Escola Politècnica Superior de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE CARRERA

TÍTULO DEL TFC: Implementación del precondicionador SAI como herramient aceleradora en el cálculo de la Sección Recta Radar de blancos conductores

TITULACIÓN: Enginyeria Tècnica de Telecomunicació, especialitat Sistemes de Telecomunicació

AUTOR: Adam Cresswell Baldellou

DIRECTOR: Eduard Úbeda Farré

DATA: 17 de julio de 2009

Título: Implementación del precondicionador SAI como herramienta aceleradora en el cálculo de la Sección Recta Radar de blancos conductores

Autor: Adam Cresswell Baldellou

Director: Eduard Úbeda Farré

Data: 17 de julio de 2009

Resumen

En este trabajo se ha implementado el precondicionador SAI (Sparse Approximative Inverse) en el contexto de la búsqueda iterativa de la solución a un problema de análisis numérico de la dispersión electromagnética asociada a un blanco Radar conductor al ser incidido por una onda plana. Se ha comparado su capacidad de aceleración en la búsqueda de la solución, la corriente eléctrica distribuida sobre el conductor, con respecto al uso del precondicionador convencional ILU (Inclomplete Lower-Upper Transform). El algoritmo iterativo adoptado ha sido el Gradiente Biconjugado (BiCG). También, se ha implementado una mejora en el precondicionador SAI, que hemos denominado SAI imaginario. Esta mejora disminuye drásticamente el tiempo de cálculo del precondicionador. Hemos valorado para qué tipo de problemas la implementación del SAI imaginario no supone una merma apreciable en la capacidad de acceleración respecto del precondicionador SAI.

Para estudiar el comportamiento de los precondicionadores SAI , ILU y SAI imaginario hemos creado una interficie en Matlab que nos permite mallar, obtener la matriz de impedancias y analizar el problema, así como hacer pruebas con diferentes objetos, direcciones y polarizaciones del campo incidente.

Las conclusiones a las que hemos llegado son:

-El SAI no puede competir con el precondicionador ILU pero el primero es paralelizable mientras que el segundo no.
-El tiempo de cálculo del precondicionador SAI imaginario es más rápido que el SAI pero aun así no es tan rápido como el precondicionador ILU.
- El comportamiento del SAI imaginario respecto del número de iteraciones es mejor cuanto más pequeño eléctricamente es el objeto analizado.
-El SAI trabaja mejor con matrices dispersas que la ILU.

Title: SAI preconditioner implementation as accelerating tool in the Radar Cross Section computation for perfectly conducting targets

Author: Adam Cresswell Baldellou

Director: Eduard Úbeda Farré

Data: July 17th 2009

Overview

In this work the SAI preconditioner (Sparse Approximative Inverse) has been implemented in the context of the iterative search of the solution associated to the electromagnetic scattering of a radar target when impinged by a plane wave. The acceleration produced by the SAI preconditioner in the iterative search of the solution has been compared with conventional tools such as the ILU preconditioner (Incomplete Lower-Upper Transform). The iterative algorithm adopted is the BiConjugate Gradient (BiCG). Furthermore, it has been carried out an improvement of the SAI preconditioner, the immaginary SAI preconditioner. This modification reduces drastically the required computational time. Moreover, we have assessed the type of problems for which the Immaginary SAI preconditioner does not involve any remarkable loss in the capacity of acceleration respect to the SAI preconditioner.

To study the behavior of the SAI, ILU and Immaginary SAI preconditioners, we have programmed an interface in Matlab that allows for meshing the radar target, for computing the matrix impedance elements and for solving the problem, either directly or iteratively.

These are the conclusions from our comparative study:

- The SAI preconditioner cannot compete with the ILU preconditioner although it can be parallellized much better.
- The time for computing the preconditioner is faster for the Immaginari SAI than for the conventional SAI. However, it is less fast than for the ILU preconditioner.
-The immaginary SAI preconditioner performance in the convergence rate becomes better as the electrical dimensions of the object decrease.
- SAI preconditioning schemes perform better than ILU with sparse matrices.

INDICE

INTRODUCCIÓN.....8

CAPÍTULO 1. ANÁLISIS NUMÉRICO..... 10

1.1. Planteamiento y Discretización..... 10

1.2. Clasificación de los Métodos Numéricos.....11

1.3. Métodos integrales.....11

1.4. Método de momentos.....13

1.5. Funciones base y peso.....14

1.6. Solución del sistema matricial.....16

1.6.1. Inversión directa 16
1.6.2. Algoritmo iterativo 16
1.6.3. Condicionamiento de la matriz 17

CAPÍTULO 2. PRECONDICIONADOR.....18

2.1. Precondicionador ILU.....19

2.2. Precondicionador SAI.....20

2.3. Precondicionador SAI Imaginario.....22

CAPÍTULO 3. INTERFICIE MATLAB.....23

CAPÍTULO 4. RESULTADOS 26

4.1. Objeto: Esfera.....28

4.1.1. Dimensiones eléctricas moderadas.....28
4.1.2. Dimensiones eléctricas pequeñas.....34

4.2. Objeto: Cubo.....40

4.1.1. Dimensiones eléctricas moderadas.....40
4.1.2. Dimensiones eléctricas pequeñas.....43

4.2. Objeto: SRR en resonancia.....46

CAPÍTULO 5. CONCLUSIONES.....53

BIBLIOGRAFIA.....54

INTRODUCCIÓN

El análisis electromagnético de un problema arbitrario requiere la discretización o mallado del problema y la implementación numérica de un método de resolución. En este trabajo, calculamos la RCS (Sección Recta Radar) de un blanco conductor mediante la discretización en Método de Momentos de la Ecuación Integral de Campo Eléctrico (EFIE). Se obtiene una matriz llena, la matriz de impedancias, que da cuenta de las interacciones entre todas las funciones base que discretizan el problema original. La inversión directa de la matriz de impedancias permite obtener la corriente equivalente en la superficie del conductor. Este proceso resulta muy costoso en términos de tiempo y de recursos de memoria a medida que las dimensiones eléctricas del objeto aumentan o requieren mayor nivel de detalle. En este contexto, hay que implementar algoritmos iterativos de búsqueda de la solución, como por ejemplo el Gradiente BiConjugado (ver [1]).

En matrices derivadas de la EFIE, el número de iteraciones en las trayectorias de búsqueda de la solución crecen mucho con el número de incógnitas. La solución, pues, de un problema dado puede ser inabordable porque el algoritmo se estanque y no converja. Así pues, es preceptiva la implementación en el agoritmo iterativo de los precondicionadores, que son matrices que pre-multiplican el sistema a ambos lados y aceleran la convergencia; es decir, disminuyen el número de iteraciones necesarias para converger.

En este trabajo se ha implementado el precondicionador SAI (Sparse Approximative Inverse) (ver [1][2][3]) . Se ha comparado su capacidad de aceleración en la búsqueda de la solución, la corriente eléctrica distribuida sobre el conductor, con respecto a las situaciones de uso del precondicionador convencional *ILU* (Inclomplete Lower-Upper Transform) y de ausencia de precondicionador. También, se ha implementado una mejora en el precondicionador *SAI*, que hemos denominado *SAI imaginario*. Esta mejora disminuye drásticamente el tiempo de cálculo del precondicionador. Hemos valorado para qué tipo de problemas la implementación del *SAI imaginario* no supone una merma apreciable en la capacidad de acceleración respecto del precondicionador *SAI*.

Para estudiar el comportamiento de los precondicionadores *SAI* , *ILU* y *SAI imaginario* hemos creado una interficie en Matlab que nos permite mallar, obtener la matriz de impedancias y analizar el problema, así como, hacer pruebas con diferentes objetos, direcciones y polarizaciones del campo incidente.

De nuestro estudio, hemos llegado a las siguientes conclusiones:

(1) El precondicionador *SAI* requiere más tiempo para ser calculado que el precondicionador *ILU*.

(2) El tiempo de cálculo del precondicionador *SAI imaginario* es mucho más rápido que el precondicionador *SAI* pero aun así no es tan rápido como el precondicionador *ILU*. Sin embargo, ambos precondicionadores *SAI*, a diferencia del precondicionador *ILU*, son muy fácilmente paralelizables.

(3) El número de iteraciones de convergencia asociadas al precondicionador *SAI imaginario* se parecen más a las del precondicionador *SAI* cuanto más pequeñas son las dimensiones eléctricas del objeto analizado.

(4) El precondicionador *SAI* es más robusto que el precondicionador *ILU* al grado de dispersión de la matriz.

CAPÍTULO 1. ANÁLISIS NUMÉRICO

1.1. Planteamiento y Discretización

Un objeto con determinadas propiedades físicas es susceptible de dispersar la energía electromagnética incidente, ya sea a través de una diferencia de tensión aplicada en sus terminales (antena), ya sea a través de la incidencia de una campo eléctrico determinado (blanco Radar).

Se induce pues, sobre la superficie del objeto, una corriente eléctrica, llamada corriente equivalente, que es la generadora del campo dispersado por el objeto. Nuestro objetivo en los problemas de análisis numérico analizados en este trabajo es encontrar esta corriente equivalente inducida.

La solución analítica (solución exacta) del problema es solamente posible para algunas geometrías muy particulares, llamadas canónicas. Ejemplos típicos son la difracción de una onda incidente por objetos cilíndricos infinitos (2D) o esféricos (3D). Como en general los elementos que estudiaremos no responden a este tipo de geometría, no es posible una solución analítica exacta y es necesario recurrir a la solución numérica del problema. Esta solución no es la solución exacta sino una aproximación y suele realizarse a través de los siguientes pasos:

1) ***Formulación del problema electromagnético*****:** naturalmente, se parte siempre de las ecuaciones de Maxwell. En este trabajo, se opta por una formulación integral de campo eléctrico (EFIE) derivada de las funciones de Maxwell a partir del conocimiento de la función de Green en el espacio libre.

2) ***Discretización de la formulación matemática*****:** como el cálculo numérico mediante ordenador sólo puede procesar números y no funciones, es necesario discretizar en vectores numéricos todas las funciones. Así, se transforma un sistema de ecuaciones funcionales en un sistema de ecuaciones algebraicas o matriciales de dimensión finita. En este trabajo, se aplica una discretización en Método de Momentos, lo cual implica adoptar un conjunto de *funciones base* para desarrollar la corriente y un conjunto de *funciones test* para ponderar el campo y así imponer es que la componente tangencial del campo eléctrico total en la superficie sea cero.

3) Solución del problema: La parte final del proceso pasa por obtener la incógnita, en nuestro caso, los coeficientes de las funciones base desarrollantes de la corriente eléctrica, a partir de la inversión de la matriz, la cual, en el contexto de Método de Momentos aplicado a la EFIE, se llama matriz de impedancias.

1.2. Clasificación de los Métodos Numéricos

Existen diferentes maneras de clasificar los Métodos numéricos de análisis electromagnético. Según el dominio en que se trabaje: *Métodos en el dominio del tiempo,* utilizan la formulación temporal del problema electromagnético, o *Métodos en el dominio de la frecuencia,* los cuales utilizan la formulación en régimen permanente senoidal. Asimismo, son *Métodos diferenciales* si discretizan directamente las ecuaciones de Maxwell y son *Métodos Integrales* si optan por la formulación integro-diferencial equivalente resultante de convolucionar las corrientes equivalentes con la función de Green

El coste computacional (cantidad de memoria y tiempo de cálculo) requerido por los métodos numéricos anteriores para el análisis de una antena aumenta enormemente con la frecuencia de trabajo (el tamaño de la celda ha de ser como máximo de un décimo de la longitud de onda λ). Por tanto, el número de incógnitas crece exponencialmente con las dimensiones eléctricas del objeto a analizar. Así pues, queda impuesto un límite superior al tamaño eléctrico de los objetos que es posible analizar con la tecnología informática disponible. Los *Métodos de Alta Frecuencia* implementan aproximaciones, solamente válidas en la región de alta frecuencia, que permiten el análisis de estructuras de tamaño eléctrico grande.

En este Trabajo, nosotros resolvemos nuestro problema a través de las ecuaciones integrales, con la Ecuación Integral de Campo Eléctrico (EFIE), en el dominio de la frecuencia. Además, llevamos a cabo una discretización basada en el Método de los Momentos.

1.3 Métodos integrales

Se basan en la aplicación del teorema de equivalencia para el cálculo del campo en el exterior o en el interior del objeto mediante la integral de radiación de las corrientes equivalentes y en la imposición de las condiciones de contorno al campo resultante.

Las expresiones integrales de los campos dispersados resultan de la integración de las corrientes con la función de Green. La función de Green es la respuesta impulsional del sistema a una fuente de corriente puntual. En el espacio libre, marco en el que residen la mayoría de problemas de dispersión electromagnética. La expresión de la función de Green es:

$$G(\vec{r},\vec{r}\,')=\exp\left(\frac{-jkR}{R}\right) \tag{1.1}$$

donde $\vec{r}\,'$ es el punto de fuente donde está definida la corriente, $\vec{r}$ se corresponde con el punto de campo, donde está definido el campo y *k* es el número de onda (2π/λ). *R* denota la distancia entre el punto de fuente y el punto de campo.

El problema original de dispersión electromagnética a cargo de un objeto conductor es heterogéneo porque las propiedades eléctricas del objeto son diferentes a las del espacio libre que lo envuelve. El Teorema de Equivalencia (Superficial) (ver [4]) permite definir un problema equivalente, es decir, con la misma solución que el problema original, pero con todas las magnitudes definidas en un solo medio, homogéneo, con las propiedades físicas del espacio libre (ver Fig. 1.1). En el problema equivalente aparecen unas corrientes equivalentes (no reales) radiantes en el espacio libre. Estas corrientes representan la incógnita del problema y radian el mismo campo que existe en el problema real. Ello permite definir las integrales de radiación como la convolución de la función de Green en el espacio libre con la corriente equivalente.

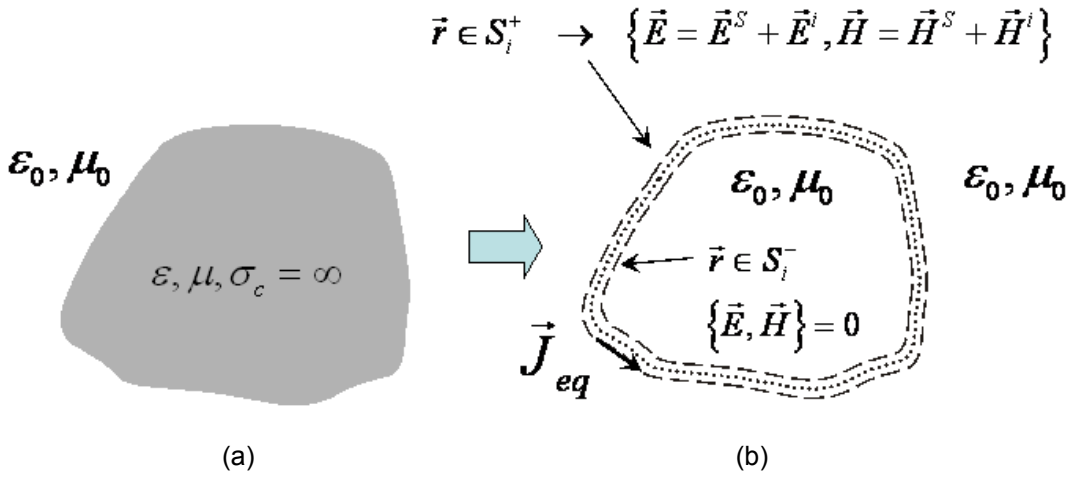


Figura 1.1. Teorema de equivalencia superficial a) problema original, b) problema equivalente superficial donde, E^S denota el campo eléctrico dispersado, H^S el campo magnético dispersado y E^i H^i son los campos eléctrico y magnético incidentes. ε_0 y μ_0 representan, respectivamente, la permitividad dieléctrica y la permeabilidad magnética en el vacío. σ_c denota la contuctividad en la región conductora

En general, las ecuaciones integrales pueden representarse en la forma general

$$LX = Y \tag{1.2}$$

donde *L* puede ser uno de los operadores lineales resultantes de aplicar la integral de campo eléctrico o de magnético dispersado, E^S y H^S , ver Fig. 1.1, a la corriente equivalente. La función *X* es la incógnita (corrientes equivalentes) y la función *Y* representa el término independiente (campo incidente). Para

resolver (1.2) mediante ordenador es necesario discretizar las funciones y operadores y convertir la ecuación funcional en una ecuación matricial, mediante el método llamado *Método de momentos.*

1.4 Método de momentos

Primero aproximamos nuestra función incógnita *X* a través de una combinación lineal de N funciones base xj

$$X_N = \sum_j^N a_j x_j \approx X \tag{1.3}$$

X_N representa, pues, una aproximación de *X.*

Al substituir (1.3) en la ecuación funcional (1.2) se obtiene

$$Y_N = \sum_j^N a_j L x_j \tag{1.4}$$

Donde Y_N representa una aproximación del campo incidente Y en la superficie. Se define, pues el residuo *R* como la diferencia entre el campo incidente aproximado y el real en la superficie del conductor*:* .

$$R = Y - Y_N = Y - \sum_j^N a_j \; L x_j \tag{1.5}$$

Para convertir la ecuación funcional (1.4) en un sistema de N ecuaciones con N incógnitas, se hace nulo el residuo ponderado con N *funciones peso* W_i

$$\langle W_i \, , \, R \rangle = 0 \qquad i = 1 \; ... \; N \tag{1.6}$$

donde el producto esta definido como el producto escalar

$$\langle f \, , \, g \rangle = \int_{D_{(L)}} f \, (\vec{r}) \cdot g \, (\vec{r}) \, \vec{dr} \tag{1.7}$$

El sistema de N ecuaciones lineales con N incógnitas que resulta es:

$$\langle W_i , \, Y \rangle = \sum_j^N a_j \, \langle W_i , Lx_j \rangle \qquad i = 1 \; ... \; N \tag{1.8}$$

que puede escribirse en forma matricial como:

$$[Z]J = e \tag{1.9}$$

Donde *Z* es una matriz llena de dimensiones NxN, conocida como matriz de impedancias, cuyos elementos son $Z_{ij}=\langle W_i, Lx_j \rangle$. Además, *J* es el vector columna formado por los elementos a_j de la ec. (1.3) y e es el vector columna $b_{ij}=\langle W_i, Y \rangle$

Si despejamos la incógnita *J*, obtenemos:

$$J = [Z]^{-1} e \tag{1.10}$$

Dependiendo de los recursos computacionales disponibles, podemos resolver esta ecuación por inversión directa o a través de un algoritmo iterativo de búsqueda de la solución. En la mayoría de implementaciones en Método de Momentos el conjunto de funciones base, (1.3), y de funciones peso, (1.6), son las mismas, procedimiento conocido como *Galerkin*.

1.5. Funciones base y peso

Para la discretización de las corrientes y la geometría de la superficie o volumen sobre la que se aplican las condiciones de contorno es necesario utilizar en la ecuación (1.2) unas funciones base adecuadas. El mallado a través de parches o celdas triangulares permite discretizar cualquier tipo de superficie (ver Fig. 1.2) .

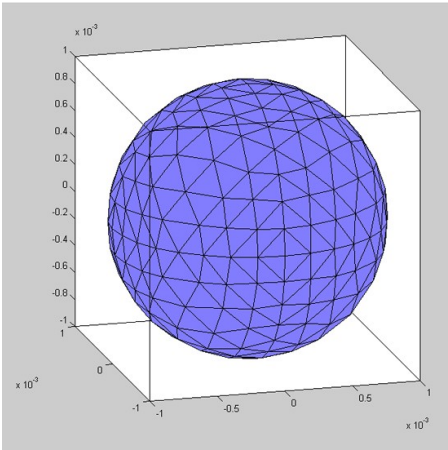


Figura 1.2. Mallado en parches triangulares de una esfera

Las funciones base utilizadas en este Trabajo para discretizar una superficie 3D están basadas en los triángulos de Rao, Glisson y Wilton (RWG) (ver [5]). Se definen las funciones base RWG como funciones vectoriales $\vec{f}_n$ asociadas a una arista n de la discretización, de longitud l_n que es común a dos triángulos T_n^+ y T_n^-

$$\vec{f}_n(\vec{r}) = \begin{cases} \dfrac{\ell_n}{2A_n^+}\vec{\rho}_n^+ & \text{si } \vec{r} \in T_n^+ \\ \dfrac{\ell_n}{2A_n^-}\vec{\rho}_n^- & \text{si } \vec{r} \in T_n^- \end{cases} \tag{1.11}$$

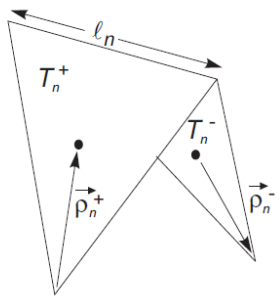


Figura 1.3. Función base RWG asociada a la n-ésima arista del mallado con triángulos

donde p_n^+ es el vector desde el vertice de T_n^+ opuesto a l_n hasta $\vec{r}$ y p_n^- desde $\vec{r}$ hasta el vértice de T_n^- opuesto a l_n. A_n^+ y A_n^- representan, respectivamente, las áreas de los triángulo positivo y negativo, T_n^+ y T_n^-. La componente de J normal a l_n se toma con signo positivo cuando fluye de T_n^+ a T_n^-.

Así pues, a la vista de la ec. (1.3), nuestro desarrollo de la corriente en función de las funciones RWG queda

$$\vec{J} \approx \vec{J}_N = \sum_{n=1}^{N} I_n \vec{f}_{n(\vec{r})} \tag{1.12}$$

donde N es el número de los lados comunes a dos triángulos, es decir, el número de aristas, y los coeficientes I_n son el valor de la corriente que atraviesa el lado l_n en la dirección normal. El valor de la corriente dentro de cada triángulo es una interpolación lineal vectorial de los tres valores I_n de las componentes de J la corriente normales a los tres lados del triángulo.

1.6.Solución del sistema matricial

1.6.1 Inversión directa

La *inversión directa* consiste en encontrar la inversa de *Z* a partir de los procedimientos habituales de descomposición *LU* (triangular superior e inferior) y eliminación gaussiana. El tamaño de la matriz está directamente relacionado con las dimensiones eléctricas del objeto y también con el nivel de detalle y complejidad que tenga el objeto. Dado que los parches triangulares tienen dimensiones máximás de $\lambda/10$, cuanto más grandes son las dimensiones eléctricas del objeto, más mallado está el objeto y por tanto la matriz de impedancias resultante tiene unas dimensiones mayores.

El tiempo de cálculo requerido para efectuar la inversión directa de la matriz crece cúbicamente con el número de incógnitas. Por tanto, a medida que el objeto crece en dimensiones eléctricas, requiere más y más recursos, de tal manera que en un cierto punto, la inversión directa de la matriz es inabordable con los recursos de memoria disponibles.

Por tanto, para abordar la solución de este tipo de problemas hay que optar por un algoritmo iterativo de búsqueda de la solución. Cada iteración representa un tiempo de computación con dependencia cuadrática respecto del número de incógnitas.

1.6.2 Algoritmo iterativo

Los *algoritmos iterativos* son algoritmos que buscan la solución de manera secuencial, iteración tras iteración. Dos de los métodos más comunes son el Gradiente conjugado, solamente válido para Matrices simétricas, y el Bigradiente conjugado, que es la generalización del Gradiente conjugado a cualquier tipo de matriz. El algoritmo del Gradiente Biconjugado es el adoptado para los problemas presentados en este Trabajo.

La búsqueda iterativa de la solución puede resultar más o menos ágil. Puede acarrear más o menos iteraciones en función de las propiedades de la matriz. La propiedad que determina la velocidad de convergencia del algoritmo iterativo es el condicionamiento de la matriz. Normalmente, el algoritmo iterativo de búsqueda de la solución se trunca una vez el error en la solución está por debajo de un umbral o tolerancia prefijado de antemano. En la interficie Matlab programada, se denomina esta cantidad como *tol.*

1.6.3 Condicionamiento de la matriz

A la hora de resolver un problema a través de un algoritmo iterativo nos podemos encontrar con que el número de iteraciones requerido sea muy elevado o que incluso, el algoritmo no converja. Es conocido que la discretización de la la Ecuación Integral del campo Eléctrico (EFIE), que es la ecuación integral desarrollada en este Trabajo, produce matrices mal condicionadas. El condicionamiento de una matriz indica el grado de singularidad o de casi-singularidad de la matriz. A efectos prácticos, el condicionamiento determina el grado de inmunidad del proceso de búsqueda iterativa de la solución a errores inherentes de truncamiento, aproximación o redondeo. En efecto, pueden surgir problemas porque el algoritmo iterativo es muy sensible a pequeños errores de truncamiento que se arrastran a lo largo del algoritmo y que desvían la trayectoria correcta de convergencia. En algunos casos, esta desviación puede ser incorregible. Por tanto, para compensar las posibles deficiencias de convergencia en el algoritmo iterativo, normalmente se añade un precondicionador a la ecuación matricial original.

CAPÍTULO 2. PRECONDICIONADOR

El precondicionador es una matriz que pre-multiplica (o-post-multiplica) el sistema matricial de manera que el condicionamiento del nuevo sistema matricial mejora. Por tanto, el número de condicionamiento de la nueva matriz producto y el número de iteraciones para converger también disminuyen.

El número de condicionamiento es un valor que determina el grado de singularidad de la matriz. Cuanto más pequeño, la matriz está mejor condicionada y más rápido convergerá a través de de un algoritmo iterativo. Por ejemplo, la identidad tiene el número de condicionamiento mínimo e igual a 1, lo cual se corresponde con un condicionamiento óptimo. Cuanto más elevado sea el número de condicionamiento de la matriz, la matriz está peor condicionada y, en consecuencia, las trayectorias de convergencia trazadas para el algoritmo iterativo son cada vez más sensibles a los errores de redondeo y truncamiento arrastrados a lo largo del proceso. De hecho, a partir de un cierto margen de valores que depende de la precisión de la máquina, la convergencia es imposible ya que la máquina no es lo suficientemente precisa para ajustar la trayectoria de convergencia.

Un precondicionador *P* debe ser tal que su producto con la matriz de impedancias *Z* se parezca a la identidad. Así, se consigue que el número condicionamiento y el número de iteraciones para converger del nuevo sistema matricial disminuyan. Si el producto del precondicionador por la matriz de impedancias ha de parecerse a la identidad, el precondicionador ha de parecerse por fuerza a la inversa de *Z*.

El proceso de precondicionado no implica ninguna mejora en precisión, ya que el precondicionador *P* multiplica ambos lados del sistema matricial original. Sí que supone sin embargo una aceleración del algoritmo iterativo en su búsqueda de la solución. Si tomamos la ec. 1.9 que deseamos resolver y multiplicamos a la izquierda en ambos lados de la ecuación, obtenemos

$$[\boldsymbol{P}]\,[Z]\,J = [\boldsymbol{P}]\,e \tag{2.1}$$

donde la matriz que rige el sistema viene dada por el producto de *P* y *Z,* y el término independiente es el producto de *P* por el término independiente original.

Si se producto se realiza a la derecha, en cambio, el sistema queda

$$[Z]\,[\boldsymbol{P}]\,y = e \tag{2.2}$$

donde resolvermos la ecuación igualmente, siendo la nueva matriz el producto de *Z* y *P.* A comentar que en el precondicionado a derechas, una vez encontrada *y* , se obtiene *J* resolviendo la siguiente ecuación

$$J = [\boldsymbol{P}]\, y \tag{2.3}$$

En este Trabajo, hemos optado por multiplicar el precondicionador a izquierdas. En el caso límite que definamos el precondicionador *P* como la inversa de *Z*, el algoritmo iterativo converge en una sola iteración. Efectivamente, Para el precondicionador situado a la izquierda: la solución se obtiene fácilmente en una sola iteración

$$[\boldsymbol{P}]\,[Z]J = [\boldsymbol{P}]\, e \;\rightarrow\; [Z]^{-1}\,[Z]\,J = [Z]^{-1}\, e \;\rightarrow\; J = [Z]^{-1}\, e \tag{2.4}$$

En la práctica no se escoge $P=Z^{-1}$ porque, lógicamente, implica un gran tiempo de cálculo para los problemas en que se requiere el uso del algoritmo iterativo. Así pues, hay que encontrar una matriz *P* que sea *parecida*, en la medida de lo posible, a la inversa de *Z* y que se calcule mucho más rápidamente. El tiempo total en encontrar la solución con el precondicionador, que incluye el tiempo de cálculo del precondicionador más el tiempo de convergencia del algoritmo iterativo, tiene que ser menor que el tiempo de convergencia solo del algoritmo iterativo sin precondicionador. Éste es el escenario ideal con matrices de grandes dimensiones ya que el precondicionador, si está bien diseñado, disminuye el número de iteraciones respecto del sistema original sin precondicionador.

Los precondicionadores se pueden clasificar en *implícitos*, si resuelven el sistema a través de la eliminación Gaussiana (*backward-forward substitution*) previa descomposición *LU* de la matriz, o *explícitos*, si efectúan una aproximación explícita de la inversa de la matriz. Como el método implícito resuelve un solo sistema lineal, no es fácilmente paralelizable. Como en el método explícito el precondicionador se calcula como producto de matrices, es fácilmente paralelizable.

El método implícito más conocido es la *Incomplete Lower-Upper decomposition* (ILU). El precondicionador explícito que se ha desarrollado en este Trabajo es el *Sparse Approximative Inverse* (SAI).

2.1.Precondicionador *ILU*

Los precondicionadores *ILU* se basan en la descomposición de *Z* como producto de matrices triangulares inferior (L) y superior (U). Para minimizar el número de operaciones, y por tanto el tiempo de cálculo, la matriz *Z* la hacemos dispersa descartando ciertos valores de la matriz de impedancias poco relevantes.

El cálculo de la matriz de impedancias en Método de Momentos siempre resulta en una matriz llena como consecuencia de computar las interacciones de campo eléctrico entre todas las funciones base con todas las funciones peso. Dado que el cálculo del precondicionador tiene que ser rápido, conviene descartar los elementos de esta matriz de impedancias *Z* menos relevantes en el resultado final. Así pues, antes de aplicar la descomposición *LU*, descartamos los elementos de la matriz de impedancias correspondientes a las interacciones menos cercanas (por encima de una cierta distancia umbral fijada por nosotros de antemano) (Fig. 2.1).

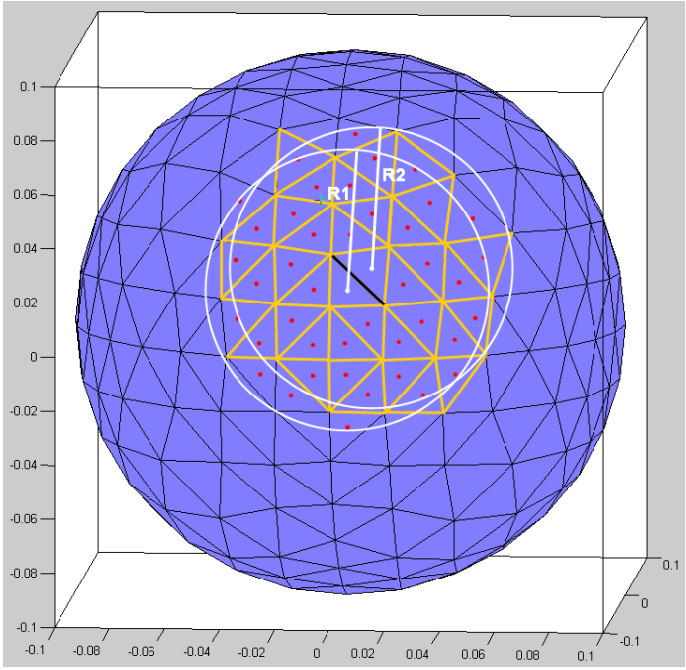


Figura 2.1. Donde R1=R2 y son la distancia del centro del triángulo superior/inferior de la arista que estamos calculando, si el centro de uno de los triángulos de fuente está dentro de estas dos distancias, tendremos en cuenta toda la función base a la hora de calcular Z.

En el programa creado en Matlab, el radio que incluye las interacciones a retener en la matriz dispersa lo denominamos *Rdi_tol*. Además, el proceso de backward-forward substitution también permite ser acelerado mediante una cierta tolerancia (*Drop_tol*) que descarta también algunas contribuciones poco relevantes. Se define así la *incomplete LU decomposition*. Para el caso límite en que *Rdi_tol* es superior a las dimensiones físicas máximas del objeto y *Drop_tol* es cero, con el precondicionador *ILU* estamos calculando exactamente la inversa de la matriz de impedancias *Z*.

2.2.Precondicionador *SAI*

El precondicionador *SAI* es una matriz dispersa que aproxima explícitamente la inversa de *Z*, y se calcula como la matriz *P* que, en su versión a izquierdas, minimiza la norma de Frobenius de la siguiente cantidad matricial

$$[P][Z]-Id \tag{2.5}$$

donde Id es la matriz identidad y Z es la matriz de impedancias.

Por definición, la norma de Frobenius de una matriz *A* es

$$\|A\|_F=\sqrt{\sum_{i,j=1}^{N}|a_{ij}|^2} \tag{2.6}$$

donde a_{ij} denota los elementos de la matriz. De la aplicación del cuadrado de la norma de Frobenius a la cantidad matricial en (2.5) obtenemos

$$\|[P][Z]-Id\|_F^2=\sum_{j=1}^{N}\|([P][Z]-Id)e_j\|_2^2=\sum_{j=1}^{N}\|p_j[Z]-e_j\|_2^2 \tag{2.7}$$

donde ej y pj se corresponden, respectivamente, con los vectores fila j-ésimos de la matrices Id y P . En (2.7) se observa la obtención de la norma de Frobenius al cuadrado de una matriz como sumatorio de las normas cuadráticas al cuadrado de las filas que componen la matriz.

Para encontrar los valores de la matriz de precondicionamiento de manera rápida, hay que ignorar el cálculo de los elementos menos relevantes de *P* así como descartar las contribuciones de los elementos menos importantes de *Z.* En general, dado que se busca un número de incógnitas por fila de *P* inferior al número de ecuaciones planteadas que minimicen la norma cuadrática para esa fila de la matriz de la cantidad matricial en (2.5) en cuestión, hay que resolver un sistema de mínimos cuadrados para cada fila. Así pues, hay que resolver N ecuaciones de mínimos cuadrados; es decir, cada uno de las contribuciones, de signo positivo, de los sumandos en (2.7), lo cual es muy fácilmente paralelizable.

$$\min_{pj}\|p_j[Z]-e_j\|_2 \quad j=1,2.....N \tag{2.8}$$

En el programa creado en Matlab, agilizamos el cálculo del precondicionador mediante dos parámetros: *Rdi_row*, que descarta las posiciones en el vector fila p_j menos relevantes y *Rdi_col*, que descarta las posiciones menos relevantes en el vector fila e_j. *Rdi-row* es siempre menor o igual que *Rdi_col* para disponer siempre de más ecuaciones que incógnitas y no dejar el problema indeterminado. Para escoger las posiciones de los vectores que tendremos en cuenta usaremos el mismo método que el usado en el precondicionador *ILU*, que es de quedarnos con las iteraciones que se encuentran más cerca de la arista *j,* correspondiente al vector fila j-ésimo que se esté considerando (ver ec. 2.8).

2.3.Precondicionador *SAI imaginario*

El cálculo del precondicionador *SAI* supone, si no se dispone de herramientas de paralelización, un tiempo de cálculo excesivo puesto que hay que minimizar consecutivamente la norma cuadrática de cada fila de la expresión matricial en (2.5). Se nos ha ocurrido una manera de mitigar esta excesiva carga computacional observando la expresión de los elementos de la matriz de impedancias Z_{mn} debida a la Ecuación integral EFIE aplicada a las funciones base *RWG*

$$\begin{aligned} Z_{mn} = & \left(\frac{j\omega\mu}{4\pi}\right)\int_S\int_S \vec{f}_m^S(\vec{r})\cdot\vec{f}_n^S(\vec{r}\,')Gd\vec{r}\,'\,d\vec{r} \\ & -\left(\frac{j}{4\pi\omega\varepsilon}\right)\int_S\int_S (\nabla_S\cdot\vec{f}_m^S)(\nabla_S\cdot\vec{f}_n^S)Gd\vec{r}\,'\,d\vec{r} \end{aligned} \tag{2.9}$$

siendo *G* la función de Green en espacio libre (ver ec. 1.1).

Si efectuamos el desarrollo de Taylor del numerador de la función de Green y simplificamos, obtenemos

$$G = \exp\left(\frac{-jkR}{R}\right) \approx \frac{1}{R} - jk - \frac{Rk^2}{2} + \ldots \tag{2.10}$$

donde R es la distancia entre el punto de campo $\vec{r}$ y el punto de fuente $\vec{r}\,'$

Cuanto más cerca estén $\vec{r}$ de $\vec{r}\,'$, más pequeño será *R* y por tanto más importante será la contribución del primer término en (2.10). Esta propiedad puede ser ventajosa en el cálculo de los precondicionadores, puesto que consideran solamente las interacciones más cercanas. La contribución del primer término en (2.10) en los elementos de la matriz de impedancias en (2.9) es asimismo imaginaria pura, lo cual nos permite agilizar muchísimo el cálculo del precondicionador SAI ya que la minimización de mínimos cuadrados ya no es compleja sino real.

CAPÍTULO 3. INTERFICIE MATLAB

En este apartado explicamos como esta organizado el programa que se ha creado en Matlab y donde se han insertado los distintos precondicionadores.

La interficie en Matlab tiene la forma:

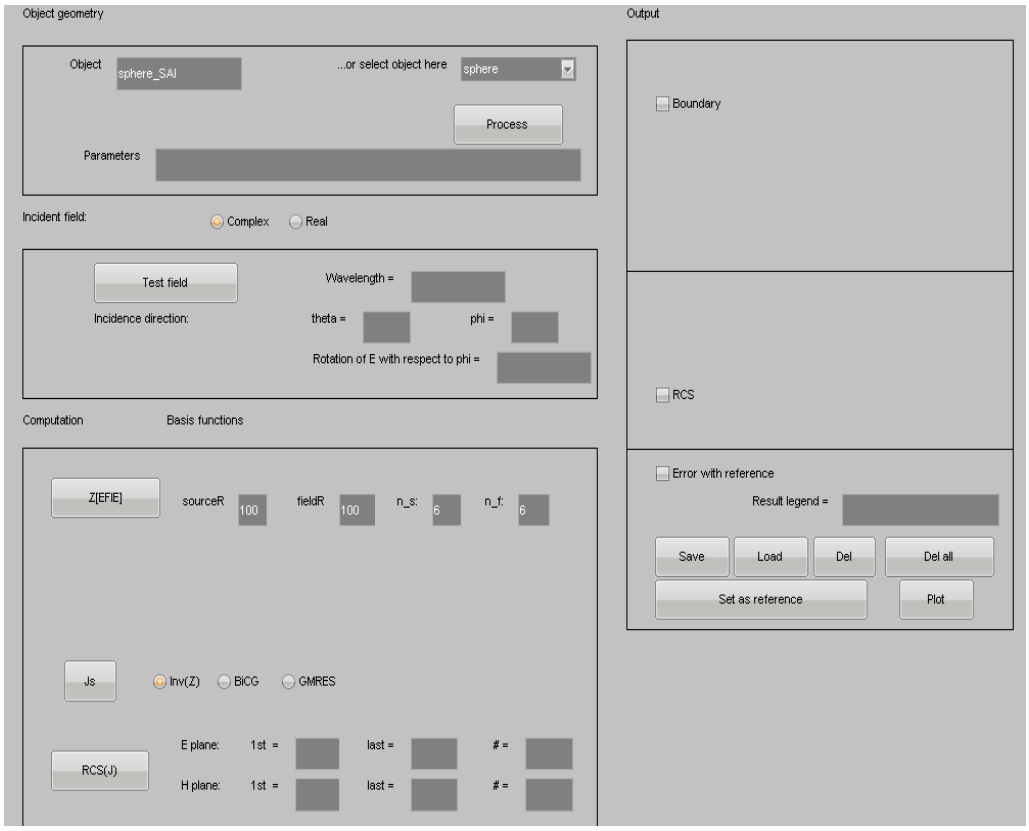


Figura 3.1. Interficie gráfica Matlab

Si analizamos un poco más a fondo la interficie,

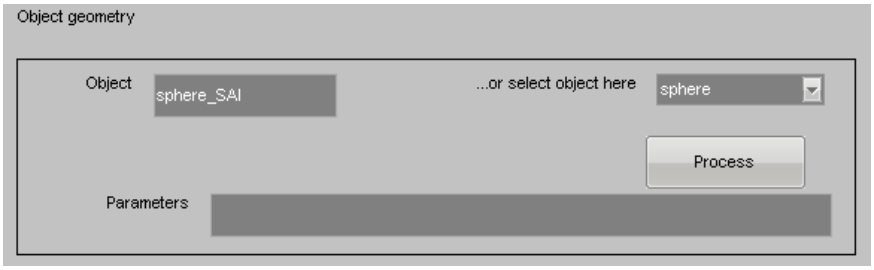


Figura 3.2. Características del objeto a estudiar

La Figura 3.2. muestra las características del objeto que vamos a estudiar o crear, ya que tenemos la posibilidad de crear esferas y prismas entre otros objetos. Para crear dichos objetos tenemos que poner en “Object” el objeto que deseamos crear y en “Parameter” las características del objeto, como son el número de aristas o las dimensiones.

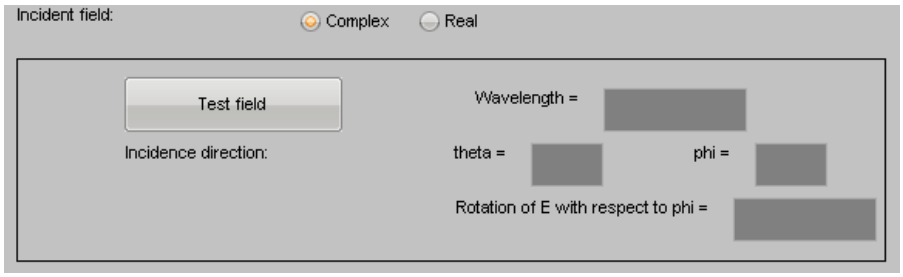


Figura 3.3. Características campo incidente

En la Figura 3.3 especificamos las características del campo incidente: la longitud de onda, los ángulos en coordenadas esféricas que definen la dirección de propagación de la onda incidente y la dirección de polarización del campo eléctrico incidente.

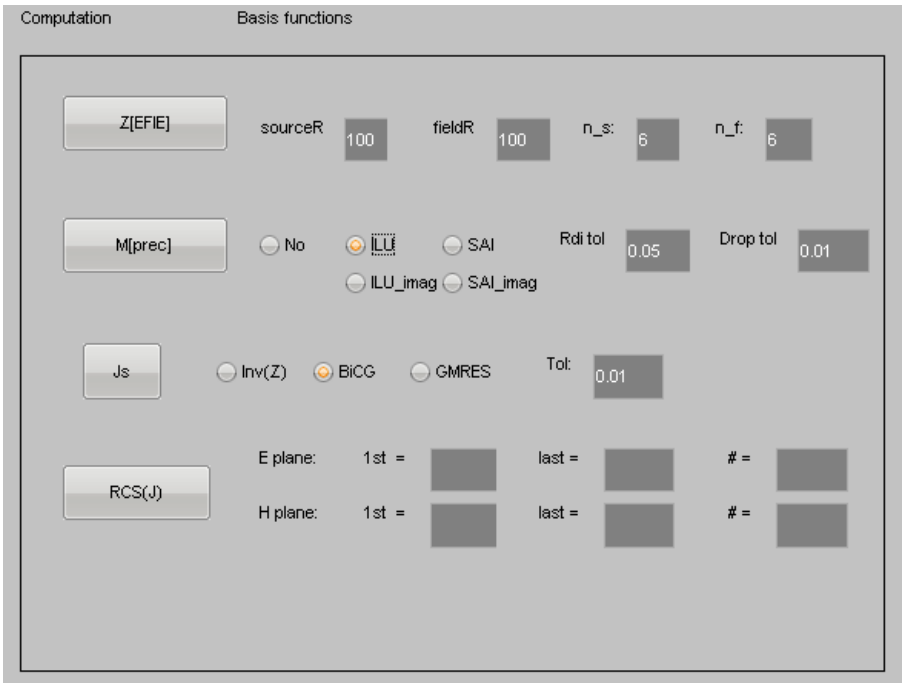


Figura 3.4. Funciones básicas de análisis numérico

En la Figura 3.4. aparece la pantalla con los detalles de las funciones para calcular la matriz de impedancias, seleccionar el precondicionador según si es *ILU* o *SAI*, obtener la corriente en el objeto a través de la inversión directa o mediante el logaritmo iterativo *BICG* y calcular la *RCS* (Radar Cross Section). La RCS indica la capacidad de un objeto para reflejar o dispersar señales radar incidentes en una dirección determinada del espacio.

En el siguiente capitulo el de resultados Prácticos, usaremos un término para medir el error entre vectores de corriente que denominamos EQM (error cuadrático medio en %). Nos indica el error cuadrático que estamos cometiendo en la corriente con el precondicionador y el algoritmo iterativo respecto del cálculo directo de la inversa de la matriz *Z*.

$$EQM = 100 \frac{\sqrt{\sum |J - Jinv|^2}}{\sqrt{\sum |Jinv|^2}} \tag{3.1}$$

donde *J* es la corriente calculada mediante el precondicionador y el algoritmo iterativo y *Jinv* es la corriente obtenida con la inversión directa de *Z*.

El EQM está directamente relacionado con la tolerancia del algoritmo iterativo, cuanto más baja sea la tolerancia menor será el error respecto a la inversión directa, en nuestro caso hemos escogido tolerancia (*tol*) 0,01.

CAPÍTULO 4. RESULTADOS

A continuación mostramos algunos de los resultados prácticos que hemos obtenido. Se han escogido los resultados y objetos que hemos considerado más significativos para explicar el funcionamiento de los precondicionadores.

Hemos realizado las pruebas en un ordenador con sistema operativo Microsoft Windows XP, un procesador AMD Athlon64 X2 Dual Core Processor 2,41GHz y una memoria RAM de 3,5 Gbyte.

Consideramos objetos eléctricamente pequeños aquellos en los que la dimensión máxima del objeto es menor a $\lambda/10$.

Se han dividido los resultados según el objeto y las dimensiones eléctricas, tal y como se muestra a continuación:

4.1.Objeto:Esfera

4.1.1.Dimensiones eléctricas moderadas
- Esfera mallada con 768 aristas
- Esfera mallada con 3072 aristas

4.1.2.Pequeñas dimensiones eléctricas
- Esfera mallada con 768 aristas
- Esfera mallada con 3072 aristas

4.2.Objeto:Cubo

4.2.1.Moderadas dimensiones eléctricas
- Cubo mallado con 882 aristas

4.2.2.Pequeñas dimensiones eléctricas
- Cubo mallado con 882 aristas

4.3.Objeto:Split Ring Resonator (SRR) en resonancia

- SRR
- Agrupación de 4 SRR's

Observaciones:

- El tiempo de cálculo del precondicionador *ILU* es menor que el del precondicionador *SAI Imaginario,* que a su vez es menor que el del precondicionador *SAI.* Podemos observar este comportamiento, para

todas las geometrías analizadas, en las tablas: (4.4) vs (4.5) vs (4.6), (4.13) vs (4.14) vs (4.15), (4.21) vs (4.22) vs (4.23), (4.30) vs (4.31) vs (4.32),(4.38) vs (4.39) vs (4.40), (4.47) vs (4.48) vs (4.49), (4.56) vs (4.57) vs (4.58),(4.65) vs (4.66) vs (4.67).

- En general en el precondicionador *ILU*, cuanto más pequeño hacemos *Rdi_tol* y *Drop_tol*, más aumenta el número de iteraciones y el tiempo en converger. Podemos observar este comportamiento en las tablas: (4.1), (4.10), (4.18), (4.27), (4.35), (4.44), (4.53), (4.62).

- Siempre observamos que en los precondicionadores *SAI*, cuanto más disminuimos *Rdi_row* y *Rdi_col*, más aumenta el número de iteraciones, en contraposición con el tiempo, que, a diferencia del precondicionador *ILU*, disminuye cada vez más. Podemos observar este comportamiento en las tablas: (4.2), (4.3), (4.11), (4.12), (4.19), (4.20), (4.28), (4.29), (4.36), (4.37), (4.45), (4.46), (4.54), (4.55), (4.63), (4.64). Este comportamiento se debe a que al dispersar el precondicionador *ILU* el número de iteraciones aumenta mucho más rápido que al dispersar los precondicionadores *SAI*.

- Cuando trabajamos con objetos eléctricamente pequeños, el *EQM* de la corriente obtenida en los precondicionadores *SAI* es el mismo (ver Tablas (4.25) vs (4.26) y (4.51) vs (4.52)). También hemos observado este comportamiento en los objetos que trabajan con *SRR´s.* Ver Tablas (4.60) vs (4.61) y (4.69) vs (4.70).

- Cuando trabajamos con objetos eléctricamente pequeños, el precondicionador *ILU* al converger produce corrientes con EQM muy elevado (ver tablas (4.24), (4.33), (4.50)).

- Cuando trabajamos con objetos eléctricamente pequeños, el precondicionador *SAI* y *SAI imaginario* se parecen mucho en el número de iteraciones. Podemos observar este comportamiento en las tablas: (4.19) vs (4.20), (4.28) vs (4.29), (4.45) vs (4.46).

- Asimismo, cuando trabajamos con objetos eléctricamente moderados, los precondicionadores *SAI* y *SAI imaginario* no se parecen tanto en el número de iteraciones como cuando trabajamos con objetos eléctricamente pequeños. Podemos observar este comportamiento en las tablas:(4.11) vs (4.12),(4.36) vs (4.37)

A continuación mostramos las Tablas de resultados para cada objeto.

4.1.Objeto: Esfera

4.1.1. Dimensiones eléctricas moderadas

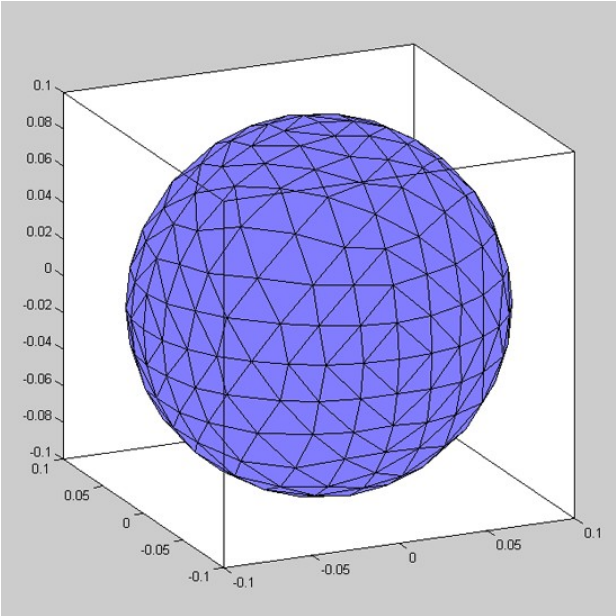


Figura 4.1. Objeto geometría esférica con radio 0,1 m y 768 aristas. λ=1m

Rdi_tol	**0,2**	**0,18**	**0,15**	**0,1**	**0,075**	**0,05**	**0,15**	**0,1**	**0,075**	**0,05**
Drop_tol	**0**	**0**	**0**	**0**	**0**	**0**	**0,001**	**0,001**	**0,001**	**0,001**
iteraciones	1	3	3	10	27	44	9	30	168	164
Tiempo it (seg)	0,094	0,25	0,25	0,797	2,031	3,203	0,297	0,844	5	4,6099

Tabla 4.1. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *ILU*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_t*ol y *Drop_tol*, el objeto analizado es el de la Fig.4.1

Rdi_row	**0,1**	**0,075**	**0,05**	**0,1**	**0,075**	**0,05**
Rdi_col	**max**	**max**	**max**	**0,125**	**0,1**	**0,075**
iteraciones	4	4	6	4	4	6
Tiempo it (seg)	4,485	2,578	1,75	4,562	2,594	1,797

Tabla 4.2. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.1

Rdi_row	0,1	0,075	0,05	0,1	0,075	0,05
Rdi_col	max	max	max	0,125	0,1	0,075
iteraciones	4	4	6	4	4	6
Tiempo it (seg)	4,343	2,5	1,703	4,313	2,5	1,703

Tabla 4.3. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI imaginario*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.1

Rdi_tol	0,2	0,18	0,15	0,1	0,075	0,05	0,15	0,1	0,075	0,05
Drop_tol	0	0	0	0	0	0	0,001	0,001	0,001	0,001
Tiempo cálculo prec.	2,172	2,146	2,162	2,165	2,163	2,17	2,055	1,997	1,972	1,95

Tabla 4.4. Tabla que contiene el tiempo de cálculo del precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.1

Rdi_row	0,1	0,075	0,05	0,1	0,075	0,05
Rdi_col	max	max	max	0,125	0,1	0,075
Tiempo cálculo prec.	165,055	65,167	15,85	89,353	25,97	6,299

Tabla 4.5. Tabla que contiene el tiempo de cálculo del precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.1

Rdi_row	0,1	0,075	0,05	0,1	0,075	0,05
Rdi_col	max	max	max	0,125	0,1	0,075
Tiempo cálculo prec.	45,611	19,075	6	14,524	5,798	2,8

Tabla 4.6. Tabla que contiene el tiempo de cálculo del precondicionador *SAI imaginario* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.1

Rdi_tol	**0,2**	**0,18**	**0,15**	**0,1**	**0,075**	**0,05**	**0,15**	**0,1**	**0,075**	**0,05**
Drop_tol	**0**	**0**	**0**	**0**	**0**	**0**	**0,001**	**0,001**	**0,001**	**0,001**
EQM para tol=0.01	1,28e-11	0,1084	0,979	1,765	5,460	2,267	3,985	0,507	18,975	4,413

Tabla 4.7. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador I*LU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.1

Rdi_row	**0,1**	**0,075**	**0,05**	**0,1**	**0,075**	**0,05**
Rdi_col	**max**	**max**	**max**	**0,125**	**0,1**	**0,075**
EQM para tol=0.01	0,0366	0,207	0,07	0,0376	0,327	0,139

Tabla 4.8. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.1

Rdi_row	**0,1**	**0,075**	**0,05**	**0,1**	**0,075**	**0,05**
Rdi_col	**max**	**max**	**max**	**0,125**	**0,1**	**0,075**
EQM para tol=0.01	0,02481	0,277	0,14	0,0260	0,323	0,14

Tabla 4.9. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI imaginario* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.1

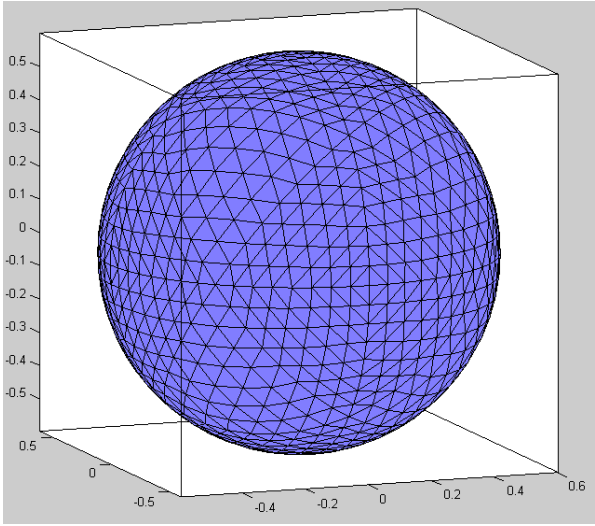


Figura 4.2. Objeto geometría espera radio 0,6
3072 vertices. λ= 1m

Rdi_tol	**1**	**0,8**	**0,5**	**0,2**	**1**	**0,8**
Drop_tol	**0**	**0**	**0**	**0**	**0,01**	**0,01**
iteraciones	6	8	8	12	28	30
Tiempo it (seg)	12,567	16,54	15,672	19,39	8,412	8

Tabla 4.10. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *ILU*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_tol* y *Drop_tol*, el objeto analizado es el de la Fig.4.2.

Rdi_row	**0,4**	**0,3**	**0,2**
Rdi_col	**0,5**	**0,4**	**0,3**
iteraciones	8	9	13
Tiempo it (seg)	280,359	171,14	111,5

Tabla 4.11. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.2

Rdi_row	**0,4**	**0,3**	**0,2**
Rdi_col	**0,5**	**0,4**	**0,3**
iteraciones	12	14	20
Tiempo it (seg)	429,937	281,547	186,357

Tabla 4.12. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI imaginario*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.2

Rdi_tol	**1**	**0,8**	**0,5**	**0,2**	**1**	**0,8**
Drop_tol	**0**	**0**	**0**	**0**	**0,01**	**0,01**
Tiempo cálculo prec.	115,35	116	107,684	244,97	12	10,32

Tabla 4.13. Tabla que contiene el tiempo de cálculo del precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.2

Rdi_row	**0,4**	**0,3**	**0,2**
Rdi_col	**0,5**	**0,4**	**0,3**
Tiempo cálculo prec.	3375,52	1123,04	357,583

Tabla 4.14. Tabla que contiene el tiempo de cálculo del precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.2

Rdi_row	**0,4**	**0,3**	**0,2**
Rdi_col	**0,5**	**0,4**	**0,3**
Tiempo cálculo prec.	367,616	144,277	43,356

Tabla 4.15. Tabla que contiene el tiempo de cálculo del precondicionador *SAI imaginario* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.2

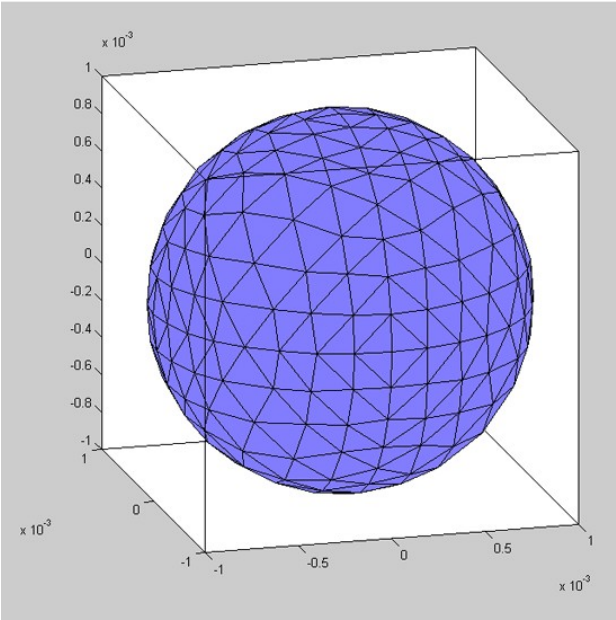
Rdi_tol	**1**	**0,8**	**0,5**	**0,2**	**1**	**0,8**
Drop_tol	**0**	**0**	**0**	**0**	**0,01**	**0,01**
EQM para tol=0,01	0,347	0,417	0,688	1,228	1,978	1,023

Tabla 4.16. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.2

Rdi_row	**0,4**	**0,3**	**0,2**
Rdi_col	**0,5**	**0,4**	**0,3**
EQM para tol=0.01	0,647	0,851	0,448

Tabla 4.17. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.2

4.1.2. Dimensiones eléctricas pequeñas



Figur4.3. Objeto geometría esférica con radio 0,001 m y 768 vertices. λ= 1m

Rdi_tol	**0,002**	**0,0018**	**0,0015**	**0,001**	**0,00075**	**0,0005**	**0,0015**	**0,001**	**0,00075**	**0,0005**
Drop_tol	**0**	**0**	**0**	**0**	**0**	**0**	**0,001**	**0,001**	**0,001**	**0,001**
iteraciones	1	4	52	35	8	21	1	1	3	510
Tiempo it (seg)	0,078	0,344	4,266	2,766	0,593	1,516	0,047	0,047	0,093	14,203

Tabla 4.18. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *ILU*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_tol y Drop_tol*, el objeto analizado es el de la Fig.4.3

Rdi_row	**0,001**	**0,00075**	**0,0005**	**0,001**	**0,00075**	**0,0005**
Rdi_col	**max**	**max**	**max**	**0,00125**	**0,001**	**0,00075**
iteraciones	2	2	4	2	2	4
Tiempo it (seg)	2,687	1,531	1,25	2,687	1,532	1,25

Tabla 4.19. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.3

Rdi_row	0,001	0,00075	0,0005	0,001	0,00075	0,0005
Rdi_col	max	max	max	0,00125	0,001	0,00075
iteraciones	2	2	4	2	2	4
Tiempo it (seg)	2,578	1,5	1,219	2,578	1,5	1,219

Tabla 4.20. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI Imaginario*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.3

Rdi_tol	0,002	0,0018	0,0015	0,001	0,00075	0,0005	0,0015	0,001	0,00075	0,0005
Drop_tol	0	0	0	0	0	0	0,001	0,001	0,001	0,001
Tiempo cálculo prec.	0,855	0,91	0,760	0,938	0,6781	0,658	0,867	0,840	0,815	0,765

Tabla 4.21. Tabla que contiene el tiempo de cálculo del precondicionador *ILU* para unos ciertos valores de *Rdi_tol y Drop_tol*,el objeto analizado es el de la Fig.4.3

Rdi_row	0,001	0,00075	0,0005	0,001	0,00075	0,0005
Rdi_col	max	max	max	0,00125	0,001	0,00075
Tiempo cálculo prec.	165,327	65,413	15,83	89,743	26,338	6,465

Tabla 4.22. Tabla que contiene el tiempo de cálculo del precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.3

Rdi_row	0,001	0,00075	0,0005	0,001	0,00075	0,0005
Rdi_col	max	max	max	0,00125	0,001	0,00075
Tiempo cálculo prec.	45,505	18,405	5,893	14,576	5,784	2,824

Tabla 4.23. Tabla que contiene el tiempo de cálculo del precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.3

Rdi_tol	0,002	0,0018	0,0015	0,001	0,00075	0,0005	0,0015	0,001	0,00075	0,0005
Drop_tol	0	0	0	0	0	0	0,001	0,001	0,001	0,001
EQM para tol=0.01	1,12e-7	99,902	20,77	61,752	102,92	90,766	683,78	237,07	122,99	255,24

Tabla 4.24. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.3

Rdi_row	0,001	0,00075	0,0005	0,001	0,00075	0,0005
Rdi_col	max	max	max	0,00125	0,001	0,00075
EQM para tol=0.01	0,0976	0,274	0,0409	0,103	0,312	0,0429

Tabla 4.25. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.3

Rdi_row	0,001	0,00075	0,0005	0,001	0,00075	0,0005
Rdi_col	max	max	max	0,00125	0,001	0,00075
EQM para tol=0.01	0,0976	0,274	0,0409	0,103	0,312	0,0429

Tabla 4.26. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.3

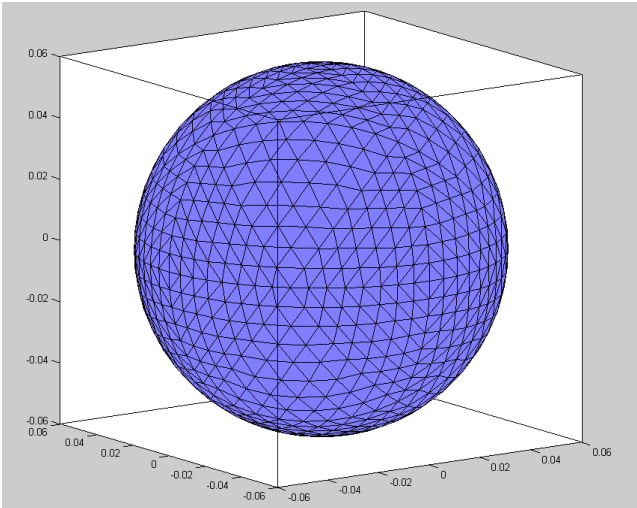


Figura 4.4. Objeto geometría espera radio 0,06 m
3072 vertices. λ= 1m

Rdi_tol	**0,12**	**0,1**	**0,08**	**0,05**	**0,02**	**0,12**	**0,1**	**0,08**
Drop_tol	**0**	**0**	**0**	**0**	**0**	**0,01**	**0,01**	**0,01**
iteraciones	1	3	11	192	111	1	1	1
Tiempo it (seg)	2,078	6,078	22	370,531	176,453	0,485	0,453	0.422

Tabla 4.27. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *ILU*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_tol* y *Drop_tol*, el objeto analizado es el de la Fig.4.4

Rdi_row	**0,04**	**0,03**	**0,02**
Rdi_col	**0,05**	**0,04**	**0,03**
iteraciones	3	5	7
Tiempo it (seg)	123,766	102,515	63,422

Tabla 4.28 Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.4

Rdi_row	**0,04**	**0,03**	**0,02**
Rdi_col	**0,05**	**0,04**	**0,03**
iteraciones	3	5	7
Tiempo it (seg)	140,36	112,22	71,094

Tabla 4.29. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI imaginario*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.4

Rdi_tol	**0,12**	**0,1**	**0,08**	**0,05**	**0,02**	**0,12**	**0,1**	**0,08**
Drop_tol	**0**	**0**	**0**	**0**	**0**	**0,01**	**0,01**	**0,01**
Tiempo cálculo prec.	113,721	113	126,375	105,833	242,491	19,074	13,234	14,426

Tabla 4.30. Tabla que contiene el tiempo de cálculo del precondicionador *ILU* para unos ciertos valores de *Rdi_tol y Drop_tol*,el objeto analizado es el de la Fig.4.4

Rdi_row	**0,04**	**0,03**	**0,02**
Rdi_col	**0,05**	**0,04**	**0,03**
Tiempo cálculo prec.	3400	1117,16	353,171

Tabla 4.31. Tabla que contiene el tiempo de cálculo del precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.4

Rdi_row	**0,04**	**0,03**	**0,02**
Rdi_col	**0,05**	**0,04**	**0,03**
Tiempo cálculo prec.	375,39	151,236	191,372

Tabla 4.32. Tabla que contiene el tiempo de cálculo del precondicionador *SAI Imaginario* para unos ciertos valores de Rdi_row y Rdi_tol,el objeto analizado es el de la Fig.4.4

Rdi_tol	**0,12**	**0,1**	**0,08**	**0,05**	**0,02**	**0,12**	**0,1**	**0,08**
Drop_tol	**0**	**0**	**0**	**0**	**0**	**0,01**	**0,01**	**0,01**
EQM para tol=0,01	2,487e-10	3,278	0,563	2,181	34,0135	192,735	244,207	339,471

Tabla 4.33. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.4

Rdi_row	**0,04**	**0,03**	**0,02**
Rdi_col	**0,05**	**0,04**	**0,03**
EQM para tol=0.01	0,773	0.349	0,159

Tabla 4.34. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.4

4.2.Objeto:Cubo

La onda plana incidente se propaga en una dirección perpendicular a una de las caras del cubo, con los campos eléctrico y magnético polarizados en dirección paralela a los lados de dicha cara.

4.2.1.Dimensiones eléctricas moderadas

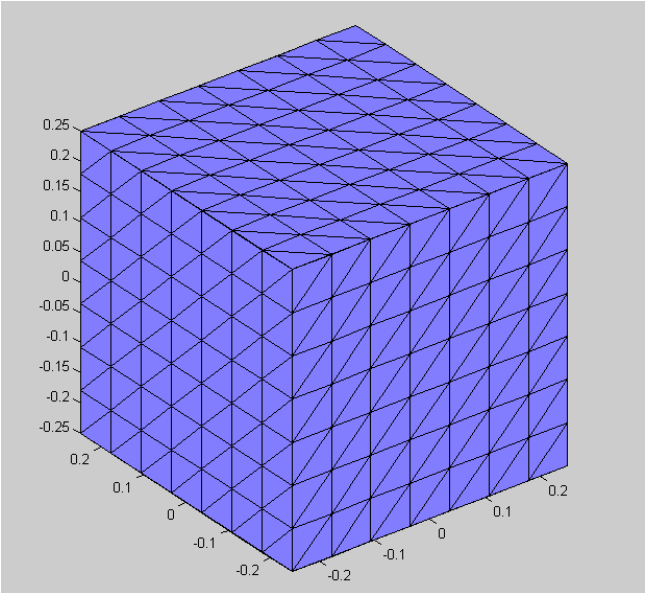


Figura 4.5 Objeto geometría prisma con longitud 0,5 m y 882 aristas. λ= 1m

Rdi_tol	**max**	**0,7**	**0,3**	**0,7**	**0,3**	**0,7**	**0,3**
Drop_tol	**0**	**0**	**0**	**0,001**	**0,001**	**0,01**	**0,01**
iteraciones	1	2	6	2	8	16	14
Tiempo it (seg)	0,125	0,235	0,485	0,219	0,515	0,641	0,406

Tabla 4.35. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *ILU*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_tol* y *Drop_tol*, el objeto analizado es el de la Fig.4.5

Rdi_row	0,25	0,2	0,15	0,1
Rdi_col	max	max	max	max
iteraciones	8	11	13	37
Tiempo it (seg)	7,281	6,359	4,391	6,015

Tabla 4.36. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI,* y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.5

Rdi_row	0,25	0,2	0,15	0,1
Rdi_col	max	max	max	max
iteraciones	15	13	17	53
Tiempo it (seg)	12,234	6,922	5,25	7,75

Tabla 4.37. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI Imaginario*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.5

Rdi_tol	max	0,7	0,3	0,7	0,3	0,7	0,3
Drop_tol	0	0	0	0,001	0,001	0,01	0,01
Tiempo cálculo prec.	3,138	3,114	2,831	4,1	1,732	1,132	1

Tabla 4.38. Tabla que contiene el tiempo de cálculo del precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.5

Rdi_row	0,25	0,2	0,15	0,1
Rdi_col	max	max	max	max
Tiempo cálculo prec.	108,168	48,489	17,333	4,948

Tabla 4.39. Tabla que contiene el tiempo de cálculo del precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.5

Rdi_row	0,25	0,2	0,15	0,1
Rdi_col	max	max	max	max
Tiempo cálculo prec.	34,338	15	7,263	4

Tabla 4.40.Tabla que contiene el tiempo de cálculo del precondicionador *SAI Imaginario* para unos ciertos valores de *Rdi_row y Rdi_tol*,el objeto analizado es el de la Fig.4.5

Rdi_tol	max	0,7	0,3	0,7	0,3	0,7	0,3
Drop_tol	0	0	0	0,001	0,001	0,01	0,01
EQM para tol=0.01	2,11e-12	0,218	0,694	0,772	0,360	1,131	1,044

Tabla 4.41. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *ILU* para unos ciertos valores de *Rdi_tol y Drop_tol*,el objeto analizado es el de la Fig.4.5

Rdi_row	0,25	0,2	0,15	0,1
Rdi_col	max	max	max	max
EQM para tol=0.01	0,403	0,238	0,265	1,147

Tabla 4.42. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI* para unos ciertos valores de *Rdi_row y Rdi_tol*,el objeto analizado es el de la Fig.4.5

Rdi_row	0,25	0,2	0,15	0,1
Rdi_col	max	max	max	max
EQM para tol=0.01	0,269	0,621	0,322	0,439

Tabla 4.43. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI Imaginario* para unos ciertos valores de *Rdi_row y Rdi_tol*,el objeto analizado es el de la Fig.4.5

4.2.2. Dimensiones eléctricas pequeñas

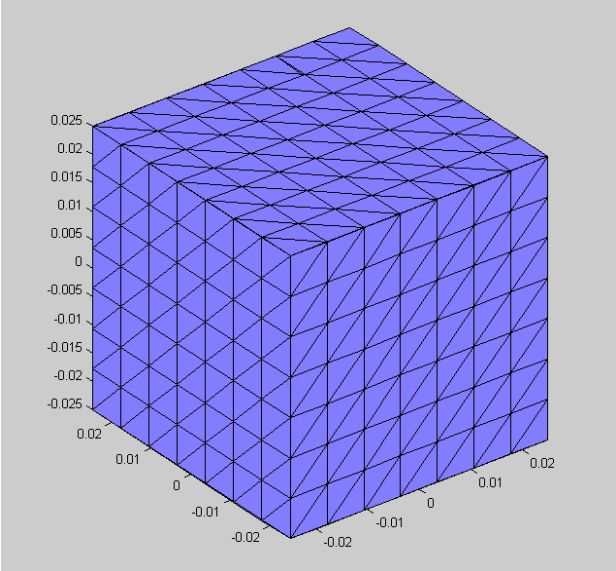


Figura 4.6. Objeto geometría prisma con longitud 0,05 m 882 vertices. λ= 1m

Rdi_tol	**max**	**0,07**	**0,06**	**0,07**	**0,07**
Drop_tol	**0**	**0**	**0**	**0,001**	**0,01**
iteraciones	1	4	35	4	1
Tiempo it (seg)	0,125	0,468	4,016	0,282	0,063

Tabla 4.44. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *ILU*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_to*l y *Drop_to*l, el objeto analizado es el de la Fig.4.6

Rdi_row	**0,025**	**0,02**	**0,015**	**0,01**
Rdi_col	**max**	**max**	**max**	**max**
iteraciones	4	5	6	12
Tiempo it (seg)	4,016	3,172	2,172	2,046

Tabla 4.45. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.6

Rdi_row	0,025	0,02	0,015	0,01
Rdi_col	max	max	max	max
iteraciones	4	5	6	12
Tiempo it (seg)	3,797	2,953	2,06	1,859

Tabla 4.46. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI imaginario*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_t*ol, el objeto analizado es el de la Fig.4.6

Rdi_tol	max	0,07	0,06	0,07	0,07
Drop_tol	0	0	0	0,001	0,01
Tiempo cálculo prec.	3,221	3,2	3,081	2,113	1,131

Tabla 4.47. Tabla que contiene el tiempo de cálculo del precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.6

Rdi_row	0,025	0,02	0,015	0,01
Rdi_col	max	max	max	max
Tiempo cálculo prec.	105,859	45,985	17,846	4,983

Tabla 4.48. Tabla que contiene el tiempo de cálculo del precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.6

Rdi_row	0,025	0,02	0,015	0,01
Rdi_col	max	max	max	max
Tiempo cálculo prec.	34,3	15	7,306	3,947

Tabla 4.49.Tabla que contiene el tiempo de cálculo del precondicionador *SAI imaginario* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.6

Rdi_tol	max	0,07	0,06	0,07	0,07
Drop_tol	0	0	0	0,001	0,01
EQM para tol=0.01	1,64e-10	0,382	2,77	387,375	115,612

Tabla 4.50. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.6

Rdi_row	0,025	0,02	0,015	0,01
Rdi_col	max	max	max	max
EQM para tol=0.01	0,395	0,419	0,708	0,629

Tabla 4.51. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.6

Rdi_row	0,025	0,02	0,015	0,01
Rdi_col	max	max	max	max
EQM para tol=0.01	0,395	0,419	0,708	0,629

Tabla 4.52. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI Imaginario* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.6

4.3. Objeto:SRR en resonancia

El *Split Ring Resonator* (SRR) es una estructura formada por dos espiras rectangulares (ver Fig. 4.7). En la práctica, el SRR se utiliza para la fabricación de materiales artificiales. A la frecuencia de funcionamiento, el SRR debe estar en resonancia. La frecuencia de resonancia es aquélla en que el SRR dispersa mayor campo. La dirección de la onda incidente debe ser tal que induzca la mayor magnitud de corriente. Por tanto, la dirección de incidencia de la onda en el SRR, o en una agrupación de SRR's, es rasante al plano del SRR con el campo magnético orientado en dirección perpendicular al plano formado por las espiras rectangulares.

Encontramos la frecuencia de resonancia del objeto mediante la observación de las gráficas de plano E y plano H de la *RCS* (R*adar Cross Section*) a diferentes frecuencias (ver Fig. 4.8). Vemos que la frecuencia de resonancia se encuentra entre 3,5GHz y 6,5GHz.

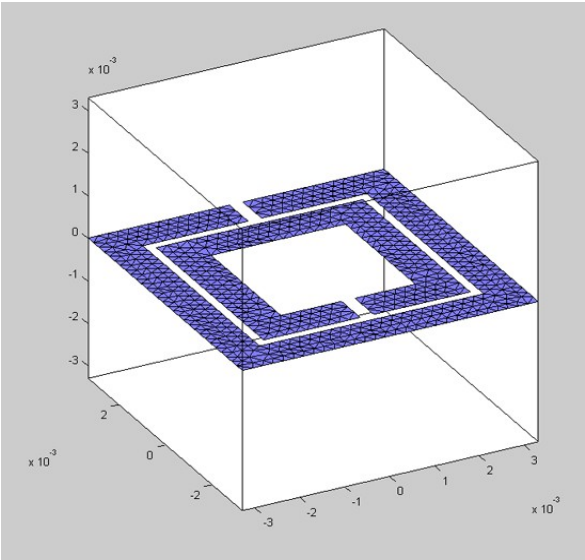


Figura 4.7. Objeto geometría SRR con longitud 0,0066 m y 1749 aristas. λ=0,0545 m

En el eje de ordenadas de las gráficas *RCS* tenemos la potencia en dBsm (*Decibels relative to a square meter*) y en el eje de abscisas, el ángulo de observación de la RCS respecto de la dirección de incidencia de la onda incidente. Como se puede observar que en la Figura 4.8, la frecuencia de resonancia es de 5,5 GHz.

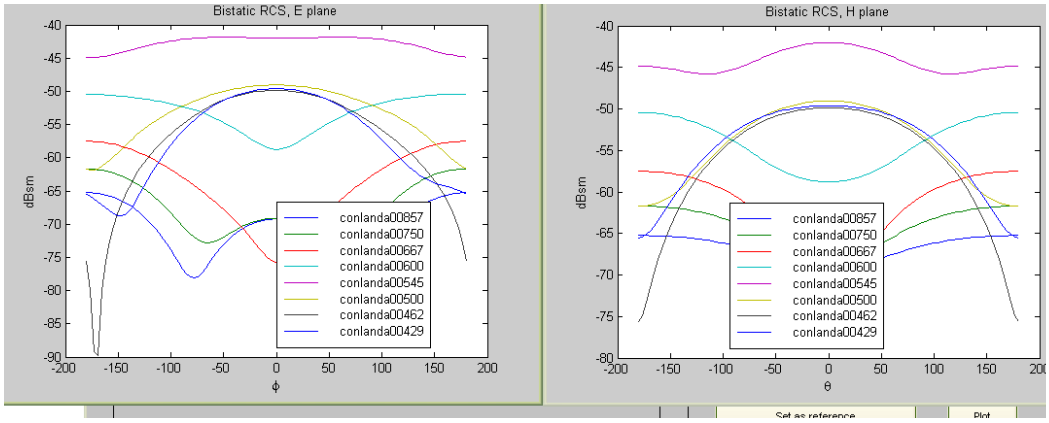


Figura 4.8. *RCS* para una frecuencia de 6,5GHz o longitud de onda 0,0462

Rdi_tol	**0,006**	**0,005**	**0,004**	**0,003**	**0,006**	**0,005**	**0,004**	**0,003**
Drop_tol	**0**	**0**	**0**	**0**	**0,01**	**0,01**	**0,01**	**0,01**
iteraciones	7	5	34	28	1	1	2	inf
Tiempo it (seg)	3,219	2,234	14,859	12,188	0,109	0,11	0,172	inf

Tabla 4.53. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *ILU*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_tol* y *Drop_tol*, el objeto analizado es el de la Fig.4.7.

Rdi_row	**0,002**	**0,0015**	**0,001**
Rdi_col	**0,003**	**0,003**	**0,003**
iteraciones	16	20	27
Tiempo it (seg)	188,89	149,735	94,594

Tabla 4.54. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.7.

Rdi_row	**0,002**	**0,0015**	**0,001**
Rdi_col	**0,003**	**0,003**	**0,003**
iteraciones	16	20	27
Tiempo it (seg)	148,031	119,656	78,5

Tabla 4.55. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI Imaginario*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.7.

Rdi_tol	**0,006**	**0,005**	**0,004**	**0,003**	**0,006**	**0,005**	**0,004**	**0,003**
Drop_tol	**0**	**0**	**0**	**0**	**0,01**	**0,01**	**0,01**	**0,01**
Tiempo cálculo prec.	22,091	21,848	21,606	21,367	22,173	21,871	21,602	21,364

Tabla 4.56. Tabla que contiene el tiempo de cálculo del precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.7.

Rdi_row	**0,002**	**0,0015**	**0,001**
Rdi_col	**0,003**	**0,003**	**0,003**
Tiempo cálculo prec.	1350	582,094	150,844

Tabla 4.57. Tabla que contiene el tiempo de cálculo del precondicionador *SAI* para unos ciertos valores de *Rdi_row y Rdi_tol*,el objeto analizado es el de la Fig.4.7.

Rdi_row	**0,002**	**0,0015**	**0,001**
Rdi_col	**0,003**	**0,003**	**0,003**
Tiempo cálculo prec.	213,229	113,604	43,465

Tabla 4.58. Tabla que contiene el tiempo de cálculo del precondicionador *SAI Imaginario* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.7.

Rdi_tol	**0,006**	**0,005**	**0,004**	**0,003**	**0,006**	**0,005**	**0,004**	**0,003**
Drop_tol	**0**	**0**	**0**	**0**	**0,01**	**0,01**	**0,01**	**0,01**
EQM para tol=0,01	0,397	0.116	0,0169	0.0572	100.917	100.031	100.043 3	100.747

Tabla 4.59. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.7.

Rdi_row	**0,002**	**0,0015**	**0,001**
Rdi_col	**0,003**	**0,003**	**0,003**
EQM para tol=0.01	0,0353	0,0451	0,0435

Tabla 4.60. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI* para unos ciertos valores de *Rdi_row y Rdi_tol*,el objeto analizado es el de la Fig.4.7.

Rdi_row	**0,002**	**0,0015**	**0,001**
Rdi_col	**0,003**	**0,003**	**0,003**
EQM para tol=0.01	0,0354	0,0416	0,0434

Tabla 4.61. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI Imaginario* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.7.

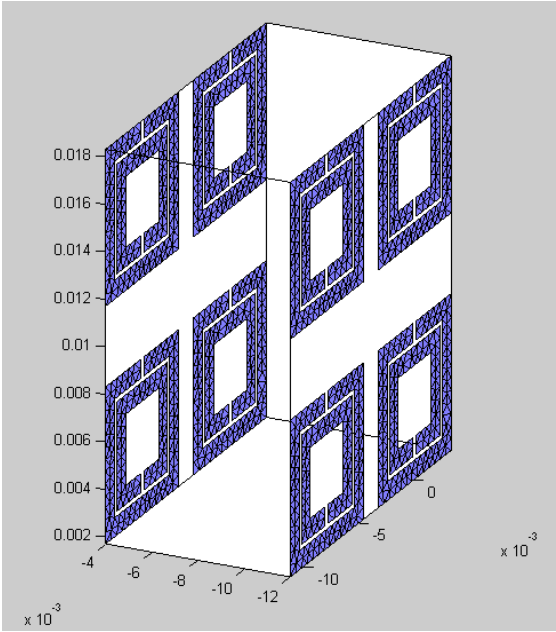


Figura 4.9. Objeto formado por una agrupación de 4 SRR mallado con un total de 1749 aristas

Rdi_tol	**0,01**	**0,0075**	**0,005**	**0,003**	**0,01**	**0,0075**	**0,005**
Drop_tol	**0**	**0**	**0**	**0**	**0,001**	**0,001**	**0,001**
iteraciones	3	3	9	96	79	82	113
Tiempo it (seg)	4,407	1,765	3,938	26,234	17,969	16,39	21,359

Tabla 4.62. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *ILU*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_tol y Drop_tol*, el objeto analizado es el de la Fig.4.9.

Rdi_row	**0,005**	**0,003**	**0,001**
Rdi_col	**0,007**	**0,005**	**0,003**
iteraciones	10	21	44
Tiempo it (seg)	230,61	190,063	86,313

Tabla 4.63. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.9.

Rdi_row	**0,005**	**0,003**	**0,001**
Rdi_col	**0,007**	**0,005**	**0,003**
iteraciones	10	21	44
Tiempo it (seg)	195,719	170,313	82,892

Tabla 4.64. Tabla que contiene el número de iteraciones y el tiempo total, para un precondicionador *SAI Imaginario*, y un algoritmo iterativo de tolerancia 0,01 respecto a unos valores *Rdi_row* y *Rdi_tol*, el objeto analizado es el de la Fig.4.9.

Rdi_tol	**0,01**	**0,0075**	**0,005**	**0,003**	**0,01**	**0,0075**	**0,005**
Drop_tol	**0**	**0**	**0**	**0**	**0,001**	**0,001**	**0,001**
Tiempo cálculo prec.	74	65,26	10	4,653	8,1	3,7	4

Tabla 4.65. Tabla que contiene el tiempo de cálculo del precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.9.

Rdi_row	**0,005**	**0,003**	**0,001**
Rdi_col	**0,007**	**0,005**	**0,003**
Tiempo cálculo prec.	1279,57	200	38,89

Tabla 4.66. Tabla que contiene el tiempo de cálculo del precondicionador *SAI* para unos ciertos valores de *Rdi_row y Rdi_tol*,el objeto analizado es el de la Fig.4.9.

Rdi_row	**0,005**	**0,003**	**0,001**
Rdi_col	**0,007**	**0,005**	**0,003**
Tiempo cálculo prec.	264.125	71,236	41,789

Tabla 4.67. Tabla que contiene el tiempo de cálculo del precondicionador *SAI Imaginario* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.9.

Rdi_tol	0,01	0,0075	0,005	0,003	0,01	0,0075	0,005
Drop_tol	0	0	0	0	0,001	0,001	0,001
EQM para tol=0,01	0,882	0,796	0,285	1,59	0,484	0,851	1,063

Tabla 4.68. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *ILU* para unos ciertos valores de *Rdi_tol* y *Drop_tol*,el objeto analizado es el de la Fig.4.9

Rdi_row	0,005	0,003	0,001
Rdi_col	0,007	0,005	0,003
EQM para tol=0.01	0,522	0,973	1,09

Tabla 4.69. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.9

Rdi_row	0,005	0,003	0,001
Rdi_col	0,007	0,005	0,003
EQM para tol=0.01	0,522	0,973	1,089

Tabla 4.70. Tabla que contiene el error cuadrático medio para una tolerancia del algoritmo iterativo de 0,01 en un precondicionador *SAI Imaginario* para unos ciertos valores de *Rdi_row* y *Rdi_tol*,el objeto analizado es el de la Fig.4.9

CAPÍTULO 5. CONCLUSIONES

(1) El precondicionador *SAI* requiere más tiempo para ser calculado que el precondicionador *ILU*.

(2) El tiempo de cálculo del precondicionador *SAI imaginario* es mucho más rápido que el precondicionador *SAI* pero aun así no es tan rápido como el precondicionador *ILU*. Sin embargo, ambos precondicionadores *SAI*, a diferencia del precondicionador *ILU*, son muy fácilmente paralelizables.

(3) El número de iteraciones de convergencia asociadas al precondicionador *SAI imaginario* se parecen más a las del precondicionador *SAI* cuanto más pequeñas eléctricamente son las dimensiones eléctricas del objeto analizado.

(4) El precondicionador *SAI* es más robusto que el precondicionador *ILU* al grado de dispersión de la matriz.

(5) Precondicionador es poco fiable en objetos eléctricamente pequeños, ya que muchas veces converge pero con un EQM muy grande.

(6).El precondicionador *ILU* es muy sensible a los valores de Drop_tol y mucho más cuando trabajamos con objetos eléctricamente pequeños.

La conclusión final a la que hemos llegado es que el *SAI* es un buen precondicionador, que puede ser mejorado a través del precondicionador *SAI Imaginario*. Es muy robusto ya sea por que puede trabajar con matrices muy dispersas o ya sea por que no se ve afectado negativamente por las dimensiones eléctricas del objeto. Y su único punto débil, que sería el tiempo que tarda en calcular el precondicionador, puede solucionarse si lo paralelizamos.

Bibliografía

[1] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996

[2] T. *Huckle, Approximate Inverse Preconditioning and Parallel Computing*. http://www.aueb.gr/pympe/hercma/proceedings2005/H05-ABSTRACTS-1/HUCKLE-2.pdf

[3] G. Alléon, M.Benzi and L.Giraud, *Sparse Approximate Inverse Preconditioning for Dense Linear Systems Arising in Computational Electromagnetics*,http://www.google.com/search?sourceid=navclient&ie=UTF-8&rls=GGLR,GGLR:2006-19,GGLR:en&q=Sparse+Approximate+Preconditioning+for+Dense+Linear+Systems+Arising+in+Computational+Electromagnetics

[4] A. Cardama, L. Jofre, J. M. Rius, J. Romeu, S. Blanch, *Antenas*, Edicions UPC, Barcelona, 1998

[5] S. M. Rao, D. R. Wilton and A. W. Glisson, *Electromagnetic Scattering by surfaces of Arbitrary Shape*, IEEE Transactions on Antennas and Propagation, vol. AP-30, No. 3, pp. 409-418, May 1982