conocida como teorema de Gauss, la cual establece que el flujo de ciertos campos a través de una superficie cerrada es proporcional a la magnitud de las fuentes de dicho campo que hay en el interior de la misma superficie. Esto se logra a través de una relación entre el campo eléctrico y la densidad de carga. El valor ε_0 corresponde a la permitividad eléctrica del vacío, la cual se definió anteriormente.

La ecuación 5.2 corresponde a la "Ley de Gauss" para campos magnéticos, en la cual se indica que la divergencia de estos campos es siempre 0.

La ecuación 5.3 corresponde a la "Ley de Faraday" y en ella se establece que el voltaje inducido en un circuito cerrado es directamente proporcional a la rapidez con que cambia en el tiempo el flujo magnético que atraviesa una superficie cualquiera con el circuito como borde.

Por último, la ecuación 5.4, corresponde a la "Ley de Ampere", relaciona un campo magnético estático con una corriente eléctrica, siendo la corriente mencionada la que genera el campo. En este caso μ_0 corresponde a la permeabilidad magnética del vacío y **J** es la densidad volumétrica de corriente.

En regiones donde no existen cargas ($\rho = 0$) ni corrientes ($\mathbf{J} = 0$), como ocurre en el vacío, las ecuaciones de Maxwell se reducen a:

$$\begin{aligned} &(i)\nabla \cdot \mathbf{E} = 0 \qquad (ii)\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \\ &(iii)\nabla \cdot \mathbf{B} = 0 \qquad (iv)\nabla \times \mathbf{B} = \mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}, \end{aligned} \tag{5.5}$$

las cuales son ecuaciones diferenciales parciales de primer orden.

Aplicando rotor a (ii) y (iv), y usando la identidad del rotor ($\nabla \times \nabla \times \mathbf{X} = \nabla(\nabla \cdot X) - \nabla^2 X$), se pueden desacoplar estas ecuaciones, obteniéndose:

$$\begin{aligned} \nabla \times \nabla \times \mathbf{E} = \nabla(\nabla \cdot E) - \nabla^2 E &= \nabla \times \left(-\frac{\partial \mathbf{B}}{\partial t} \right) \\ &= -\frac{\partial}{\partial t}(\nabla \times \mathbf{B}) = -\mu_0 \varepsilon_0 \frac{\partial^2 \mathbf{E}}{\partial t^2} \end{aligned}$$

$$\nabla \times \nabla \times \mathbf{B} = \nabla(\nabla \cdot B) - \nabla^2 B = \nabla \times \left(-\mu_0 \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \right)$$
$$= -\mu_0 \varepsilon_0 \frac{\partial}{\partial t}(\nabla \times \mathbf{E}) = -\mu_0 \varepsilon_0 \frac{\partial^2 \mathbf{B}}{\partial t^2},$$

para lo cual, como se mencionó anteriormente, $\nabla \cdot E = 0$ y $\nabla \cdot B = 0$, simplificando las ecuaciones anteriores y obteniéndose:

$$\nabla^2 E = \mu_0 \varepsilon_0 \frac{\partial^2 E}{\partial t^2} \tag{5.6}$$

$$\nabla^2 B = \mu_0 \varepsilon_0 \frac{\partial^2 B}{\partial t^2} \tag{5.7}$$

Finalmente se obtienen las ecuaciones análogas a (5.6) y (5.7):

$$\nabla^2 E - \frac{1}{c^2} \frac{\partial^2 E}{\partial t^2} = 0 \tag{5.8}$$

$$\nabla^2 B - \frac{1}{c^2} \frac{\partial^2 B}{\partial t^2} = 0. \tag{5.9}$$

dado que:

$$c = \frac{1}{\sqrt{\mu_0 \varepsilon_0}} \left[\frac{m}{s} \right]$$

Estas ecuaciones anteriores se pueden reescribir de manera general de la forma:

$$\nabla^2 \mathbf{u} - \frac{1}{c^2} \frac{\partial^2 \mathbf{u}}{\partial t^2} = 0. \tag{5.10}$$

la cual corresponde a una ecuación de onda que representa tanto la parte eléctrica como magnética y que será estudiada más adelante.

5.1. Ecuación de Helmholtz

La ecuación de Helmholtz es de uso frecuente en problemas de la física que involucran ecuaciones diferenciales parciales tanto en espacio como tiempo. La ecuación, que representa una forma de la ecuación de onda independiente del tiempo, resulta de la separación de variables para reducir la complejidad del análisis. Es por esto, que se considerar la ecuación de onda 5.10 y se realizará una separación de variables asumiendo que la ecuación $u(r,t)$ es, en verdad, separable.
La separación que se supondrá será:

$$\mathbf{u}(\mathbf{r},t) = A(\mathbf{r})T(t).$$

Sustituyendo en la ecuación de onda (5.10) y luego simplificando, se obtiene:

$$\frac{\nabla^2 A(\mathbf{r})}{A(\mathbf{r})} = \frac{1}{T(t)c^2}\frac{\partial^2 \mathbf{u}}{\partial t^2}. \tag{5.11}$$

Es posible notar que la parte izquierda de la expresión depende únicamente de r, mientras que la parte derecha depende únicamente de t. Como resultado de lo anterior, la ecuación (5.11) es válida únicamente en el caso en que ambos lados de la ecuación son igualmente a una constante. Visto de esta forma, se pueden obtener dos ecuaciones, una para A y otra para T, obteniéndose así:

$$\frac{\nabla^2 A(\mathbf{r})}{A(\mathbf{r})} = -k^2$$
$$\frac{1}{T(t)c^2}\frac{\partial^2 \mathbf{u}}{\partial t^2} = -k^2$$

donde se elige $-k^2$ como constante, respetando la nomenclatura universal. Luego, arreglando la primera ecuación, se obtiene finalmente la ecuación de Helmholtz:

$$\nabla^2 A(\mathbf{r}) + k^2 A(\mathbf{r}) = (\nabla^2 + k^2)A(\mathbf{r}) = 0 \tag{5.12}$$

Tanto campo eléctrico como magnético deben satisfacer una ecuación cuya solución representa una onda, por lo tanto, si se supone que el campo tiene una dependencia armónica de la forma $\psi = Re(\psi_0 e^{-iwt})$, que para este caso corresponde a $T(t) = e^{-i\omega t}$, se puede determinar el valor de k, debido a que w corresponde a la frecuencia angular y v_p es la velocidad de fase de la onda:

$$k = \frac{\omega}{v_p}$$

donde k al "número de onda" y v_p se puede calcular de la forma:

$$v_p = \frac{1}{\sqrt{\mu_r \mu_0 \varepsilon_r \varepsilon_0}}$$

.

Finalmente, con las ecuaciones y variables anterior es posible construir las ecuaciones de Helmholtz necesarias para el problema. Para el campo eléctrico resulta:

$$\nabla^2 E(\mathbf{r}) + \frac{w^2}{v_p^2} E(\mathbf{r}) = 0$$

o

$$\nabla^2 E(\mathbf{r}) + k^2 E(\mathbf{r}) = 0 \tag{5.13}$$

Análogamente, para el campo magnético se obtiene:

$$\nabla^2 B(\mathbf{r}) + k^2 B(\mathbf{r}) = 0 \tag{5.14}$$

Como se mencionó anteriormente, ahora se tiene una ecuación que únicamente depende de la posición, tanto para el campo magnético como para el campo eléctrico, dejando a un lado la variable tiempo.

5.2. Formulando una ecuación integral en el borde

En primer lugar, se debe tener en consideración la condición de radiación de Sommerfield, en la cual se establece que "las fuentes deben ser fuentes, no sumideros de energía. La energía que es irradiada desde la fuente debe continuar hasta el infinito, y ninguna energía que provenga del infinito debe irradiar hacia el campo". Si la onda es armónica y el valor k es mayor que cero, se define matemáticamente como:

$$\lim_{|\mathbf{r}|\to\infty} |\mathbf{r}|^{\frac{n-1}{2}} \left(\frac{\partial}{\partial |\mathbf{r}|} - ik \right) u(\mathbf{r}) = 0. \tag{5.15}$$

Teniendo esto en cuenta, se procede a resolver el caso particular de estudio. Dado que se tiene una onda incidente en un microhilo y esta se asumió armónica, el problema se reduce a una onda moviéndose con la misma frecuencia angular w después de un periodo inicial transitorio. Se propone una solución de la ecuación de la forma:

$$\mathbf{U}(\mathbf{r},t) = u(\mathbf{r})e^{-i\omega t}$$

donde u corresponde a la variable que se quiera estudiar, ya sea el campo eléctrico E o el campo magnético B.

La solución fundamental de este problema corresponde a una función de Green "$g(r-r')$" (en la cual es importante mencionar que r no se restringe a un volumen, y la variable de integración será r'), que se da reemplazando el término de fuente, por un delta de Dirac, como fuente en el punto "r_0".

Al utilizar la ecuación de Helmholtz obtenida anteriormente y multiplicar por la solución fundamental, se obtiene:

$$\int_\Omega (\nabla^2 u(\mathbf{r'}) + k^2 u(\mathbf{r'}))g(\mathbf{r}-\mathbf{r'})d\Omega(\mathbf{r'}) = 0 \tag{5.16}$$

donde Ω corresponde al volumen de estudio. Luego, reescribiendo la ecuación se logra obtener:

$$\int_\Omega \nabla\cdot(g(\mathbf{r}-\mathbf{r'})\nabla u(\mathbf{r'})) - \nabla u(\mathbf{r'})\cdot\nabla g(\mathbf{r}-\mathbf{r'})d\Omega + \int_\Omega k^2 u(\mathbf{r'})g(\mathbf{r}-\mathbf{r'})d\Omega = 0 \tag{5.17}$$

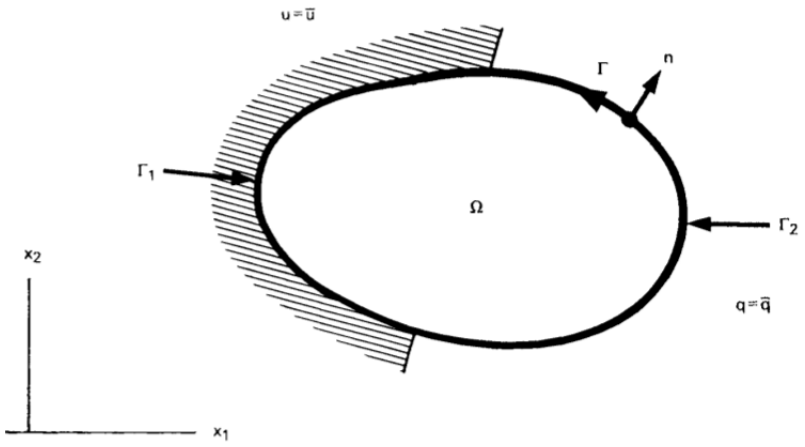


Figura 16: Volumen y superficie de una geometría en estudio.

Dado que el teorema de la divergencia (o teorema de Green) indica que:

$$\int_{\Omega}(\nabla \cdot v)d\Omega = \oint_{\Gamma} v \cdot d\Gamma$$

, se puede reescribir la ecuación 5.17, si este teorema se utiliza en la primera integral, como:

$$\int_{\Gamma} g\nabla u \cdot \mathbf{n} d\Gamma - \int_{\Omega} \nabla u \cdot \nabla g d\Omega + \int_{\Omega} k^2 u g d\Omega = 0, \tag{5.18}$$

a la cual nuevamente se le aplica la regla del producto de la divergencia, aplicada en 5.16, y el teorema de la divergencia, para ser reeescrita como:

$$\int_{\Gamma} g\frac{\partial u}{\partial \mathbf{n}} d\Gamma - \int_{\Gamma} u\frac{\partial g}{\partial \mathbf{n}} d\Gamma + \int_{\Omega} u(\nabla^2 + k^2) g d\Omega = 0 \tag{5.19}$$

Teniendo en consideración que el método aplicado en el trabajo corresponde a un estudio de borde, es necesario llevar aquellas partes de la ecuación 5.19 que se estudian en el volumen, al borde. Se mencionó que la ecuación de Helmholtz es resuelta por la solución fundamental de Green (la cual no será profundizada y solo se utilizará), esta se presenta en el caso de una onda en tres dimensiones como:

$$g(\mathbf{r} - \mathbf{r'}) = \frac{e^{ik|\mathbf{r}-\mathbf{r'}|}}{4\pi|\mathbf{r}-\mathbf{r'}|}$$

con r_0 como punto fuente. Siendo solución de la ecuación de Helmholtz propuesta:

$$(\Delta + k^2) g(\mathbf{r} - \mathbf{r'}) = -f(\mathbf{r} - \mathbf{r'}) \tag{5.20}$$

Con esta solución fundamental de Green, se puede reescribir la ecuación 5.19 para la superficie Γ del volumen en estudio:

$$\boxed{u_{ext}(\mathbf{r}) = \int_{\Gamma} u_{ext}(\mathbf{r'}) \frac{\partial g(\mathbf{r}-\mathbf{r'})}{\partial \mathbf{n(r')}} d\Gamma(\mathbf{r'}) - \int_{\Gamma} g(\mathbf{r}-\mathbf{r'}) \frac{\partial u_{ext}(\mathbf{r'})}{\partial \mathbf{n(r')}} d\Gamma(\mathbf{r'}),} \tag{5.21}$$

la cual corresponde a la ecuación base del problema, ya que en ella sólo se estudiará el borde de la geometría requerida. Para este caso en particular, se utilizarán las normales de todas las geometrías implicadas hacia el exterior.

Con esto en mente, por fin se puede pensar en el problema principal de una onda incidente en la superficie de un objeto. Para esto se deben acordar ciertas nomenclaturas. Si el campo "u" se elige igual al campo de onda incidente constante, se obtiene una solución de la forma:

$$u = u_{inc} + u_{scat} \tag{5.22}$$

donde u_{inc} corresponde a la onda incidente al objeto y u_{scat} es el campo dispersado o dispersor, del inglés "scattered".

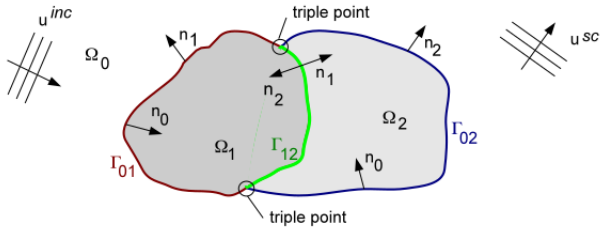


Figura 17: Onda incidente y dispersor en geometrías, con sus respectivas normales.

Además, como se mencionará en la sección siguiente, con la biblioteca que proporciona BEM++, se tienen operadores que corresponden a los que se utilizarán para la resolución de la ecuación de Helmholtz. Estos son el operador de capa simple y el operador de capa doble (Single layer potential y Double layer potential, en inglés):

$$\begin{aligned} [S\psi](\mathbf{r}) &= \int_\Gamma g(\mathbf{r}-\mathbf{r'})\psi(\mathbf{r'})d\Gamma(\mathbf{r'}) \\ [D\phi](\mathbf{r}) &= \int_\Gamma \frac{\partial}{\partial n(\mathbf{r'})} g(\mathbf{r}-\mathbf{r'})\phi(\mathbf{r'})d\Gamma(\mathbf{r'}) \\ \mathbf{r} &\in \Omega \end{aligned} \tag{5.23}$$

Cuando se hace tender un punto que se encuentra fuera de la geometría, a un punto definido en el borde, como se muestra en las figuras 18 y 19, en las cuales el $r = \varepsilon$ se hace tender a cero, se genera una condición particular en el caso del operador de capa doble:

$$\lim_{\mathbf{r}\to\mathbf{0}} \int_\Gamma u(\mathbf{r'}) \frac{\partial g(\mathbf{r}-\mathbf{r'})}{\partial \mathbf{n}(\mathbf{r'})} d\Gamma(\mathbf{r'}) = \int_\Gamma u(\mathbf{r'}) \frac{\partial g(\mathbf{r}-\mathbf{r'})}{\partial \mathbf{n}(\mathbf{r'})} d\Gamma(\mathbf{r'}) + \frac{1}{2}u(\mathbf{r})$$

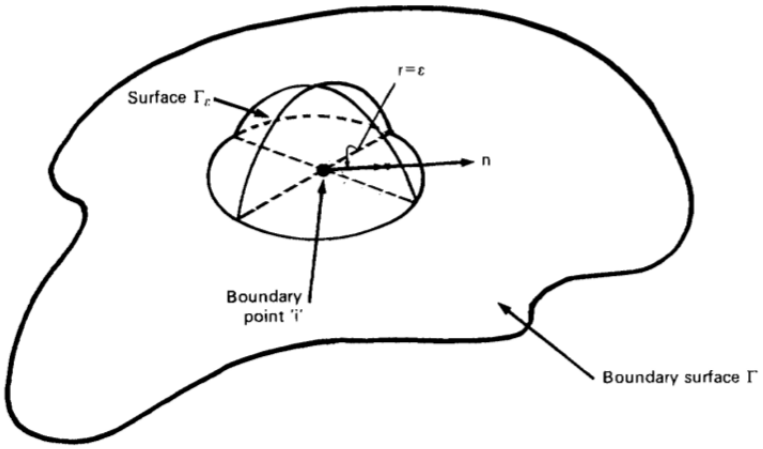


Figura 18: Punto exterior a la geometría que se acercará al borde.

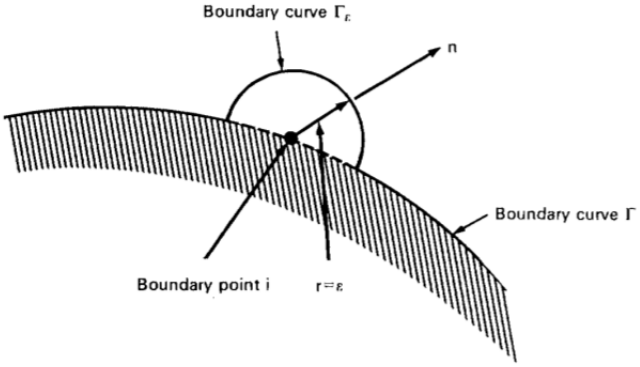


Figura 19: Punto exterior en la geometría que se acercará al borde.

Análogamente, se puede hacer el mismo procedimiento para el caso interior, haciendo tender desde un punto interior del volumen, con tan solo un cambio de signos:

$$\lim_{\mathbf{r}\to\mathbf{0}} \int_\Gamma u(\mathbf{r'}) \frac{\partial g(\mathbf{r}-\mathbf{r'})}{\partial \mathbf{n(r')}} d\Gamma(\mathbf{r'}) = \int_\Gamma u(\mathbf{r'}) \frac{\partial g(\mathbf{r}-\mathbf{r'})}{\partial \mathbf{n(r')}} d\Gamma(\mathbf{r'}) - \frac{1}{2} u(\mathbf{r})$$

Luego, si se reemplaza 5.22 en 5.21 y se tiene en cuenta la condición para el operador de capa doble, se puede reescribir la ecuación de la forma:

$$u^{ext}(\mathbf{r}) = \frac{1}{2} u_s^{ext}(\mathbf{r}) + \int_\Gamma u_s^{ext}(\mathbf{r'}) \frac{\partial g(\mathbf{r}-\mathbf{r'})}{\partial \mathbf{n(r')}} d\Gamma(\mathbf{r'}) - \int_\Gamma g(\mathbf{r}-\mathbf{r'}) \frac{\partial u_s^{ext}(\mathbf{r'})}{\partial \mathbf{n(r')}} d\Gamma(\mathbf{r'}) + u_{inc}(\mathbf{r}), \quad (5.24)$$

o en su forma de operador:

$$u^{ext}(\mathbf{r}) = \frac{I}{2} u_s^{ext}(\mathbf{r}) + \left[D u_s^{ext}\right](\mathbf{r}) - \left[S \frac{\partial u_s^{ext}}{\partial n}\right](\mathbf{r}) + u_{inc}(\mathbf{r}), \quad (5.25)$$

donde u_s corresponde al dispersor u_{scat}, que a su vez se divide segun su superíndice en *exterior* e *interior*, y donde I corresponde al operador de la matriz identidad.

Dado que en el interior de la geometría no hay onda incidente, la ecuación en el interior es más sencilla y es de la forma:

$$u^{int}(\mathbf{r}) = \frac{I}{2}u_s^{int}(\mathbf{r}) - \left[Du_s^{int}\right](\mathbf{r}) + \left[S\frac{\partial u_s^{int}}{\partial n}\right](\mathbf{r}) \tag{5.26}$$

Aún así, es necesario establecer una segunda ecuación para la coherencia de ecuaciones e incógnitas, dado que se tiene una sola ecuación para las dos incógnitas u y $\frac{\partial u}{\partial n}$. Para esto, se realiza una derivada con respecto a la normal de la geometría sobre las ecuaciones anteriores, siempre con respecto a r, obteniéndose:

$$\frac{\partial u^{ext}}{\partial n}(\mathbf{r}) = \frac{I}{2}\frac{\partial u_s^{ext}}{\partial n}(\mathbf{r}) + \left[D'u_s^{ext}\right](\mathbf{r}) - \left[S'\frac{\partial u_s^{ext}}{\partial n}\right](\mathbf{r}) + \frac{\partial u_{inc}}{\partial n}(\mathbf{r}) \tag{5.27}$$

$$\frac{\partial u^{int}}{\partial n}(\mathbf{r}) = \frac{I}{2}\frac{\partial u_s^{int}}{\partial n}(\mathbf{r}) + \left[S'\frac{\partial u_s^{int}}{\partial n}\right](\mathbf{r}) - \left[D'u_s^{int}\right](\mathbf{r}), \tag{5.28}$$

donde S' corresponde al operador adjunto de doble capa (Adjoint Double Layer Potential, en inglés) y D' al operador hipersingular (Hypersingular, en inglés), como se mencionará en la sección siguiente. Estos se definen como:

$$[S'\psi](\mathbf{r}) = \int_\Gamma \frac{\partial}{\partial n(\mathbf{r})}g(\mathbf{r}-\mathbf{r'})\psi(\mathbf{r'})d\Gamma(\mathbf{r'}) \quad \text{y} \quad [D'\phi](\mathbf{r}) = -\frac{\partial}{\partial n(\mathbf{r})}\left[\int_\Gamma \frac{\partial}{\partial n(\mathbf{r'})}g(\mathbf{r}-\mathbf{r'})\phi(\mathbf{r'})d\Gamma(\mathbf{r'})\right]$$

Además, dado que el campo exterior es igual al campo interior en un punto del borde, la ecuación exterior e interior 5.27 y 5.28 se pueden igualar, obteniéndose así:

$$\frac{I}{2}(u^{int}-u^{ext})(\mathbf{r}) - \left[D_{ext}u^{ext} + D_{int}u^{int}\right](\mathbf{r}) + \left[S_{int}\frac{\partial u^{int}}{\partial n} + S_{ext}\frac{\partial u^{ext}}{\partial n}\right](\mathbf{r}) = u_{inc}(\mathbf{r}), \tag{5.29}$$

cuando se estudia en el borde.

Dado que existe un traspaso del campo hacia el interior del objeto, es necesario establecer ciertas condiciones de transmisión. Estas son las siguientes:

$$u^i = u^e$$
$$\frac{\partial u^i}{\partial n} = \frac{\mu_i}{\mu_e}\frac{\partial u^e}{\partial n}$$

,
donde para ser consecuente con la nomenclatura encontrada en los textos, se define:

$$\frac{\mu_i}{\mu_e} = \alpha$$

.

Con estas condiciones, se pueden reescribir las ecuaciones (5.29), (5.27) y (5.28) de la forma:

$$[-D_e - D_i]\, u^e(\mathbf{r}) + [\alpha S_i + S_e]\, \frac{\partial u^e}{\partial n}(\mathbf{r}) = u_{inc}(\mathbf{r}) \tag{5.30}$$

$$(\frac{\alpha - 1}{2})\frac{\partial u^e}{\partial n}(\mathbf{r}) + \left[-D'_e - D'_i\right] u^e(\mathbf{r}) + \left[\alpha S'_i + S'_e\right] \frac{\partial u^e}{\partial n}(\mathbf{r}) = \frac{\partial u_{inc}}{\partial n}(\mathbf{r}). \tag{5.31}$$

El cual es el sistema de ecuaciones que da paso al cálculo de dispersión en objetos, tambien escrito como matriz:

$$\begin{bmatrix} -D_e - D_i & \alpha S_i + S_e \\ -D'_e - D'_i & (\frac{\alpha-1}{2}) + \alpha S'_i + S'_e \end{bmatrix} \begin{bmatrix} u^e \\ \frac{\partial u^e}{\partial n} \end{bmatrix} = \begin{bmatrix} u_{inc} \\ \frac{\partial u_{inc}}{\partial n} \end{bmatrix}. \tag{5.32}$$

5.3. Dispersión en múltiples objetos o Multiple Scattering

En la sección anterior, se llegó a ecuaciones que permitían calcular la dispersión en un objeto, debido a que el campo que llega a él se separa en lo que incide y lo que se dispersa, ya que respetando la condición de radiación de Sommerfield, no debiese llegar nuevamente al objeto ni modificar la fuente. Pero eso se cumple únicamente cuando existe una sola geometría en cuestión, lo que se aleja del problema real a resolver, debido a que en este no sólo se requiere que sean múltiples geometrías, si no que además se necesita que los microhilos se posicionen dentro de un compósito que los contendrá, lo que aumentará la dificultad de las expresiones y de su cálculo.

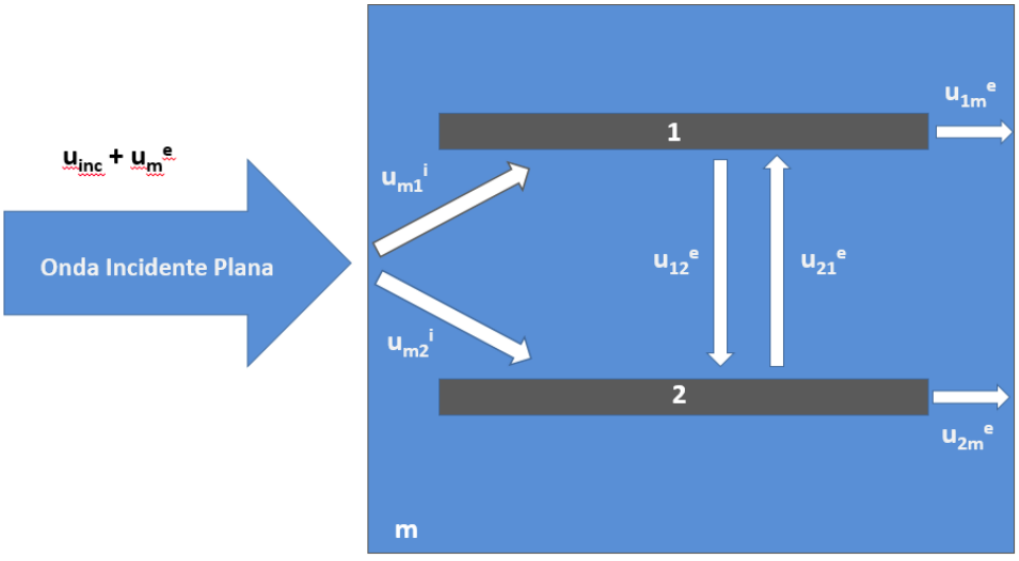


Figura 20: Nomenclatura a utilizar en dispersión sobre múltiples objetos. 1 y 2 corresponden a dos hilos y "m" corresponde a la matriz.

Como antes, es necesario establecer cierta nomenclatura utilizada dado que se utilizarán múltiples volúmenes. El subíndice "11" se referirá a un operador actuando desde el volumen 1, a sí mismo. El operador "12" significará que se está integrando con los operadores desde la geometría

1 al borde de la geometría 2. Para el caso de las incógnitas, sólo significará la geometría a la que pertenecen, como por ejemplo, u_1.

Teniendo lo anterior en cuenta, acomodando las ecuaciones para dos geometrías, se obtiene:

$$\left[-D_e^{11}-D_i^{11}\right]u_1^e(\mathbf{r})+\left[\alpha S_i^{11}+S_e^{11}\right]\frac{\partial u_1^e}{\partial n}(\mathbf{r})+\left[D_e^{21}u_2^e\right](\mathbf{r})-\left[S_e^{21}\frac{\partial u_2^e}{\partial n}\right](\mathbf{r})=u_{inc}^1(\mathbf{r}) \tag{5.33}$$

$$\left[-D_e^{22}-D_i^{22}\right]u_2^e(\mathbf{r})+\left[\alpha S_i^{22}+S_e^{22}\right]\frac{\partial u_2^e}{\partial n}(\mathbf{r})+\left[D_e^{12}u_1^e\right](\mathbf{r})-\left[S_e^{12}\frac{\partial u_1^e}{\partial n}\right](\mathbf{r})=u_{inc}^2(\mathbf{r}) \tag{5.34}$$

$$\begin{aligned}(\frac{\alpha-1}{2})\frac{\partial u_1^e}{\partial n}(\mathbf{r})+&\left[-D_e^{'11}-D_i^{'11}\right]u_1^e(\mathbf{r})+\left[\alpha S_i^{'11}+S_e^{'11}\right]\frac{\partial u_1^e}{\partial n}(\mathbf{r})\\&+\left[D_e^{21}u_2^e\right](\mathbf{r})-\left[S_e^{21}\frac{\partial u_2^e}{\partial n}\right](\mathbf{r})=\frac{\partial u_{inc}^1}{\partial n}(\mathbf{r})\end{aligned} \tag{5.35}$$

$$\begin{aligned}(\frac{\alpha-1}{2})\frac{\partial u_2^e}{\partial n}(\mathbf{r})+&\left[-D_e^{'22}-D_i^{'22}\right]u_2^e(\mathbf{r})+\left[\alpha S_i^{'22}+S_e^{'22}\right]\frac{\partial u_2^e}{\partial n}(\mathbf{r})\\&+\left[D_e^{12}u_1^e\right](\mathbf{r})-\left[S_e^{12}\frac{\partial u_1^e}{\partial n}\right](\mathbf{r})=\frac{\partial u_{inc}^2}{\partial n}(\mathbf{r}).\end{aligned} \tag{5.36}$$

las cuales representan al sistema de ecuaciones a resolver para 2 geometrías, y como se hizo anteriormente, se pueden ordenar de forma matricial:

$$\begin{bmatrix} -D_e^{11}-D_i^{11} & \alpha S_i^{11}+S_e^{11} & D_e^{21} & -S_e^{21} \\ -D_e^{'11}-D_i^{'11} & (\frac{\alpha-1}{2})+\alpha S_i^{'11}+S_e^{'11} & D_e^{'21} & -S_e^{'21} \\ D_e^{12} & -S_e^{12} & -D_e^{22}-D_i^{22} & \alpha S_i^{22}+S_e^{22} \\ D_e^{'12} & -S_e^{'12} & -D_e^{'22}-D_i^{'22} & (\frac{\alpha-1}{2})+\alpha S_i^{'22}+S_e^{'22} \end{bmatrix}\begin{bmatrix} u_1^e \\ \frac{\partial u_1^e}{\partial n} \\ u_2^e \\ \frac{\partial u_2^e}{\partial n} \end{bmatrix}=\begin{bmatrix} u_{inc}^1 \\ \frac{\partial u_{inc}^1}{\partial n} \\ u_{inc}^2 \\ \frac{\partial u_{inc}^2}{\partial n} \end{bmatrix}. \tag{5.37}$$

Para el caso en que se tiene una geometría dentro de otra, se debe formular con cuidado debido a que en este ejemplo, la geometría 2 se encuentra contenida en la geometría 1, y las ecuaciones sólo son válidas para ese caso. Se presenta como:

$$\begin{bmatrix} -D_e^{11}-D_i^{11} & S_i^{11}+\frac{1}{\alpha}S_e^{11} & D_e^{21} & -S_e^{21} \\ -D_e^{'11}-D_i^{'11} & (\frac{\alpha-1}{2\alpha})+S_i^{'11}+\frac{1}{\alpha}S_e^{'11} & D_e^{'21} & -S_e^{'21} \\ -D_e^{12} & S_e^{12} & -D_e^{22}-D_i^{22} & \alpha S_i^{22}+S_e^{22} \\ -D_e^{'12} & S_e^{'12} & -D_e^{'22}-D_i^{'22} & (\frac{\alpha-1}{2\alpha})+\alpha S_i^{'22}+S_e^{'22} \end{bmatrix}\begin{bmatrix} u_1^i \\ \frac{\partial u_1^i}{\partial n} \\ u_2^e \\ \frac{\partial u_2^e}{\partial n} \end{bmatrix}=\begin{bmatrix} u_{inc}^1 \\ \frac{\partial u_{inc}^1}{\partial n} \\ 0 \\ 0 \end{bmatrix}. \tag{5.38}$$

6. Método de elementos de borde y BEM++

6.1. BEM

El método de elementos de borde ha surgido como una alternativa poderosa a los elementos finitos en los casos donde se requiere mayor precisión, como es el caso de concentración de esfuerzos, o donde el dominio se extiende al infinito. La característica más importante del método de elemento de bordes es que la metodología de formulación de las ecuaciones integrales de borde permite describir problemas de condiciones de borde conocidos o desconocidos. Por ende, solo se requiere una discretización de la superficie y no del volumen, reduciendo las dimensiones del problema en una dimensión. En consecuencia, el esfuerzo necesario para discretizar es bastante menor y las mallas son fáciles de generar y modificar. El método de elementos de borde es particularmente ventajoso en el caso de problemas de dominio infinito o semi-infinito (exterior domain problems, en inglés).

6.1.1. Ventajas de BEM

- *Menor tiempo de preparación de la información:* Esto es un resultado directo del hecho de modelar únicamente la superficie. Por lo tanto, el tiempo de preparación y de revisión de la información del problema es menor.

- *Alta resolución de esfuerzos:* Los esfuerzos son muy precisos debido a que no hay mayor aproximación impuesta en la solución de los puntos interiores, por lo tanto, la solución es exacta y completamente continua en el interior del dominio.

- *Menor tiempo y almacenamiento del computador:* Para la misma precisión, BEM usa un menor número de nodos y elementos (aunque una matriz muy llena). Para lograr resultados comparables en valores de esfuerzos, las mallas de FEM requieren mayores divisiones de borde que las mallas de BEM.

- *Menos información no deseada:* En la mayoría de los problemas ingenieriles, la peor situación (como fracturas, concentración de esfuerzos, choques térmicos, etc.) usualmente ocurre en las superficies. Por lo que modelar un cuerpo completo en tres dimensiones y calcular la información requerida en cada nodo del problema es bastante ineficiente. Usar un método de bordes disminuye los recursos necesarios, y ya que los puntos internos de BEM son opcionales, se hace énfasis en la parte querida y no en todo el interior.

6.1.2. Desventajas de BEM

- *Matemáticas compleja:* Utiliza matemáticas compleja, aunque no difícil de aprender. Por otro lado, los procedimientos numéricos de FEM pueden ser utilizados directamente a las soluciones de BEM.

- *Problemas no lineales:* En estos, el interior debe ser modelado, inevitablemente.
- *Matrices complejas y no simétricas:* La matriz solución de los problemas de BEM es asimétrica y llena de coeficientes distintos a cero, mientras que en FEM la matriz es más grande, pero escasamente populada. Esto significa que en BEM, la matriz debe ser almacenada por completa en la memoria del computador. Aún asi, esto no es una gran desventaja debido a que para obtener el mismo nivel de precisión que en FEM, BEM requiere un menor número de nodos y elementos, como se mencionó anteriormente.
- *Herramienta pobre para análisis de estructuras finas en tres dimensiones:* Esto debido al cuociente elevado entre superficie y volumen y la alta proximidad entre puntos nodales en la estructura. Esto provoca imprecisiones en las integraciones numéricas.

6.1.3. Estructuración de BEM

Una aplicación típica del método en cuestión se da de la siguiente áorma, la cual se justificará con ejemplos mas adelante:

- Modelo Matemático
- Fórmula de representación
- Ecuación integral de bordes
- Elementos de borde
- Ecuaciones discretizadas
- Solución lineal de los sistemas
- Análisis e interpretación

6.2. Utilizando BEM++ como biblioteca

Como se mencionó, BEM++ es una biblioteca de C++ con uso de python. La biblioteca esta compuesta de cinco grandes partes, esquematizadas en la figura 21.

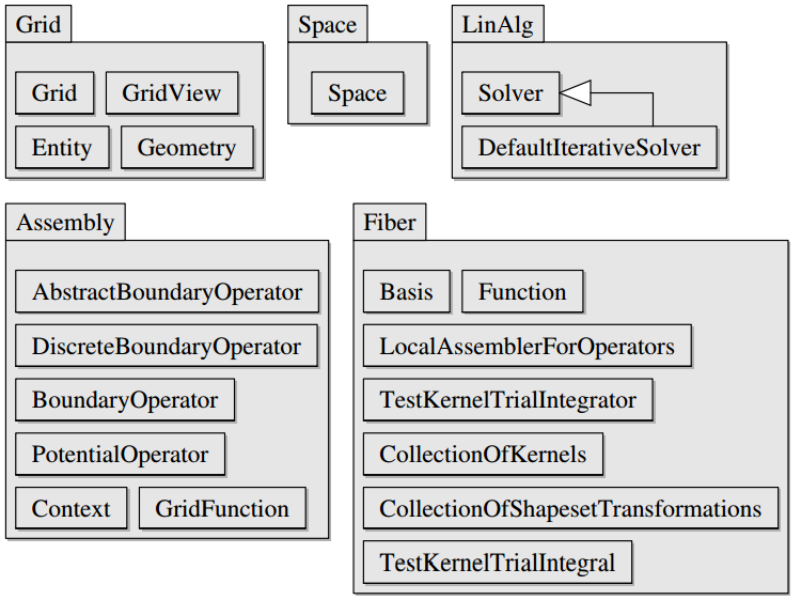


Figura 21: Módulos de BEM++ con sus clases más importantes.

El módulo *Grid* es responsable del manejo de las mallas, las cuales existen en la biblioteca para geometrías simples, o bien pueden ser importadas en el formato adecuado.

Las *Fiber* son rutinas rápidas de integración de elementos de borde, las cuales son un componente esencial debido a que este módulo es responsable de la evaluación de integrales de elementos de borde en un elemento solo o en pares de elementos, sin tomar en cuenta su conectividad.

El módulo *Space* representa el espacio de funciones definidas en los elementos de una malla. Provee un mapeo entre elementos y actúa como un regulador de los grados de libertad, utilizando su conocimiento respecto a la conectividad elemento-elemento.

El módulo *Assembly* es la parte más larga de la biblioteca. Define clases representando a los integradores operacionales y funciones definidas en las mallas, conteniendo en general, el código responsable del ensamblaje de todas aquellas matrices de operadores discretizados desde integrales elementales producidas en el módulo *Fiber*.

Finalmente esta el módulo *LinAlg*, el cual provee interfaces para un amplio rango de solucionadores lineales.

Name	Description
`PiecewiseConstantScalarSpace`	Space $S_h^{(0)}$ of piecewise constant functions.
`PiecewiseConstantDualGridScalarSpace`	Space of piecewise constant functions defined on the dual grid.
`PiecewiseLinearContinuousScalarSpace`	Space $S_h^{(1)}$ of continuous piecewise linear functions.
`PiecewiseLinearDiscontinuousScalarSpace`	Space of element-wise linear functions.
`PiecewisePolynomialContinuousScalarSpace`	Space of continuous piecewise polynomial functions.
`PiecewisePolynomialDiscontinuousScalarSpace`	Space of element-wise polynomial functions.
`RaviartThomas0VectorSpace`	Space of lowest-order Raviart-Thomas basis functions.
`UnitScalarSpace`	Space of globally constant functions.

Figura 22: Principales espacios de la librería de BEM++

Function	Weak form
`identityOperator()`	$\int_\Gamma \overline{\phi}(\mathbf{x})\,\psi(\mathbf{x})\,\mathrm{d}\Gamma(\mathbf{x})$
`maxwell3dIdentityOperator()`	$\int_\Gamma \overline{\boldsymbol{\phi}}(\mathbf{x})\cdot[\boldsymbol{\psi}(\mathbf{x})\times\boldsymbol{n}(\mathbf{x})]$
`laplaceBeltrami3dOperator()`	$\int_\Gamma \boldsymbol{\nabla}_\Gamma\overline{\phi}(\mathbf{x})\cdot\boldsymbol{\nabla}_\Gamma\psi(\mathbf{x})\,\mathrm{d}\Gamma(\mathbf{x})$
`laplace3dSingleLayerBoundaryOperator()`	$\int_\Gamma\int_\Sigma \overline{\phi}(\mathbf{x})\,g(\mathbf{x},\mathbf{y})\,\psi(\mathbf{y})\,\mathrm{d}\Gamma(\mathbf{x})\,\mathrm{d}\Sigma(\mathbf{y})$
`laplace3dDoubleLayerBoundaryOperator()`	$\int_\Gamma\int_\Sigma \overline{\phi}(\mathbf{x})\,\partial_{\mathbf{n}(\mathbf{y})}g(\mathbf{x},\mathbf{y})\,\psi(\mathbf{y})\,\mathrm{d}\Gamma(\mathbf{x})\,\mathrm{d}\Sigma(\mathbf{y})$
`laplace3dAdjointDoubleLayerBoundaryOperator()`	$\int_\Gamma\int_\Sigma \overline{\phi}(\mathbf{x})\,\partial_{\mathbf{n}(\mathbf{x})}g(\mathbf{x},\mathbf{y})\,\psi(\mathbf{y})\,\mathrm{d}\Gamma(\mathbf{x})\,\mathrm{d}\Sigma(\mathbf{y})$
`laplace3dHypersingularBoundaryOperator()`	$\int_\Gamma\int_\Sigma \mathbf{curl}_\Gamma\,\overline{\phi}(\mathbf{x})\cdot g(\mathbf{x},\mathbf{y})\;\mathbf{curl}_\Sigma\,\psi(\mathbf{y})\,\mathrm{d}\Gamma(\mathbf{x})\,\mathrm{d}\Sigma(\mathbf{y})$
`helmholtz3dSingleLayerBoundaryOperator()`	$\int_\Gamma\int_\Sigma \overline{\phi}(\mathbf{x})\,g_k(\mathbf{x},\mathbf{y})\,\psi(\mathbf{y})\,\mathrm{d}\Gamma(\mathbf{x})\,\mathrm{d}\Sigma(\mathbf{y})$
`helmholtz3dDoubleLayerBoundaryOperator()`	$\int_\Gamma\int_\Sigma \overline{\phi}(\mathbf{x})\,\partial_{\mathbf{n}(\mathbf{y})}g_k(\mathbf{x},\mathbf{y})\,\psi(\mathbf{y})\,\mathrm{d}\Gamma(\mathbf{x})\,\mathrm{d}\Sigma(\mathbf{y})$
`helmholtz3dAdjointDoubleLayerBoundaryOperator()`	$\int_\Gamma\int_\Sigma \overline{\phi}(\mathbf{x})\,\partial_{\mathbf{n}(\mathbf{x})}g_k(\mathbf{x},\mathbf{y})\,\psi(\mathbf{y})\,\mathrm{d}\Gamma(\mathbf{x})\,\mathrm{d}\Sigma(\mathbf{y})$
`helmholtz3dHypersingularBoundaryOperator()`	$\int_\Gamma\int_\Sigma g_k(\mathbf{x},\mathbf{y})[\mathbf{curl}_\Gamma\,\overline{\phi}(\mathbf{x})\cdot\mathbf{curl}_\Sigma\,\psi(\mathbf{y})$ $-k^2\,\overline{\phi}(\mathbf{x})\,\mathbf{n}(\mathbf{x})\cdot\psi(\mathbf{y})\,\mathbf{n}(\mathbf{y})]\,\mathrm{d}\Gamma(\mathbf{x})\,\mathrm{d}\Sigma(\mathbf{y})$
`maxwell3dSingleLayerBoundaryOperator()`	$\int_\Gamma\int_\Sigma g_k(\mathbf{x},\mathbf{y})[-\mathrm{i}k\,\overline{\boldsymbol{\phi}}(\mathbf{x})\cdot\boldsymbol{\psi}(\mathbf{y})$ $-\frac{1}{\mathrm{i}k}\,\mathrm{div}_\Gamma\,\overline{\phi}(\mathbf{x})\;\mathrm{div}_\Sigma\,\psi(\mathbf{y})]\,\mathrm{d}\Gamma(\mathbf{x})\,\mathrm{d}\Sigma(\mathbf{y})$
`maxwell3dDoubleLayerBoundaryOperator()`	$\int_\Gamma\int_\Sigma \boldsymbol{\nabla}_{\mathbf{x}}g_k(\mathbf{x},\mathbf{y})\cdot[\overline{\boldsymbol{\phi}}(\mathbf{x})\times\boldsymbol{\psi}(\mathbf{y})]\,\mathrm{d}\Gamma(\mathbf{x})\,\mathrm{d}\Sigma(\mathbf{y})$

Figura 23: Funciones que construyen los operadores de la biblioteca.

Para la resolución de problemas, BEM++ utiliza variados tipos de espacios:

- *Discontinuous polynomial spaces* (DP), estos espacios de funciones son polinomios a través de cada elemento pero discontinuos entre elementos. El orden máximo es de 10.
- *Polynomial spaces* (P), estos espacios de funciones son polinomios a través de cada elemento y continuos entre elementos. El orden mínimo es 0. El máximo es 10.
- *Polynomial spaces on barycentric grids* (B-P), Estos son los mismos espacios que los *Polynomial spaces* y el mismo número de grados de libertad, pero esta vez el refinamiento de la malla es baricéntrico. El único orden permito actualmente es de 1.
- *Discontinuous polynomial spaces on barycentric grids* (B-DP), similar al espacio B-P pero los polinomios de los espacios de funciones son discontinuos entre elementos. Igualmente solo se puede utilizar orden 1.
- *Dual spaces of constant functions* (DUAL), este es un espacio de funciones constantes definidas en una malla dual.
- *Raviart-Thomas Vector Space* (RT), estos son espacios de funciones de Raviart-Thomas. Estos son necesarios para operadores integrales en scattering electromagnético. Sólo existen de orden 0.

Otra carácterística de BEM++, es que cuenta con numerosos operadores que se basan en el concepto de operadores de borde.

Un operador de borde

$$A\colon D \to R$$

es un mapeo desde un dominio D hasta un recorrido R, donde ambos son definidos en una malla.BEM++ no trabaja directamente con el operador de borde A, sino que con su forma débil,

$$a(u,v) = \int_\Gamma [Au](\mu)\overline{v(\mu)}d\mu \quad u\varepsilon D, v\varepsilon V$$

Donde V es el espacio dual a espacio de alcance. Los operadores en BEM++ por lo general utilizan como argumento D, R y V. Actualmente la librería dispone de los siguientes operadores integrales,

- Single-layer boundary operator

$$[S\psi](x) = \int_\Gamma g(x,y)\psi(y)d\Gamma(y)$$

- Double-layer boundary operator

$$[K\phi](x) = \int_\Gamma \frac{\partial}{\partial n(y)} g(x,y)\phi(y)d\Gamma(y)$$

- Adjoint double layer boundary operator

$$[K'\psi](x) = \int_\Gamma \frac{\partial}{\partial n(x)} g(x,y)\psi(y)d\Gamma(y)$$

- Hypersingular

$$[D\phi](x) = -\frac{\partial}{\partial n(x)}\left[\int_\Gamma \frac{\partial}{\partial n(y)} g(x,y)\phi(y)d\Gamma(y)\right]$$

Estos son posible de implementar a través de las siguientes soluciones fundamentales,

- Laplace ($-\triangle u = 0$)

$$g(x,y) = \frac{1}{4\pi|x-y|}$$

- Hemholtz modificado ($-\triangle u + \omega^2 u = 0$)

$$g(x,y) = \frac{e^{-\omega|x-y|}}{4\pi|x-y|}$$

- Helmholtz ($-\triangle u - k^2 u = 0$)

$$g(x,y) = \frac{e^{ik|x-y|}}{4\pi|x-y|}$$

Además, BEM++ soporta la solución de la ecuación de Maxwell de la forma:

$$\nabla \times \nabla \times u - k^2 u = 0$$

con los siguientes operadores potenciales,

- Operador potencial de campo eléctrico de Maxwell

$$[\varepsilon\phi](x) = ik\int_\Gamma g(x,y)\phi(y)ds - \frac{1}{ik}\nabla_x\int_\Gamma g(x,y)(\nabla_\Gamma\cdot\phi)(y)ds(y)$$

- Operador potencial de campo magnético de Maxwell

$$[M\phi](x) = \nabla_x\times\int_\Gamma g(x,y)\phi(y)ds$$

y operadores de borde:

- Operador de campo eléctrico de Maxwell de borde

$$s(\phi,\psi) = \int_\Gamma\int_\Gamma g(x,y)\left[-ik\overline{\psi(x)}\cdot\phi(y) - \frac{1}{ik}(\nabla_\Gamma\cdot\overline{\psi})(x)(\nabla_\Gamma\cdot\phi)(y)\right]ds(x)ds(y)$$

- Operador de campo magnético de Maxwell de borde

$$c(\phi,\psi)) = \int_\Gamma\int_\Gamma \nabla_x g(x,y)\left[\overline{\psi(x)}\times\phi(y)\right]ds(x)ds(y)$$

.

Es también posible utilizar los operadores identidad y de LaPlace-Beltrami.

Fundamentalmente el ensamblaje de los operadores integrales de borde en BEM++ están basados en una evaluación de integrales del tipo:

$$I = \int_{T1}\int_{T2} g(x,y)\phi(y)\overline{\psi(x)}ds(y)ds(x)$$

donde $T1$ y $T2$ son triángulos pertenecientes a la malla de integración. $\psi(x)$ y $\phi(y)$ son funciones en $T1$ y $T2$. Se diferencian 2 casos:

- $T1$ y $T2$ no tienen conexión.
- $T1$ y $T2$ comparten un vértice, cara o son idénticos.

El primer caso es resuelto a través de una cuadratura Gaussiana de la forma:

$$I \approx \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} g(x_i,y_j)\phi(y_j)\overline{\psi(x_i)}\omega_i^{(1)}\omega_j^{(2)}$$

.

Para el segundo caso es necesario tomar en cuenta la singularidad y se debe tomar particular atención. Es posible cambiar el orden de la cuadratura utilizada.

Esta información, y bastante más, se puede encontrar en el sitio oficial del software, en su documentación.

7. Presentación del problema a resolver

Como se mencionó anteriormente, para la resolución y estudio de campos electromagnéticos irradiadios sobre compósitos con microhilos en la frecuencia de microondas, se requieren numerosas propiedades que son difíciles de obtener experimentalmente, por lo que es de gran ayuda una herramienta computacional que permita aproximar estos valores de acuerdo a una aproximación matemática en base a los resultados empíricos existentes. En algunas publicaciones científicas se pueden encontrar valores con respecto a las permeabilidades y permitividades efectivas de ciertos compósitos, los cuales son válidos únicamente para cierta configuración y proporción de hilo/-compósito. Gracias a valores empíricos se establecerá una herramienta que permita calcular las permeabilidades y permitividades de los hilos en cuestión, para luego comprobar que el código desarrollado sea válido para la resolución de estos problemas. Luego de obtener las propiedades de los hilos, se utilizará el compósito sin hilos, con sus propiedades efectivas, y se comparará con las básicas por separado en el compósito con hilos.

Se trabajará con microhilos en un compósito de silicona, en disposición de pantalla como se mostró anteriormente, por lo que las formulaciones anteriores no son del todo útiles, dado que se explicó el primer caso de dispersión en dos cuerpos y el segundo caso de dispersión en un cuerpo dentro de otro. En el problema en estudio, existe una combinación de los dos, ya que se encuentran numerosos microhilos dentro de un compósito, por lo que será necesario generalizar la matriz de ecuaciones a usar.

En el presente trabajo se trabajará con 3 y 6 hilos, de 15 y $10[\mu m]$ de diámetro, respectivamente, separados por $2[mm]$ cada uno y de $1[cm]$ de largo, en una matriz de $16[mm] \times 16[mm] \times 1[mm]$ para el primer caso y $12[mm] \times 12[mm] \times 0,3[mm]$ para el segundo caso. Con estas medidas se tiene una proporción volumétrica hilo/matriz del 0,0022 % y 0.01 % respectivamente, para 3 y 6 hilos.

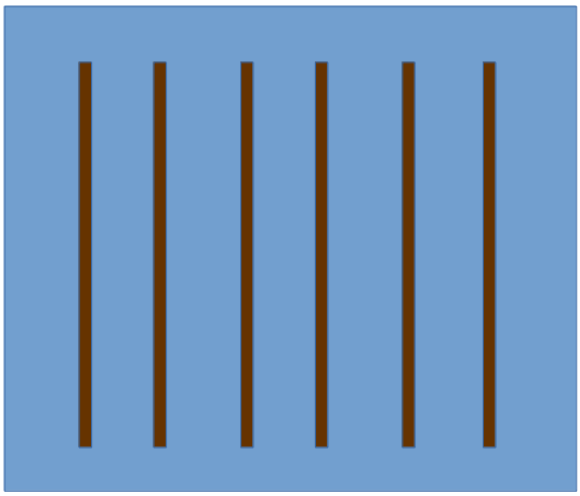


Figura 24: Disposición de los microhilos dentro del compósito.

7.1. Formulación de las ecuaciones

Uno de las variables más importantes cuando se considera dispersión de una onda sobre un objeto, es el índice de refracción, el cual corresponde al cociente de la velocidad de un fenómeno ondulatorio como luz o sonido en el de un medio de referencia, respecto a la velocidad de fase (v_p) en dicho medio:

$$n = \frac{c}{v_p}. \tag{7.1}$$

Generalmente se utiliza la velocidad de la luz en el vacío ($c = 299{,}792{,}458[m/s]$) como medio de referencia, y en este caso eso:

$$n = \sqrt{\varepsilon_r \mu_r} \tag{7.2}$$

donde como se mencionó anteriormente, aparecen variables importantes como lo son μ y ε, que en este caso son los valores relativos, por lo que si se quiere utilizar el índice de refracción para los valores que se tienen de hilos, se debe reescribir la expresión como:

$$n = \frac{\sqrt{\mu\varepsilon}}{\sqrt{\mu_0\varepsilon_0}}, \tag{7.3}$$

donde μ y ε corresponde a la permeabilidad y permitividad del hilo, mientras que μ_0 y ε_0 a los de la luz en el vacío, los cuales ya se mencionaron en las secciones anteriores.

Además, anteriormente se mencionó la variable α en las condiciones de transmisión del campo en los hilos, la cual para el caso presente se define como:

$$\alpha = \frac{\mu_{int}}{\mu_{ext}}. \tag{7.4}$$

Es por lo anterior, que es de suma importancia conocer el valor de estas variables para la simulación de dispersión.

Finalmente, con estas variables en consideración, y teniendo en cuenta que el problema a resolver será con más de un microhilo, es necesario replantear las ecuaciones para generalizar la matriz. Antes, se debe establecer una nomenclatura a seguir. Como en el caso anterior, el subíndice "*i*" indica interior, el subíndice "*e*" indica exterior, "*m*" se refiere a la matriz (*mm* indica efecto de matriz sobre sí misma) y la enumeración indica el número del hilo que se esta relacionando $(1, 2, ..., n)$. Por ejemplo, S_{ext}^{2m} corresponde al operador de capa simple exterior que opera con respecto a la influencia del hilo número 2 sobre la matriz.

La matriz se presenta a continuación:

$$
\begin{bmatrix}
-D_e^{mm}-D_i^{mm} & S_i^{mm}+\frac{1}{\alpha}S_e^{mm} & D_{ext}^{1m} & -S_e^{1m} & D_{ext}^{2m} & -S_e^{2m} & \cdots & D_e^{nm} & -S_e^{nm} \\
-D_e^{'mm}-D_i^{'mm} & (\frac{\alpha-1}{2\alpha})+S_i^{'mm}+\frac{1}{\alpha}S_e^{'mm} & D_e^{'1m} & -S_e^{'1m} & D_e^{'2m} & -S_e^{'2m} & \cdots & D_e^{'nm} & -S_e^{'nm} \\
-D_i^{n1} & S_i^{n1} & -D_e^{11}-D_i^{11} & \alpha S_i^{11}+S_e^{11} & D_e^{21} & -S_e^{21} & \cdots & D_e^{n1} & -S_e^{n1} \\
-D_i^{'m1} & S_i^{'m1} & -D_e^{'11}-D_i^{'11} & (\frac{\alpha-1}{2\alpha})+\alpha S_i^{'11}+S_e^{'11} & D_e^{'21} & -S_e^{'21} & \cdots & D_e^{'n1} & -S_e^{'n1} \\
-D_i^{n2} & S_i^{n2} & D_e^{12} & -S_e^{12} & -D_e^{22}-D_i^{22} & \alpha S_i^{22}+S_e^{22} & \cdots & D_e^{n2} & -S_e^{n2} \\
-D_i^{'m2} & S_i^{'m2} & D_e^{'12} & -S_e^{'12} & -D_e^{'22}-D_i^{'22} & (\frac{\alpha-1}{2\alpha})+\alpha S_i^{'22}+S_e^{'22} & \cdots & D_e^{'n2} & -S_e^{'n2} \\
\vdots & & \vdots & & \vdots & & & \vdots & \\
-D_i^{mn} & S_i^{mn} & D_e^{1n} & -S_e^{1n} & D_e^{2n} & -S_e^{2n} & \cdots & -D_e^{nn}-D_i^{nn} & \alpha S_i^{nn}+S_e^{nn} \\
-D_i^{'mn} & S_i^{'mn} & D_e^{'1n} & -S_e^{'1n} & D_e^{2n} & -S_e^{2n} & \cdots & -D_e^{'nn}-D_i^{'nn} & (\frac{\alpha-1}{2\alpha})+\alpha S_i^{'nn}+S_e^{'nn}
\end{bmatrix}
\begin{bmatrix}
u_m^i \\
\frac{\partial u_m^i}{\partial n} \\
u_1^e \\
\frac{\partial u_1^e}{\partial n} \\
u_2^e \\
\frac{\partial u_2^e}{\partial n} \\
\vdots \\
u_n^e \\
\frac{\partial u_n^e}{\partial n}
\end{bmatrix}
=
\begin{bmatrix}
u_{inc}^1 \\
\frac{\partial u_{inc}^1}{\partial n} \\
0 \\
0 \\
0 \\
0 \\
\vdots \\
0 \\
0
\end{bmatrix}
\tag{7.5}
$$

Se necesita entonces encontrar un método para obtener la permitividad y permeabilidad de los hilos. Para esto, se realizará un proceso en el cual desde valores de las permitividades efectivas del compósito se obtendrán los valores del hilo (análogamente para la permeabilidad), y luego se intentará comprobar su validez en base al código computacional que se presentará más adelante.

7.1.1. Obtención de propiedades de los hilos

En compósitos electrodinámicos, los microhilos conductores pueden generar propiedades de polarización inusual, como se mencionó anteriormente, en respuesta a radiación de distintas frecuencias. Como se propuso, los materiales serán tratados como un medio continuo, al menos en el rango del campo de radiación con una permitividad efectiva ε_{ef}.

La habilidad de estos microhilos para manipular la radiación electromagnética puede ser visto como una consecuencia de la dispersión de ε_{ef}. Estos materiales en el compósito demuestran la dispersión de la permitividad con frecuencias determinadas por la antena de resonancia $f_{res,n} = c(2n-1)/(2l\sqrt{\varepsilon})$, donde c es la velocidad de la luz, l es el largo de los hilos, ε es la permitividad de la matriz y n es un número entero. Las concentraciones volumétricas de hilos p deben cumplir con el límite $p < p_l \sim 2a/l$, donde a es el radio del microhilo. La forma general de la permitividad se da de la siguiente manera:

$$\varepsilon_{ef}(w) = \varepsilon + 4\pi p \sum_{n=1}^{\text{ínf}} \frac{A_n}{(w_{res,n}^2 - w^2) - iw\Gamma_n}, \tag{7.6}$$

donde la suma se realiza en todas las frecuencias de resonancia de antena $w_{res,n} = 2\pi f_{res,n}$, y la fuerza y relajación de los osciladores son descritos por los parámetros fenomenológicos A_n y Γ_n, respectivamente. Esto parámetros son contribuidos por pérdidas magnéticas y de resistividad.

Como se mencionó anteriormente, cuando el campo eléctrico es irradiado paralelamente sobre los microhilos, se crea un campo magnético circular en el radio. La dependencia de la impedancia en la superficie en las propiedades magnéticas se toman en consideración ya que se puede establecer una relación, en la cual la polarización η y la permitivdad efectiva ε_{ef} toman una forma analítica simple:

$$\eta = \frac{1}{2\pi ln(l/a)(\tilde{k}/a)^2}\left(\frac{2}{\tilde{k}l}tan(\tilde{k}l/2) - 1\right) \tag{7.7}$$

y

$$\varepsilon_{ef} = \varepsilon + 4\pi p\eta, \tag{7.8}$$

donde $\tilde{k}$ es el número de onda renormalizado.

La polarización η fue derivada asumiendo que las pérdidas de radiación son insignificantes en comparación con las magnéticas y de resistividad, lo cual es razonable en un caso de "skin efect" moderado. Con esta polarización, la permitividad efectiva para ondas en los microhilos tiene una dispersión de plasma característica que resulta $\varepsilon_{ef} = \varepsilon - (w_p/w)^2$ pero con frecuencia de plasma reducida w_p,

$$w_p = \frac{2\pi c^2}{L^2 ln(L/a)}, \tag{7.9}$$

donde L es la distancia promedio entre hilos.

La permitividad efectiva con microhilos no-magnéticos es deducida utilizando una ténica de homogeinización, resolviendo las ecuaciones de Maxwell:

$$\varepsilon_{ef} = \varepsilon - p\frac{2\varepsilon_c F_1(k_c a)}{(ak_c)^2 F_1(k_c a) ln(L/a) - 1} \tag{7.10}$$

$$F_1 = J_1(x)/xJ_0(x), \tag{7.11}$$

donde $p = \pi a^2/L^2$ es la concentración volumétrica de hilos, $\varepsilon_c = 4\pi i\sigma/w$ es la permitividad dieléctrica del conductor, σ es la conductividad de los hilos, $k_c^2 = 4\pi iw\sigma/c^2$ es el número de onda en el hilo y $J_{0,1}$ son las funciones de Bessel. [13]

Con esta ecuación, es posible obtener el valor de la permitividad de los hilos ε_c con algún método de resolución de ecuaciones. En el presente caso, se utilizará un método iterativo de resolución llamado "punto fijo". Para la resolución se requiere de ciertos valores que se obtendrán del siguiente gráfico:

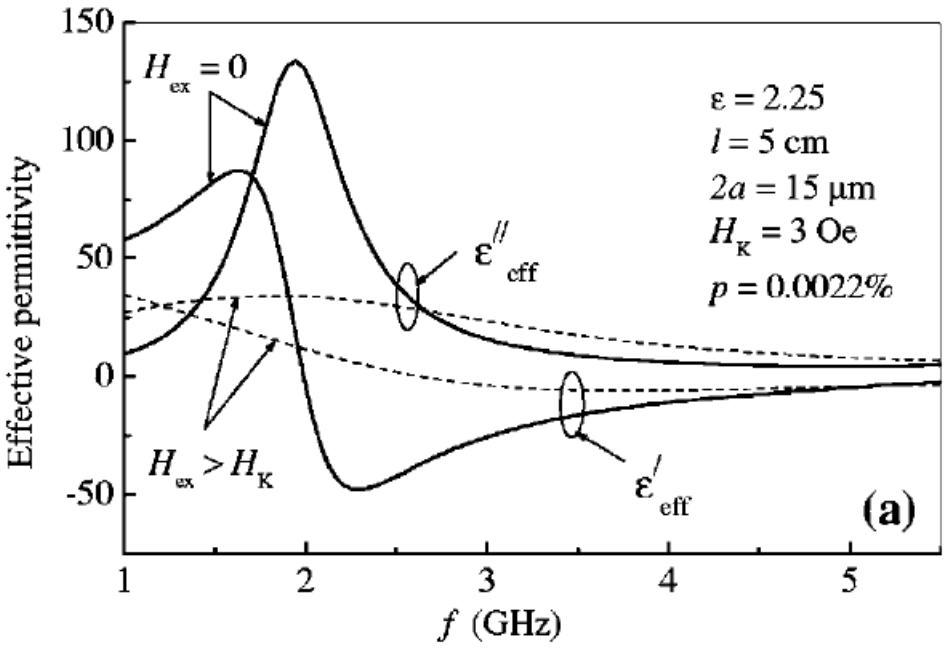


Figura 25: Permitividad efectiva con anisotropía circunferencial de acuerdo a la frecuencia.

El código es el siguiente:

```
########################## IMPORTANDO LIBRERIAS ##########################
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve
from mpmath import *
from sympy import *
```

```
from scipy import special as sp
from numba import vectorize
from matplotlib import pyplot
%matplotlib inline

######################### DEFINING VALUES ##############################
eff = np.array([95.-6.j, 115.-30.j, 90.-110.j, -15.-120.j, -30.-50.j, -15.-25.
    j, -5.-18.j, -12.j, 5.-10.j, 10.-10.j, 8.-8.j, 8.-5.j, 8.-2.j, 8.-2.j],
dtype=complex)              #Permitividad efectiva de acuerdo al grafico.
print len(eff)
eff_r = np.real(eff) #Parte real de la efectividad
eff_i = np.imag(eff) #Parte imaginaria de la efectividad


ec #Permitividad hilo, incognita
e = 16                      #Permitividad de la matriz
f = np.linspace(1e9, 14e9, len(eff))
w = f*2.*np.pi              #Frecuencia angular
a = 5e-6                    #Radio del hilo
c = 3e8                     #Velocidad de la luz
p = 0.01/100.               #Concentracion volumetrica
L = np.sqrt(np.pi*a**2/p)   #Distancia promedio entre hilos
lc= 0.01                    #Largo del hilo
largo=0.38                  #Largo composito
ancho=0.34                  #Ancho composito
prof=0.3/1000.              #Espesor composito
volm=largo*ancho*prof       #Volumen composito
volc=np.pi*a**2*lc          #Volumen hilos
volct=volm*p                #Volumen todos los hilos
n=volct/volc                #Numero de hilos
nh=n*1e-4/(largo*ancho)     #Numero hilos por area de composito

print "Mid distance between wires =",L, '[m2]'
print "Wire volume =", volc, '[m3]'
print "All wires sum volume =", volct, '[m3]'
print "Composite volume =", volm, '[m3]'
print "Number of wires =", n, '[-]'
print "Number of wires in 1[cm2] of composite =", nh

######################### SOLVER ######################################
x = symbols('x')
ec = np.ones(len(eff_r), dtype=complex)


for i in range(len(eff_r)):
err_r=10.
err_i=10.
cont=0
while True:
if err_r>1e-3 or err_i>1e-3:
ecp = ec[i]
J1 = sp.jv(1,(complex(ec[i])**0.5)*w[i]*a/c)
J0 = sp.jv(0,(complex(ec[i])**0.5)*w[i]*a/c)
```

```
F1 = J1/((complex(ec[i])**0.5)*w[i]*a*J0/c)
sol = solve(e - (np.pi*a**2/L**2)*(2*x*F1)/((((x**0.5)*w[i]*a/c)**2*F1*np.log(
    L/a))-1)
- complex(eff_r[i], eff_i[i]) , x)
ec[i] = 0.5*sol[0]+0.5*ecp
err_r = abs(np.real(ecp)-np.real(ec[i]))
err_i = abs(np.imag(ecp)-np.imag(ec[i]))
cont+=1
if cont%100==0:
print 'lleva', cont, 'iteraciones'
#            print 'y un valor de permitividad:', ec[i]
elif err_r<1e-3 and err_i<1e-3:
#            print 'Number of iterations=', cont
#            print sol
break
ec[i] = complex(sol[0])
print ec
print '\nla permitividad en', f[9]/1e9, '[GHz] es', ec[9]
```

El código entrega lo siguiente:

```
Mid distance between wires = 0.000886226925453 [m2]
Wire volume = 7.85398163397e-13 [m3]
All wires sum volume = 3.876e-09 [m3]
Composite volume = 3.876e-05 [m3]
Number of wires = 4935.07647539 [-]
Number of wires in 1[cm2] of composite = 3.81971863421


[ 771953.17862464  -57274.99537698j  893285.99037402 -239829.12848419j
795285.07037763 -722452.12725188j  400430.87476208-1139694.31394582j
-204325.42013928 -875821.97824659j -270671.57149929 -482813.69844903j
-178998.88620416 -327148.92844371j -156531.18078631 -224024.14403977j
-94874.78994346 -167197.23778068j  -25589.15565186 -131017.27392027j
-57515.68388428 -135643.73179934j  -87641.35948409 -104633.97246396j
-124678.71900071  -54409.48184288j -135299.67983216  -67014.30968642j]

La permitividad en 10.0 [GHz] es (-25589.1556519-131017.27392j),
```

con lo cual se puede plotear todas las soluciones con respecto a las distintas frecuencias, obteniéndose finalmente:

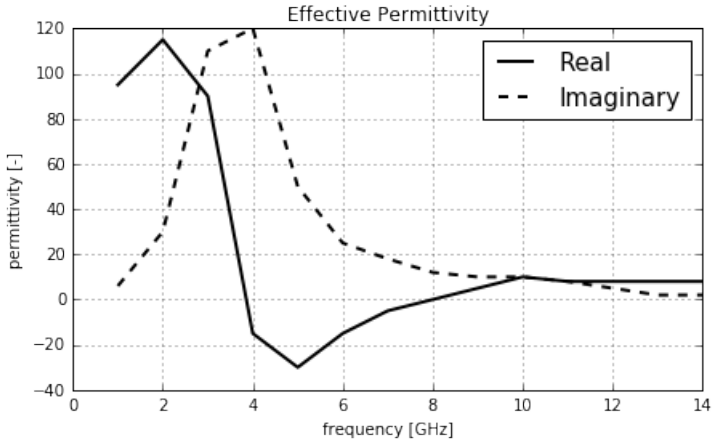


Figura 26: Permitividad efectiva con la solución análitica.

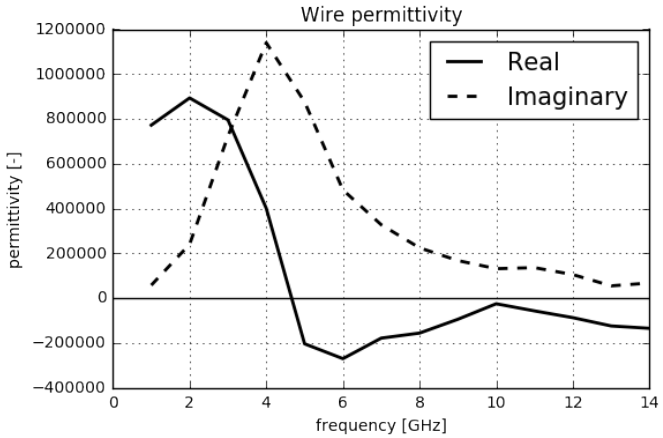


Figura 27: Permitividad de los hilos para distintas frecuencias.

Ahora es necesario obtener el valor de la permeabilidad de los hilos para distintas frecuencias. Considerando aproximaciones lineales respecto a parámetros E,B y M (magnetización) y encontrando las soluciones a las ecuaciones de Maxwell, para lo cual no se entrará en detalle, se obtienen la siguiente expresion:

$$\tilde{\chi} = \frac{\omega_M(\omega_2 - i\tau\omega) + 4\pi\omega_M^2}{(\omega_1 - i\tau\omega)(\omega_2 + 4\pi\omega_M - i\tau\omega) - \omega^2}, \tag{7.12}$$

Se necesita saber entonces:

$$
\begin{aligned}
\chi_1 &= \omega_M(\omega_1 - i\tau\omega)/\Delta \\
\chi_2 &= \omega_M(\omega_2 - i\tau\omega)/\Delta \\
\chi_a &= \omega\omega_M/\Delta \\
\Delta &= (\omega_2 - i\tau\omega)(\omega_1 - i\tau\omega) - \omega^2 \\
\omega_1 &= \gamma[H_{ex}cos(\theta) + H_k cos2(\psi - \theta)] \\
H_k &= 2K/M_0 \\
\omega_2 &= \gamma[H_{ex}cos(\theta) + H_k cos^2(\psi - \theta)] \\
\omega_M &= \gamma M_0,
\end{aligned}
\tag{7.13}
$$

donde γ es la constante giromagnética, τ es el parámetro de relajación, H_k es ek campo de anisotropía, H_{ex} es el campo magnético externo a lo largo del eje del hilo, K es la constante de anisotropía, θ es el ángulo entre la magnetización estática M_0 y el eje del hilo.

Una vez obtenidos estos valores, es posible comenzar a calcular la permeabilidad del hilo de la forma:

$$\mu = 1 + 4\pi\tilde{\chi}, \tag{7.14}$$

Para esto, se requieren ciertos valores:

- $H_k = 2[Oe]$
- $H_{ex} = 0[Oe]$
- Constante giromagnetica $(\gamma) = 2 \times 10^7 [\frac{rad/s}{Oe}]$
- Parámetro de relajación de spin $(\tau) = 0{,}06$
- Magnetización estática = $500[G]$

Además, es necesario conocer los ángulos de magnetización θ y de anisotropía ψ, que se obtienen del siguiente gráfico:

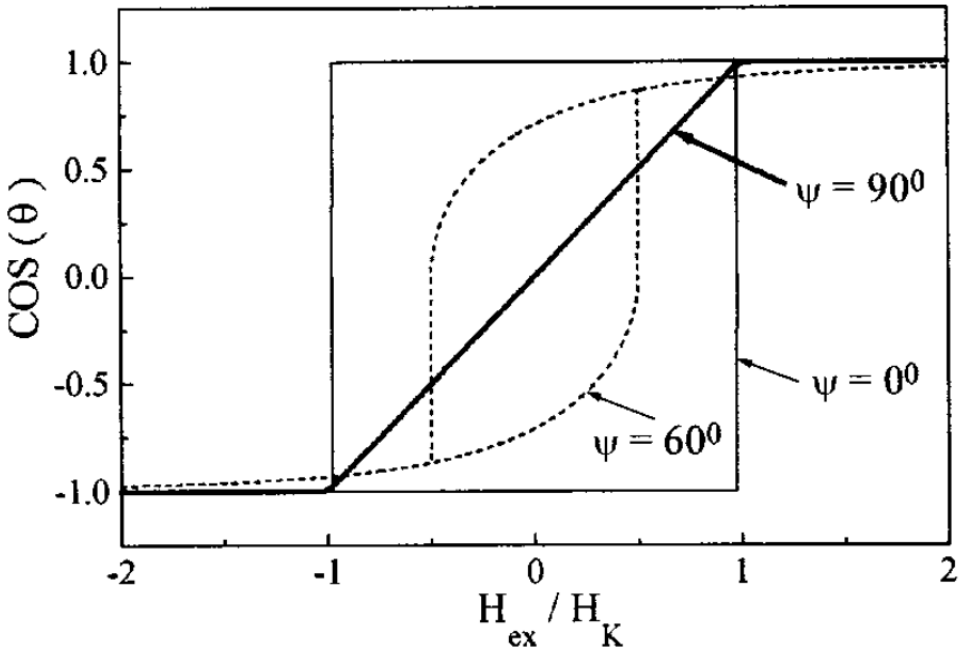


Figura 28: Rotación típica de curvas de histéresis para diferentes clases de anisotropía: longitudinal ($\psi = 0°$), circunferencial ($\psi = 90°$), y helicoidal ($\psi = 60°$).

Finalmente, se procede a calcular la permeabilidad con el código siguiente:

```
#### IMPORTANDO LIBRERIAS ####
from matplotlib import pyplot as pl
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve
from mpmath import *
from sympy import *
from scipy import special as sp
from matplotlib import rcParams

####PLOTEAR EN EL NOTEBOOK####
%matplotlib inline

#### EXPRESIONES Y VALORES ####
f=np.linspace(10,10e9,100)
w = np.array(f)*np.pi*2  #Frecuencia angular
gamma = 2e7              #[rad/s/Oe] Giromagnetic constant
Hex = 0.                 #No external field
Hk = 2.                  #[Oe] Anisotropy field
psi = np.pi/2.           #90 grados
theta = np.pi/2.         #Grafico
w1 = gamma*(Hex*np.cos(theta)+Hk*np.cos(2*(psi-theta)))
w2 = gamma*(Hex*np.cos(theta)+Hk*(np.cos(psi-theta))**2)
M0 = 500. #[G] saturation magnetization
```

```
wm = gamma*M0
tau = 0.06
n=len(w)
#dummy=np.linspace(1,1e9,n)
valores=[]
for i in range(n):
valores = np.append(valores,(wm*(w2-1j*tau*w[i])+4*np.pi*(wm**2))/((w1-1j*tau*
    w[i])*(w2+4*np.pi*wm-1j*tau*w[i])-(w[i]**2)))

mu=1+4*np.pi*valores


rcParams['font.family'] = 'serif'
rcParams['font.size'] = 20

pl.figure(figsize=(15,10))
#pl.xticks(dummy[::100], omega[::100] , rotation=75)
pl.title('Microwire Permeability \n', fontsize=30)
pl.plot(f/1e9,np.real(mu),'k-', linewidth=3,label='Real')
pl.plot(f/1e9,np.imag(mu),'k--', linewidth=3,label='Imaginary')
pl.legend(loc='best', fontsize=40)
pl.xlabel('Frecuency [GHz]', fontsize=20)
pl.ylabel('Permeability [-]', fontsize=20)
pl.grid();
```

El código anterior entrega un gráfico de la permeabilidad del hilo en las distintas frecuencias.

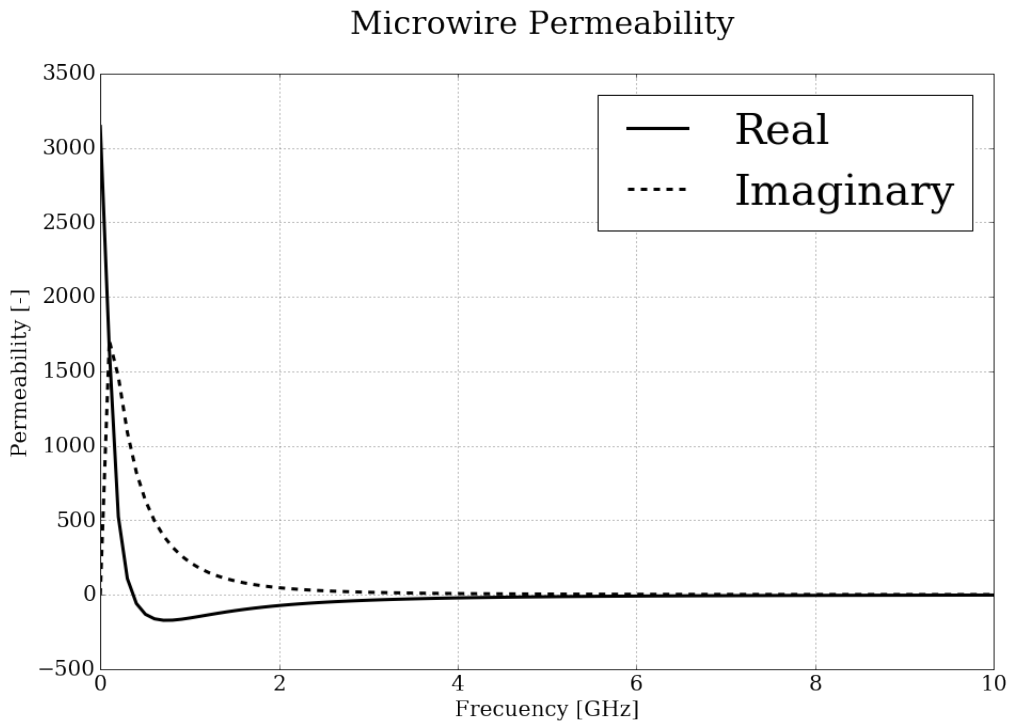


Figura 29: Permeabilidad de un hilo para distintas frecuencias.

Teniendo estos valores, se procede al cálculo de los campos involucrados.

7.2. Utilización del código para la resolución

Se explicará paso por paso el proceso a seguir para la verificación de la herramienta. En primer lugar, el código presenta una ventana de datos de entrada:

```
################################################################################
#### ESTE ARCHIVO CONTIENE LOS PARAMETROS DE ENTRADA PARA RESOLVER EL PROBLEMA ####
################################################################################

#PROPIEDADES MAGNETICAS (relativas)
#AMPLITUD CAMPO  FRECUENCIA  MU_MEDIO   EPS_MEDIO     MU_HILO          EPS_HILO          DIST_RECEPTOR
1.               12e9        (1.)      (8.-5.j)  (-1.734+0.37j) (-87641.359-104633.972j) [[1e4],[0.],[0.]]

#IMPORTAR MALLAS (/direccion/nombre.msh)
#### MALLA DEL MEDIO ###########################################################
'/home/milan/matriz_12x12x300_E16772.msh'

#### MALLAS DE LOS HILOS #######################################################
#'/home/milan/PH1_a5_l10_E5550_D2.msh'
#'/home/milan/PH2_a5_l10_E5550_D2.msh'
#'/home/milan/PH3_a5_l10_E5550_D2.msh'
#'/home/milan/PH4_a5_l10_E5550_D2.msh'
#'/home/milan/PH5_a5_l10_E5550_D2.msh'
#'/home/milan/PH6_a5_l10_E5550_D2.msh'
################################################################################
```

Figura 30: Datos de entrada para el código computacional.

Una vez ingresados los datos, se pueden tener tres distintos escenarios. El primero, corresponde al código llamado “conector”, el cual recibe la información y genera el código que se ejecutará

(Anexo A). El segundo corresponde al código con 6 hilos y una matriz contenedora (Anexo B). Finalmente, el tercer código corresponde a la matriz sin hilos, pero con valores efectivos (Anexo C). Es importante mencionar que tanto el código de la data como el código del Anexo A, son comúnes para los códigos con y sin hilos.

El código base contiene importantes partes de acuerdo a lo mencionado anteriormente en la sección de BEM++.

Primero, es necesario importar la biblioteca mencionada:

```
import bempp.api
```

Luego se requiere importar las mallas de la forma:

```
malla= = bempp.api.import_grid('nombremalla.msh')
```

para todas las geometrías utilizadas.

Luego se representa la onda incidente de la dispersión:

```
def fun(x, n, domain_index, result):
  result[0] = 1.*np.exp(1j*k*x[0])
```

donde en ella se indica la amplitud del campo y su dirección.

Una de las partes más importantes del código es la generación de los operadores a utilizar con funciones para problemas de frontera de Dirichlet y de Neumann. Esto se puede juntar en un operador multitrazo, el cual contiene a los demás operadores como SLP, DLP, HYP, ADJ.

```
operadormultritrace = bempp.api.operators.boundary.helmholtz.
    multitrace_operator(malla, n*k)
```

Luego, estos operadores multitrazo, como se notó en la formulación de la matriz con las ecuaciones, deben agruparse en bloques para su resolución, lo cual BEM++ permite de la forma:

```
blocked = bempp.api.BlockedOperator(filas,columnas)
```

Se puede simular el campo incidente que se definió arriba de la forma

```
grid_fun = bempp.api.GridFunction(space, fun=fun)
```

Luego, se discretizan los bloques creados con un método llamado “strong form” incluido en la bibliteca, el cual realiza una discretización de coeficientes directamente.

Ya teniendo lo anterior, se arma la parte del lado derecho de la ecuación (7.5) con los valores de la onda incidente:

```
grid_fun.coefficients
```

y se resuelve el sistema de ecuaciones con GMRES (Generalized minimal residual method), que corresponde a un método iterativo para la solución numérica de sistemas de ecuaciones lineales no-simétricas.

Se reemplaza la solución del sistema de ecuaciones en las ecuaciones

$$\begin{aligned} u_{ext}(\mathbf{r}) &= [Du_{ext}](\mathbf{r}) - \left[S\frac{\partial u_{ext}}{\partial n}\right](\mathbf{r}) \\ u_{int}(\mathbf{r}) &= \left[S\frac{\partial u_{int}}{\partial n}\right](\mathbf{r}) - [Du_{int}](\mathbf{r}), \end{aligned} \tag{7.15}$$

las cuales se obtienen con el uso de operadores en la ecuación (5.23), y se obtiene el valor del campo utilizado en el lugar que se propuso, en este caso 1 [cm] del compósito.

7.3. Entrada de datos y resultados

7.3.1. Caso 1

Se ingresan los datos necesarios, que para el primer caso corresponde a una matriz con 3 hilos de 15 $[\mu m]$ de diámetro y una proporción de 0,0022 %, como se explicó al principio del trabajo, y utilizando valores de los gráficos se ingresan:

- Amplitud de campo = 1. $[N \cdot C^{-1}]$
- Frecuencia = 5,5 $[GHz]$
- Permeabilidad matriz contenedora = 1.
- Permitividad de la matriz = 12
- Permeabilidad hilo = -5.763+6.665j
- Permitividad hilo = 47871.53 -264695.2595j

Una vez utilizado el código se obtiene:

```
Numero de onda exterior: 0.000115270253615
Indice de refraccion matriz: 3.46410161514
Indice de refraccion conductor: (1388.95385172+663.989853106j)
Numero de onda interior matriz: 0.000399307871724
Numero de onda interior conductor: (0.554620206453+0.26513637509j)
Indice de transmision matriz: 1.0
Indice de transmision conductor: (-5.763+6.665j)
Longitud de onda: 54508.2977624 micras
Shape of matrix: (33586, 33586)
```

```
El sistema fue resuelto en 153 iteraciones
Valor del campo en receptor: [ 0,47456856 + 1,13788863j]
```

Como se puede ver, el valor del campo obtenido es de 0,47456856 + 1,13788863j $[N \cdot C^{-1}]$. Este valor corresponde al campo medido a 1 $[cm]$ del compósito, con 3 hilos dentro de la matriz, y tomando cada componente por separado, utilizando las ya obtenidas anteriormente propiedades de los hilos.

El siguiente paso es comprobar que esto es cierto. Para eso, se procede a utilizar los valores de permitividad y permeabilidad efectivas para el compósito, pensando que éste es el compósito con hilos. Luego, se irradiará el campo nuevamente, y se comparará la respuesta obtenida con el caso anterior con hilos.

Ahora los datos de entrada son:

- Amplitud de campo = 1. $[N \cdot C^{-1}]$
- Frecuencia = 5,5 $[GHz]$
- Permeabilidad efectiva = $2 - 1j$
- Permitividad efectiva = $0 - 5j$

y se debe tener cuidado con quitar los hilos de la prueba.

Una vez finalizado el código, se entregan los resultados:

```
Numero de onda exterior: 0.000115270253615
Indice de refraccion matriz: (1.75788792127-2.84432240503j)
Indice de refraccion conductor: (1388.95385172+663.989853106j)
Numero de onda interior matriz: (0.000202632186511-0.00032786576499j)
Numero de onda interior conductor: (0.499146297071-0.320844701373j)
Indice de transmision matriz: (2-1j)
Indice de transmision conductor: (-3.6382+1.5134j)
Longitud de onda: 54508.2977624 micras
Shape of matrix: (16776, 16776)
El sistema fue resuelto en 15 iteraciones
Valor del campo en receptor: [ 0.4709458 + 0.71906953j]
```

donde se puede ver que el campo resultante es muy similar al obtenido con 3 hilos y las propiedades por separado.

Si se calcula el error según la ecuación

$$e_{rel} = |\frac{f_h - f_{sh}}{f_{sh}}| \cdot 100\,\%, \tag{7.16}$$

donde f_h es el valor con hilos y f_{sh} sin hilos, se obtiene un $0,7\,\%$.

7.3.2. Caso 2

Los valores obtenidos anteriormente del campo fueron comprobados con datos y condiciones similares a las del paper donde se encontraba información sobre como obtener permeabilidad y permitividad de los hilos. [13] Es por eso, que ahora se cambiarán algunas condiciones del problema, y se realizarán los mismos procedimientos anteriores para 6 hilos de 10 $[\mu m]$ de diámetro y una proporción de 0,01 % para diferentes condiciones de trabajo. [11]

Los datos son:

- Amplitud de campo = 1. $[N \cdot C^{-1}]$
- Frecuencia = 12 $[GHz]$
- Permeabilidad matriz contenedora = 1.
- Permitividad de la matriz = 16
- Permeabilidad hilo = -1.733+0.3725j
- Permitividad hilo = -87641.359 -104633.972j

Para lo cual se obtiene:

```
Numero de onda exterior: 0.000251498735159
Indice de refraccion matriz: 4.0
Indice de refraccion conductor: (465.126197069+160.18018933j)
Numero de onda interior matriz: 0.00100599494064
Numero de onda interior conductor: (0.467914601009+0.161140460057j)
Indice de transmision matriz: 1.0
Indice de transmision conductor: (-1.734+0.37j)
Longitud de onda: 24982.9698078 micras
Shape of matrix: (50100, 50100)
El sistema fue resuelto en 2709 iteraciones
Valor del campo en receptor: [-1.06694707-0.05291927j]
```

donde se ve que el valor del campo es de -1.06694707-0.05291927j $[N \cdot C^{-1}]$. Ahora con el mismo procedimiento anterior para el compósito sin hilos, se ingresan los datos:

- Amplitud de campo = 1. $[N \cdot C^{-1}]$
- Frecuencia = 12 $[GHz]$
- Permeabilidad efectiva = $1 - 0j$
- Permitividad efectiva = $8 - 5j$

Se obtiene entonces:

```
1
Numero de onda exterior: 0.000251498735159
3 Indice de refraccion matriz: (2.95245500661-0.846752954544j)
Indice de refraccion conductor: (465.126197069+160.18018933j)
5 Numero de onda interior matriz: (0.000742538699776-0.00021295729706j)
Numero de onda interior conductor: (0.379485741766+0.0198879717955j)
7 Indice de transmision matriz: 1.0
Indice de transmision conductor: (-1.734+0.37j)
9 Longitud de onda: 24982.9698078 micras
Shape of matrix: (16776, 16776)
11 El sistema fue resuelto en 11 iteraciones
Valor del campo en receptor: [-0.97682803+0.7664753j]
```

Se obtiene un valor del campo de -0.97682803+0.7664753j $[N \cdot C^{-1}]$. Si se compara este valor con el compósito con hilos, se obtiene un error relativo de un **9,2 %**.

Para el **caso número 1**, el valor del campo en la posición medida sin compósito que lo afecte, corresponde a **0.40601917+0.91386456j** $[N \cdot C^{-1}]$ y para el **caso número 2** corresponde a **-0.81002282+0.58639836j** $[N \cdot C^{-1}]$.

Con esto se puede medir el porcentaje de cambio del campo cuando no hay objeto que se interponga en su camino, y cuando existe un un compósito. Se obtienen los siguientes resultados:

- Caso 1: 15,99 %
- Caso 2: 20,59 %

Además se puede medir el error de cambio en los módulos de los valores del campo, considerando la parte imaginaria:

- Caso 1: 43 %
- Caso 2: 13,96 %

8. Análisis y conclusiones

En el trabajo presente se comentó la utilización de microhilos para la medición de esfuerzos mecánicos en aplicaciones de ingeniería. Se establecieron formulaciones matemáticas para su desarrollo y se diseño un código computacional para el cálculo de las propiedaes más importantes de los microhilos, como lo son la permeabilidad y la permitividad. En la sección anterior se realizaron pruebas de validez para el código computacional desarrollado.

Los valores de frecuencia que se eligieron corresponden a valores de los gráficos en que la permitividad de los microhilos y del compósito no son tan altos, y su parte compleja no se aleja demasiado del cero. En estos valores, se logró demostrar que los resultados obtenidos de la simulación son bastante cercanos a los datos experimentales obtenidos de los papers citados, con errores incluso menores al 1 %. Esto se logró obteniendo las propiedades de los microhilos en base a propiedades efectivas, para luego utilizar ambas por separados y ser comparadas. Si bien existen

referencias que contienen valores de permitividades mas pequeñas, estas no consideran el hecho de obtener las propiedades analíticamente, por lo que el problema se restringío a las referencias que si validaban el trabajo actual.

Se desarrollaron códigos para la simulación de la dispersión del campo electromagnético en los compósitos, que corresponde al código de mayor complejidad, para la obtención de la permitividad del hilo y la permeabilidad del hilo. Los resultados obtenidos para las propiedades de los microhilos se obtuvieron con metodos iterativos simples y se les tiene para distintas frecuencias.

Sin embargo, para frecuencias distintas a las mencionadas anteriormente, se generaron numerosos problemas de convergencia de los códigos y/o valores muy alejados de la realidad, llegando incluso a varias miles de iteraciones, lo cual no es aceptable. En ciertas ocaciones, algunos valores escapaban de valores reales y se hacían infinitos, imposibilitando la resolución o iteración de soluciones.

Hay que tener ciertas consideraciones en cuenta, que pudiesen haber afectado ya sea la convergencia o los valores obtenidos. Los compósitos utilizados en los experimentos reales, eran de hasta incluso $34 \times 38 cm^2$ (en el trabajo se utilizó un compósito 300 veces más pequeño), para lo cual, según la proporción de hilos/compósito, implicaría utilizar cientos de hilos para el problema. Si bien esto es posible de desarrollar, implicaría que el aumento de la memoria RAM utilizada por el computador incrementara de manera absurda, escapándose de toda herramienta computacional de fácil acceso. En los códigos utilizados para 6 hilos ya se ocupaban valores cercanos a los 12 Gb de RAM. La relación de RAM con el trabajo va de la mano de los elementos de las geometrías, por lo que si esta aumenta 300 veces, el problema se hace complejo.

Otra causa de los problemas de convergencia, fueron los valores complejos utilizados. Al utilizar un complejo muy alto, las iteraciones aumentaban considerablemente, incluso estancándose en algún valor fijo. Esto podía darse también con problemas en las mallas, las cuales si bien fueron revisadas y corregidas numerosas veces, nunca son perfectas.

Como se mencionó en el trabajo, se utilizó la ecuación de Helmholtz en el borde para realizar la aproximación numérica del problema. Si bien la aproximación es aceptable, esta deja consideraciones de lado que son importantes, como por ejemplo, polarización. Para resolver el problema de la polarización, una de las opciones habría sido realizar la formulación matemática utilizando la ecuación armónica en el tiempo de Maxwell, la cual como se mostró en secciones anteriores, es soportada por BEM++. Sin embargo, eso significa tomar una perspectiva completamente distinta, y desarrollar el trabajo desde cero, dejándo de lado una ecuación únicamente dependiente del espacio.

Es importante mencionar además, que como se explicó en secciones anteriores, no se consideraron campos externos sobre el problema, los cuales podrían cambiar considerablemente los resultados, y en el caso de la impedancia, esta se consideró incluida en las propiedades de permeabilida y permitividad del material. Como se mencionó, la impedancia magnética genera en el material un campo magnético circular que altera el campo eléctrico, alterando las propiedades. Dado que los valores de las propiedades no se tenían, se debió considerar que estos estímulos de impedancia se encontraban en ellos, lo que podría implicar cambios en los resultados.

Finalmente, es posible decir que, si bien la simulación de dispersión sobre los compósitos da resultados aceptables, se deben arreglar numerosos factores que influyen sobre todo en la convergencia. La ecuación de Helmhotz resultó una herramienta sencilla para una primera aproximación del problema de dispersión. Esta ecuación permitía, además, facilidades para ecuaciones en los que

los bordes no se encuentraban unidos, ya que permitía afrontar el problema con solo las integrales borde, lo que permite considerar menos elementos. De todas formas, la resolución del problema con métodos de Maxwell podría significar un gran cambio y mejoría en los resultados.

Referencias

[1] David J. Griffiths. *Introduction to Electrodynamics*. 3rd Edition, 1998.

[2] W. Śmigaj, S. Arridge, T. Betcke, J. Phillips, M. Schweiger, *"Solving Boundary Integral Problems with BEM++",* ACM Trans. Math. Software 41, pp. $6:1-6:40$ (2015)

[3] S.P. Groth, A.J. Baran, T. Betcke, S. Havemann and W. Śmigaj *"The boundary element method for light scattering by ice crystals and its implementation in BEM++",* Journal of Quantitative Spectroscopy & Radiative Transfer 167 (2015) 40–52

[4] The BEM++ project,
`http://www.bempp.org/`

[5] The Gmsh project,
`http://gmsh.info/`

[6] R. E. Kleinman and G. F. Roach. *"Boundary integral equations for the three-dimensional Helmholtz equation",* SIAM Review. Vol. 16, No 2, April 1974.

[7] S. Amini and S. M. Kirkup. *"Solution of Helmholtz equation in the exterior domain by elementary boundary integral methods",* Journal of computational physics 118, 208-221 (1995).

[8] P. A. Martin. *Multiple scattering, Interaction of Time-Harmonic Waves with N obstacles*. 1st Edition, 2006.

[9] Xavier Claeys and Ralf Hiptmair. *"Multi-Trace Boundary Integral Formulation for Acoustic Scattering by Composite Structures",* Communications on Pure and Applied Mathematics, Vol. LXVI, 1163-1201 (2013).

[10] Xavier Claeys and Ralf Hiptmair. *"Electromagnetic Scattering at composite objects: A Novel Multi-Trace Boundary Integral Formulation",* ESAIM: M2AN 46 (2012) 1421-1445

[11] D. P. Markhnovskiy and L. V. Panina. *"Field dependent permittivity of composite materials containing ferromagnetic wires",* Journal of applied Physics 93, 4120 (2003).

[12] Manuel Vásquez, Anne-Lise Adenot-Engelvin *"Glass-coated amorphous ferromagnetic microwires at microwave frequencies",* Journal of Magnetism and Magnetic Materials 321 (2009) 2066-2073.

[13] D. P. Markhnovskiy and L. V. Panina. *"Experimental demonstration of tunable scattering spectra at microwave frequencies in composite media containing CoFeCrSiB glass-coated amorphous ferromagnetic wires and comparison with theory",* Physical Review B 74, 064205 (2006).

[14] Faxiang Qin, C. Brosseau, and H. X. Peng. *"In situ microwave characterization of microwire composites under mechanical stress",* Appl. Phys. Lett. 99, 252902 (2011).

[15] L. Liu, K.N. Rozanov and M. Abshinova *"Tunable properties of microwire composites at microwave frequency",* Appl. Phys. A (2013) 110:275-279.

Apéndice A: Código base que genera el código final con la entrada de datos

```
2 ################################################################
2 ###### ACA SE LEE EL PROGRAMA QUE CONTIENE LA INFORMACION ######
3 ################################################################
4 import numpy as np
5
6 info = []
7 for line in open("info.txt"):
8     li=line.strip()
9     if not li.startswith("#"): #Comentarios empiezan con '#'
10         info.append(line.split())
11
12 info = filter(None, info)
13 #print info
14 N_hilos = len(info) - 2
15 print '\nEl composito contiene', N_hilos, 'microhilos\n'
16
17
18 ################################################################
19 ###### ACA SE ESCRIBE EL PROGRAMA PARA EJECUTAR EL CALCULO #####
20 ################################################################
21 exe = open('ejecutor.py','w')
22
23 ###### IMPORTANDO LIBRERIAS ###################################
24 exe.write('import numpy as np\n')
25 exe.write('import bempp.api\n')
26
27 ###### PARAMETROS DE ENTRADA ##################################
28 #exe.write('bempp.api.global_parameters.quadrature.double_singular = 7\n')
   #Orden de cuadratura
29 exe.write('omega = 2.*np.pi*' + info[0][1] + '\n') #frecuencia angular
30 exe.write('e0 = 8.854*1e-12*1e-18\n') #permitividad del vacio
31 exe.write('mu0 = 4.*np.pi*1e-7*1e6\n') #permeabilidad del vacio
32 exe.write('mue = ' + info[0][2] + '*mu0\n') #permeabilidad de la matriz
33 exe.write('ee = ' + info[0][3] + '*e0\n') #permitividad de la matriz
34 exe.write('mui = ' + info[0][4] + '*mu0\n') #permeabilidad del conductor
35 exe.write('ei = ' + info[0][5] + '*e0\n') #permitividad del conductor
36 exe.write('k = omega*np.sqrt(e0*mu0)\n') #numero de onda exterior
37 exe.write('lam = 2*np.pi/k\n') #longitud de onda al exterior
38 exe.write('nm = np.sqrt((ee*mue)/(e0*mu0))\n') #indice de refraccion matriz
39 exe.write('nc = np.sqrt((ei*mui)/(e0*mu0))\n') #indice de refraccion
   conductor
40 exe.write('alfa_m = mue/mu0\n') #indice de transmision matriz
41 exe.write('alfa_c = mui/mue\n') #indice de transmision conductor
42 exe.write('antena = np.array('+info[0][6]+')\n') #Punto antena receptor
43
44 ###### ESCRIBIR VALORES DE INTERES ############################
```

```
45 exe.write('print "Numero de onda exterior:", k\n')
46 exe.write('print "Indice de refraccion matriz:", nm\n')
47 exe.write('print "Indice de refraccion conductor:", nc\n')
48 exe.write('print "Numero de onda interior matriz:", nm*k\n')
49 exe.write('print "Numero de onda interior conductor:", nm*nc*k\n')
50 exe.write('print "Indice de transmision matriz:", alfa_m\n')
51 exe.write('print "Indice de transmision conductor:", alfa_c\n')
52 exe.write('print "Longitud de onda:", lam, "micras"\n')
53
54 ###### IMPORTANDO MALLAS ########################################
55 exe.write('\n#Importando mallas\n')
56
57 exe.write('matriz = bempp.api.import_grid('+info[1][0]+')\n') #Malla de la
    matriz
58 for i in range(N_hilos): #Mallas de hilos
59     exe.write('grid_'+str(i)+' = bempp.api.import_grid('+info[i+2][0]+')\n'
    )
60
61 ###### FUNCIONES DIRICHLET Y NEUMANN ############################
62 exe.write('\n#Funciones de dirichlet y neumann\n')
63
64 exe.write('def dirichlet_fun(x, n, domain_index, result):\n')
65 exe.write('\tresult[0] = '+info[0][0]+'*np.exp(1j*k*x[0])\n')
66
67 exe.write('def neumann_fun(x, n, domain_index, result):\n')
68 exe.write('\tresult[0] = '+info[0][0]+'*1j*k*n[0]*np.exp(1j*k*x[0])\n')
69
70 ###### OPERADORES EN EL BORDE ###################################
71 exe.write('\n#Operadores identidad\n')
72
73 #exe.write('ident_m = bempp.api.operators.boundary.sparse.
    multitrace_identity(matriz)\n') #Operador identidad matriz
74 #exe.write('IT_m = 0.5*ident_m + 0.5*ident_m*(1./alfa_m)\n') #Identidad
    matriz en el borde y condiciones de transmision
75 #for i in range(N_hilos): #Operadores identidad en hilos
76 #    exe.write('ident_'+str(i)+' = bempp.api.operators.boundary.sparse.
    multitrace_identity(grid_' + str(i) + ')\n') #Identidad
77 #    exe.write('IT_'+str(i)+' = 0.5*ident_'+str(i)+' + 0.5*alfa_c*ident_'+
    str(i)+'\n') #Borde y transmision
78
79 exe.write('\n#Operadores multitrazo\n')
80 exe.write('Ai_m = bempp.api.operators.boundary.helmholtz.
    multitrace_operator(matriz, nm*k)\n') #Operador multitrazo interior matriz
81 exe.write('Ae_m = bempp.api.operators.boundary.helmholtz.
    multitrace_operator(matriz, k)\n') #Operador multitrazo exterior matriz
82 for i in range(N_hilos): #Operadores multitrazo hilos
83     exe.write('Ai_'+str(i)+' = bempp.api.operators.boundary.helmholtz.
    multitrace_operator(grid_' + str(i) + ',nm*nc*k)\n') #Interior hilos
84     exe.write('Ae_'+str(i)+' = bempp.api.operators.boundary.helmholtz.
    multitrace_operator(grid_' + str(i) + ',nm*k)\n') #Exterior hilos
85
86 exe.write('\n#Transmision en Multitrazo\n')
87
88
```

```
88 #exe.write('Ai_m[0,1] = Ai_m[0,1]*(1./alfa_m)\n') #Transmision en matriz
   interior
89 #exe.write('Ai_m[0,0] = Ai_m[0,0]*(1./alfa_m)\n') #Transmision en matriz
   interior
90 exe.write('Ae_m[0,1] = Ae_m[0,1]*(1./alfa_m)\n') #Transmision en matriz
   exterior
91 exe.write('Ae_m[1,1] = Ae_m[1,1]*(1./alfa_m)\n') #Transmision en matriz
   exterior
92 for i in range(N_hilos): #Transmision en Multitrazo en hilos
93     exe.write('Ai_'+str(i)+'[0,1] = Ai_'+str(i)+'[0,1]*alfa_c\n') #
   Transmision interior hilos
94     exe.write('Ai_'+str(i)+'[1,1] = Ai_'+str(i)+'[1,1]*alfa_c\n') #
   Transmision interior hilos
95 #    exe.write('Ae_'+str(i)+'[0,0] = Ae_'+str(i)+'[0,0]*alfa_c\n') #
   Transmision exterior hilos
96 #    exe.write('Ae_'+str(i)+'[0,1] = Ae_'+str(i)+'[0,1]*alfa_c\n') #
   Transmision exterior hilos
97
98 exe.write('\n#Acople interior y exterior\n')
99 exe.write('op_m = (Ai_m + Ae_m)\n') #Interior + exterior matriz
100 for i in range(N_hilos): #Interior + exterior hilos
101     exe.write('op_'+str(i)+' = (Ai_'+str(i)+' + Ae_'+str(i)+')\n')
102
103 exe.write('\n#Espacios\n')
104 exe.write('dirichlet_space_m = Ai_m[0,0].domain\n') #Espacio de dirichlet
   en matriz
105 exe.write('neumann_space_m = Ai_m[0,1].domain\n') #Espacio de neumann en
   matriz
106 for i in range(N_hilos): #Espacios en hilos
107     exe.write('dirichlet_space_'+str(i)+' = Ai_'+str(i)+'[0,0].domain\n')
   #Espacio de dirichlet en hilos
108     exe.write('neumann_space_'+str(i)+' = Ai_'+str(i)+'[0,1].domain\n') #
   Espacio de neumann en hilos
109
110 #operadores identidad
111 exe.write('ident_m = bempp.api.operators.boundary.sparse.identity(
   neumann_space_m, neumann_space_m, neumann_space_m)\n') #Operador identida
     d matriz
112 for i in range(N_hilos): #Operadores identidad en hilos
113     exe.write('ident_'+str(i)+' = bempp.api.operators.boundary.sparse.
   identity(neumann_space_' + str(i) + ', neumann_space_' + str(i) + ',
   neumann_space_' + str(i) + ')\n') #Identidad
114
115 #operadores diagonales
116 exe.write('op_m[1,1] = op_m[1,1] + 0.5 * ident_m * ((alfa_m -1)/alfa_m)\n'
   )
117 for i in range(N_hilos):
118     exe.write('op_'+str(i)+'[1,1] = op_' + str(i) + '[1,1] + 0.5 * ident_'
    + str(i) + '* (alfa_c - 1)\n')
119
120 #Operadores entre mallas
121 exe.write('\n#Operadores entre mallas\n')
122 for i in range(N_hilos): #Operadores entre mallas
```

```
123     exe.write('SLP_m_'+str(i)+' = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_m, dirichlet_space_'+str(i)+', dirichlet_space_'+str(i)+', nm*k)\n') #Operadores matriz-hilos single layer
124     exe.write('SLP_'+str(i)+'_m = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_'+str(i)+', dirichlet_space_m, dirichlet_space_m, nm*k)\n') #Operadores matriz-hilos single layer
125
126     exe.write('DLP_m_'+str(i)+' = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_m, dirichlet_space_'+str(i)+', dirichlet_space_'+str(i)+', nm*k)\n') #Operadores matriz-hilos double layer
127     exe.write('DLP_'+str(i)+'_m = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_'+str(i)+', dirichlet_space_m, dirichlet_space_m, nm*k)\n') #Operadores matriz-hilos double layer
128
129     exe.write('ADLP_m_'+str(i)+' = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_m, neumann_space_'+str(i)+', neumann_space_'+str(i)+', nm*k)\n') #Operadores matriz-hilos adjoint double layer
130     exe.write('ADLP_'+str(i)+'_m = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_'+str(i)+', neumann_space_m, neumann_space_m, nm*k)\n') #Operadores matriz-hilos adjoint double layer
131
132     exe.write('HYP_m_'+str(i)+' = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_m, neumann_space_'+str(i)+', neumann_space_'+str(i)+', nm*k)\n') #Operadores matriz-hilos hypersingular
133     exe.write('HYP_'+str(i)+'_m = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_'+str(i)+', neumann_space_m, neumann_space_m, nm*k)\n') #Operadores matriz-hilos hypersingular
134
135     for j in range(N_hilos): #Interaccion entre hilos
136         if i!=j:
137             exe.write('SLP_'+str(i)+'_'+str(j)+' = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_'+str(i)+', dirichlet_space_'+str(j)+', dirichlet_space_'+str(j)+', nm*k)\n') #Single-layer interaccion entre hilos
138             exe.write('DLP_'+str(i)+'_'+str(j)+' = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_'+str(i)+', dirichlet_space_'+str(j)+', dirichlet_space_'+str(j)+', nm*k)\n') #Double-layer interaccion entre hilos
139             exe.write('ADLP_'+str(i)+'_'+str(j)+' = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_'+str(i)+', neumann_space_'+str(j)+', neumann_space_'+str(j)+', nm*k)\n') #Adjoint interaccion entre hilos
140             exe.write('HYP_'+str(i)+'_'+str(j)+' = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_'+str(i)+', neumann_space_'+str(j)+', neumann_space_'+str(j)+', nm*k)\n') #Hypersingular interaccion entre hilos
141
142
143 ###### ENSAMBLANDO MATRIZ DE OPERADORES ########################
144 exe.write('\n#Matriz de operadores\n')
145 exe.write('blocked = bempp.api.BlockedOperator('+str(2*(N_hilos+1))+','+str(2*(N_hilos+1))+')\n') #Tamano bloque de operadores
```

```
146
147 exe.write('\n#Diagonal\n')
148 exe.write('blocked[0,0] = op_m[0,0]\n') #Diagonal matriz
149 exe.write('blocked[0,1] = op_m[0,1]\n') #Diagonal matriz
150 exe.write('blocked[1,0] = op_m[1,0]\n') #Diagonal matriz
151 exe.write('blocked[1,1] = op_m[1,1]\n') #Diagonal matriz
152
153 c=0
154 for i in range(2, 2*(N_hilos+1)-1, 2): #Diagonal hilos
155     exe.write('blocked['+str(i)+','+str(i)+'] = op_'+str(c)+'[0,0]\n')
156     exe.write('blocked['+str(i)+','+str(i+1)+'] = op_'+str(c)+'[0,1]\n')
157     exe.write('blocked['+str(i+1)+','+str(i)+'] = op_'+str(c)+'[1,0]\n')
158     exe.write('blocked['+str(i+1)+','+str(i+1)+'] = op_'+str(c)+'[1,1]\n')
159     c+=1
160
161 exe.write('\n#Contribucion hilos-matriz\n')
162 c=0
163 for i in range(2, 2*N_hilos+1, 2): #Contribucion hilos en matriz
164     exe.write('blocked[0,'+str(i)+'] = DLP_'+str(c)+'_m\n') #Double-layer
    hilos-matriz
165     exe.write('blocked[0,'+str(i+1)+'] = -SLP_'+str(c)+'_m\n') #Single-
    layer hilos-matriz
166     exe.write('blocked[1,'+str(i)+'] = -HYP_'+str(c)+'_m\n') #
    Hypersingular hilos-matriz
167     exe.write('blocked[1,'+str(i+1)+'] = -ADLP_'+str(c)+'_m\n') #Adjoint
    hilos-matriz
168     c+=1
169
170 c1=0
171 for i in range(2, 2*(N_hilos+1)-1, 2): #Contribucion hilos-hilos
172     c2=0
173     for j in range(2, 2*(N_hilos+1)-1, 2):
174         exe.write('\n#Contribucion hilos-hilos\n')
175         if i<j:
176             exe.write('blocked['+str(i)+','+str(j)+'] = DLP_'+str(c2+1)+'_
    '+str(c1)+'\n') #Double-layer hilo-hilo
177             exe.write('blocked['+str(i)+','+str(j+1)+'] = -SLP_'+str(c2+1)
    +'_'+str(c1)+'\n') #Single-layer hilo-hilo
178             exe.write('blocked['+str(i+1)+','+str(j)+'] = -HYP_'+str(c2+1)
    +'_'+str(c1)+'\n') #Hypersingular hilo-hilo
179             exe.write('blocked['+str(i+1)+','+str(j+1)+'] = -ADLP_'+str(c2
    +1)+'_'+str(c1)+'\n') #Adjoint hilo-hilo
180             c2+=1
181         elif i>j:
182             exe.write('blocked['+str(i)+','+str(j)+'] = DLP_'+str(c2)+'_'+
    str(c1)+'\n') #Double-layer hilo-hilo
183             exe.write('blocked['+str(i)+','+str(j+1)+'] = -SLP_'+str(c2)+'
    _'+str(c1)+'\n') #Single-layer hilo-hilo
184             exe.write('blocked['+str(i+1)+','+str(j)+'] = -HYP_'+str(c2)+'
    _'+str(c1)+'\n') #Hypersingular hilo-hilo
185             exe.write('blocked['+str(i+1)+','+str(j+1)+'] = -ADLP_'+str(c2
    )+'_'+str(c1)+'\n') #Adjoint hilo-hilo
186             c2+=1
```

```
187
188     exe.write('\n#Contribucion matriz-hilos\n') #Contribucion matriz en hilos
189     exe.write('blocked['+str(i)+',0] = -DLP_m_'+str(c1)+'\n') #Double-layer matriz-hilos
190     exe.write('blocked['+str(i)+',1] = SLP_m_'+str(c1)+'\n') #Single-layer matriz-hilos
191     exe.write('blocked['+str(i+1)+',0] = HYP_m_'+str(c1)+'\n') #Hypersingular matriz-hilos
192     exe.write('blocked['+str(i+1)+',1] = ADLP_m_'+str(c1)+'\n') #Adjoint matriz-hilos
193     c1+=1
194
195 ###### CONDICIONES DE BORDE ###################################
196 exe.write('\n#Condiciones de borde\n') #Condiciones en el borde de la matriz
197 exe.write('dirichlet_grid_fun_m = bempp.api.GridFunction(dirichlet_space_m, fun=dirichlet_fun)\n')
198 exe.write('neumann_grid_fun_m = bempp.api.GridFunction(neumann_space_m, fun=neumann_fun)\n')
199
200 ###### DISCRETIZACION ##########################################
201 exe.write('\n#Discretizacion lado izquierdo\n') #Lado izquierdo
202 exe.write('blocked_discretizado = blocked.strong_form()\n')
203 #exe.write('blocked_discr_2 = blocked_discretizado*blocked_discretizado\n')
204
205 exe.write('\n#Discretizacion lado derecho\n') #Lado derecho con onda incidente
206 exe.write('rhs = np.concatenate([')
207 exe.write('dirichlet_grid_fun_m.coefficients, neumann_grid_fun_m.coefficients,')
208 for i in range(N_hilos):
209     exe.write('np.zeros(dirichlet_space_'+str(i)+'.global_dof_count), np.zeros(neumann_space_'+str(i)+'.global_dof_count)')
210     if i!=N_hilos-1:
211         exe.write(', ')
212 exe.write('])\n')
213 #exe.write('rhs_2 = blocked_discr_2*rhs\n')
214
215
216 ###### SISTEMA DE ECUACIONES ###################################
217 exe.write('\n#Sistema de ecuaciones\n')
218 exe.write('import inspect\n')
219 exe.write('from scipy.sparse.linalg import gmres\n')
220
221 exe.write('it_count = 0\n') #numero de iteraciones
222 exe.write('def iteration_counter(x):\n') #Contador
223 exe.write('\tglobal it_count\n')
224 #exe.write('\tprint it_count\n') #Mostrar contador en cada iteracion
225 exe.write('\tit_count += 1\n')
226 exe.write('\tframe = inspect.currentframe().f_back\n')
227 exe.write('\tprint it_count, frame.f_locals["resid"]\n')
```

```
228
229 exe.write('print("Shape of matrix: {0}".format(blocked_discretizado.shape))\n') #Tamano de la matriz
230 #exe.write('print("Shape of matrix: {7}".format(blocked_discr_2.shape))\n') #Tamano de la matriz
231 exe.write('x,info = gmres(blocked_discretizado, rhs, tol=1e-5, callback = iteration_counter, maxiter = 150000)\n') #GMRES para resolver el sistema lineal
232 #exe.write('x,info = gmres(blocked_discr_2, rhs_2, callback= iteration_counter)\n') #GMRES para resolver el sistema lineal
233 exe.write('print("El sistema fue resuelto en {0} iteraciones".format(it_count))\n') #Numero de iteraciones
234
235 exe.write('np.savetxt("Solucion.out", x, delimiter=",")\n') #Guardando solucion en archivo txt
236
237 ###### SEPARACION DE LA SOLUCION ################################
238 exe.write('\n#Campo interior\n') #Separar la solucion del sistema solo para la matriz
239 exe.write('interior_field_dirichlet_m = bempp.api.GridFunction(dirichlet_space_m, coefficients=x[:dirichlet_space_m.global_dof_count])\n')
240 exe.write('interior_field_neumann_m = bempp.api.GridFunction(neumann_space_m,coefficients=x[dirichlet_space_m.global_dof_count:dirichlet_sp    ace_m.global_dof_count + neumann_space_m.global_dof_count])\n')
241
242 ###### CAMPO EXTERIOR ###########################################
243 exe.write('\n#Campo exterior\n') #Aplicar condiciones de transmision para obtener campo exterior
244 exe.write('exterior_field_dirichlet_m = interior_field_dirichlet_m\n')
245 exe.write('exterior_field_neumann_m = interior_field_neumann_m*(1./alfa_m)\n')
246
247 ###### CALCULO DEL CAMPO EN ANTENA ##############################
248 exe.write('\n#Calculo campo en antena\n')
249
250 ###### CAMPO EXTERIOR A LA MATRIZ ###############################
251 exe.write('slp_pot_ext_m = bempp.api.operators.potential.helmholtz.single_layer(dirichlet_space_m, antena, k)\n') #Single-layer exterior
252 exe.write('dlp_pot_ext_m = bempp.api.operators.potential.helmholtz.double_layer(dirichlet_space_m, antena, k)\n') #Double-layer exterior
253
254 exe.write('Campo_en_antena = (dlp_pot_ext_m * exterior_field_dirichlet_m - slp_pot_ext_m * exterior_field_neumann_m).ravel() + np.exp(1j*k*    antena[0])\n') #Calculo del campo en la antena
255
256 exe.write('print "Valor del campo en receptor:", Campo_en_antena\n') #Imprimir resultados
257
258 ###### CERRAMOS ARCHIVO .PY CREADO ##############################
259 exe.close
260
261 ################################################################
```

Apéndice B: Código final para 6 hilos en un compósito

```
1
2
3  import numpy as np
4  import bempp.api
5  omega = 2.*np.pi*12e9
6  e0 = 8.854*1e-12*1e-18
7  mu0 = 4.*np.pi*1e-7*1e6
8  mue = (1.)*mu0
9  ee = (16.)*e0
10 mui = (-1.734+0.37j)*mu0
11 ei = (-87641.359-104633.972j)*e0
12 k = omega*np.sqrt(e0*mu0)
13 lam = 2*np.pi/k
14 nm = np.sqrt((ee*mue)/(e0*mu0))
15 nc = np.sqrt((ei*mui)/(e0*mu0))
16 alfa_m = mue/mu0
17 alfa_c = mui/mue
18 antena = np.array([[1e4],[0.],[0.]])
19 print "Numero de onda exterior:", k
20 print "Indice de refraccion matriz:", nm
21 print "Indice de refraccion conductor:", nc
22 print "Numero de onda interior matriz:", nm*k
23 print "Numero de onda interior conductor:", nm*nc*k
24 print "Indice de transmision matriz:", alfa_m
25 print "Indice de transmision conductor:", alfa_c
26 print "Longitud de onda:", lam, "micras"
27
28 #Importando mallas
29 matriz = bempp.api.import_grid('/home/milan/matriz_12x12x300_E16772.msh')
30 grid_0 = bempp.api.import_grid('/home/milan/PH1_a5_l10_E5550_D2.msh')
31 grid_1 = bempp.api.import_grid('/home/milan/PH2_a5_l10_E5550_D2.msh')
32 grid_2 = bempp.api.import_grid('/home/milan/PH3_a5_l10_E5550_D2.msh')
33 grid_3 = bempp.api.import_grid('/home/milan/PH4_a5_l10_E5550_D2.msh')
34 grid_4 = bempp.api.import_grid('/home/milan/PH5_a5_l10_E5550_D2.msh')
35 grid_5 = bempp.api.import_grid('/home/milan/PH6_a5_l10_E5550_D2.msh')
36
37 #Funciones de dirichlet y neumann
38 def dirichlet_fun(x, n, domain_index, result):
39     result[0] = 1.*np.exp(1j*k*x[0])
40 def neumann_fun(x, n, domain_index, result):
41     result[0] = 1.*1j*k*n[0]*np.exp(1j*k*x[0])
42
43 #Operadores identidad
44
45 #Operadores multitrazo
46 Ai_m = bempp.api.operators.boundary.helmholtz.multitrace_operator(matriz,
   nm*k)
```

```
45 Ae_m = bempp.api.operators.boundary.helmholtz.multitrace_operator(matriz, k)
46
47 Ae_0 = bempp.api.operators.boundary.helmholtz.multitrace_operator(grid_0,nm*k)
48 Ai_1 = bempp.api.operators.boundary.helmholtz.multitrace_operator(grid_1,nm*nc*k)
49 Ae_1 = bempp.api.operators.boundary.helmholtz.multitrace_operator(grid_1,nm*k)
50 Ai_2 = bempp.api.operators.boundary.helmholtz.multitrace_operator(grid_2,nm*nc*k)
51 Ae_2 = bempp.api.operators.boundary.helmholtz.multitrace_operator(grid_2,nm*k)
52 Ai_3 = bempp.api.operators.boundary.helmholtz.multitrace_operator(grid_3,nm*nc*k)
53 Ae_3 = bempp.api.operators.boundary.helmholtz.multitrace_operator(grid_3,nm*k)
54 Ai_4 = bempp.api.operators.boundary.helmholtz.multitrace_operator(grid_4,nm*nc*k)
55 Ae_4 = bempp.api.operators.boundary.helmholtz.multitrace_operator(grid_4,nm*k)
56 Ai_5 = bempp.api.operators.boundary.helmholtz.multitrace_operator(grid_5,nm*nc*k)
57 Ae_5 = bempp.api.operators.boundary.helmholtz.multitrace_operator(grid_5,nm*k)
58
59 #Transmision en Multitrazo
60 Ae_m[0,1] = Ae_m[0,1]*(1./alfa_m)
61 Ae_m[1,1] = Ae_m[1,1]*(1./alfa_m)
62 Ai_0[0,1] = Ai_0[0,1]*alfa_c
63 Ai_0[1,1] = Ai_0[1,1]*alfa_c
64 Ai_1[0,1] = Ai_1[0,1]*alfa_c
65 Ai_1[1,1] = Ai_1[1,1]*alfa_c
66 Ai_2[0,1] = Ai_2[0,1]*alfa_c
67 Ai_2[1,1] = Ai_2[1,1]*alfa_c
68 Ai_3[0,1] = Ai_3[0,1]*alfa_c
69 Ai_3[1,1] = Ai_3[1,1]*alfa_c
70 Ai_4[0,1] = Ai_4[0,1]*alfa_c
71 Ai_4[1,1] = Ai_4[1,1]*alfa_c
72 Ai_5[0,1] = Ai_5[0,1]*alfa_c
73 Ai_5[1,1] = Ai_5[1,1]*alfa_c
74
75 #Acople interior y exterior
76 op_m = (Ai_m + Ae_m)
77 op_0 = (Ai_0 + Ae_0)
78 op_1 = (Ai_1 + Ae_1)
79 op_2 = (Ai_2 + Ae_2)
80 op_3 = (Ai_3 + Ae_3)
81 op_4 = (Ai_4 + Ae_4)
82 op_5 = (Ai_5 + Ae_5)
83
84 #Espacios
85 dirichlet_space_m = Ai_m[0,0].domain
```

```
86 neumann_space_m = Ai_m[0,1].domain
87 dirichlet_space_0 = Ai_0[0,0].domain
88 neumann_space_0 = Ai_0[0,1].domain
89 dirichlet_space_1 = Ai_1[0,0].domain
90 neumann_space_1 = Ai_1[0,1].domain
91 dirichlet_space_2 = Ai_2[0,0].domain
92 neumann_space_2 = Ai_2[0,1].domain
93 dirichlet_space_3 = Ai_3[0,0].domain
94 neumann_space_3 = Ai_3[0,1].domain
95 dirichlet_space_4 = Ai_4[0,0].domain
96 neumann_space_4 = Ai_4[0,1].domain
97 dirichlet_space_5 = Ai_5[0,0].domain
98 neumann_space_5 = Ai_5[0,1].domain
99 ident_m = bempp.api.operators.boundary.sparse.identity(neumann_space_m,
    neumann_space_m, neumann_space_m)
100 ident_0 = bempp.api.operators.boundary.sparse.identity(neumann_space_0,
    neumann_space_0, neumann_space_0)
101 ident_1 = bempp.api.operators.boundary.sparse.identity(neumann_space_1,
    neumann_space_1, neumann_space_1)
102 ident_2 = bempp.api.operators.boundary.sparse.identity(neumann_space_2,
    neumann_space_2, neumann_space_2)
103 ident_3 = bempp.api.operators.boundary.sparse.identity(neumann_space_3,
    neumann_space_3, neumann_space_3)
104 ident_4 = bempp.api.operators.boundary.sparse.identity(neumann_space_4,
    neumann_space_4, neumann_space_4)
105 ident_5 = bempp.api.operators.boundary.sparse.identity(neumann_space_5,
    neumann_space_5, neumann_space_5)
106 op_m[1,1] = op_m[1,1] + 0.5 * ident_m * ((alfa_m -1)/alfa_m)
107 op_0[1,1] = op_0[1,1] + 0.5 * ident_0* (alfa_c - 1)
108 op_1[1,1] = op_1[1,1] + 0.5 * ident_1* (alfa_c - 1)
109 op_2[1,1] = op_2[1,1] + 0.5 * ident_2* (alfa_c - 1)
110 op_3[1,1] = op_3[1,1] + 0.5 * ident_3* (alfa_c - 1)
111 op_4[1,1] = op_4[1,1] + 0.5 * ident_4* (alfa_c - 1)
112 op_5[1,1] = op_5[1,1] + 0.5 * ident_5* (alfa_c - 1)
113
114 #Operadores entre mallas
115 SLP_m_0 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_m, dirichlet_space_0, dirichlet_space_0, nm*k)
116 SLP_0_m = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_0, dirichlet_space_m, dirichlet_space_m, nm*k)
117 DLP_m_0 = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_m, dirichlet_space_0, dirichlet_space_0, nm*k)
118 DLP_0_m = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_0, dirichlet_space_m, dirichlet_space_m, nm*k)
119 ADLP_m_0 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_m, neumann_space_0, neumann_space_0, nm*k)
120 ADLP_0_m = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_0, neumann_space_m, neumann_space_m, nm*k)
121 HYP_m_0 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_m, neumann_space_0, neumann_space_0, nm*k)
122 HYP_0_m = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_0, neumann_space_m, neumann_space_m, nm*k)
```

```
SLP_0_1 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_0, dirichlet_space_1, dirichlet_space_1, nm*k)
DLP_0_1 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_0, dirichlet_space_1, dirichlet_space_1, nm*k)
ADLP_0_1 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_0, neumann_space_1, neumann_space_1, nm*k)
HYP_0_1 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_0, neumann_space_1, neumann_space_1, nm*k)
SLP_0_2 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_0, dirichlet_space_2, dirichlet_space_2, nm*k)
DLP_0_2 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_0, dirichlet_space_2, dirichlet_space_2, nm*k)
ADLP_0_2 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_0, neumann_space_2, neumann_space_2, nm*k)
HYP_0_2 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_0, neumann_space_2, neumann_space_2, nm*k)
SLP_0_3 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_0, dirichlet_space_3, dirichlet_space_3, nm*k)
DLP_0_3 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_0, dirichlet_space_3, dirichlet_space_3, nm*k)
ADLP_0_3 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_0, neumann_space_3, neumann_space_3, nm*k)
HYP_0_3 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_0, neumann_space_3, neumann_space_3, nm*k)
SLP_0_4 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_0, dirichlet_space_4, dirichlet_space_4, nm*k)
DLP_0_4 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_0, dirichlet_space_4, dirichlet_space_4, nm*k)
ADLP_0_4 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_0, neumann_space_4, neumann_space_4, nm*k)
HYP_0_4 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_0, neumann_space_4, neumann_space_4, nm*k)
SLP_0_5 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_0, dirichlet_space_5, dirichlet_space_5, nm*k)
DLP_0_5 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_0, dirichlet_space_5, dirichlet_space_5, nm*k)
ADLP_0_5 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_0, neumann_space_5, neumann_space_5, nm*k)
HYP_0_5 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_0, neumann_space_5, neumann_space_5, nm*k)
SLP_m_1 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_m, dirichlet_space_1, dirichlet_space_1, nm*k)
SLP_1_m = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_1, dirichlet_space_m, dirichlet_space_m, nm*k)
DLP_m_1 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_m, dirichlet_space_1, dirichlet_space_1, nm*k)
DLP_1_m = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_1, dirichlet_space_m, dirichlet_space_m, nm*k)
ADLP_m_1 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_m, neumann_space_1, neumann_space_1, nm*k)
ADLP_1_m = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_1, neumann_space_m, neumann_space_m, nm*k)
```

```
149 HYP_m_1 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_m, neumann_space_1, neumann_space_1, nm*k)
150 HYP_1_m = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_1, neumann_space_m, neumann_space_m, nm*k)
151 SLP_1_0 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_1, dirichlet_space_0, dirichlet_space_0, nm*k)
152 DLP_1_0 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_1, dirichlet_space_0, dirichlet_space_0, nm*k)
153 ADLP_1_0 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_1, neumann_space_0, neumann_space_0, nm*k)
154 HYP_1_0 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_1, neumann_space_0, neumann_space_0, nm*k)
155 SLP_1_2 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_1, dirichlet_space_2, dirichlet_space_2, nm*k)
156 DLP_1_2 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_1, dirichlet_space_2, dirichlet_space_2, nm*k)
157 ADLP_1_2 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_1, neumann_space_2, neumann_space_2, nm*k)
158 HYP_1_2 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_1, neumann_space_2, neumann_space_2, nm*k)
159 SLP_1_3 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_1, dirichlet_space_3, dirichlet_space_3, nm*k)
160 DLP_1_3 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_1, dirichlet_space_3, dirichlet_space_3, nm*k)
161 ADLP_1_3 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_1, neumann_space_3, neumann_space_3, nm*k)
162 HYP_1_3 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_1, neumann_space_3, neumann_space_3, nm*k)
163 SLP_1_4 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_1, dirichlet_space_4, dirichlet_space_4, nm*k)
164 DLP_1_4 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_1, dirichlet_space_4, dirichlet_space_4, nm*k)
165 ADLP_1_4 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_1, neumann_space_4, neumann_space_4, nm*k)
166 HYP_1_4 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_1, neumann_space_4, neumann_space_4, nm*k)
167 SLP_1_5 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_1, dirichlet_space_5, dirichlet_space_5, nm*k)
168 DLP_1_5 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_1, dirichlet_space_5, dirichlet_space_5, nm*k)
169 ADLP_1_5 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_1, neumann_space_5, neumann_space_5, nm*k)
170 HYP_1_5 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_1, neumann_space_5, neumann_space_5, nm*k)
171 SLP_m_2 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_m, dirichlet_space_2, dirichlet_space_2, nm*k)
172 SLP_2_m = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_2, dirichlet_space_m, dirichlet_space_m, nm*k)
173 DLP_m_2 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_m, dirichlet_space_2, dirichlet_space_2, nm*k)
174 DLP_2_m = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_2, dirichlet_space_m, dirichlet_space_m, nm*k)
```

```
175 ADLP_m_2 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_m, neumann_space_2, neumann_space_2, nm*k)
176 ADLP_2_m = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_2, neumann_space_m, neumann_space_m, nm*k)
177 HYP_m_2 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_m, neumann_space_2, neumann_space_2, nm*k)
178 HYP_2_m = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_2, neumann_space_m, neumann_space_m, nm*k)
179 SLP_2_0 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_2, dirichlet_space_0, dirichlet_space_0, nm*k)
180 DLP_2_0 = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_2, dirichlet_space_0, dirichlet_space_0, nm*k)
181 ADLP_2_0 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_2, neumann_space_0, neumann_space_0, nm*k)
182 HYP_2_0 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_2, neumann_space_0, neumann_space_0, nm*k)
183 SLP_2_1 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_2, dirichlet_space_1, dirichlet_space_1, nm*k)
184 DLP_2_1 = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_2, dirichlet_space_1, dirichlet_space_1, nm*k)
185 ADLP_2_1 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_2, neumann_space_1, neumann_space_1, nm*k)
186 HYP_2_1 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_2, neumann_space_1, neumann_space_1, nm*k)
187 SLP_2_3 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_2, dirichlet_space_3, dirichlet_space_3, nm*k)
188 DLP_2_3 = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_2, dirichlet_space_3, dirichlet_space_3, nm*k)
189 ADLP_2_3 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_2, neumann_space_3, neumann_space_3, nm*k)
190 HYP_2_3 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_2, neumann_space_3, neumann_space_3, nm*k)
191 SLP_2_4 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_2, dirichlet_space_4, dirichlet_space_4, nm*k)
192 DLP_2_4 = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_2, dirichlet_space_4, dirichlet_space_4, nm*k)
193 ADLP_2_4 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_2, neumann_space_4, neumann_space_4, nm*k)
194 HYP_2_4 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_2, neumann_space_4, neumann_space_4, nm*k)
195 SLP_2_5 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_2, dirichlet_space_5, dirichlet_space_5, nm*k)
196 DLP_2_5 = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_2, dirichlet_space_5, dirichlet_space_5, nm*k)
197 ADLP_2_5 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_2, neumann_space_5, neumann_space_5, nm*k)
198 HYP_2_5 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_2, neumann_space_5, neumann_space_5, nm*k)
199 SLP_m_3 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_m, dirichlet_space_3, dirichlet_space_3, nm*k)
200 SLP_3_m = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_3, dirichlet_space_m, dirichlet_space_m, nm*k)
```

```
DLP_m_3 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_m, dirichlet_space_3, dirichlet_space_3, nm*k)
DLP_3_m = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_3, dirichlet_space_m, dirichlet_space_m, nm*k)
ADLP_m_3 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_m, neumann_space_3, neumann_space_3, nm*k)
ADLP_3_m = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_3, neumann_space_m, neumann_space_m, nm*k)
HYP_m_3 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_m, neumann_space_3, neumann_space_3, nm*k)
HYP_3_m = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_3, neumann_space_m, neumann_space_m, nm*k)
SLP_3_0 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_3, dirichlet_space_0, dirichlet_space_0, nm*k)
DLP_3_0 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_3, dirichlet_space_0, dirichlet_space_0, nm*k)
ADLP_3_0 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_3, neumann_space_0, neumann_space_0, nm*k)
HYP_3_0 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_3, neumann_space_0, neumann_space_0, nm*k)
SLP_3_1 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_3, dirichlet_space_1, dirichlet_space_1, nm*k)
DLP_3_1 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_3, dirichlet_space_1, dirichlet_space_1, nm*k)
ADLP_3_1 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_3, neumann_space_1, neumann_space_1, nm*k)
HYP_3_1 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_3, neumann_space_1, neumann_space_1, nm*k)
SLP_3_2 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_3, dirichlet_space_2, dirichlet_space_2, nm*k)
DLP_3_2 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_3, dirichlet_space_2, dirichlet_space_2, nm*k)
ADLP_3_2 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_3, neumann_space_2, neumann_space_2, nm*k)
HYP_3_2 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_3, neumann_space_2, neumann_space_2, nm*k)
SLP_3_4 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_3, dirichlet_space_4, dirichlet_space_4, nm*k)
DLP_3_4 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_3, dirichlet_space_4, dirichlet_space_4, nm*k)
ADLP_3_4 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_3, neumann_space_4, neumann_space_4, nm*k)
HYP_3_4 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_3, neumann_space_4, neumann_space_4, nm*k)
SLP_3_5 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_3, dirichlet_space_5, dirichlet_space_5, nm*k)
DLP_3_5 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_3, dirichlet_space_5, dirichlet_space_5, nm*k)
ADLP_3_5 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_3, neumann_space_5, neumann_space_5, nm*k)
HYP_3_5 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_3, neumann_space_5, neumann_space_5, nm*k)
```

```
227 SLP_m_4 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_m, dirichlet_space_4, dirichlet_space_4, nm*k)
228 SLP_4_m = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_4, dirichlet_space_m, dirichlet_space_m, nm*k)
229 DLP_m_4 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_m, dirichlet_space_4, dirichlet_space_4, nm*k)
230 DLP_4_m = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_4, dirichlet_space_m, dirichlet_space_m, nm*k)
231 ADLP_m_4 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_m, neumann_space_4, neumann_space_4, nm*k)
232 ADLP_4_m = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_4, neumann_space_m, neumann_space_m, nm*k)
233 HYP_m_4 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_m, neumann_space_4, neumann_space_4, nm*k)
234 HYP_4_m = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_4, neumann_space_m, neumann_space_m, nm*k)
235 SLP_4_0 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_4, dirichlet_space_0, dirichlet_space_0, nm*k)
236 DLP_4_0 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_4, dirichlet_space_0, dirichlet_space_0, nm*k)
237 ADLP_4_0 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_4, neumann_space_0, neumann_space_0, nm*k)
238 HYP_4_0 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_4, neumann_space_0, neumann_space_0, nm*k)
239 SLP_4_1 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_4, dirichlet_space_1, dirichlet_space_1, nm*k)
240 DLP_4_1 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_4, dirichlet_space_1, dirichlet_space_1, nm*k)
241 ADLP_4_1 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_4, neumann_space_1, neumann_space_1, nm*k)
242 HYP_4_1 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_4, neumann_space_1, neumann_space_1, nm*k)
243 SLP_4_2 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_4, dirichlet_space_2, dirichlet_space_2, nm*k)
244 DLP_4_2 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_4, dirichlet_space_2, dirichlet_space_2, nm*k)
245 ADLP_4_2 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_4, neumann_space_2, neumann_space_2, nm*k)
246 HYP_4_2 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_4, neumann_space_2, neumann_space_2, nm*k)
247 SLP_4_3 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_4, dirichlet_space_3, dirichlet_space_3, nm*k)
248 DLP_4_3 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_4, dirichlet_space_3, dirichlet_space_3, nm*k)
249 ADLP_4_3 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_4, neumann_space_3, neumann_space_3, nm*k)
250 HYP_4_3 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_4, neumann_space_3, neumann_space_3, nm*k)
251 SLP_4_5 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_4, dirichlet_space_5, dirichlet_space_5, nm*k)
252 DLP_4_5 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_4, dirichlet_space_5, dirichlet_space_5, nm*k)
```

```
253 ADLP_4_5 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_4, neumann_space_5, neumann_space_5, nm*k)
254 HYP_4_5 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_4, neumann_space_5, neumann_space_5, nm*k)
255 SLP_m_5 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_m, dirichlet_space_5, dirichlet_space_5, nm*k)
256 SLP_5_m = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_5, dirichlet_space_m, dirichlet_space_m, nm*k)
257 DLP_m_5 = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_m, dirichlet_space_5, dirichlet_space_5, nm*k)
258 DLP_5_m = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_5, dirichlet_space_m, dirichlet_space_m, nm*k)
259 ADLP_m_5 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_m, neumann_space_5, neumann_space_5, nm*k)
260 ADLP_5_m = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_5, neumann_space_m, neumann_space_m, nm*k)
261 HYP_m_5 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_m, neumann_space_5, neumann_space_5, nm*k)
262 HYP_5_m = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_5, neumann_space_m, neumann_space_m, nm*k)
263 SLP_5_0 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_5, dirichlet_space_0, dirichlet_space_0, nm*k)
264 DLP_5_0 = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_5, dirichlet_space_0, dirichlet_space_0, nm*k)
265 ADLP_5_0 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_5, neumann_space_0, neumann_space_0, nm*k)
266 HYP_5_0 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_5, neumann_space_0, neumann_space_0, nm*k)
267 SLP_5_1 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_5, dirichlet_space_1, dirichlet_space_1, nm*k)
268 DLP_5_1 = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_5, dirichlet_space_1, dirichlet_space_1, nm*k)
269 ADLP_5_1 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_5, neumann_space_1, neumann_space_1, nm*k)
270 HYP_5_1 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_5, neumann_space_1, neumann_space_1, nm*k)
271 SLP_5_2 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_5, dirichlet_space_2, dirichlet_space_2, nm*k)
272 DLP_5_2 = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_5, dirichlet_space_2, dirichlet_space_2, nm*k)
273 ADLP_5_2 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_5, neumann_space_2, neumann_space_2, nm*k)
274 HYP_5_2 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_5, neumann_space_2, neumann_space_2, nm*k)
275 SLP_5_3 = bempp.api.operators.boundary.helmholtz.single_layer(
    neumann_space_5, dirichlet_space_3, dirichlet_space_3, nm*k)
276 DLP_5_3 = bempp.api.operators.boundary.helmholtz.double_layer(
    dirichlet_space_5, dirichlet_space_3, dirichlet_space_3, nm*k)
277 ADLP_5_3 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
    neumann_space_5, neumann_space_3, neumann_space_3, nm*k)
278 HYP_5_3 = bempp.api.operators.boundary.helmholtz.hypersingular(
    dirichlet_space_5, neumann_space_3, neumann_space_3, nm*k)
```

```
279 SLP_5_4 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_5, dirichlet_space_4, dirichlet_space_4, nm*k)
280 DLP_5_4 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_5, dirichlet_space_4, dirichlet_space_4, nm*k)
281 ADLP_5_4 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_5, neumann_space_4, neumann_space_4, nm*k)
242 HYP_4_1 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_4, neumann_space_1, neumann_space_1, nm*k)
243 SLP_4_2 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_4, dirichlet_space_2, dirichlet_space_2, nm*k)
244 DLP_4_2 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_4, dirichlet_space_2, dirichlet_space_2, nm*k)
245 ADLP_4_2 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_4, neumann_space_2, neumann_space_2, nm*k)
246 HYP_4_2 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_4, neumann_space_2, neumann_space_2, nm*k)
247 SLP_4_3 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_4, dirichlet_space_3, dirichlet_space_3, nm*k)
248 DLP_4_3 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_4, dirichlet_space_3, dirichlet_space_3, nm*k)
249 ADLP_4_3 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_4, neumann_space_3, neumann_space_3, nm*k)
250 HYP_4_3 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_4, neumann_space_3, neumann_space_3, nm*k)
251 SLP_4_5 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_4, dirichlet_space_5, dirichlet_space_5, nm*k)
252 DLP_4_5 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_4, dirichlet_space_5, dirichlet_space_5, nm*k)
253 ADLP_4_5 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_4, neumann_space_5, neumann_space_5, nm*k)
254 HYP_4_5 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_4, neumann_space_5, neumann_space_5, nm*k)
255 SLP_m_5 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_m, dirichlet_space_5, dirichlet_space_5, nm*k)
256 SLP_5_m = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_5, dirichlet_space_m, dirichlet_space_m, nm*k)
257 DLP_m_5 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_m, dirichlet_space_5, dirichlet_space_5, nm*k)
258 DLP_5_m = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_5, dirichlet_space_m, dirichlet_space_m, nm*k)
259 ADLP_m_5 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_m, neumann_space_5, neumann_space_5, nm*k)
260 ADLP_5_m = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_5, neumann_space_m, neumann_space_m, nm*k)
261 HYP_m_5 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_m, neumann_space_5, neumann_space_5, nm*k)
262 HYP_5_m = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_5, neumann_space_m, neumann_space_m, nm*k)
263 SLP_5_0 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_5, dirichlet_space_0, dirichlet_space_0, nm*k)
264 DLP_5_0 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_5, dirichlet_space_0, dirichlet_space_0, nm*k)
```

```
265 ADLP_5_0 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_5, neumann_space_0, neumann_space_0, nm*k)
266 HYP_5_0 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_5, neumann_space_0, neumann_space_0, nm*k)
267 SLP_5_1 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_5, dirichlet_space_1, dirichlet_space_1, nm*k)
268 DLP_5_1 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_5, dirichlet_space_1, dirichlet_space_1, nm*k)
269 ADLP_5_1 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_5, neumann_space_1, neumann_space_1, nm*k)
270 HYP_5_1 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_5, neumann_space_1, neumann_space_1, nm*k)
271 SLP_5_2 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_5, dirichlet_space_2, dirichlet_space_2, nm*k)
272 DLP_5_2 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_5, dirichlet_space_2, dirichlet_space_2, nm*k)
273 ADLP_5_2 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_5, neumann_space_2, neumann_space_2, nm*k)
274 HYP_5_2 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_5, neumann_space_2, neumann_space_2, nm*k)
275 SLP_5_3 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_5, dirichlet_space_3, dirichlet_space_3, nm*k)
276 DLP_5_3 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_5, dirichlet_space_3, dirichlet_space_3, nm*k)
277 ADLP_5_3 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_5, neumann_space_3, neumann_space_3, nm*k)
278 HYP_5_3 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_5, neumann_space_3, neumann_space_3, nm*k)
279 SLP_5_4 = bempp.api.operators.boundary.helmholtz.single_layer(neumann_space_5, dirichlet_space_4, dirichlet_space_4, nm*k)
280 DLP_5_4 = bempp.api.operators.boundary.helmholtz.double_layer(dirichlet_space_5, dirichlet_space_4, dirichlet_space_4, nm*k)
281 ADLP_5_4 = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(neumann_space_5, neumann_space_4, neumann_space_4, nm*k)
282 HYP_5_4 = bempp.api.operators.boundary.helmholtz.hypersingular(dirichlet_space_5, neumann_space_4, neumann_space_4, nm*k)
283
284 #Matriz de operadores
285 blocked = bempp.api.BlockedOperator(14,14)
286
287 #Diagonal
288 blocked[0,0] = op_m[0,0]
289 blocked[0,1] = op_m[0,1]
290 blocked[1,0] = op_m[1,0]
291 blocked[1,1] = op_m[1,1]
292 blocked[2,2] = op_0[0,0]
293 blocked[2,3] = op_0[0,1]
294 blocked[3,2] = op_0[1,0]
295 blocked[3,3] = op_0[1,1]
296 blocked[4,4] = op_1[0,0]
297 blocked[4,5] = op_1[0,1]
298 blocked[5,4] = op_1[1,0]
299 blocked[5,5] = op_1[1,1]
```

```
300 blocked[6,6] = op_2[0,0]
301 blocked[6,7] = op_2[0,1]
302 blocked[7,6] = op_2[1,0]
303 blocked[7,7] = op_2[1,1]
304 blocked[8,8] = op_3[0,0]
305 blocked[8,9] = op_3[0,1]
306 blocked[9,8] = op_3[1,0]
307 blocked[9,9] = op_3[1,1]
308 blocked[10,10] = op_4[0,0]
309 blocked[10,11] = op_4[0,1]
310 blocked[11,10] = op_4[1,0]
311 blocked[11,11] = op_4[1,1]
312 blocked[12,12] = op_5[0,0]
313 blocked[12,13] = op_5[0,1]
314 blocked[13,12] = op_5[1,0]
315 blocked[13,13] = op_5[1,1]
316
317 #Contribucion hilos-matriz
318 blocked[0,2] = DLP_0_m
319 blocked[0,3] = -SLP_0_m
320 blocked[1,2] = -HYP_0_m
321 blocked[1,3] = -ADLP_0_m
322 blocked[0,4] = DLP_1_m
323 blocked[0,5] = -SLP_1_m
324 blocked[1,4] = -HYP_1_m
325 blocked[1,5] = -ADLP_1_m
326 blocked[0,6] = DLP_2_m
327 blocked[0,7] = -SLP_2_m
328 blocked[1,6] = -HYP_2_m
329 blocked[1,7] = -ADLP_2_m
330 blocked[0,8] = DLP_3_m
331 blocked[0,9] = -SLP_3_m
332 blocked[1,8] = -HYP_3_m
333 blocked[1,9] = -ADLP_3_m
334 blocked[0,10] = DLP_4_m
335 blocked[0,11] = -SLP_4_m
336 blocked[1,10] = -HYP_4_m
337 blocked[1,11] = -ADLP_4_m
338 blocked[0,12] = DLP_5_m
339 blocked[0,13] = -SLP_5_m
340 blocked[1,12] = -HYP_5_m
341 blocked[1,13] = -ADLP_5_m
342
343 #Contribucion hilos-hilos
344
345 #Contribucion hilos-hilos
346 blocked[2,4] = DLP_1_0
347 blocked[2,5] = -SLP_1_0
348 blocked[3,4] = -HYP_1_0
349 blocked[3,5] = -ADLP_1_0
350
351 #Contribucion hilos-hilos
352 blocked[2,6] = DLP_2_0
```

```
353 blocked[2,7] = -SLP_2_0
354 blocked[3,6] = -HYP_2_0
355 blocked[3,7] = -ADLP_2_0
356
357 #Contribucion hilos-hilos
358 blocked[2,8] = DLP_3_0
359 blocked[2,9] = -SLP_3_0
360 blocked[3,8] = -HYP_3_0
361 blocked[3,9] = -ADLP_3_0
362
363 #Contribucion hilos-hilos
364 blocked[2,10] = DLP_4_0
365 blocked[2,11] = -SLP_4_0
366 blocked[3,10] = -HYP_4_0
367 blocked[3,11] = -ADLP_4_0
368
369 #Contribucion hilos-hilos
370 blocked[2,12] = DLP_5_0
371 blocked[2,13] = -SLP_5_0
372 blocked[3,12] = -HYP_5_0
373 blocked[3,13] = -ADLP_5_0
374
375 #Contribucion matriz-hilos
376 blocked[2,0] = -DLP_m_0
377 blocked[2,1] = SLP_m_0
378 blocked[3,0] = HYP_m_0
379 blocked[3,1] = ADLP_m_0
380
381 #Contribucion hilos-hilos
382 blocked[4,2] = DLP_0_1
383 blocked[4,3] = -SLP_0_1
384 blocked[5,2] = -HYP_0_1
385 blocked[5,3] = -ADLP_0_1
386
387 #Contribucion hilos-hilos
388
389 #Contribucion hilos-hilos
390 blocked[4,6] = DLP_2_1
391 blocked[4,7] = -SLP_2_1
392 blocked[5,6] = -HYP_2_1
393 blocked[5,7] = -ADLP_2_1
394
395 #Contribucion hilos-hilos
396 blocked[4,8] = DLP_3_1
397 blocked[4,9] = -SLP_3_1
398 blocked[5,8] = -HYP_3_1
399 blocked[5,9] = -ADLP_3_1
400
401 #Contribucion hilos-hilos
402 blocked[4,10] = DLP_4_1
403 blocked[4,11] = -SLP_4_1
404 blocked[5,10] = -HYP_4_1
405 blocked[5,11] = -ADLP_4_1
```

```
406
407 #Contribucion hilos-hilos
408 blocked[4,12] = DLP_5_1
409 blocked[4,13] = -SLP_5_1
410 blocked[5,12] = -HYP_5_1
411 blocked[5,13] = -ADLP_5_1
412
413 #Contribucion matriz-hilos
414 blocked[4,0] = -DLP_m_1
415 blocked[4,1] = SLP_m_1
416 blocked[5,0] = HYP_m_1
417 blocked[5,1] = ADLP_m_1
418
419 #Contribucion hilos-hilos
420 blocked[6,2] = DLP_0_2
421 blocked[6,3] = -SLP_0_2
422 blocked[7,2] = -HYP_0_2
423 blocked[7,3] = -ADLP_0_2
424
425 #Contribucion hilos-hilos
426 blocked[6,4] = DLP_1_2
427 blocked[6,5] = -SLP_1_2
428 blocked[7,4] = -HYP_1_2
429 blocked[7,5] = -ADLP_1_2
430
431 #Contribucion hilos-hilos
432
433 #Contribucion hilos-hilos
434 blocked[6,8] = DLP_3_2
435 blocked[6,9] = -SLP_3_2
436 blocked[7,8] = -HYP_3_2
437 blocked[7,9] = -ADLP_3_2
438
439 #Contribucion hilos-hilos
440 blocked[6,10] = DLP_4_2
441 blocked[6,11] = -SLP_4_2
442 blocked[7,10] = -HYP_4_2
443 blocked[7,11] = -ADLP_4_2
444
445 #Contribucion hilos-hilos
446 blocked[6,12] = DLP_5_2
447 blocked[6,13] = -SLP_5_2
448 blocked[7,12] = -HYP_5_2
449 blocked[7,13] = -ADLP_5_2
450
451 #Contribucion matriz-hilos
452 blocked[6,0] = -DLP_m_2
453 blocked[6,1] = SLP_m_2
454 blocked[7,0] = HYP_m_2
455 blocked[7,1] = ADLP_m_2
456
457 #Contribucion hilos-hilos
458 blocked[8,2] = DLP_0_3
```

```
459 blocked[8,3] = -SLP_0_3
460 blocked[9,2] = -HYP_0_3
461 blocked[9,3] = -ADLP_0_3
462
463 #Contribucion hilos-hilos
464 blocked[8,4] = DLP_1_3
465 blocked[8,5] = -SLP_1_3
466 blocked[9,4] = -HYP_1_3
467 blocked[9,5] = -ADLP_1_3
468
469 #Contribucion hilos-hilos
470 blocked[8,6] = DLP_2_3
471 blocked[8,7] = -SLP_2_3
472 blocked[9,6] = -HYP_2_3
473 blocked[9,7] = -ADLP_2_3
474
475 #Contribucion hilos-hilos
476
477 #Contribucion hilos-hilos
478 blocked[8,10] = DLP_4_3
479 blocked[8,11] = -SLP_4_3
480 blocked[9,10] = -HYP_4_3
481 blocked[9,11] = -ADLP_4_3
482
483 #Contribucion hilos-hilos
484 blocked[8,12] = DLP_5_3
485 blocked[8,13] = -SLP_5_3
486 blocked[9,12] = -HYP_5_3
487 blocked[9,13] = -ADLP_5_3
488
489 #Contribucion matriz-hilos
490 blocked[8,0] = -DLP_m_3
491 blocked[8,1] = SLP_m_3
492 blocked[9,0] = HYP_m_3
493 blocked[9,1] = ADLP_m_3
494
495 #Contribucion hilos-hilos
496 blocked[10,2] = DLP_0_4
497 blocked[10,3] = -SLP_0_4
498 blocked[11,2] = -HYP_0_4
499 blocked[11,3] = -ADLP_0_4
500
501 #Contribucion hilos-hilos
502 blocked[10,4] = DLP_1_4
503 blocked[10,5] = -SLP_1_4
504 blocked[11,4] = -HYP_1_4
505 blocked[11,5] = -ADLP_1_4
506
507 #Contribucion hilos-hilos
508 blocked[10,6] = DLP_2_4
509 blocked[10,7] = -SLP_2_4
510 blocked[11,6] = -HYP_2_4
511 blocked[11,7] = -ADLP_2_4
```

```
512
513 #Contribucion hilos-hilos
514 blocked[10,8] = DLP_3_4
515 blocked[10,9] = -SLP_3_4
516 blocked[11,8] = -HYP_3_4
517 blocked[11,9] = -ADLP_3_4
518
519 #Contribucion hilos-hilos
520
521 #Contribucion hilos-hilos
522 blocked[10,12] = DLP_5_4
523 blocked[10,13] = -SLP_5_4
524 blocked[11,12] = -HYP_5_4
525 blocked[11,13] = -ADLP_5_4
526
527 #Contribucion matriz-hilos
528 blocked[10,0] = -DLP_m_4
529 blocked[10,1] = SLP_m_4
530 blocked[11,0] = HYP_m_4
531 blocked[11,1] = ADLP_m_4
532
533 #Contribucion hilos-hilos
534 blocked[12,2] = DLP_0_5
535 blocked[12,3] = -SLP_0_5
536 blocked[13,2] = -HYP_0_5
537 blocked[13,3] = -ADLP_0_5
538
539 #Contribucion hilos-hilos
540 blocked[12,4] = DLP_1_5
541 blocked[12,5] = -SLP_1_5
542 blocked[13,4] = -HYP_1_5
543 blocked[13,5] = -ADLP_1_5
544
545 #Contribucion hilos-hilos
546 blocked[12,6] = DLP_2_5
547 blocked[12,7] = -SLP_2_5
548 blocked[13,6] = -HYP_2_5
549 blocked[13,7] = -ADLP_2_5
550
551 #Contribucion hilos-hilos
552 blocked[12,8] = DLP_3_5
553 blocked[12,9] = -SLP_3_5
554 blocked[13,8] = -HYP_3_5
555 blocked[13,9] = -ADLP_3_5
556
557 #Contribucion hilos-hilos
558 blocked[12,10] = DLP_4_5
559 blocked[12,11] = -SLP_4_5
560 blocked[13,10] = -HYP_4_5
561 blocked[13,11] = -ADLP_4_5
562
563 #Contribucion hilos-hilos
564
```

```
565 #Contribucion matriz-hilos
566 blocked[12,0] = -DLP_m_5
567 blocked[12,1] = SLP_m_5
568 blocked[13,0] = HYP_m_5
569 blocked[13,1] = ADLP_m_5
570
571 #Condiciones de borde
572 dirichlet_grid_fun_m = bempp.api.GridFunction(dirichlet_space_m, fun=
    dirichlet_fun)
573 neumann_grid_fun_m = bempp.api.GridFunction(neumann_space_m, fun=
    neumann_fun)
574
575 #Discretizacion lado izquierdo
576 blocked_discretizado = blocked.strong_form()
577
578 #Discretizacion lado derecho
579 rhs = np.concatenate([dirichlet_grid_fun_m.coefficients,
    neumann_grid_fun_m.coefficients,np.zeros(dirichlet_space_0.global_dof_count
    ), np.z    eros(neumann_space_0.global_dof_count), np.zeros(
    dirichlet_space_1.global_dof_count), np.zeros(neumann_space_1.
    global_dof_count), np.zeros(    dirichlet_space_2.global_dof_count), np.
    zeros(neumann_space_2.global_dof_count), np.zeros(dirichlet_space_3.
    global_dof_count), np.zeros(neu    mann_space_3.global_dof_count), np.zeros
    (dirichlet_space_4.global_dof_count), np.zeros(neumann_space_4.
    global_dof_count), np.zeros(dirichle    t_space_5.global_dof_count), np.
    zeros(neumann_space_5.global_dof_count)])
580
581 #Sistema de ecuaciones
582 import inspect
583 from scipy.sparse.linalg import gmres
584 it_count = 0
585 def iteration_counter(x):
586     global it_count
587     it_count += 1
588     frame = inspect.currentframe().f_back
589     print it_count, frame.f_locals["resid"]
590 print("Shape of matrix: {0}".format(blocked_discretizado.shape))
591 x,info = gmres(blocked_discretizado, rhs, tol=1e-5, callback =
    iteration_counter, maxiter = 150000)
592 print("El sistema fue resuelto en {0} iteraciones".format(it_count))
593 np.savetxt("Solucion.out", x, delimiter=",")
594
595 #Campo interior
596 interior_field_dirichlet_m = bempp.api.GridFunction(dirichlet_space_m,
    coefficients=x[:dirichlet_space_m.global_dof_count])
597 interior_field_neumann_m = bempp.api.GridFunction(neumann_space_m,
    coefficients=x[dirichlet_space_m.global_dof_count:dirichlet_space_m.globa
      l_dof_count + neumann_space_m.global_dof_count])
598
599 #Campo exterior
600 exterior_field_dirichlet_m = interior_field_dirichlet_m
601 exterior_field_neumann_m = interior_field_neumann_m*(1./alfa_m)
602
```

```
603 #Calculo campo en antena
604 slp_pot_ext_m = bempp.api.operators.potential.helmholtz.single_layer(
    dirichlet_space_m, antena, k)
605 dlp_pot_ext_m = bempp.api.operators.potential.helmholtz.double_layer(
    dirichlet_space_m, antena, k)
606 Campo_en_antena = (dlp_pot_ext_m * exterior_field_dirichlet_m -
    slp_pot_ext_m * exterior_field_neumann_m).ravel() + np.exp(1j*k*antena[0])
607 print "Valor del campo en receptor:", Campo_en_antena
```

Apéndice C: Código final para compósito como un todo, sin la inclusión de hilos.

```
##############################################
import numpy as np
import bempp.api
omega = 2.*np.pi*12e9
e0 = 8.854*1e-12*1e-18
mu0 = 4.*np.pi*1e-7*1e6
mue = (-20.-20.j)*mu0
ee = (8.-5.j)*e0
mui = (-1.734+0.37j)*mu0
ei = (-87641.359-104633.972j)*e0
k = omega*np.sqrt(e0*mu0)
lam = 2*np.pi/k
nm = np.sqrt((ee*mue)/(e0*mu0))
nc = np.sqrt((ei*mui)/(e0*mu0))
alfa_m = mue/mu0
alfa_c = mui/mue
antena = np.array([[1e4],[0.],[0.]])
print "Numero de onda exterior:", k
print "Indice de refraccion matriz:", nm
print "Indice de refraccion conductor:", nc
print "Numero de onda interior matriz:", nm*k
print "Numero de onda interior conductor:", nm*nc*k
print "Indice de transmision matriz:", alfa_m
print "Indice de transmision conductor:", alfa_c
print "Longitud de onda:", lam, "micras"

#Importando mallas
matriz = bempp.api.import_grid('/home/milan/matriz_12x12x300_E16772.msh')

#Funciones de dirichlet y neumann
def dirichlet_fun(x, n, domain_index, result):
    result[0] = 1.*np.exp(1j*k*x[0])
def neumann_fun(x, n, domain_index, result):
    result[0] = 1.*1j*k*n[0]*np.exp(1j*k*x[0])

#Operadores identidad

#Operadores multitrazo
Ai_m = bempp.api.operators.boundary.helmholtz.multitrace_operator(matriz,
    nm*k)
Ae_m = bempp.api.operators.boundary.helmholtz.multitrace_operator(matriz, k
    )

#Transmision en Multitrazo
Ae_m[0,1] = Ae_m[0,1]*(1./alfa_m)
Ae_m[1,1] = Ae_m[1,1]*(1./alfa_m)
```

```
44
#Acople interior y exterior
op_m = (Ai_m + Ae_m)

#Espacios
dirichlet_space_m = Ai_m[0,0].domain
neumann_space_m = Ai_m[0,1].domain
ident_m = bempp.api.operators.boundary.sparse.identity(neumann_space_m, neumann_space_m, neumann_space_m)
op_m[1,1] = op_m[1,1] + 0.5 * ident_m * ((alfa_m -1)/alfa_m)

#Operadores entre mallas

#Matriz de operadores
blocked = bempp.api.BlockedOperator(2,2)

#Diagonal
blocked[0,0] = op_m[0,0]
blocked[0,1] = op_m[0,1]
blocked[1,0] = op_m[1,0]
blocked[1,1] = op_m[1,1]

#Contribucion hilos-matriz

#Condiciones de borde
dirichlet_grid_fun_m = bempp.api.GridFunction(dirichlet_space_m, fun=dirichlet_fun)
neumann_grid_fun_m = bempp.api.GridFunction(neumann_space_m, fun=neumann_fun)

#Discretizacion lado izquierdo
blocked_discretizado = blocked.strong_form()

#Discretizacion lado derecho
rhs = np.concatenate([dirichlet_grid_fun_m.coefficients, neumann_grid_fun_m.coefficients,])

#Sistema de ecuaciones
import inspect
from scipy.sparse.linalg import gmres
it_count = 0
def iteration_counter(x):
    global it_count
    it_count += 1
    frame = inspect.currentframe().f_back
    print it_count, frame.f_locals["resid"]
print("Shape of matrix: {0}".format(blocked_discretizado.shape))
x,info = gmres(blocked_discretizado, rhs, tol=1e-5, callback = iteration_counter, maxiter = 150000)
print("El sistema fue resuelto en {0} iteraciones".format(it_count))
np.savetxt("Solucion.out", x, delimiter=",")

#Campo interior
```

```
92 interior_field_dirichlet_m = bempp.api.GridFunction(dirichlet_space_m,
    coefficients=x[:dirichlet_space_m.global_dof_count])
93 interior_field_neumann_m = bempp.api.GridFunction(neumann_space_m,
    coefficients=x[dirichlet_space_m.global_dof_count:dirichlet_space_m.globa
      l_dof_count + neumann_space_m.global_dof_count])
94
95 #Campo exterior
96 exterior_field_dirichlet_m = interior_field_dirichlet_m
97 exterior_field_neumann_m = interior_field_neumann_m*(1./alfa_m)
98
99 #Calculo campo en antena
100 slp_pot_ext_m = bempp.api.operators.potential.helmholtz.single_layer(
    dirichlet_space_m, antena, k)
101 dlp_pot_ext_m = bempp.api.operators.potential.helmholtz.double_layer(
    dirichlet_space_m, antena, k)
102 Campo_en_antena = (dlp_pot_ext_m * exterior_field_dirichlet_m -
    slp_pot_ext_m * exterior_field_neumann_m).ravel() + np.exp(1j*k*antena[0])
103 print "Valor del campo en receptor:", Campo_en_antena
106 ##########################################
```