industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Estudio de sistemas de resolución paralelos multigrid geométricos y algebraicos en problemas de análisis estructural y optimización topológica.

TRABAJO FIN DE MASTER

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

Autor: Miguel Ángel Gómez Nadal
Director: David Herrero Pérez

Cartagena, 2021

Universidad Politécnica de Cartagena

Resumen

En la actualidad, existen problemas tecnológicos que impiden incrementar la eficiencia de los procesadores, entre los que cabe destacar las limitaciones en la frecuencia a la que pueden operar por problemas de disipación térmica. Debido a esta problemática, los fabricantes optan por utilizar arquitecturas multinúcleo para seguir incrementando las prestaciones computacionales. Las ventajas que este cambio de paradigma ofrece son obvias, reduciendo considerablemente el tiempo de computación en problemas complejos que serían inabordables desde el punto de vista de la computación serializada. Si bien, es cierto que, utilizar los recursos computacionales de forma adecuada es un aspecto clave, también lo es el método de resolución utilizado. Por ello, se presenta una evaluación de la implementación en paralelo de los métodos de resolución multigrid, ya que estas técnicas son las más eficientes para resolver problemas de cierta complejidad en ingeniería, sin ningún competidor existente. En este proyecto, se estudian los métodos multigrid algebraicos y geométricos como precondicionadores de solvers iterativos, así como las técnicas existentes para su paralelización, enfocándose en las ventajas y desventajas que estos métodos ofrecen y realizando simulaciones enfocadas al análisis estructural y la optimización topológica.

Abstract

Currently, there are technological problems that do not allow increase the processors' efficiency and we can highlight the limitations on the frequency they can operate heat dissipation problems. Due of this issue, manufacturers decide to use multicore architectures to keep increasing computing performance. The advantages of this paradigm shift are obvious, the computational time in complex problems that would be intractable from the serialized computing perspective could be considerably reduced. Even though using computing resources properly is a key aspect, the resolution method we use is also important. Thereupon, we present an evaluation of the parallel implementation of multigrid resolution methods, since these techniques are the most efficients to solve complex problems in engineering and do not have competitors. In this project, algebraic and geometric multigrid methods are studied as preconditioners of iterative solvers. We also study the existent techniques for its parallelization, focusing in the advantages and desadvantages that these methods offer and doing structural analysis and topological optimization simulations.

ÍNDICE

TABLA DE FIGURAS

TABLA DE GRÁFICAS

1 Introducción

Realizar simulaciones de forma eficiente es un aspecto clave en ingeniería. El aumento de la complejidad de dichas simulaciones requiere el uso de técnicas computacionales que respondan a la necesidad de aumento de eficiencia exigida.

En 1965 Gordon Moore realizó una predicción en la que exponía que el número de transistores utilizados en un microprocesador se duplicaría cada año. En 1975 corrigió esta predicción, enunciando como el número de transistores se duplicaría cada dos años. Esta predicción está asociada a un crecimiento exponencial de la velocidad de procesamiento obtenida y sugiriere una disminución de los costos. Dicha predicción es la llamada Ley de Moore.

La predicción se ha asemejado con bastante fidelidad al aumento de las prestaciones obtenidas. Esta fue enunciada para dos décadas, pero sigue presente hoy en día y ha marcado el progreso tecnológico seguido en la actualidad. Sin embargo, el propio Moore afirma en una entrevista en 2005 que su propia ley tiene un límite, ya que, existe una limitación natural. El tamaño de los transistores difícilmente se puede seguir reduciendo, ya que, se está alcanzando el tamaño del átomo, dificultando el crecimiento exponencial al que estamos acostumbrados. La Ley de Moore está llegando a su fin. Se necesita un cambio en el paradigma de computación para seguir aumentando las prestaciones en un presente y futuro próximo.

Además, existe cierta problemática que impide a los procesadores seguir aumentando la velocidad de procesamiento con la que estos operan, explicando la tendencia de los fabricantes a elaborar infraestructuras de cálculo de altas prestaciones basadas en arquitecturas multinúcleo. La imposibilidad de seguir aumentando el rendimiento de los procesadores se debe a una limitación conocida como Power Wall. En la que, por motivos de disipación térmica, la frecuencia a la que traban los microprocesadores queda limitada. Dicha limitación se alcanzó en el siglo xxi, produciendo la incapacidad de seguir incrementando de forma exponencial la velocidad de operación de los procesadores, como había estado sucediendo hasta dicho momento. Limitación que se alcanza en torno a los 5 GHz [1]. Además, existe otra limitación denominada memory Wall [2]. Esta se alcanzó en el siglo XX, implicando que las mejoras de los procesadores alcanzadas, quedan inutilizadas, ya que, estas sobrepasan las velocidades de acceso a memoria. Quedando los procesadores a la espera de acceder a los datos y haciendo inútiles toda mejora de las prestaciones alcanzada.

La computación en paralelo se presenta como una solución a la problemática planteada. Planteándose un cambio en el paradigma de computación, en el que, los fabricantes optan por utilizar arquitecturas multinúcleo para seguir aumentando las prestaciones en lugar de mejorar las características de los procesadores individualmente. La computación en paralelo permite dividir un problema complejo entre los distintos procesadores. Otorgando de esta forma un aumento de eficiencia y permitiendo resolver en ingeniería problemas que serían inabordables

utilizando computación tradicional, como es el caso de resolver problemas de optimización topológica.

Si bien, utilizar un método de computación que responda de manera eficiente es un aspecto sin duda importante, también lo es el método de resolución utilizado. Los métodos multigrid son los métodos de resolución más eficientes y se presentan como los métodos óptimos para resolver problemas complejos en ingeniería, sin ningún competidor existente. En este proyecto se estudiarán los métodos multigrid algebraicos y geométricos, estudiando su eficiencia tanto en sólidos 2D como 3D, y las técnicas existentes para su paralelización, realizando simulaciones enfocadas a la optimización topológica.

2 Análisis y optimización topológica en mecánica de sólidos.

2.1 Introducción

Optimizar la disposición geométrica de un cuerpo para que minimice o maximice cierta función objetivo es un problema muy común en ingeniería [3]. A diferencia de otros métodos de optimización, como son el de tamaño y el de forma, en la optimización topológica se realizan diseños conceptuales sin ninguna configuración estructural previa. Esto hace que la optimización topológica sea utilizada para alcanzar prototipos de alto rendimiento en las fases iniciales de diseño. [11]

La optimización topológica tiene como objetivo modificar la geometría de un sólido dentro de unas restricciones, realizando un aligeramiento masivo de su estructura, conservando en su totalidad sus propiedades mecánicas. Produciendo que el material se presente sólo donde se necesite según su funcionamiento y sus propiedades. Consiguiendo de esta forma un aumento muy significativo del nivel de eficiencia de material y presentándose como una tecnología clave en cualquier ámbito relacionado con ingeniería aeroespacial, automoción y optimización estructural.

Estos métodos se pueden clasificar en tres grandes categorías:

- Métodos Eulerianos. Se obtiene el diseño mediante la definición implícita de su frontera.
- Métodos Larangianos. La creación de la malla se realiza a partir de un programa CAD. En estos métodos se realiza una representación explicita de la frontera.
- Métodos basados en densidades. Consisten en la minimización de una función objetivo. Discretizando el dominio a partir de una malla fija, por el método de elementos finitos, cuyo objetivo es encontrar el material óptimo (lleno/vacío) Existen distintos métodos basados en densidades entre los que se pueden destacar el método de homogenización y el método SIMP (Solid Isotropic Material with Penalization). Este último es el que se va a implementar en este proyecto, debido a que es el más usado en los programas comerciales, probablemente por su simplicidad frente a otros métodos de optimización topológica [11].

En la optimización topológica se dan como variables conocidas el volumen final de la estructura, las condiciones de apoyo, las cargas aplicadas y restricciones adicionales, como zonas donde no se precisa que exista material. La forma o conexión de la estructura no son conocidas [4].

Un típico problema de optimización topológica es el que se presenta a continuación:

¿Cuál es la distribución de material en el dominio (Ω) que proporciona la máxima rigidez (o mínima flexibilidad) para el estado de carga impuesto y la restricción de volumen de determinados? [5]

En la figura 2.1 se puede ver un ejemplo de optimización topológica, con un volumen reducido en un 60 por ciento.

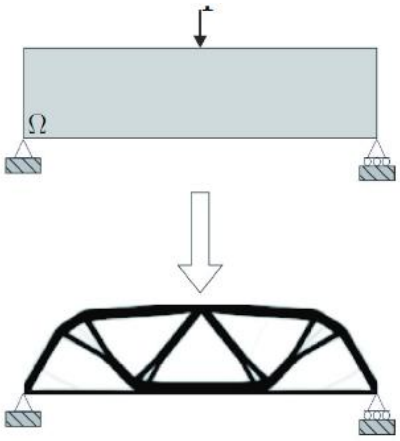


Figura 2.1: Ejemplo de optimización topológica [5]

En esta, se ha resuelto el problema de optimización topológica en un dominio determinado, sometido a un estado de carga dado (F) y unas restricciones, que en este caso serían los apoyos. Los problemas de optimización topológica dan soluciones numéricas implementando algoritmos iterativos que maximizan o minimizan una determinada función objetivo sujeta a una serie de restricciones.

Los procedimientos a nivel conceptual empleados para implementar un problema de optimización topológica son los siguientes:

- Definir un dominio inicial donde se presenta el estado de cargas y las condiciones de los apoyos.
- Se realiza una discretización del dominio por medio del método de elementos finitos
- Implementación del algoritmo iterativo, calculando en cada iteración los desplazamientos por medio del método de elementos finitos y añadiendo o eliminando material como resultado de la maximización o la minimización de la función objetivo implementando un algoritmo de optimización.
- Verificación del resultado por medio de MEF.[5]

Como la solución a estos problemas se basa es un proceso iterativo donde en cada iteración se resuelven uno o más problemas de elementos finitos, el coste computacional se

vuelve un aspecto especialmente relevante. Además, se tiene que utilizar una malla que aporte una precisión aceptable para que el error obtenido al minimizar la función objetivo sea de una gran fidelidad, y que permita obtener sólidos de un alto nivel de eficiencia. Claramente el aumento de precisión que se obtiene utilizando una malla fina que proporcione un gran número de elementos, y la resolución de problemas basados en MEF en cada iteración, tiene como consecuencia un gran aumento del coste computacional necesario. Es bien conocido que, en optimización topológica la resolución de los sistemas de ecuaciones que se crean es el principal cuello de botella, requiriendo por estas razones, un problema de optimización un coste computacional especialmente alto. Aquí es dónde la computación en paralelo cobra una gran relevancia. Permitiendo solucionar problemas de optimización con una alta fidelidad, con una malla que otorgue una discretización de una gran precisión. Permitiendo dar esta solución de una manera eficiente, aumentando la velocidad de procesamiento obtenida y consiguiendo resolver problemas de gran escala.

2.2 Análisis en mecánica de sólidos

Se define el problema de elasticidad particularizado para hacerlo más sencillo al caso bidimensional (figura 2.2).

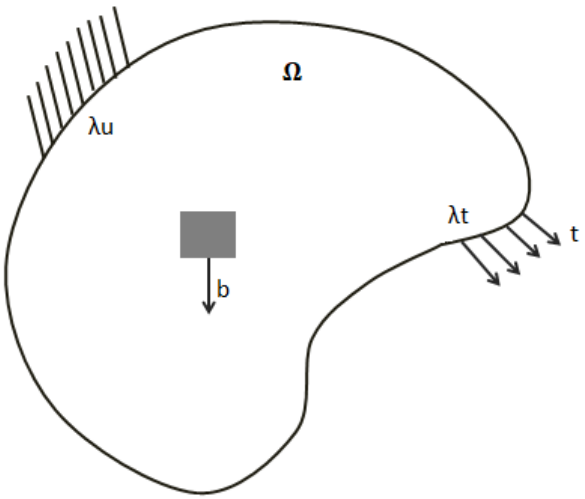


Figura 2.2 Problema de elasticidad lineal

En un problema bidimensional de teoría de elasticidad lineal se tiene una geometría no definida, es decir, una geometría que puede tener cualquier tipo de forma. En esta, están caracterizadas dos tipos de fuerzas que tienen que cumplir la ecuación de equilibrio. La primera, b, representa el conjunto de fuerzas volumétricas y las segundas, t, se aplican en la

superficie del sólido deformable. El dominio está definido como Ω, λu representa las condiciones de frontera de tipo desplazamiento y λt representa las condiciones de frontera de tipo cargas.

Usando notación matricial, la tensión se define como: $\sigma = \begin{bmatrix} \sigma x \\ \sigma y \\ \sigma xy \end{bmatrix}$ (2.1)

Siendo ∇ el operador de la matriz: $\nabla = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix}$ (2.2)

La ecuación de equilibrio se define con la ecuación diferencial de la elasticidad expresada de forma compacta, se obtiene operando con las ecuaciones 2.1 y 2.2:

$$\boldsymbol{\nabla^T(h * \sigma) + b = \theta} \quad \textbf{en } \boldsymbol{\Omega} \tag{2.3}$$

En la frontera λt, debe de haber un equilibrio entre las fuerzas internas representadas por σ y las fuerzas externas sobre la frontera denominada por t. Esto debe ser formulado introduciendo la matriz: $N = \begin{bmatrix} nx & \theta & ny \\ \theta & ny & nx \end{bmatrix}$ (2.4)

Dónde n=(nx,ny) es el vector unitario de λt.

La condición de frontera se define como:

$$\boldsymbol{N(h * \sigma) = t \quad en \quad \lambda t} \tag{2.5}$$

Esta ecuación expresada en componentes, es de la forma:

$$tx=(h^* \sigma x) * nx + (h * \sigma x y)^* ny \tag{2.6}$$

$$ty=(h^* \sigma x) * nx + (h * \sigma x y)^* ny \tag{2.7}$$

El vector de desplazamientos se expresa como: $u = \begin{bmatrix} ux \\ uy \end{bmatrix}$. En λu el cuerpo está fijado, por lo tanto, el vector de desplazamientos es cero. Por lo que, la condición de frontera en cuanto desplazamientos o condición esencial es: $u = 0$.

Además de las ecuaciones anteriores se necesita una ecuación constitutiva para la elasticidad lineal se puede escribir en función de la tensión cómo:

$$\varepsilon = \nabla u \tag{2.8}$$

La ley de ecuación constitutiva, usualmente se conoce como la ley de Hooke que se denomina como: $\sigma = \mathrm{D} * \mathrm{\Epsilon}\ (2.9)$, siendo D la matriz constitutiva. Esta matriz se denomina como:

$$D = \frac{E}{1-v^2} * \begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix} \qquad (2.10)$$

Usando esas ecuaciones se obtiene la formulación en derivadas parciales. Dicha PDE es la siguiente:

$$PDE: \begin{cases} \boldsymbol{\nabla^T(h * \sigma) + b = \theta} & \textbf{en } \boldsymbol{\Omega} \\ \boldsymbol{N(h * \sigma) = t} & \boldsymbol{en} \quad \boldsymbol{\lambda t} \\ \boldsymbol{u = 0} & \boldsymbol{en} \quad \boldsymbol{\lambda u} \end{cases} \qquad (2.11)$$

La solución que se busca solamente depende de los desplazamientos. Por lo que, se puede decir que, desde un punto de vista matemático se ha encontrado la solución al problema formalmente cuando se han resuelto los desplazamientos de este. La obtención de las tensiones se considera simplemente un postprocesamiento.

Se puede aplicar el principio de los trabajos virtuales considerando el campo de desplazamientos contino. Aplicando el principio de las fuerzas virtuales y usando el teorema de Green se obtiene el problema de elasticidad en formulación débil:

$$\int_{\Omega} (\nabla u)^T * h * D * \nabla \mathrm{u} * \mathrm{dA} = \int_{\lambda \mathrm{t}} u^T * t * ds + \int_{\Omega} u^T * b * dA \qquad (2.12)$$

Finalmente, se consideramos la energía de deformación como:

$$B(u) = \frac{1}{2} * \int_{\Omega} (\nabla u)^T * h * D * \nabla \mathrm{u} * \mathrm{dA} - \int_{\lambda \mathrm{t}} u^T * t * ds - \int_{\Omega} u^T * b * dA \ (2.13)$$

Se considera la minimización de la energía de deformación sobre el campo de desplazamientos admisible, como:

$$\begin{cases} \text{Min} \quad B(u) \\ \text{s.t} \quad u=0 \quad \text{en} \quad \lambda u \end{cases} \tag{2.14}$$

2.3 Optimización topológica basada en densidades (SIMP: Solid Isotropic with Material Penalization)

La optimización topológica tiene como objetivo optimizar la distribución de un material, sujeto a una seria de restricciones, en un determinado dominio. Idealmente tiene impuesta una condición binaria, en la que la solución está formada por todo unos o todo cero. Esta solución trae con sigo dificultades de implementación, al no poder ser aplicada sobre una variable de diseño binaria algoritmos basados en gradientes. Usualmente se "relaja" el problema incluyendo densidades intermedias entre 0 y 1, con el objetivo de poder aplicar gradientes a la variable de diseño (densidad). Esto evidentemente trae consigo la formación de estructuras con densidades intermedias, siendo este aspecto no deseable, desde el punto de vista constructivo. Por esta razón se implementan métodos como el método SIMP capaz de penalizar la variable de diseño para evitar las densidades intermedias.

A continuación, se va a introducir al método SIMP y a la formulación del problema de optimización topológica.

El problema se formula de la siguiente forma:

Encontrar [ρ] minimizando la función:

$$\begin{aligned} \text{Min} \quad & f(\rho_e, u) \\ \text{s.t:} \quad & K(\rho_e)u = f \\ : \quad & V(\rho_e) \leq V^* \\ : \quad & 0 \leq \rho_e \leq 1, \rho_e \in D \end{aligned} \tag{2.15}$$

Dónde:

La función objetivo se denomina como *f,* u son los desplazamientos, siendo estos la solución del sistema de ecuaciones, K es la matriz de rigidez y f es un vector de fuerzas. El dominio se denomina como *D*, perteneciendo el vector de densidades a dicho dominio. Se establece además la restricción de que el volumen del material sea menor que un volumen V*[11].

Como el concepto principal de la optimización topológica es la distribución del material en cierto dominio, maximizando la rigidez, se necesita de un método que otorgue la posibilidad de modelar la presencia de material. Para la realización de este proyecto se ha utilizado el modelo de un material sólido isotrópico con penalización del material (SIMP).
En este modelo los elementos discretizados tienen asociada una variable ρ. Dicha variable se asemeja a la densidad del material.

Como se ha dicho anteriormente, dicha variable es continua para poder realizar sobre esta, algoritmos basados en gradientes. Se encuentra en un rango entre el cero y uno. Dónde, el cero representa la ausencia completa de material y el uno representa la existencia de este:

$$(0 \leq \rho \leq 1)$$

Si bien es cierto que, en computación, se suele establecer un ρ_{min} un poco mayor de cero, para que, al implementar el algoritmo de resolución, no se establezcan problemas de singularidad en las matrices. El diseño final tiene densidades 0 o 1 en todos los elementos (aproximadamente), formando diseños en "blanco y negro". Para problemas donde existen limitaciones por restricción, la experiencia demuestra que la solución de los problemas de optimización da lugar a este tipo de diseños si se utilizan factores de penalización grandes, normalmente se requiere un factor de penalización de p=3. Por lo que, para obtener verdaderos "diseños 0" y "diseños 1" se requiere de un factor de penalización con un valor mínimo de 3. [6]
Esto se muestra en la siguiente gráfica:

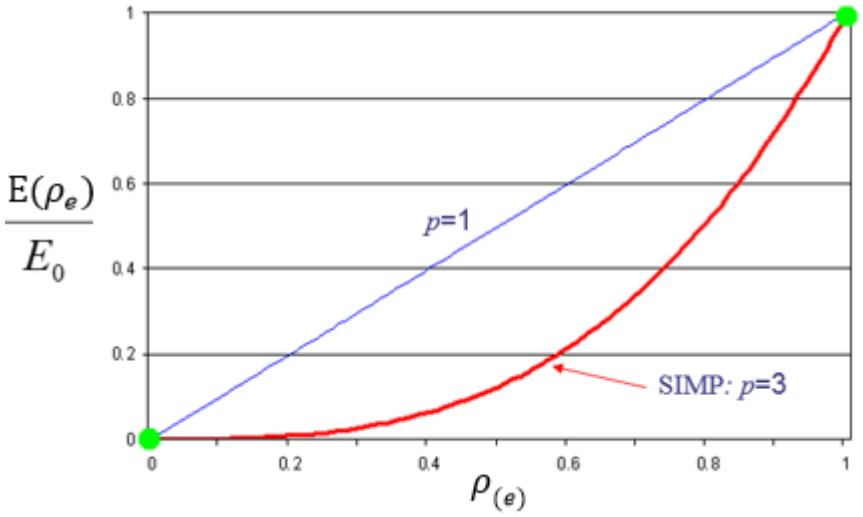


Figura 2.3Penalización[7].

Dónde:

$$E_i = E_i(x_i) = E_{min} + x_i^p(E_0 - E_{min}), \quad xi \in [0,1], \tag{2.16}$$

Siendo E el módulo elástico y p el factor de penalización. El factor de penalización p disminuye la contribución de elementos con densidades intermedias (elementos grises) a la rigidez total y dirige la solución de optimización a elementos en color negro ρ=1 o en color blanco ρ= ρmin. La explicación está en que el factor de penalización modifica las pendientes de la curva, penalizando de esta forma las pendientes intermedias mediante un algoritmo de optimización. Dicha penalización permite evitar que en la solución del problema de optimización aparezcan soluciones matemáticamente óptimas, ya que, cumplirían con los requisitos establecidos, pero que muy difícilmente podrían fabricarse. El método SIMP permite calcular la matriz de rigidez de cada elemento siguiendo la siguiente ecuación de interpolación:

$$K_e = K_{min} + \rho_e^p(K_0 - K_{min}), \tag{2.17}$$

Dónde:

K_0 es la matriz de rigidez del elemento lleno, K_{min} es la matriz del elemento vacío (mayor que cero para evitar problemas de singularidad), ρ_e es el vector de densidad y p es el factor de penalización.

En este estudio se define la función objetivo como minimización de la función “compilance”, o lo que es lo mismo maximizando la rigidez, de la siguiente forma:

$$c(\tilde{x}) = F^T U(\tilde{x}), \tag{2.18}$$

Dónde:

F es el vector de fuerzas aplicadas sobre los nudos y u es el resultado de la aplicación del método de elementos finitos siguiendo la siguiente ecuación:

$$K(\tilde{x})U(\check{x}) = F \tag{2.19}$$

Cuando se aplica un método de optimización es necesario realizar un análisis de la sensibilidad. Un análisis de sensibilidad estudia el grado de cambio de las respuestas estructurales respecto a los parámetros de diseño. Por lo que, en este caso, dicho análisis evalúa como va a afectar un cambio en la densidad de cada elemento finito (lleno o vacío) a la respuesta que se tiene en la estructura (cambio en la rigidez), teniendo en cuanta que el objetivo buscado es minimizar *compilance* o lo que es lo mismo, maximizar la rigidez. Comúnmente, los algoritmos de optimización utilizan los gradientes durante el proceso de optimización para encontrar posibles mínimos locales, es decir, realizan un análisis de la sensibilidad [13]. Existen además métodos de optimización que no están basados en gradientes. Estos normalmente necesitan un mayor número de iteraciones para conseguir una convergencia del problema, al no disponer información sobre cuál es la dirección más óptima para avanzar. Por esta razón, este estudio se centra en los métodos que utilizan el gradiente como punto inicial para realizar la optimización. La figura 2.4 muestra un ejemplo gráfico de como actuaría un método de optimización basado en gradientes.

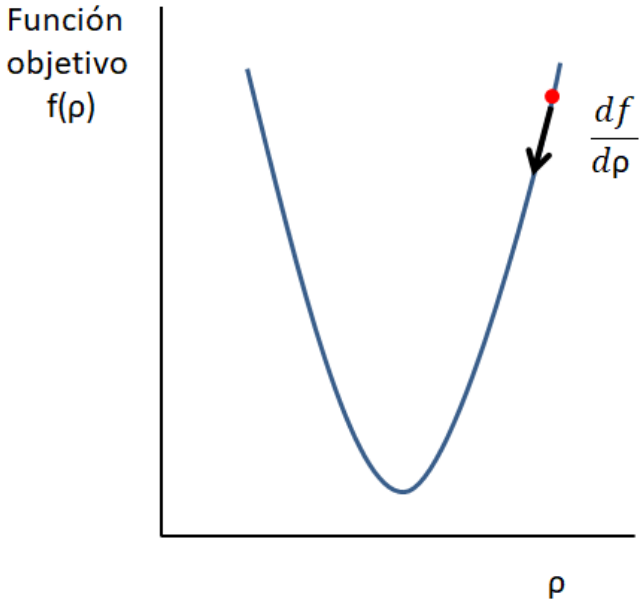


Figura 2.4Minimización mediante gradiente

En la figura se ha representado la función objetivo en función de la variable de diseño. La representación que se muestra no corresponde a una función real, es simplemente, una simplificación para explicar conceptualmente el funcionamiento de los métodos basados en gradiente. El objetivo es minimizar la función objetivo, para ello, se tiene que realizar un cálculo de la sensibilidad, calculando el gradiente de la función, para obtener la dirección en la que se puede "descender". El algoritmo de optimización, cambiará los valores de densidades, usando como base el gradiente calculado, reduciendo el valor de la función objetivo. Este proceso se repite iterativamente hasta encontrar el mínimo de la función o lo que es más común cumplir un criterio de parada. Obteniéndose como resultado un sólido optimizado, ya que, se ha encontrado una solución, en la que, la función objetivo tiene el valor más bajo posible y esta ha sido masivamente aligerada, al tener, uno o varios elementos huecos.

La expresión de la sensibilidad más sencilla es la siguiente [11]:

$$\frac{\partial f}{\partial \rho_e} = -u^{*T}\frac{\partial K}{\partial \rho_e}u = -u^{*T}p\rho_e^{p-1}(K_0 - K_{\min})u, \tag{2.20}$$

Dónde u* es el resultado de:

$$ku^* = \frac{\partial f}{\partial u} \tag{2.21}$$

Como $\frac{df}{du^*}$=f, entonces u*=u.

Las expresiones de la sensibilidad cambian según el filtro utilizado, lo que se ha expresado es la expresión usando un filtro de sensibilidad. Posteriormente se explicarán que son los filtros y para qué son utilizados. En el siguiente capítulo se explicarán las herramientas existentes para abordar el problema de optimización topológica de forma paralela, así como se introducirá al uso de filtros y los algoritmos de optimización.

3 Paralelización

3.1 Introducción

En la optimización topológica hay que resolver el problema de la elasticidad por el método de los elementos finitos. Las ecuaciones que proporciona el método de elementos finitos presentan una gran dificultad de cálculo si se plantean de manera analítica. Para resolver dichas ecuaciones se requieren métodos de computación. La resolución de un problema de optimización topológica se basa en un método iterativo, en el que se resuelven las ecuaciones del MEF en cada iteración. Esto tiene como consecuencia un coste computacional elevadamente alto. Para problemas complejos es inabordable utilizar métodos de computación tradicionales para obtener una solución al problema de optimización. Para dar una solución eficiente, es necesario utilizar métodos de computación paralela de manera que se permita distribuir el trabajo de cálculo entre los procesadores disponibles, aumentando de esta manera la velocidad de procesamiento obtenida.

En este capítulo se exponen las herramientas existentes para abordar el problema de optimización de forma paralela, en el que, se va a explicar el método de partición de grafos para distribuir el dominio entre los distintos procesadores, la herramienta ParCSR para el sistema ensamblado, los filtros y el algoritmo de optimización utilizado.

3.2 Particionamiento de grafos (Metis)

Para realizar simulaciones de forma paralela es necesario realizar un particionamiento de la malla con la que se discretiza un determinado dominio. Conseguir que esa partición se realice de forma eficiente es un aspecto clave, al utilizar dicho proceso altos recursos computacionales. El objetivo de esa partición es que se realice una distribución de ese dominio entre los distintos procesadores disponibles, para poder aumentar la eficiencia de cálculo. Esa distribución tiene que cumplir las siguientes premisas:

- *Correcto balance.* Es necesario equilibrar los cálculos entre los distintos procesadores. Si esta premisa no se cumple no se obtendrá un aumento de eficiencia, ya que los procesadores con menor trabajo de cálculo quedarán a la espera de los que han recibido una mayor distribución. Por esta razón es un aspecto fundamental realizar una partición equitativa.
- *Reducir los elementos adyacentes.* Una reducción de estos elementos supondrá una reducción de las comunicaciones necesarias entre los distintos procesadores. Hay que tener en cuenta que, a mayor comunicación necesaria para resolver un problema, mayor será el tempo requerido para dar una solución a este.

Para realizar la partición de grafos se realiza un primer mallado, discretizando el dominio existente en elementos finitos iguales, para posteriormente proceder a su partición. Metis es la herramienta que se va a utilizar en este proyecto para proceder a la partición de los grafos. “METIS es una herramienta basada en algoritmos de grafos utilizada para realizar la *partición de*

las mallas del método de elementos finitos, desarrollada por la Universidad de Minnesota y basada en el paradigma de partición de grafos multinivel por el cual permite realizar una partición y reducción de gran calidad. METIS proporciona distintos comandos para realizar las particiones y realizar la reducción de rellenos además de proporcionar una interfaz de programación que soporta C/ C++ o Fortan para invocar diversos algoritmos" [8].

Como cita el autor mencionado, Metis se basa en una partición de grafos multinivel en tres fases: reducción, partición y proyección.

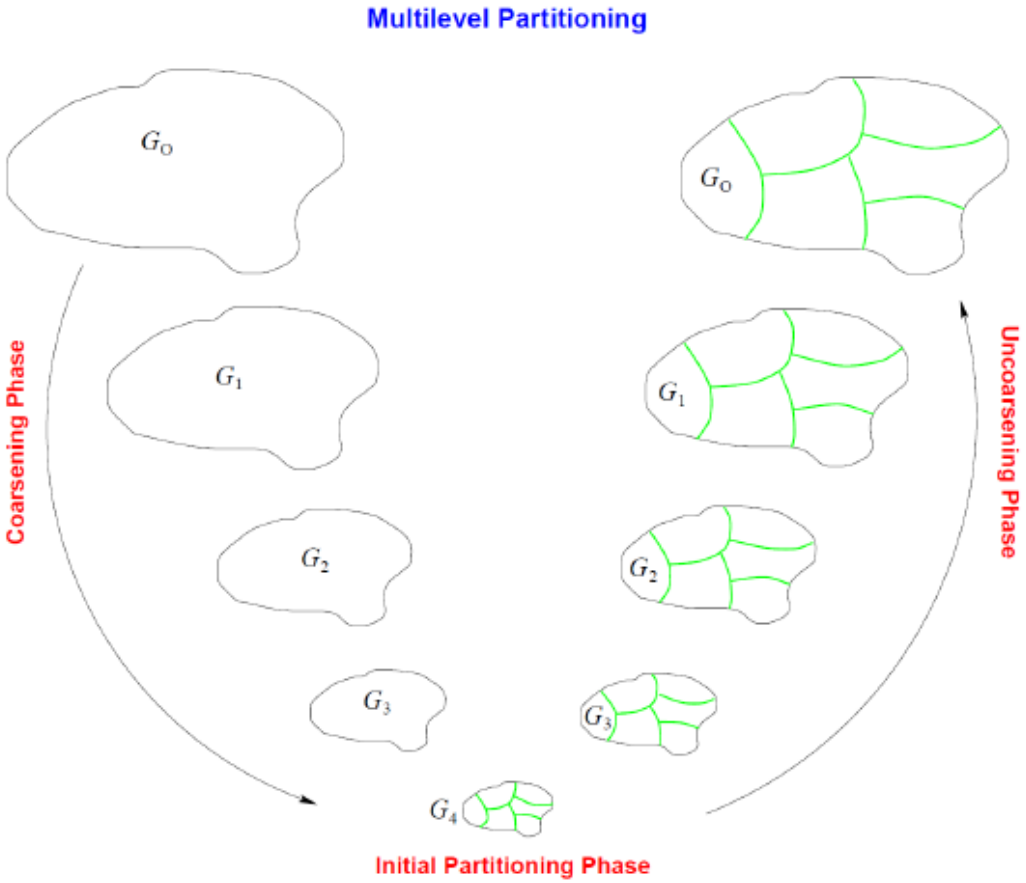


Figura 3.1Partición de grafos (Metis).[8]

- *Reducción.* Como se muestra en la figura 3.1 se parte de un grafo de gran tamaño G0. Este se reduce progresivamente mediante la unión de vértices adyacentes (representando cada vértice un elemento). Esta reducción termina en el grafo G4, cuando se ha conseguido que este sea de un tamaño mucho menor que el original, con sólo unos pocos vértices.
- *Partición.* Cuando se ha alcanzado un grafo muy reducido se procede a su partición, siendo este proceso mucho más sencillo en comparación con haber realizado la reducción sobre el grafo original que es de gran tamaño.
- *Proyección.* Una vez que se ha realizado de forma correcta la partición del grafo se procede a su proyección. En esta fase se recorre el camino inverso, en el que se produce la descombinación de los vértices de forma que al grafo va aumentando

progresivamente de tamaño. Al final de la proyección se obtiene un grado de tamaño equivalente al original pero ya particionado.

Cabe destacar que Metis es una herramienta capaz de realizar particiones de una gran calidad, centrándose en los bordes de los elementos y permitiendo una rápida partición de grafos.

3.3 Formato ParCSR

Para realizar la distribución de los datos entre los distintos procesadores hace falta de hacer uso del formato ParCSR con el objetivo de poder distribuir las matrices que se forman en el cálculo entre los distintos procesadores. Este formato pertenece a la librería Hypre.

La librería de HYPRE permite dar soluciones detalladas a simulaciones de forma más eficiente que los métodos de computación tradicionales. Este ofrece un conjunto de solvers escalables utilizados en computación científica a gran escala, proporcionando herramientas como métodos multigrid paralelos. Es una biblioteca portable y que soporta varios lenguajes [10]

La herramienta ParCSR permite almacenar matrices de grandes dimensiones de forma distribuida, asegurando de esta forma la eficiencia de la computación en paralelo. Por lo que, esta herramienta consigue distribuir las matrices resultantes del sistema de ecuaciones lineales que proporciona el MEF, entre los distintos procesadores disponibles.
Una característica del sistema de ecuaciones que se forma, es que, la matriz principal del sistema, que es la que relacionan las fuerzas con los desplazamientos, es una matriz muy dispersa. En esta se concentran la mayoría de los coeficientes en la diagonal principal, siendo nulos la mayoría de los coeficientes fuera de esa diagonal. El formato de datos CSR (Compressed Sparse Row) permite definir y distribuir de manera eficiente, optimizando el uso de memoria, matrices dispersas de grandes dimensiones como es el caso de la matriz de coeficientes. Para definir una matriz se utilizan los siguientes tres vectores.

- "*Data*". En este vector se almacenan todos los elementos no nulos de la matriz, ordenados por filas.
- "*Column_index*". En este vector se almacenan el número de columna en la que se almacena cada coeficiente.
- "Row_index". En este vector se almacenan el número de columna en la que se almacena cada coeficiente.

Con estos tres vectores se puede definir completamente la matriz de coeficientes, ya que, quedan completamente definidos los valores no nulos y la posición que cada uno ocupa dentro de la matriz.
Una matriz "A" en formato ParCSR está dividida en "p" partes formando unas submatrices denominada A1,A2,A2.An…, para que estas submatrices puedan repartirse entre los procesadores de forma distribuida. A su vez, cada submatriz de A se divide en las matrices D y 0 (figura 3.2) [9]

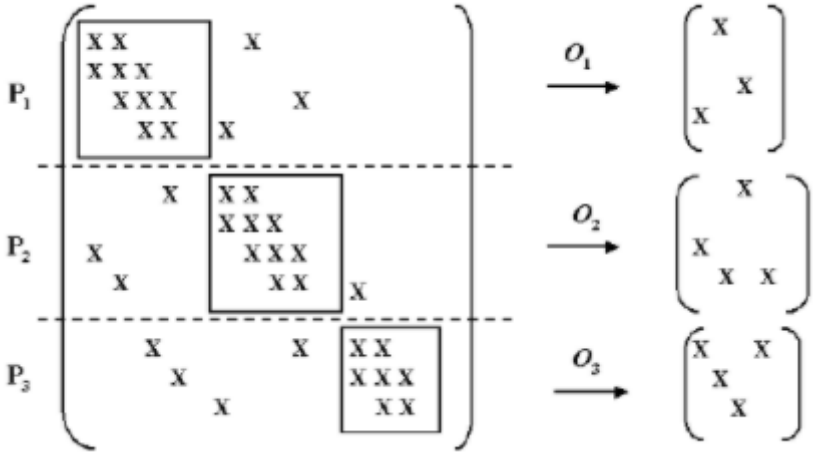


Figura 3.2Distribución de la matriz en tres procesadores.[9]

En esta figura se puede observar como una matriz dispersa "A" se divide en Pk partes, siendo k en número de procesadores. A su vez, las submatrices que se han formado se dividen en una matriz Dk y una matriz Ok. La matriz Dk está formada por los elementos próximos a la diagonal de la matriz A, dentro de Pk. En la matriz Ok se incluyen los elementos dispersos de la matriz original, que no se habían incluido anteriormente. Se tiene que incluir un vector que contenga los números de las columnas de los coeficientes de Ok. Dicho vector se denomina "COL_MAP_O_k". En este caso COL_MAP_O_1= (5,6,8), COL_MAP_O_2= (1,2,4,9) y COL_MAP_O_3= (3,4,5,8).

En esta estructura de datos hay que diferenciarla entre la "local" y la "distribuida". En la estructura local los datos necesarios se encuentran dentro del propio procesador. En la distribuida se requiere de un mecanismo de comunicación. En este proyecto se utiliza el mecanismo de comunicaciones basado en el estándar MPI. En este estándar se realiza el intercambio de datos directamente entre los procesadores mediante tareas de recepción y envío. Se reduce de esta forma el costo computacional, ya que, el número de vecinos y la cantidad de información a transmitir es independiente del número de procesadores. [11]

3.4 Paralelización de filtros

En optimización topológica, la función objetivo que se quiere minimizar no es una función regular. Eso dificulta el uso de métodos basados en gradientes para obtener una solución del problema (figura 3.3).

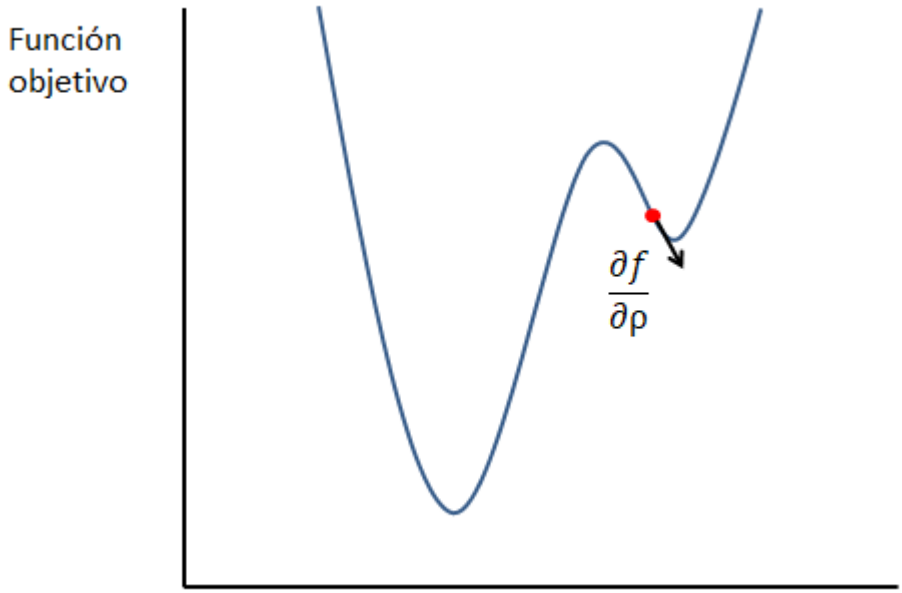


Figura 3.3Necesidad del uso de flitros

En la figura 3.3 se ha representado la función objetivo frente a la variable de diseño de forma conceptual. En esta función objetivo, los métodos basados en gradientes no son útiles ya que pueden dar como solución un mínimo local que no es el buscado, otorgando de esta forma una solución poco optimizada. Además, existe un problema comúnmente llamado como la inestabilidad del tablero de ajedrez. La inestabilidad del tablero de ajedrez es una solución del problema de optimización dónde el cuerpo no es continuo y este es claramente inestable [12]. Los elementos de color negro ρ=1 (lleno) se intercalan con los elementos de color blanco vacío (figura_). Esta solución cumple con el criterio de convergencia y con las restricciones impuestas, a pesar que la solución al problema sea inestable.

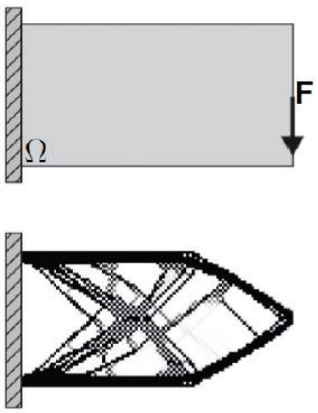


Figura 3.4Inestablilidad tablero de ajedrez[12]

La solución a estos problemas es utilizar una regularización, también denominada como filtrado del campo de variables de diseño. El uso de filtros produce un efecto de suavizamiento sobre la función objetivo, permitiendo obtener la convergencia deseada, mediante métodos basados en gradientes. Existe una alta variedad de filtros y son de uso casi obligatorio en la resolución de cualquier problema de optimización. Destacando los filtros basados en distancias y el filtro de Helmholtz.

Los filtros de distancias trabajan sobre la función objetivo cambiando cada valor, por el promedio de las que se encuentra en un círculo de radio r min. Como se observa en la figura 3.5 al ir aumentando el valor de r_{min}, desaparecen las inestabilidades y se obtienen estructuras más simples.

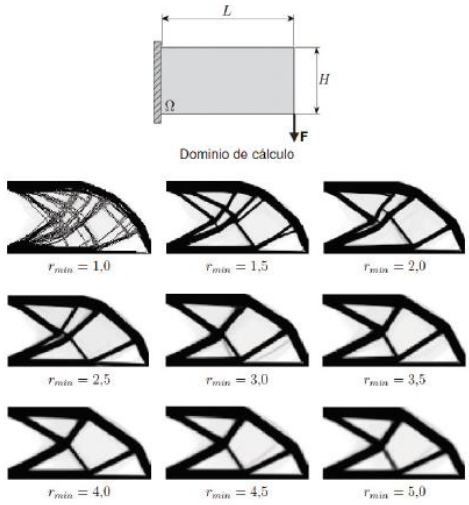


Figura 3.5 Filtro basado en distancias[12]

El filtro de Helmholtz en vez de considerar un radio mínimo resuelve una EDP.

Dicha EDP es la siguiente:

$$-r^2\nabla^2\hat{g} + \hat{g} = g \qquad \hat{g} \in \Omega \tag{3.1}$$

$$\frac{\partial \hat{g}}{\partial n} = 0 \qquad \hat{g} \in d\Omega$$

Dónde: g es la variable de diseño sin filtrar, gˆ es la variable de diseño filtrada, y r es un parámetro de longitud que juega un papel similar al del radio de la integral de convolución para calcular el filtro de sensibilidad. [11]. Siendo:

$$g = \rho_e \frac{\partial f}{\partial \rho_e} \tag{3.2}$$

Para resolver esta PDE se produce un ensamblaje de forma distribuida formando un sistema de ecuaciones similar al obtenido al resolver la PDE de la elasticidad lineal. Esta ecuación tiene la siguiente forma:

$$K * \hat{g} = g \tag{3.3}$$

La solución de la ecuación 3.3 es el campo de densidades filtrado, destaca que la matriz ensamblada K está en formato ParCSR.

Desde el punto de vista de la computación en serie, este filtro no trae ninguna ventaja, ya que, es más sencillo utilizar un filtro basado en distancias y hacer el promedio de los valores incluidos en ese círculo, que resolver una EDP, con el coste computacional que conlleva. Pero, en la computación paralela hay que tener en cuenta las comunicaciones que se tienen que realizar. Al estar el dominio repartido entre los distintos procesadores, hacer el promedio en un radio que incluya a dos o más subdominios incrementa el coste computacional del problema. La razón está en todas las comunicaciones que se tendrían que realizar entre estos procesadores, teniéndose de esta forma una pérdida de eficiencia (figura 3.6).

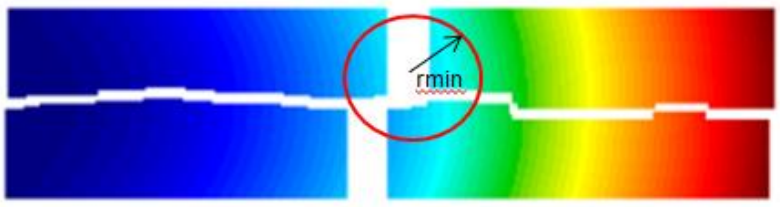


Figura 3.6 Problemas de los filtros de cónicos en computación paralela.

El filtro de Helmholtz necesita ser ensamblado y ser resuelto mediante un método iterativo, además, al realizar el ensamblaje también se necesitan realizar comunicaciones entre los procesadores existentes. El filtro óptimo depende en gran medida de la naturaleza del problema a optimizar. Siendo de manera general más óptimo el filtro de Helmholtz, si se compara con el de distancias, cuando la r_{min} tiene un valor muy grande. Si se utiliza un valor de r_{min} pequeño, es más óptimo, por regla general un filtro de distancias. Lo más usual al realizar simulaciones de una alta precisión, con una malla que genere un número pequeño de elementos, es utilizar un valor bajo de rmin, ya que, no tiene sentido dividir un problema en un gran número de elementos para luego hacer el promedio de este. Por esta razón en este estudio se utilizará un filtro cónico (filtro de distancias).

3.5 Algoritmo de optimización (Optimality criteria / MMA)

Se necesita un algoritmo que permita actualizar las variables de diseño. De forma que sea posible encontrar la distribución óptima del material, minimizando en cada iteración la función objetivo. Este estudio se centra en los métodos que utilizan el gradiente como punto

inicial para realizar la optimización. Se va a introducir al criterio de optimalidad (0C) y el método de las asíntotas móviles (MMA).

- ***Criterio de optimalidad de Benso.*** Este es el enfoque clásico utilizado en un problema de optimización debido a su simplicidad numérica y eficiencia [23]. Se basa un método iterativo cuyo algoritmo es el siguiente:

$$x_i^{new} = \begin{cases} \max(0, x_i - m), \text{ if } x_i\, B_i^n \leq \max(0, x_i - m) \\ \max(1, x_i + m), \text{ if } x_i\, B_i^n \geq \min(1, x_i - m) \\ (x_i B_i^n)^q, \text{ otherrwise} \end{cases} \tag{3.4}$$

Dónde: m es un límite de movimiento que presenta un valor positivo, n representa el coeficiente de amortiguamiento (elegir valores de m=0.2 y n=0.5 es recomendable para problemas de cumplimientos mínimos [23]), q es un factor de penalización (típicamente q = 2). Be sigue la siguiente expresión:

$$B_e = -\frac{\partial c(\tilde{x})}{\partial x_e}\left(\lambda \frac{\partial v(\tilde{x})}{\partial x_e}\right)^{-1} \tag{3.5}$$

Las condiciones KKT se cumplen cuando Be=1. Siendo λ el multiplicador de Larange asociado a la variable $v(\tilde{x})$. El único valor a determinar en el algoritmo presentado es el multiplicador de Larange(λ), que cumple con la siguiente expresión:

$$v\left(\tilde{x}\left(x^{new}(\lambda)\right)\right) = 0 \tag{3.6}$$

Dicha ecuación se resuelve mediante un método de busca de raíces, como es el método de la bisección. Este es un método iterativo que permite encontrar soluciones a ecuaciones no lineales.

Este algoritmo de desarrolla de forma iterativa, actualizando el valor de la variable de diseño en cada iteración. Finalmente este converge al cumplir con el criterio de parada. Este criterio se cumple si el valor absoluto de la variable de diseño actualizada menos la variable de diseño sin actualizar es menor que la tolerancia elegida, o si se han superado un número máximo de iteraciones.

Método de las asíntotas móviles (MMA). Método iterativo de programación lineal que se caracteriza por generar un problema convexo que es resuelto en cada iteración y actualizar las asíntotas del problema de manera que se acelere la convergencia del mismo [15].

El proceso que sigue este algoritmo de forma conceptual es el siguiente:

1) Se elige un punto inicial x^k, siendo x la variable de diseño y k=0.

2) Se calcula la función objetivo $f_i(x^k)$ y el gradiente $\nabla f_i(x^k)$.

3) Se genera un subproblema convexo $P^{(k)}$ mediante funciones de aproximación actualizando las asíntotas de este.

4) Se resuelve el subproblema, encontrando un valor de x más próximo al buscado que el inicial, se actualiza la variable volviendo al paso 2 con k=1.

El procedimiento se repite de forma iterativa hasta que se cumple con el criterio de parada.

En las figuras 3.7 se muestran un ejemplo de cómo es el funcionamiento del algoritmo. Se muestra una primera iteración, en la que se ha representado en negro la función que se quiere minimizar (función objetivo), en rojo el subproblema generado por aproximación convexa, y el azul el problema antes de la aproximación. Se empezaría con la curva convexa azul, con asíntotas L_i y U_i y a partir de este, mediante métodos de aproximación se genera el subproblema convexo (color rojo). Se soluciona este subproblema encontrando su mínimo, descendiendo en la curva convexa a través del cálculo del gradiente. Posteriormente, se actualiza la variable de diseño y se produce el movimiento de las asíntotas. Se comprueba el criterio de parada. Si este cumple el proceso ha llegado a su fin. Si no cumple, se repite el proceso, siendo ahora la curva roja la anterior curva inicial y generando a través de esta un nuevo subproblema convexo, volviendo a mover las asíntotas, y acercando de forma iterativa la variable de diseño al mínimo de la función objetivo

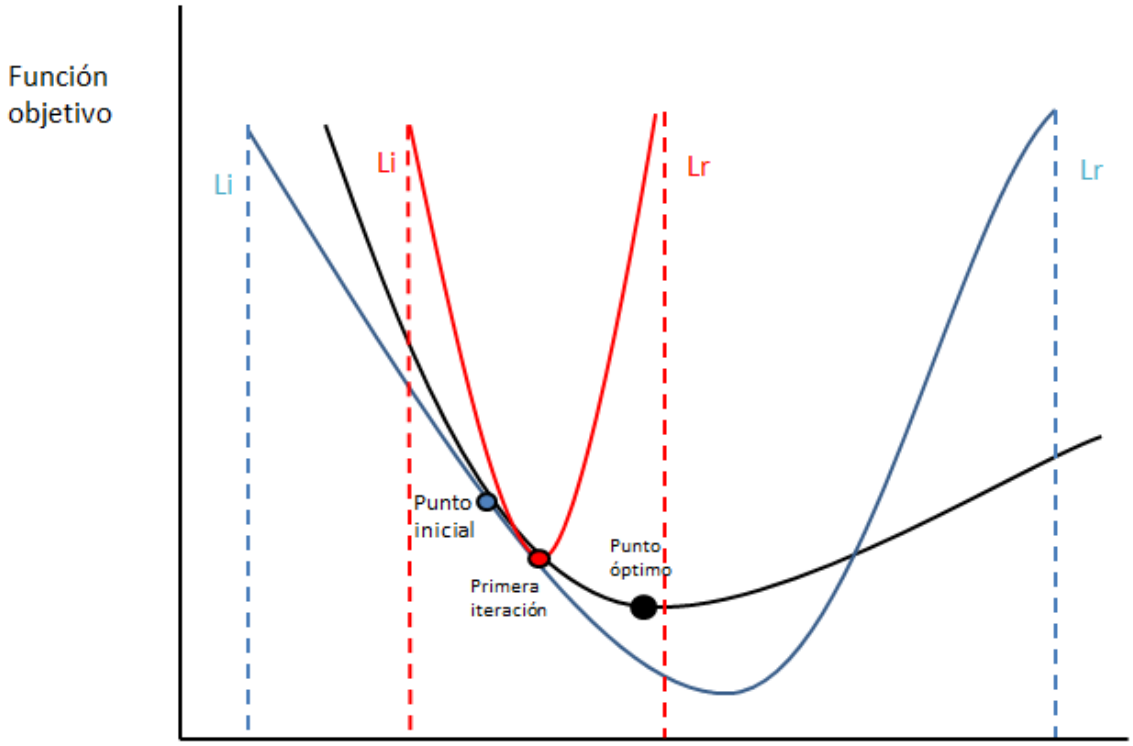


Figura 3.7.MMA primera iteración

El MMA será el algoritmo usado en este proyecto por su rápida convergencia y porque este, tiene la ventaja de tener que utilizar el filtro solo una vez por iteración. El algoritmo OC tiene que aplicar el filtro por cada iteración del método de la bisección, lo que da lugar a que se tenga que resolver más de un filtro por cada iteración principal. Pudiendo esto suponer que se tarde más tiempo en utilizar el filtro y pudiendo crear un posible cuello de botella.

3.6 Enfoque paralelizado de optimización topológica.

En este apartado se resumen las estrategias de paralelización utilizadas en este proyecto para resolver problemas de optimización topológica. Hay que destacar, como ya se ha mencionado anteriormente, que un problema de optimización topológica destaca por sus grandes necesidades computacionales. Por esa razón, este estudio tiene un enfoque paralelo.

El siguiente diagrama resume el proceso de optimización del método SIMP [11]:

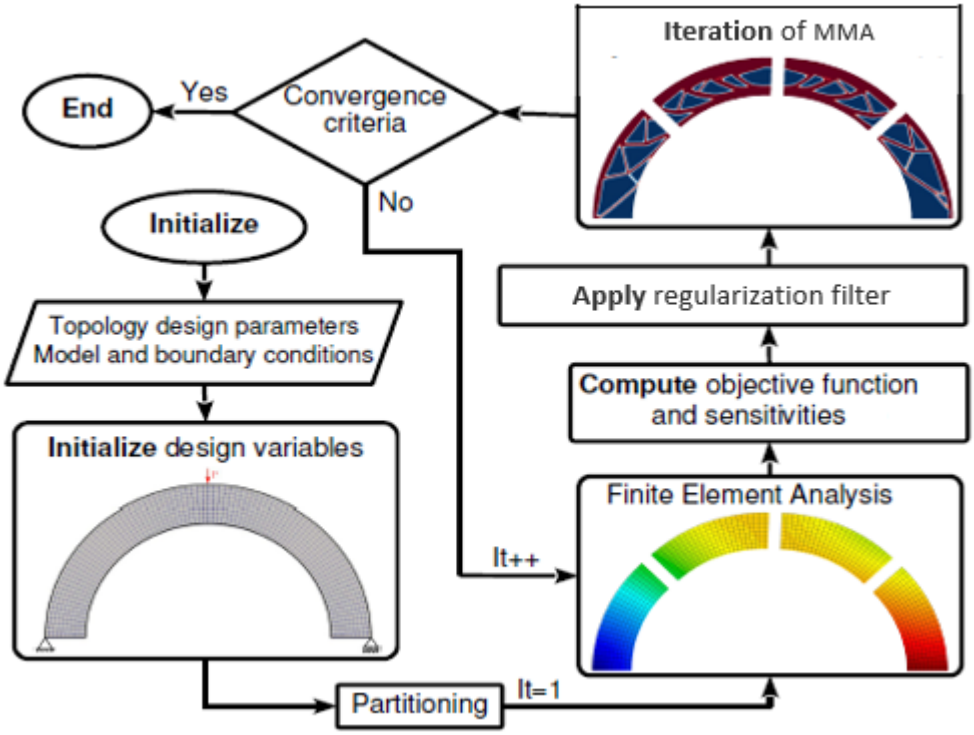


Figura 3.8 Enfoque paralelizado de optimización topológica [11]

En un inicio, se introducen los parámetros de diseño y las condiciones de contorno (cargas, apoyos, el volumen máximo del material etc.). Posteriormente se realiza una partición del dominio en los diferentes subdominios, atribuyendo cada subdominio a un procesador. La partición se realiza mediante la herramienta de partición de grafos METIS. Una vez formado los distintos subdominios comienza el proceso iterativo. Se resuelve el problema de forma paralela ensamblando la matriz de rigidez y resolviendo el sistema de ecuaciones formado. A continuación, se calcula la función objetivo y el gradiente para poder aplicar algoritmos de optimización basados en gradiente. Se aplica de forma distribuida el filtro de Helmholtz, que consiste en la resolución de una PDE o el filtro cónico. Posteriormente, se realiza la actualización del vector de densidades mediante un algoritmo de optimización basado en gradientes. La definición de las condiciones iniciales y la partición se producen sólo una vez, el resto del ciclo se repite hasta que se cumple con el criterio de parada.

El resultado es la obtención de un sólido lleno-vacío, en la que se ha minimizado la función *compilance,* o lo que es lo mismo se ha maximizado la rigidez, bajo unas restricciones de partida. Obteniéndose de esta forma estructuras de alto nivel de eficiencia.

En el siguiente capítulo se va a introducir a las herramientas disponibles para resolver el método de elementos finitos de forma paralela, en el que se utilizará el método iterativo del gradiente conjugado, precondicionado con un método multigrid. Se utilizará la misma estrategia para resolver la EDP de Helmholtz, así como se discutirá la utilización de métodos multigrid algebraicos y geométricos.[17]

4 Métodos multigrid

4.1 Introducción

En este capítulo se va a introducir el método de resolución utilizado para resolver las ecuaciones que se presentan en el problema de optimización topológica y se realizará un estudio de los métodos multigrid algebraico y geométrico. Es bien sabido que los sistemas de ecuaciones que se forman en optimización topológica presentan el principal cuello de botella de este. Utilizar métodos altamente escalables y eficientes es un aspecto clave para lograr diseños de alta calidad con costes computacionales razonables.

Si bien es cierto que, utilizar los recursos computacionales disponibles de manera óptima es un aspecto muy relevante, también lo es el método de resolución que se utiliza para resolver las distintas ecuaciones que se plantean. Siendo los métodos multigrid los más eficientes, sin ningún competidor existente. Para resolver los sistemas de ecuaciones en las simulaciones se utilizará el método del gradiente conjugado de forma distribuida. El gradiente conjugado es un método iterativo que ofrece una gran ventaja frente al uso de solvers directos, al reducir de forma exponencial el coste computacional requerido.

Los métodos multigrid se pueden utilizar como solvers, pero en este estudio se utilizarán como precondicionadores del gradiente conjugado (CG) distribuido, consiguiendo de esta forma una aceleración en la convergencia del método. Por lo que, en este estudio se ha utilizado un CG con un precondicionamiento en cada iteración.

A continuación, se va a realizar una breve introducción sobre métodos multigrid.

El objetivo de los métodos multigrid en este estudio es facilitar la convergencia de un método iterativo. Como se ha comentado anteriormente estos pueden ser utilizados como solvers, pero su potencia se explota en mayor medida al ser utilizados como precondicionadores. Este método opera sobre distintos niveles de malla, realizando un ciclo en V (figura 4.1).

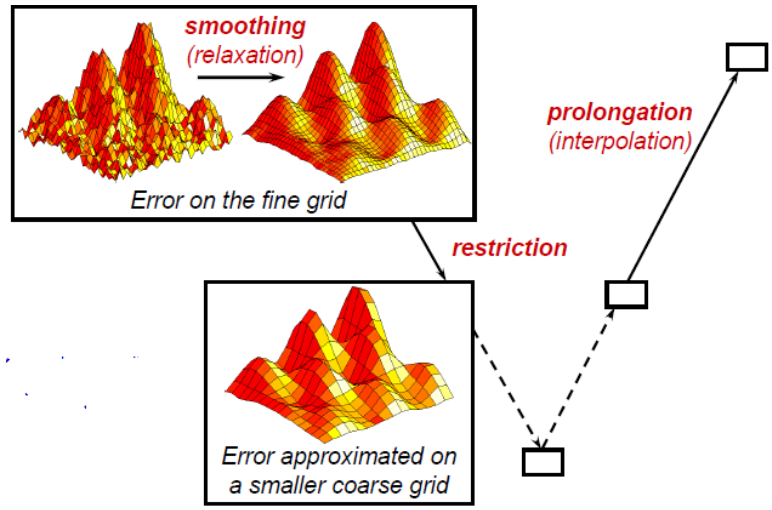


Figura 4.1 Ciclo en V [19]

El procedimiento que utiliza un método multigrid se introduce a continuación.

Se parte de un mallado muy fino, por lo que, los sistemas de ecuaciones que se forman son de un gran tamaño, lo que hace inviable su resolución mediante métodos directos por la cantidad de memoria, necesitando usar métodos numéricos basados en iteraciones. El objetivo es reducir el error del método iterativo. En un primer lugar se requiere de un suavizamiento o relajación. Este permite obtener el residuo en la malla fina y dar una primera solución con la que comenzará el método iterativo. Seguidamente mediante operadores de restricción se proyectará ese residuo a una malla más gruesa. En la figura 4.1 se ha presentado el caso más simple, en el que solo hay dos niveles de malla. Las operaciones de restricción se realizan en los distintos niveles de malla, mallas progresivamente más gruesas, hasta aproximar el error de la malla fina en una malla lo suficientemente gruesa como para resolver directamente los sistemas de ecuaciones que se forman. Se resuelven los sistemas de ecuaciones de la malla gruesa obteniéndose la corrección de malla gruesa. A continuación, se recorre el camino inverso, mediante operadores de prolongación, pasando la corrección de la malla gruesa a la malla fina. Finalmente se actualiza la primera solución obtenida sumándole esa corrección. Obteniendo de esta forma una solución más próxima a la solución real. Usualmente se denomina al ciclo descrito como ciclo en V, ya que esa es la trayectoria seguida entre la malla fina y la gruesa [19].

Existen dos filosofías en cuanto a la utilización del método multigrid, una algebraica y otra geométrica. En los siguientes apartados se presentará con más detalle cómo se realiza el precondicionamiento del CG. Además, se realizará un estudio y discusión del precondicionamiento multigrid algebraico y el geométrico.

4.2 Método del gradiente conjugado distribuido.

El método del gradiente conjugado distribuido es un método iterativo utilizado en este estudio para resolver los sistemas de ecuaciones que se plantean. Este se utiliza de forma paralela en cada subdominio formado por la herramienta de partición de grafos Metis. Esto

permite que los cálculos se realicen de forma paralela aumentando la velocidad de procesamiento. El gradiente conjugado es una modificación del método del gradiente descendiente que se muestra a continuación.

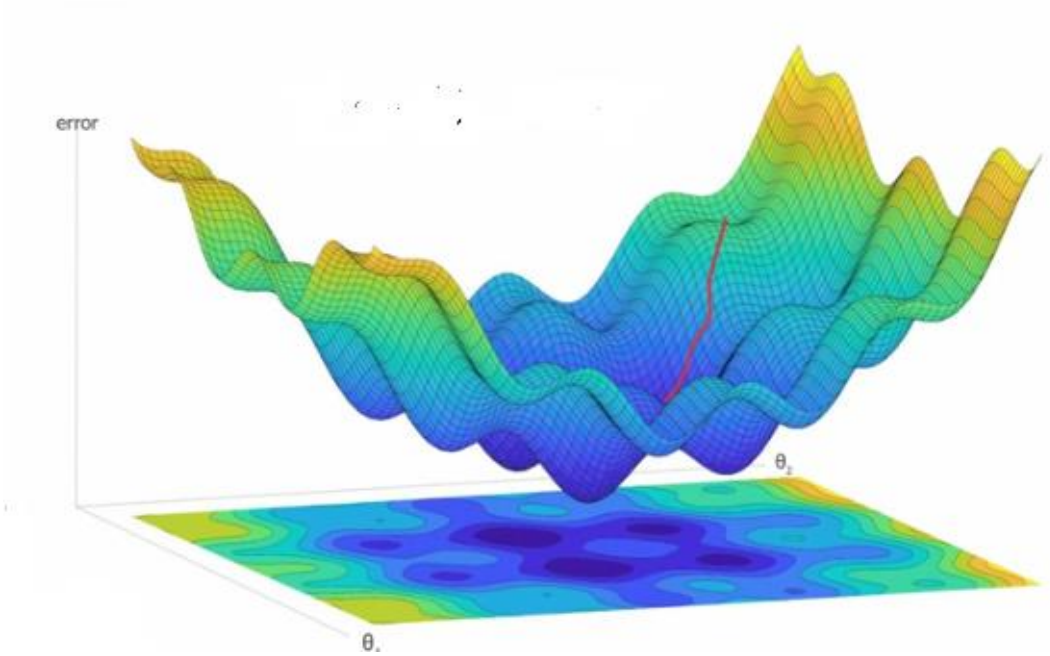


Figura 4.2 Método del gradiente descendiente

En la figura 4.1 se ha representado la función de coste de las variables θ1 y θ2. Lo que se ha representado es el error del método iterativo en función de estas dos variables.
El objetivo es encontrar la solución de un sistema de ecuaciones de la forma:

$$A * x = b \tag{4.1}$$

siendo esta la ecuación resultante de aplicar el MEF. Siendo A, la matriz de coeficientes, x, corresponde a un vector desconocido, y b, corresponde a un vector conocido. En este estudio x se correspondería con el vector de desplazamientos y b con el vector de fuerzas.

Claramente, este método iterativo puede ser aplicado a un conjunto de variables mayor, pero para hacer la explicación más visual se han utilizado únicamente dos variables. El objetico es reducir el error que se comete por lo que hay que llegar al valor mínimo de la función representada. La expresión de la función a minimizar sería la siguiente:

$$f(x) = \frac{1}{2}x^T Ax - x^T b \tag{4.2}$$

Se empieza en un punto inicial que se denomina xo, una vez situado en este punto, el método analiza cuál es la pendiente descendente máxima en ese punto, para poder encontrar el camino que permite descender de una forma más rápida. Matemáticamente la pendiente de una variable la calculamos a partir de la derivada, pero, al estar trabajando con un número mayor de variables se calcula el gradiente de la función de coste.

$$\nabla f(x) = \nabla\left(\frac{1}{2}x^t Ax - x^T b\right) = Ax - b \tag{4.3}$$

Igualamos a cero para minimizar de forma que:

$$A * x = b \tag{4.1}$$

Por lo que la ecuación 4.1 es un mínimo de la función f(x) pudiendo ser utilizada para encontrar los mínimos.

Este es el procedimiento que habría que resolver hasta que la función de error sea menor que la tolerancia exigida, alcanzando de esta forma la convergencia del método.

El método del gradiente conjugado es una modificación del método de gradiente descendiente que permite calcular de forma más eficiente las direcciones de búsqueda. Su algoritmo es el siguiente:

1 $d_{(0)} = r_{(0)} = b - \mathbf{A}x_{(0)}$
2 **for** $number\ of\ iterations$ **do**
3 $\quad \alpha_i = \frac{r_{(i)}^T r_{(i)}}{d_{(i)}^T A d_{(i)}}$
4 $\quad x_{(i+1)=x_{(i)}+\alpha_{(i)}d_{(i)}}$
5 $\quad r_{(i+1)} = r_{(i)} - \alpha_{(i)}\mathbf{A}d_{(i)}$
6 $\quad \beta_{(i+1)} = \frac{r_{(i+1)}^T r_{(i+1)}}{r_{(i)}^T r_{(i)}}$
7 $\quad d_{(i+1)} = r_{(i+1)} + \beta_{(i+1)}d_{(i)}$
8 **end for**

En este algoritmo *b* corresponde a las fuerzas externas, *A* es la matriz de coeficientes, *x* la variable en la que se busca iterativamente encontrar la solución, *α* es un factor llamado tasa de aprendizaje que permite avanzar en una dirección de descenso, *d* es la dirección de descenso y r es el vector de residuos que se quiere minimizar.

Por lo que, para obtener el mínimo como hemos visto en el proceso iterativo se define:

$$x_{i+1}=xi+\alpha_i*d_i \tag{4.4}$$

siendo αi (tasa de aprendizaje) un valor que minimiza la función. Este tiende a ser mayor en las iteraciones iniciales cuando se está lejos de la solución y más pequeño cuando el residuo es menor para aumentar la precisión cuando el algoritmo se está acercando a la solución. Se obtiene a partir de:

$$\propto_i = \frac{r_i^T r_i}{d_i^T A d_i} \tag{4.5}$$

Siendo el residuo de cada iteración:

$$ri = b - A * x_{\mathrm{i}} \tag{4.6}$$

En las dos últimas líneas del algoritmo, se calcula la nueva dirección mediante el conjugado

$$d_{\mathrm{i}} = u_{\mathrm{i}} + \sum_{K=0}^{i-1} \beta_{\mathrm{ik}} * d(k) \tag{4.7}$$

Si ui= ri

$$d_{(i+1)=r_{(i+1)}+}\beta_{(i+1)}d_{(i)} \tag{4.8}$$

Siendo:

$$\beta_{(i+1)} = \frac{r_{(i+1)}^{T} r_{(i+1)}}{r_{(i)}^{T} r_{(i)}} \tag{4.9}$$

Este método se repite hasta que el residuo obtenido sea menor que la tolerancia pudiendo admitir que el proceso ha logrado su convergencia.

Este método cumple dos condiciones:

-La dirección calculada en el método del gradiente es conjugada respecto a la matriz A, y todas las direcciones de descenso calculadas anteriormente:

$$(di)^{T} A d_j = 0 \quad 0 \leq j \leq i \tag{4.10}$$

-Los residuos son ortogonales a las direcciones de descenso de la iteración anterior.

$$r_i * d_j = 0 \quad 0 \leq j \leq i \tag{4.11}$$

[22]

4.3 Precondicionador multigrid geométrico.

Para acelerar la convergencia del gradiente conjugado distribuido, este, se precondiciona con un método multigrid. Realizando un ciclo en V al final de cada iteración del gradiente conjugado, como ya se había mencionado con anterioridad. Dicho precondicionamiento se puede realizar utilizando un multigrid geométrico o un multigrid algebraico. En este apartado, se enfoca sobre el multigrid geométrico y como este precondiciona al CG.

La principal característica de un multigrid geométrico es que este trabaja con mallas jerárquicas, teniendo que realizar distintas dicretizaciones para obtener las mallas más gruesas o utilizando alguna función o algoritmo para generar las mallas. Necesitando este método la

información de cada nivel de malla que se vaya a utilizar. En cambio, el método AMG requiere solo de la información de la malla fina para realizar todas las operaciones.

En la figura 4.3 se muestra resumido como el método multigrid precondiciona al gradiente conjugado. Se considera el caso más sencillo en el que sólo hay dos niveles de malla.

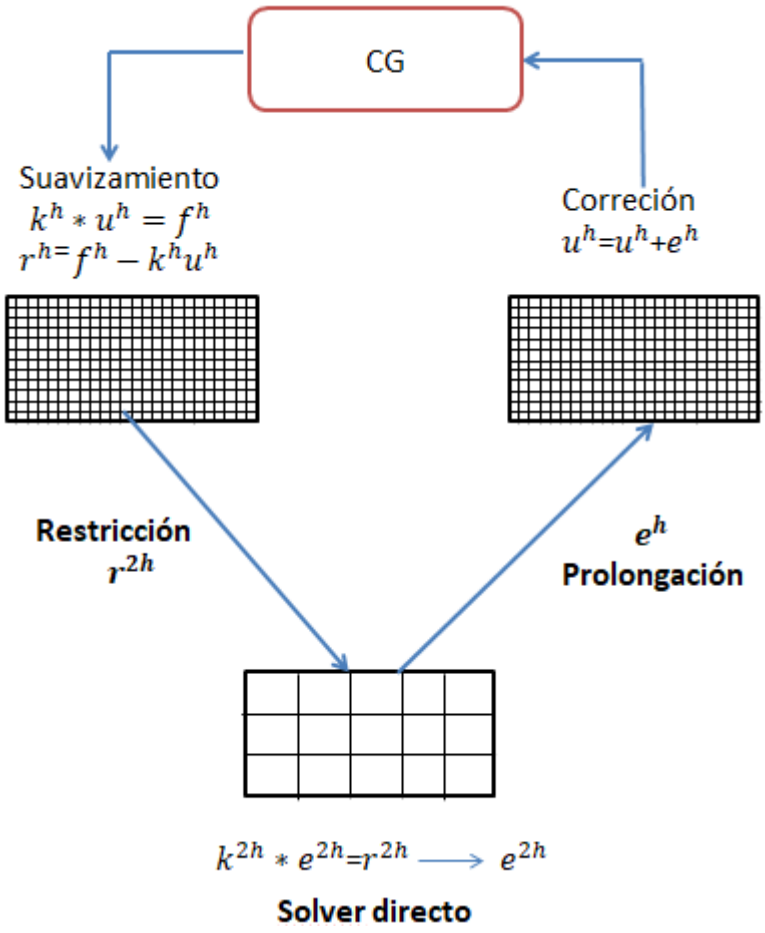


Figura 4.3 Precondicionamiento del CG

Las fases se han introducido en el comienzo del apartado. Ahora se va a entrar en detalle para el caso de un precondicionamiento multigrid geométrico. Las fases son las siguientes:

1) Relajación (suavizamiento). Se calcula el residuo del siguiente sistema de ecuaciones para el nivel inicial (malla fina):

$$k^h * u^h = f^h \tag{4.12}$$

Entonces el residuo es:

$$r^h = f^h - k^h u^h \tag{4.13}$$

El superíndice hace referencia al primer nivel, que es la malla fina y el 2h al segundo nivel, que es la malla gruesa. k^h es la matriz de coeficientes que se obtiene mediante el ensamblado, a partir de la discretización de malla fina y f^h representa el vector de fuerzas. Mediante el suavizamiento se obtiene una solución aproximada u^h. Una vez obtenida esta solución es posible calcular el residuo r^h mediante la ecuación 4.3.

2) Restricción. Esta fase tiene como objetico traspasar el residuo obtenido de la malla fina a la malla gruesa. Para ello se utiliza la matriz de restricción R. Operando de la siguiente forma

$$r^{2h} = R * r^h \tag{4.14}$$

Dónde R es obtenido a partir de funciones de interpolación entre la malla fina y la malla gruesa.

3) Cálculo del error. El error se calcula a partir de la siguiente ecuación:

$$k^{2h} * e^{2h} = r^{2h} \tag{4.15}$$

dónde k^{2h} es la matriz de coeficientes obtenida a partir de la dicretización de la malla gruesa. Estos sistemas de ecuaciones destacan por ser mucho más reducidos que los utilizados en la malla fina. Por lo tanto, a nivel de eficiencia computacional es viable utilizar un solver directo para resolverlo. El resultado es el vector de error en la malla gruesa e^{2h}.

4) Prolongación. El vector de error e^{2h} se proyecta de la malla fina a la malla gruesa siguiendo la siguiente operación.

$$e^h = \mathrm{P} * e^{2h} \tag{4.16}$$

Dónde la matriz P es usualmente la transpuesta de R.

5) *Corrección.* El vector de desplazamientos u^h se corrige de la siguiente forma:

$$u^h = u^h + e^h \tag{4.17}$$

El nuevo vector u^h otorga una mejor aproximación al resultado real que el vector original.

[20]

De esta forma se realiza un precondicionamiento utilizando un multigrid geométrico, realizándose al final de la fase de corrección una iteración del CG y repitiendo todo el proceso hasta que el residuo sea menor que la tolerancia exigida.

4.4 Precondicionador multigrid algebraico.

El método algebraico es un método alternativo al geométrico. Este sigue las mismas fases que el método geométrico, pero para realizarlas sólo requiere la información de una malla, la malla fina que proporciona el sistema de ecuaciones del primer nivel.

Este método trabaja únicamente con la matriz de coeficientes k^h. El vector de prolongación se obtiene a partir de operaciones de interpolación a partir de la matriz de coeficientes, siendo:

$$R = P^T \tag{4.18}$$

La matriz de coeficientes del nivel grueso se obtiene obtener a partir de la siguiente ecuación:

$$Ac = R * A * P = P^T * A * P \tag{4.19}$$

Esto implica que las operaciones de restricción, de prolongación y la matriz de coeficientes correspondiente a una malla gruesa ficticia y se pueden realizar completamente con la información correspondiente a la malla fina, no necesitando realizar discretizaciones a distintos niveles [21].

4.5 Discusión

La principal ventaja que ofrece el multigrid algebraico frente al geométrico es su flexibilidad. Al trabajar únicamente con la información que otorga una matriz de coeficientes, este es fácilmente adaptable a distintas aplicaciones. Pudiendo utilizarse como un algoritmo que funciona como una caja negra, sin requerir información sobre la geometría excepto en la malla fina.[16]. Otra gran ventaja es que este método puede aplicarse directamente a mallas estructuradas o desestructuradas [20].

Las desventajas de un GMG son su falta de flexibilidad al tener que utilizar la información de las mallas de cada nivel. Este se presenta como un método muy eficiente, pero con poca robustez y con un campo de aplicación más limitado.

Si bien es cierto que, se sabe a ciencia cierta que el AMG se conoce por ser un método más flexible, la eficiencia de un método, frente al otro, es un asunto a discutir. No siendo, en todos los casos más eficiente el geométrico, ni el algebraico. Al depender la eficiencia de estos de la naturaleza del problema a analizar. Además, las investigaciones que comparen estos métodos en sólidos 3D son escasas o inexistentes. Este estudio tiene como objetivo comparar el comportamiento del AMG con el GMG. Para ello, en el siguiente capítulo se van a realizar simulaciones de optimización topológica de sólidos de distinta complejidad, tanto 2D como 3D, resolviendo cada caso con un AMG y con un GMG, con la finalidad de determinar en qué casos uno es más eficiente que el otro.

5 Simulaciones

Este capítulo es el principal del estudio, en este se van a realizar simulaciones de distintos problemas de optimización topológica, resueltos mediante las herramientas y estrategias descritas anteriormente. Se van a realizar simulaciones sobre 5 modelos distintos, tanto 2D como 3D. Estos estarán discretizados con distintos tipos de malla, tanto estructuradas como desestructuradas. Dichas simulaciones se van a realizar utilizando en primer lugar un precondicionamiento mediante AMG, y posteriormente se resolverá el mismo problema precondicionado con GMG. El objetivo es estudiar estos métodos en distintos casos para poder obtener resultados y analizar en qué situaciones uno es más eficiente que el otro.

5.1 Definición de las simulaciones

En este apartado se definen los modelos objeto de estudio. Como se ha nombrado anteriormente, se van a estudiar 5 modelos, 2 modelos 2D y 3 3D.

- **Modelo 1: beam 2d**

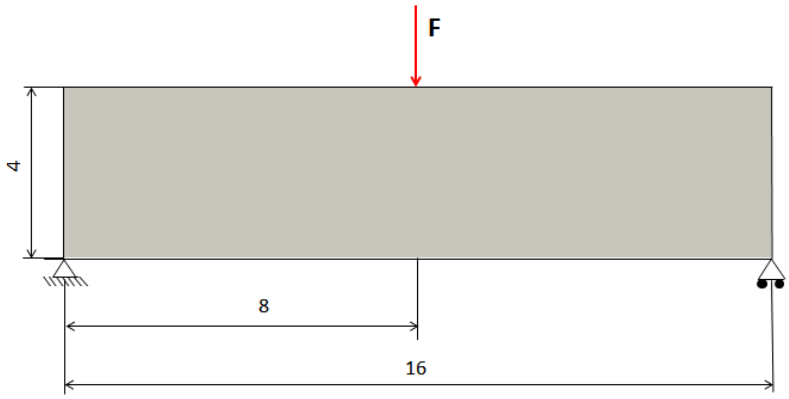


Figura 5.1 Modelo 1, beam 2D

Datos:

Modelo	sbeam2d
F	1
Dimensiones principales	16mx4m
Cores	16
Fracción de volumen	0.3
Young	1
poisson	0.29
Filtro	Cónico
Radio de filtro	0.02 m
Criterio de convergencia	MMA
Niveles multigrid geométrico	6

❖ Modelo 2: sbeam 2d con agujero

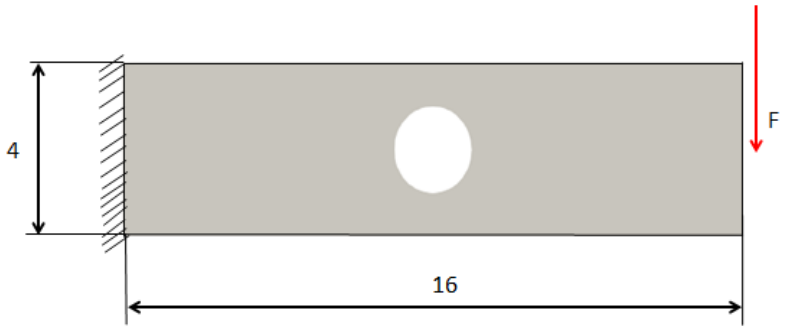


Figura 5.2 Modelo 2. sbeam 2D con agujero

Datos:

Modelo	sbeam2d con hueco
F	1
Dimensiones principales	16mx4m
Cores	16
Malla	Triangular
Fracción de volumen	0.3
Elementos	1523712
Incógnitas	1528832
Young	1
poisson	0.29
Filtro	Helmholtz
Radio de filtro	0.011719
Criterio de convergencia	MMA
Niveles multigrid geométrico	6

❖ Modelo 3: Sbeam 3d

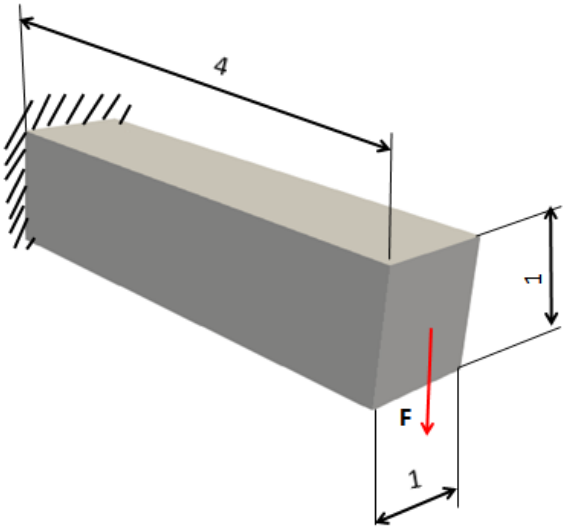


Figura 5.3 Modelo 3sbeam 3D

Datos:

Modelo	sbeam3d
F	-1e2
Dimensiones principales	4x1x1m
Cores	16
Young	1
poisson	0.3
Filtro	Cónico(estructurado) Helmholtz(desestructurado)
Radio de filtro	0,04
Criterio de convergencia	MMA
Niveles multigrid geométrico	4

❖ Modelo 4: Sbeam3d con agujero

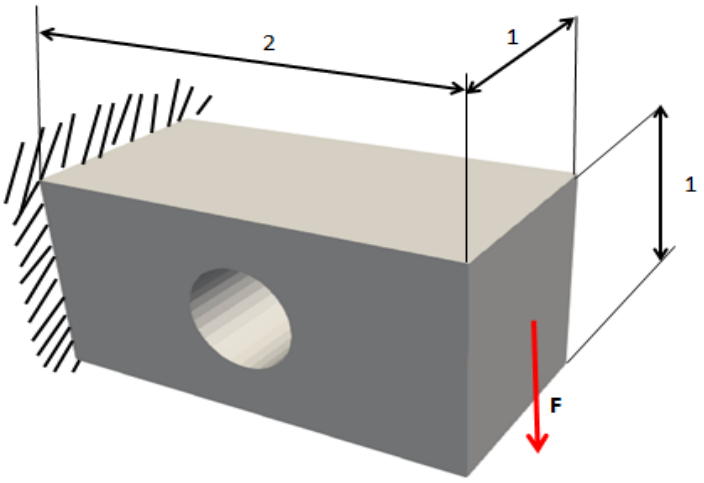


Figura 5.4 Modelo 4. Voladizo 3d con agujero

Datos:

Modelo	sbeam3d-hole
F	-1
Dimensiones principales	4x1x1m
Cores	16
Malla	Estructurada
Fracción de volumen	0.1
Elementos	61440
Incógnitas	201600
Young	1
poisson	0.3
Filtro	Cónico
Radio de filtro	0,08
Criterio de convergencia	MMA
Niveles multigrid geométrico	2

❖ Modelo 5: IPE 140

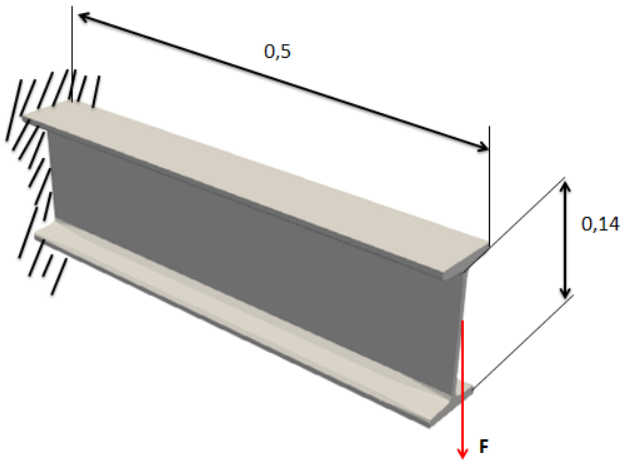


Figura 5.5 Perfil IPE

Datos:

Modelo	IPE 140
F	-1
Cores	16
Malla	Desestructurada
Fracción de volumen	0,3
Elementos	481280
Incógnitas	24707
Young	1
poisson	0.3
Filtro	Helmholtz
Radio de filtro	0,02
Criterio de convergencia	MMA
Niveles multigrid geométrico	3

5.2 Resultados de las simulaciones

En este apartado se muestran los resultados obtenidos al resolver el problema de optimización topológica para los distintos modelos. Las simulaciones han sido realizadas con AMD RAIZEN 9 3950 de 16 cores y 64 Gb de RAM. Se muestran en este apartado únicamente el resultado de la optimización topológica, sin incluir el análisis y la comparativa de los tiempos de resolución con los distintos métodos multigrid para los modelos. La comparativa del AMG y el GMG será abordada en el siguiente apartado.

❖ Modelo 1: Sbeam 2d.

El primer modelo a estudiar es un sólido en 2 D, este se ha simulado tanto con una malla estructurada (figura 5.6) cuadrada, como con una malla no estructurada triangular (figura 5.10). Los resultados se muestran a continuación:

Malla estructurada.

Elementos: 262144

Incógnitas: 526850

V=0,3

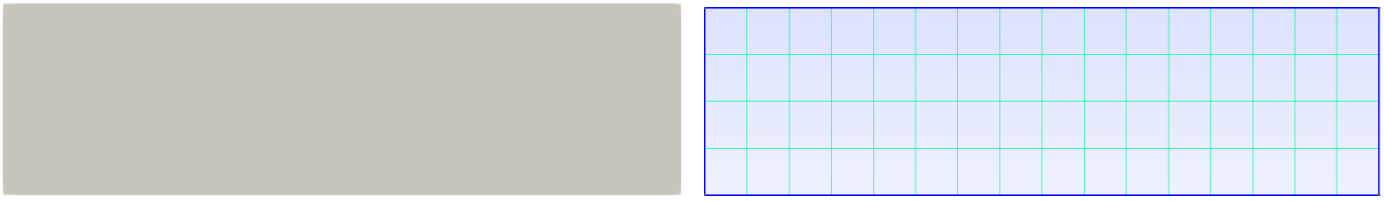


Figura 5.6 Malla beam 2D

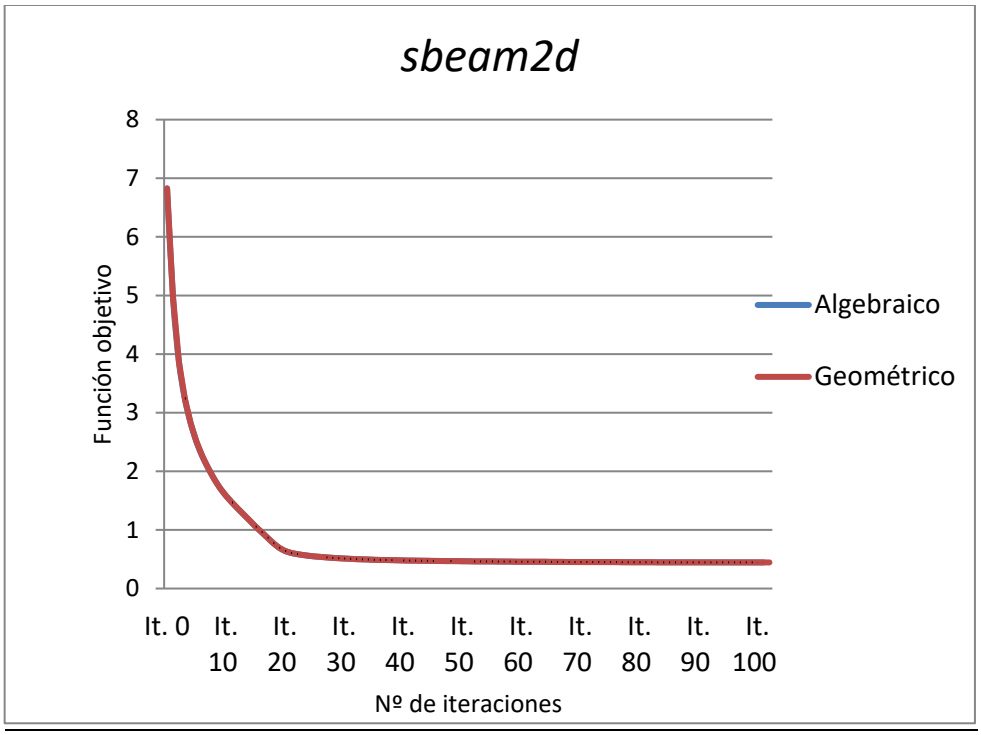


Gráfico 5.1Función objetico sbeam 2d

En la gráfica se puede ver como la función objetivo va disminuyendo en cada iteración. En cada una de las iteraciones se realiza el proceso definido en los apartados anteriores, en los que se resuelve el problema por MEF, se calcula la función objetivo y la sensibilidad, se aplica el filtro y posteriormente se realiza el algoritmo de optimización que permite actualizar las densidades, que en este caso es el MMA. Este proceso se vuelve a repetir si no se cumple el criterio de convergencia.

Los resultados obtenidos para la iteración 100 son los siguientes:

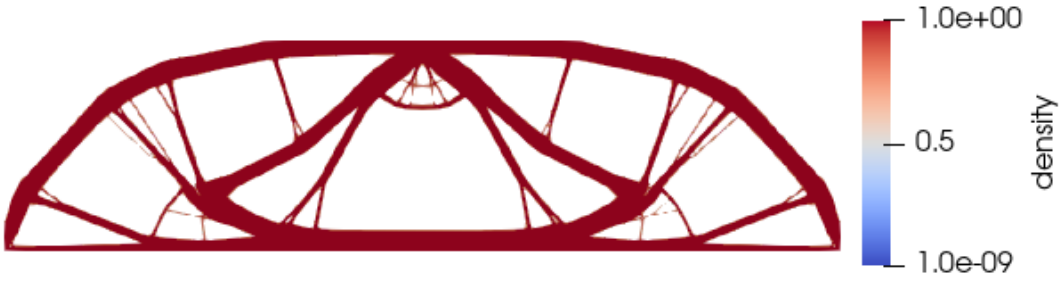


Figura 5.7 Optimización Beam 2D

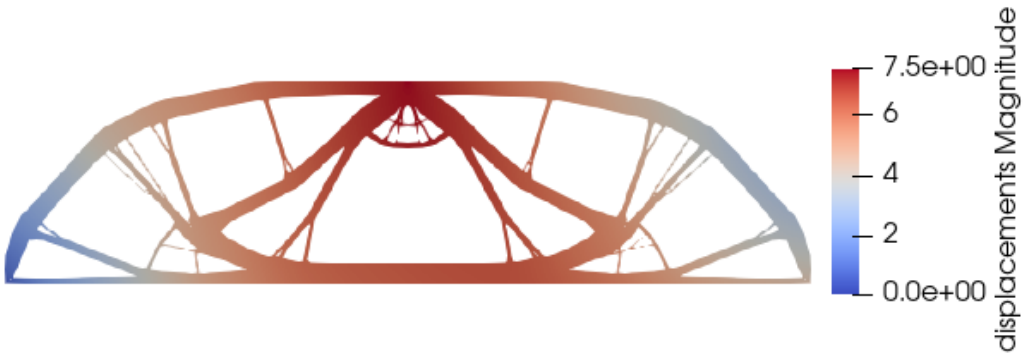


Figura 5.8 Desplazamientos en el modelo optimizado beam 2D

En la figura 5.7 se muestra el resultado de la simulación. En este se ha representado el campo de densidades. El proceso de optimización topológica ha conseguido encontrar la distribución óptima del material, consiguiendo que se obtengan elementos de densidad 1 o 0 (aproximadamente) y encontrando una solución de alta eficiencia, resultado de la minimización de la función objetivo. Dicha solución es capaz de soportar las cargas impuestas en las condiciones de contorno, con un volumen igual a 0,3 en comparación con el modelo original.

A continuación, se muestra la distribución del material:

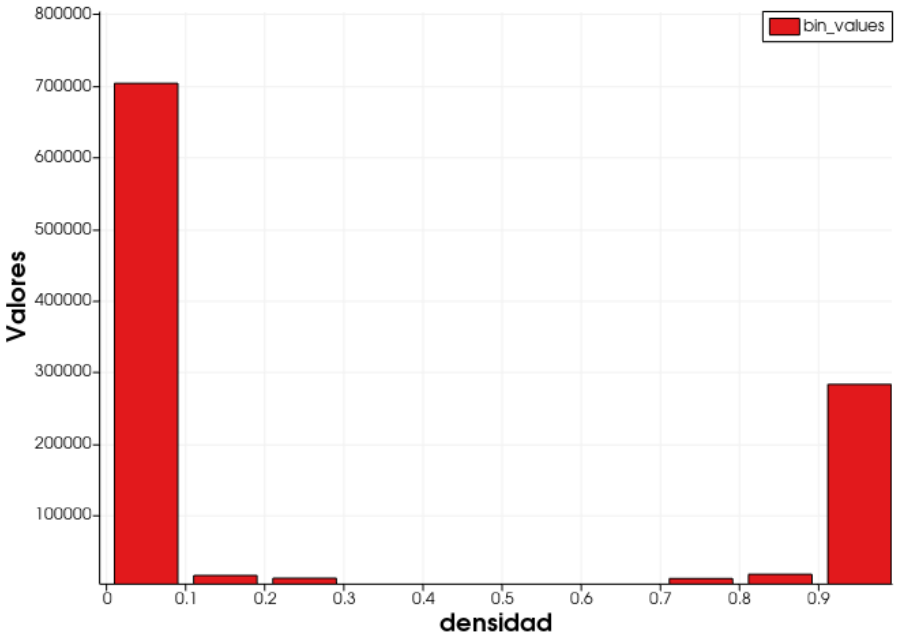


Figura 5.9Distribución del material modelo 1

En la figura se puede ver el número de elementos que tiene el modelo en función de los distintos rangos de densidades obtenidos. Destaca como no se obtienen únicamente resultados “0” o “1”. Esto se debe a que, para resolver el problema de optimización, se ha realizado una regularización, incluyéndose las densidades intermedias, con el objetivo de poder realizar

algoritmos basados en gradiente. El método SIMP realiza una penalización, favoreciendo la formación de elementos vacíos o llenos, pero a pesar de realizar dicha penalización, la formación de elementos con densidades intermedias es inevitable.

A continuación, se muestran los resultados obtenidos con la malla no estructurada.

Malla no estructurada.

Elementos: 679936

Incógnitas: 682498

V=0.3

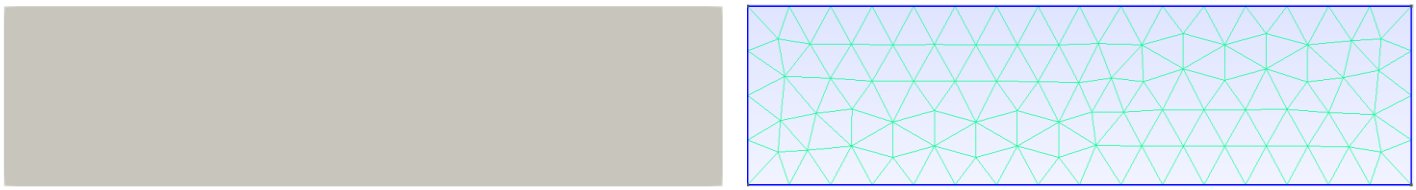


Figura 5.10 Beam 2D malla no estructurada

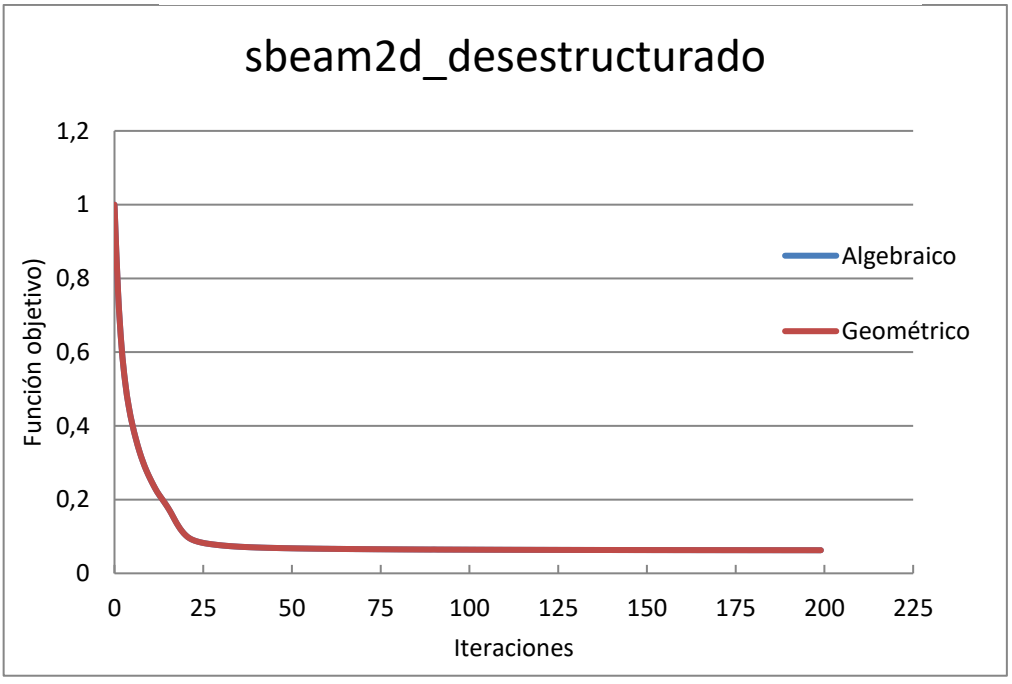


Gráfico 5.2Función objetivo sbeam 2D malla desestructurada

En este gráfico se muestra la función objetivo, esta se minimiza progresivamente en cada iteración. Destaca como el descenso no ocurre de forma proporcional en cada iteración. En las primeras 25 iteraciones se ha conseguido minimizar la función casi por completo. A partir de esta iteración la función objetivo tiene una forma asintótica, que va descendiendo poco a poco.

Elegir un criterio de parada en optimización es un asunto difícil, ya que, en muchos casos, si no se marca un número de iteraciones máximas, el problema podría no alcanzar convergencia, rondando alrededor de un valor de la función objetivo, sin alcanzar la parada. También, hay que tener en cuenta el gran coste computacional que se tiene. El problema ha sido resuelto casi por completo en la iteración 25. A partir de esta, el descenso de la función es muy bajo y el coste computacional de realizar, en este caso 175 iteraciones muy alto. No existe una regla matemática que diga cuando parar, esta decisión se tomará según los recursos computacionales o temporales y el grado de eficiencia en cuanto a resultado que se quiera conseguir.

Es importante destacar como, a pesar de ser el mismo modelo, las funciones objetivo obtenidas en una malla estructurada en comparativa con una estructurada no son idénticas. El discretizar de distinta forma hace que se estén resolviendo problemas con cierta similitud, pero distintos.

Los resultados obtenidos son los siguientes:

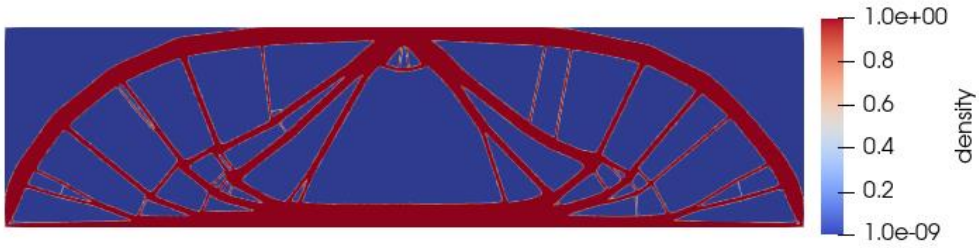


Figura 5.11 Sbeam 2D desestructurado, campo de densidades sin filtrar

Figura 5.12 Sbeam 2D desestructurado, campo de densidades filtrado

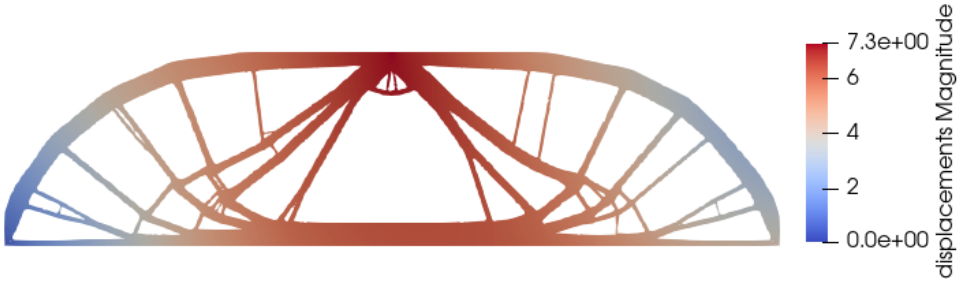


Figura 5.13 Sbeam 2D desestructurado, campo de desplazamientos

Se obtienen resultados muy similares en comparación con la malla estructurada, en la figura 5.11 se ha representado el campo de densidades sin ningún filtro visual posterior. Se puede ver como la mayoría de elementos son o de densidad mínima o "1". Pero, también hay una menor medida de elementos de densidad intermedia.

❖ Modelo 2: sbeam 2d con hueco.

El modelo 2 es similar al anterior, pero esta vez, con un hueco centrado. Se estudiará con una malla desestructurada como se muestra en la figura 5.14.

Malla no estructurada

Elementos: 1523712

Incógnitas: 1528832

Volumen: 0,3

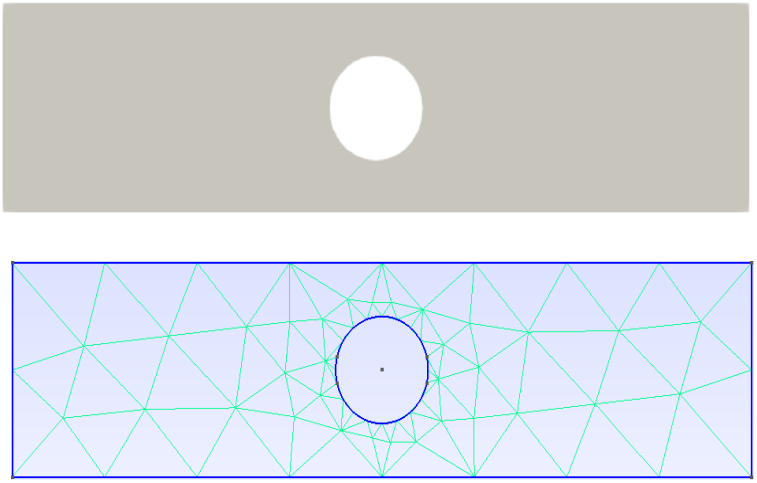


Figura 5.14 Sbeam 2D-hole malla no estructurada

En la gráfica 5.3 se muestra la función objetivo obtenida:

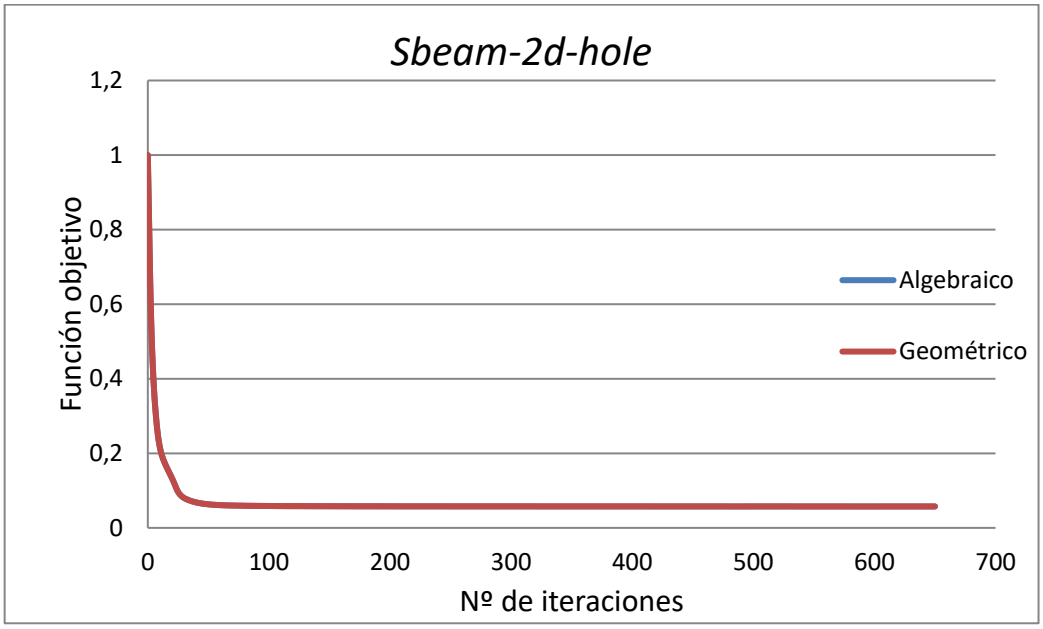


Gráfico 5.3 Beam 2D –hole, malla no estructurada

Al igual que en los casos anteriores, la función desciende rápidamente en las iteraciones iniciales y posteriormente se obtienen un comportamiento asintótico.

Los resultados que se obtienen son los siguientes:

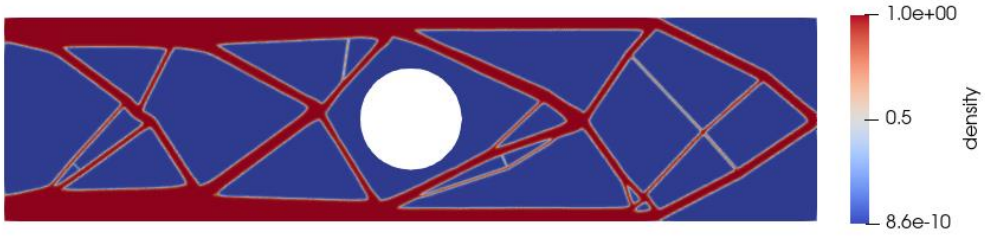


Figura 5.15 Sbeam 2D-hole, campo de densidades sin filtrar

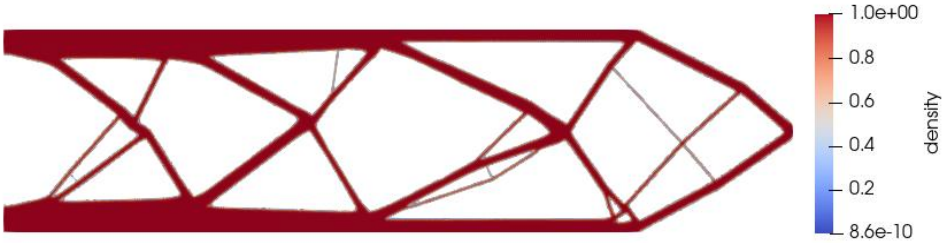


Figura 5.16 Sbeam 2D-hole, campo de densidades filtrado

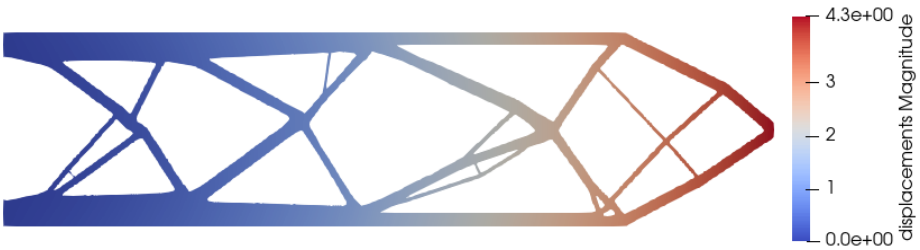


Figura 5.17 Sbeam 2D-hole, campo de desplazamientos

Se muestra como se ha producido el aligeramiento masivo de material y se obtienen resultados esperados, ya que, el problema de un voladizo sometido a una carga es un clásico en optimización topológica y la solución obtenida se asemeja mucho a otros estudios consultados.

La distribución del material se muestra en la figura 5.18.

Figura 5.18 Distribución del material modelo 2

Destaca como la penalización ha conseguido eliminar casi todos los elementos de densidades intermedias.

❖ Modelo 3: Sbeam 3d.

Una vez estudiados los modelos 2D se van a estudiar los 3D, para medir el distinto comportamiento de los métodos multigrid en estos casos.

Como se ha mencionado con anterioridad, en este apartado simplemente se va a mostrar las soluciones en cuanto a la geometría obtenida, así como la función objetivo y la distribución del material. El análisis del coste computacional de las simulaciones con ambos métodos multigrid será analizado en el siguiente apartado con el fin de realizar comparaciones de forma más ágil.

Estructurada tetraédrica

Elementos: 876544
Incógnitas: 473379
V015

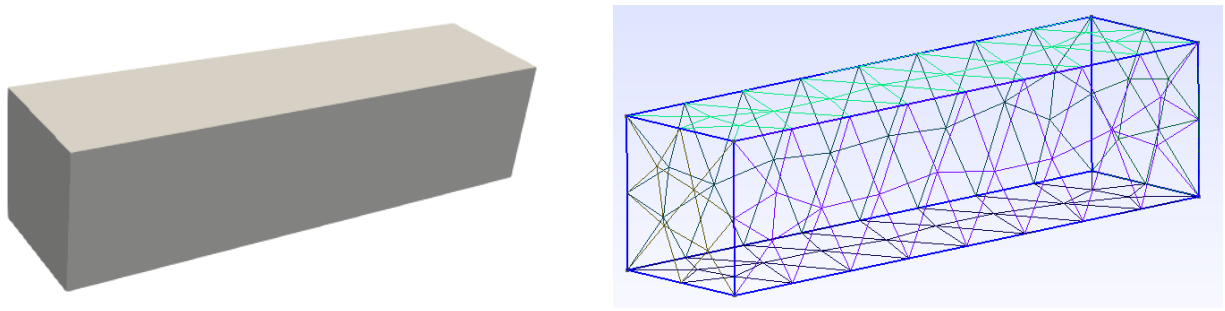


Figura 5.19 Malla Sbeam 3D, desestructurada tetraédrica

En el gráfico 5.4 se ha representado la función objetivo. Esta se ha representado en una escala logarítmica de base 5, con el fin de observar con más detalles como se minimiza la función en las primeras iteraciones. Se observa como la función se minimiza a un ritmo muy rápido en las primeras tres iteraciones. De la iteración 3 a la 25 la función se va disminuyendo más lentamente, pero la minimización es apreciable. A partir de la iteración 25 se alcanza la tendencia asintótica característica, minimizando a un ritmo casi inapreciable en la gráfica, pero consiguiendo optimizar detalles del modelo.

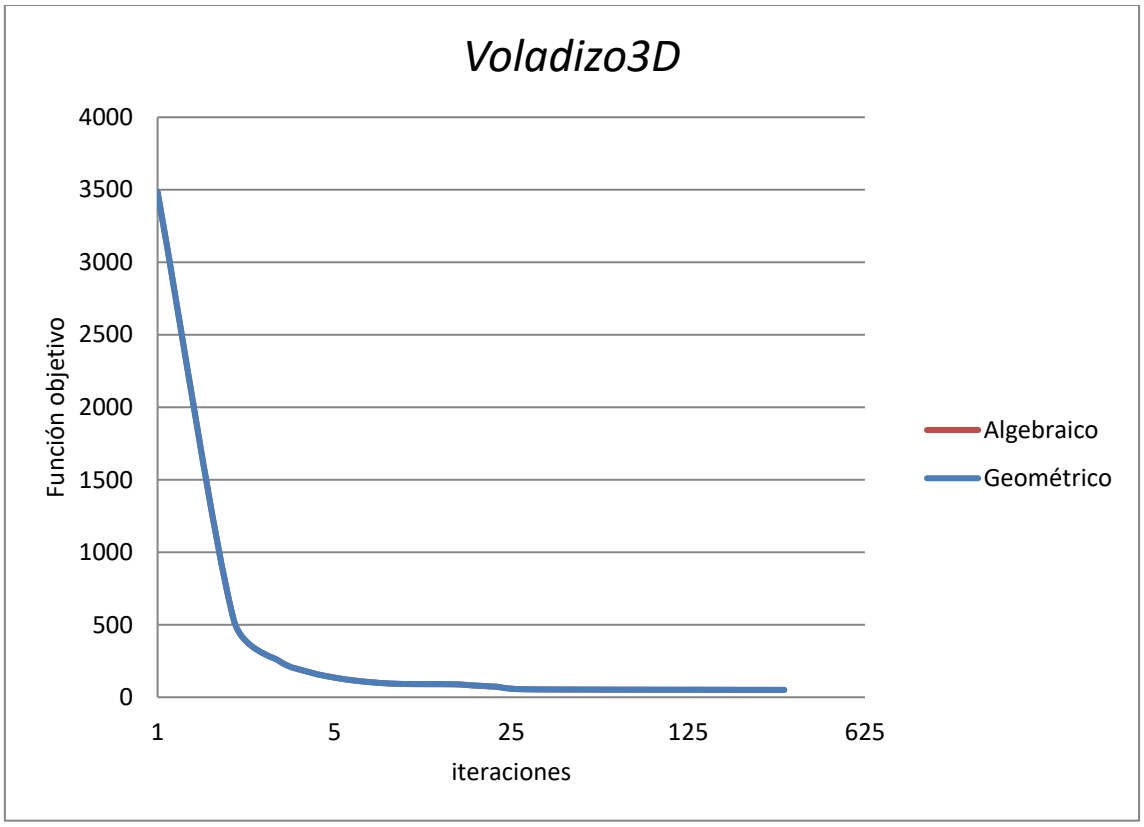


Gráfico 5.4 Función objetivo voladizo 3D

Los resultados obtenidos con una fracción de volumen de 0,15 y una malla desestructurada tetraédrica son los siguientes:

Figura 5.20Densidades voladizo 3d, malla desestructurada

Figura 5.21 Desplazamientos voladizo 3D, malla desestructurada

En la figura 5.22 se puede ver como se ha distribuido el material en función de los distintos valores de densidad, destacando que la malloría de valores son" 0" 0 "1".

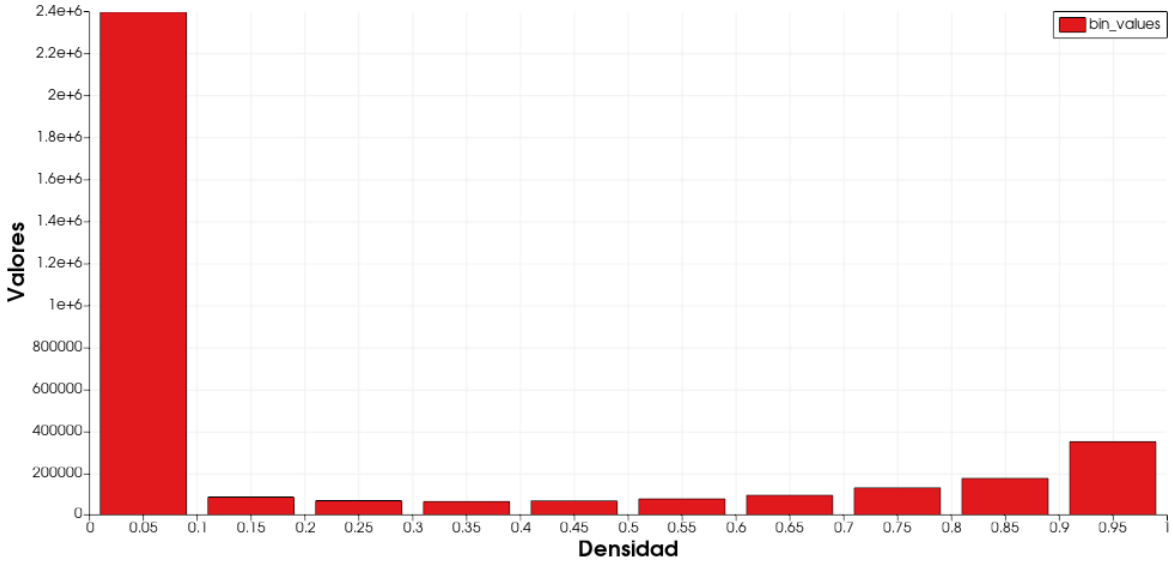


Figura 5.22 Distribución del material voladizo 3D

Estructurada cúbica

Elementos: 131072
Incógnitas: 421443
V=0,2

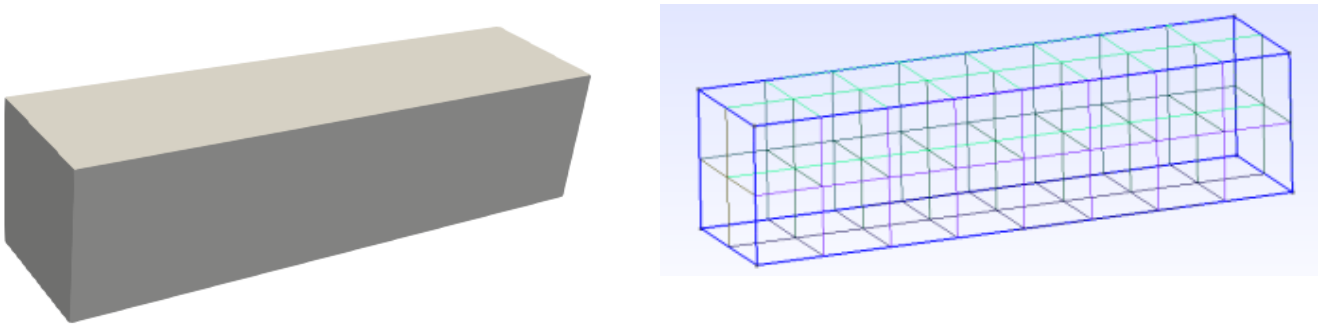


Figura 5.23Voladizo 3D, malla estructurada cúbica

Se ha estudiado el mismo modelo con una malla estructurada y una fracción de volumen de 0,2. El resultado se muestra en las figuras 5.24 y 5.25

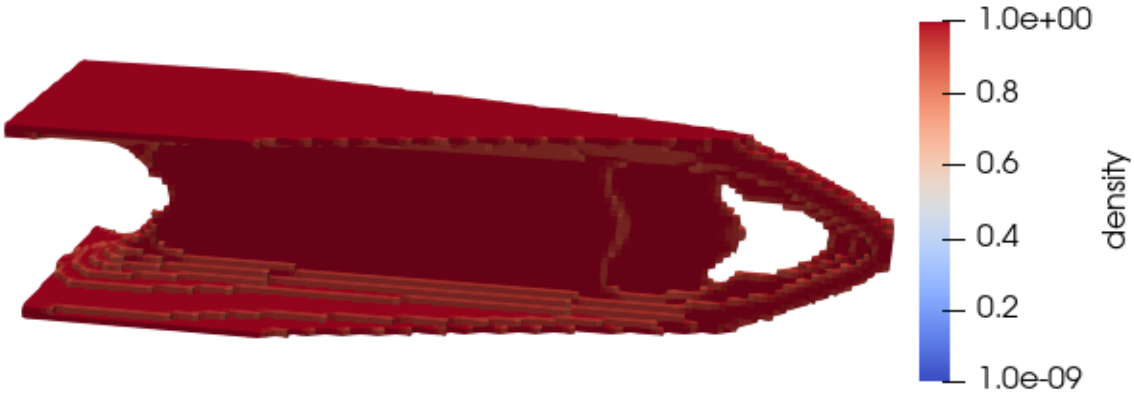


Figura 5.24 Densidades Voladizo 3D, malla estructurada

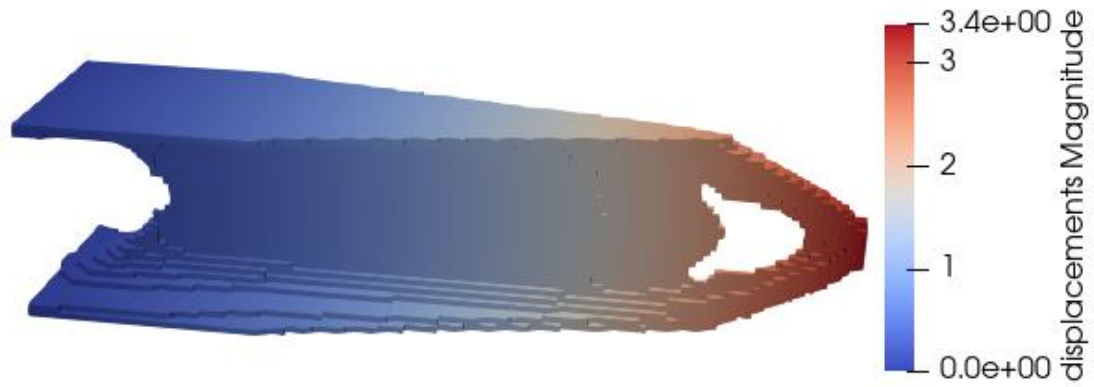


Figura 5.25Desplazamientos voladizo 3D, malla estructurada

❖ Modelo 4: sbeam 3d-Hole.

El Modelo 4 corresponde a un voladizo 3d con un agujero. Este se ha estudiado con una fracción de volumen de 0,1 y una malla estructurada cúbica (figura 5.26).

Malla estructurada

Elementos: 61440

Incógnitas: 201600

V=0,1

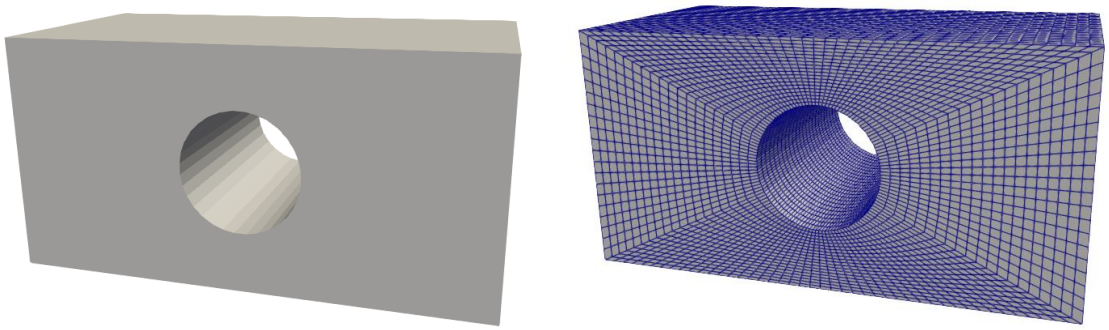


Figura 5.26Malla sbeam ·3D-hole

La función objetivo obtenida es la siguiente:

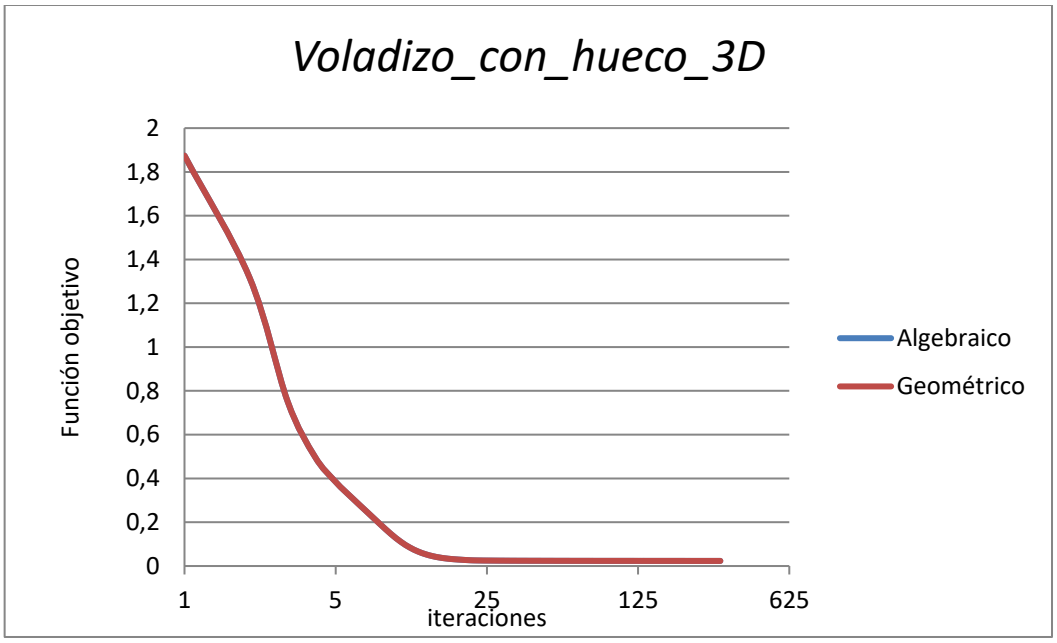


Gráfico 5.5 Función objetivo Sbeam 3d-hole

Se observa como esta se minimiza describiendo una recta descendente de pendiente casi constante en las primeras 23 iteraciones. A partir de esta iteración se alcanza el comportamiento asintótico característico.

Los resultados se muestran en las figuras 5.27 y 5.28.

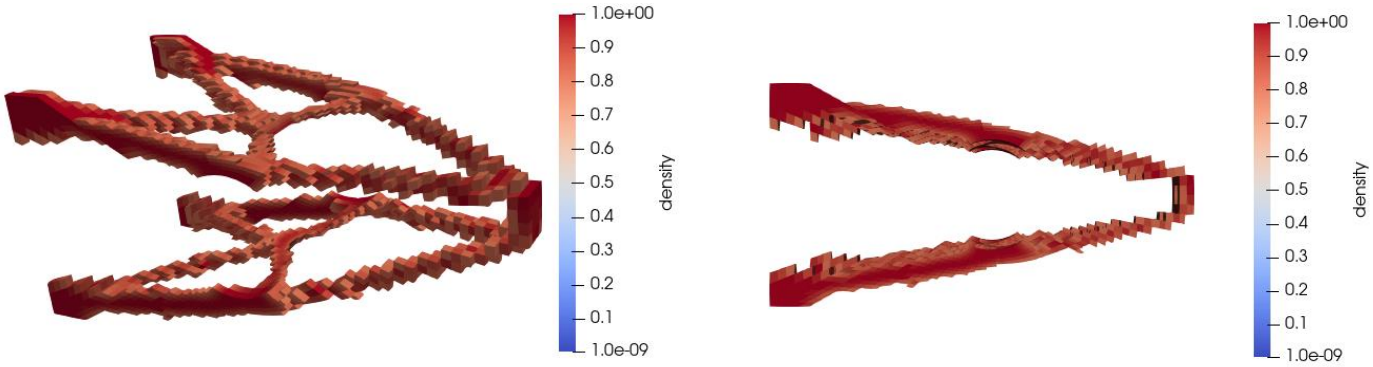


Figura 5.27 Densidad sbeam 3D-Hole

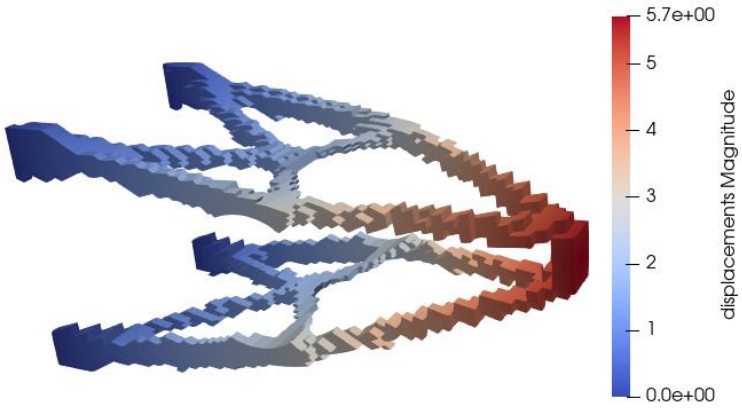


Figura 5.28 Desplazamientos sbeam 3D-Hole

La distribución del material en función de la densidad se muestra en la figura 5.29.

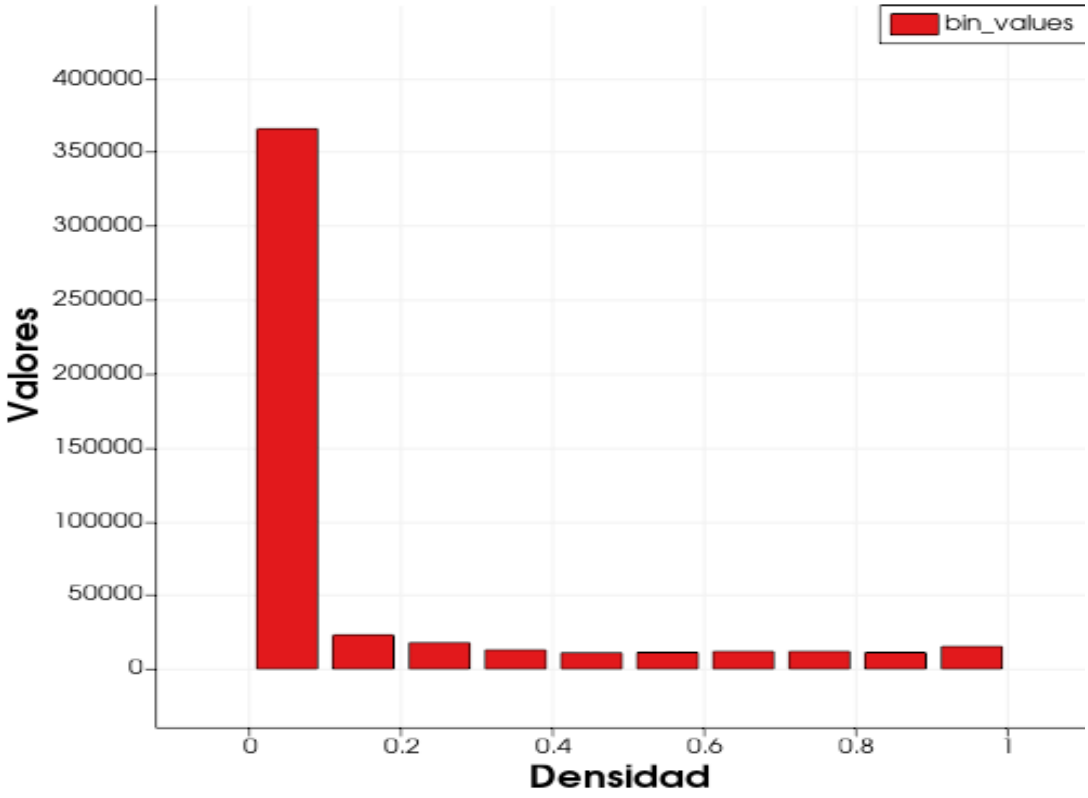


Figura 5.29 Distribución de los valores en función de la densidad sbeam 3d-hole

❖ Modelo 5: IPE

El último modelo a estudiar es un voladizo con perfil IPE 140, discretizado con una malla no estructurada (figura 5.30).

Elementos: 481280
Incógnitas: 274707
V=0,3

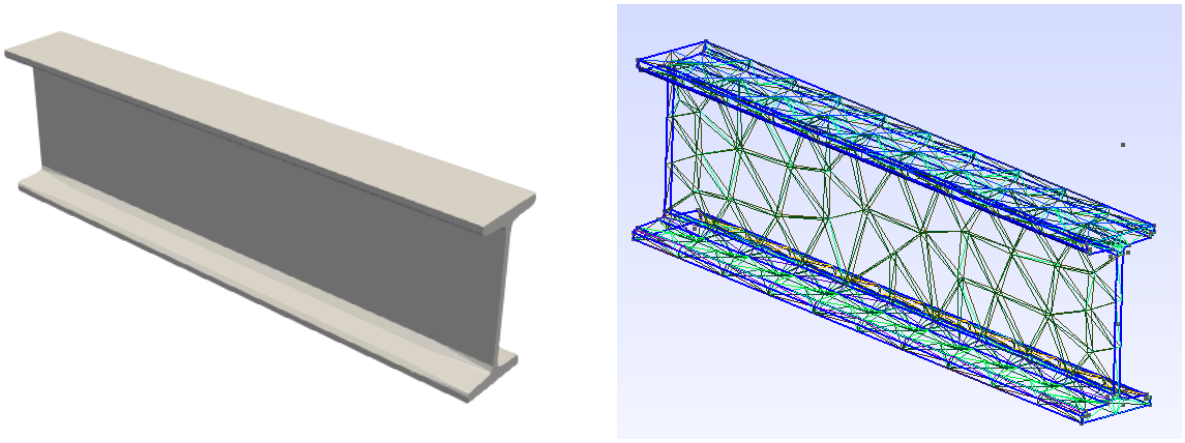


Figura 5.30 Malla perfil IPE

La función objetivo se muestra a continuación:

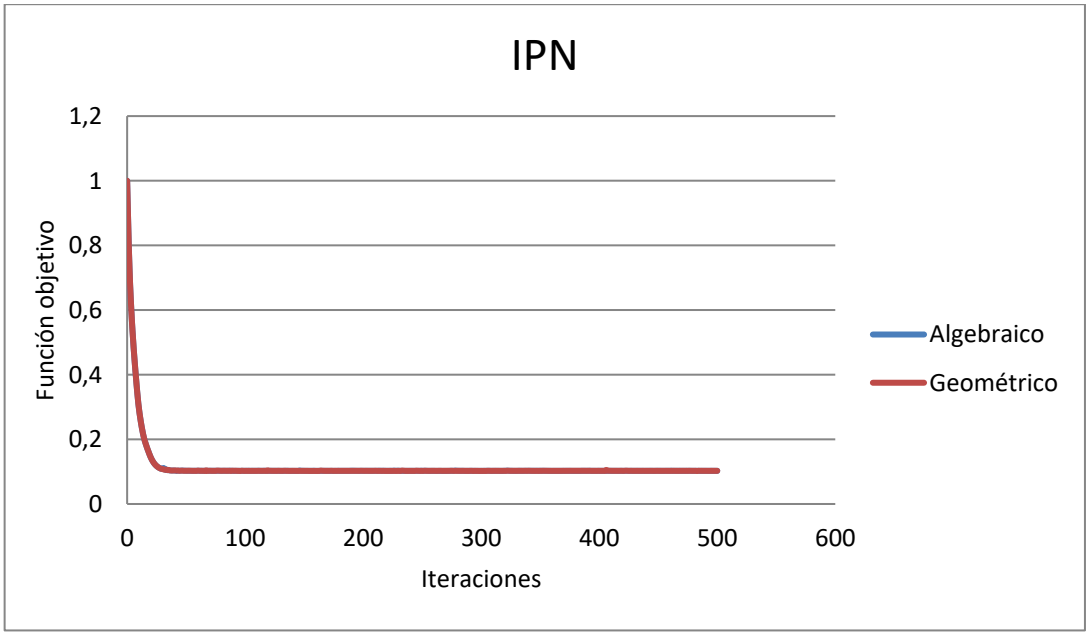


Gráfico 5.6 Función objetivo perfil IPE

El comportamiento de la función objetivo es similar a los modelos anteriores.

Los resultados, así como la distribución del material en función de la densidad se muestran en la figura 5.31, 5,32 y 5.33.

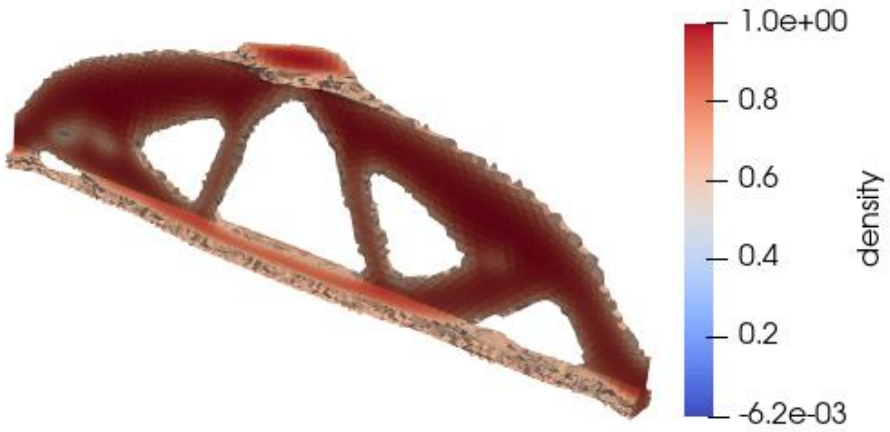


Figura 5.31: Densidad IPE

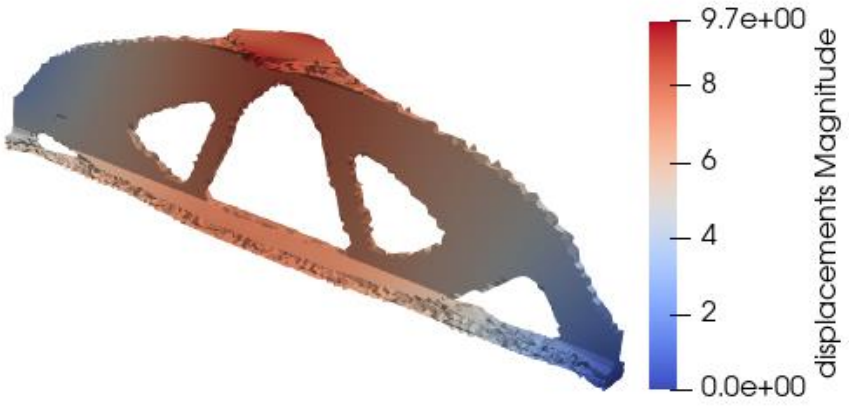


Figura 5.32: Desplazamientos IPE

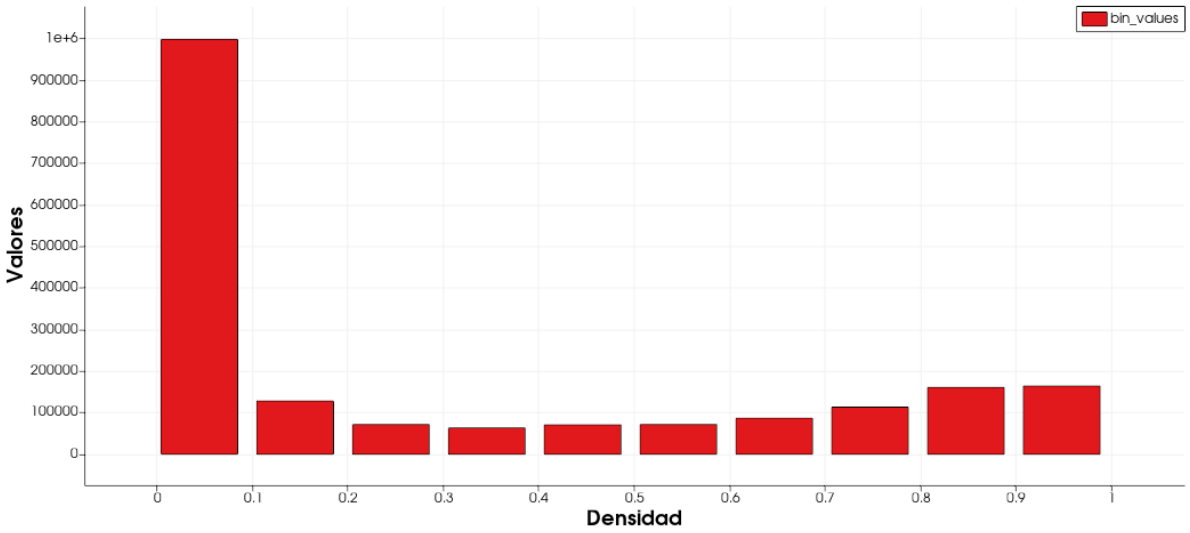


Figura 5.33 Valores en función de la densidad modelo IPE

5.3 Comparativa multigrid algebraico y geométrico

Este capítulo es el principal del estudio, en este se muestran los resultados obtenidos en cuanto a coste computacional de las simulaciones descritas en el apartado anterior. Se representarán los tiempos necesarios para realizar cada simulación mediante multigrid algebraico y geométrico y se discutirá sobre los resultados obtenidos para cada modelo.

❖ Resultados Modelo 1. Sbeam 2d

Malla estructurada

Elementos: 262144

Incógnitas: 526850

V=0,3

En el gráfico 5.7 se muestran los resultados obtenidos con la malla estructurada:

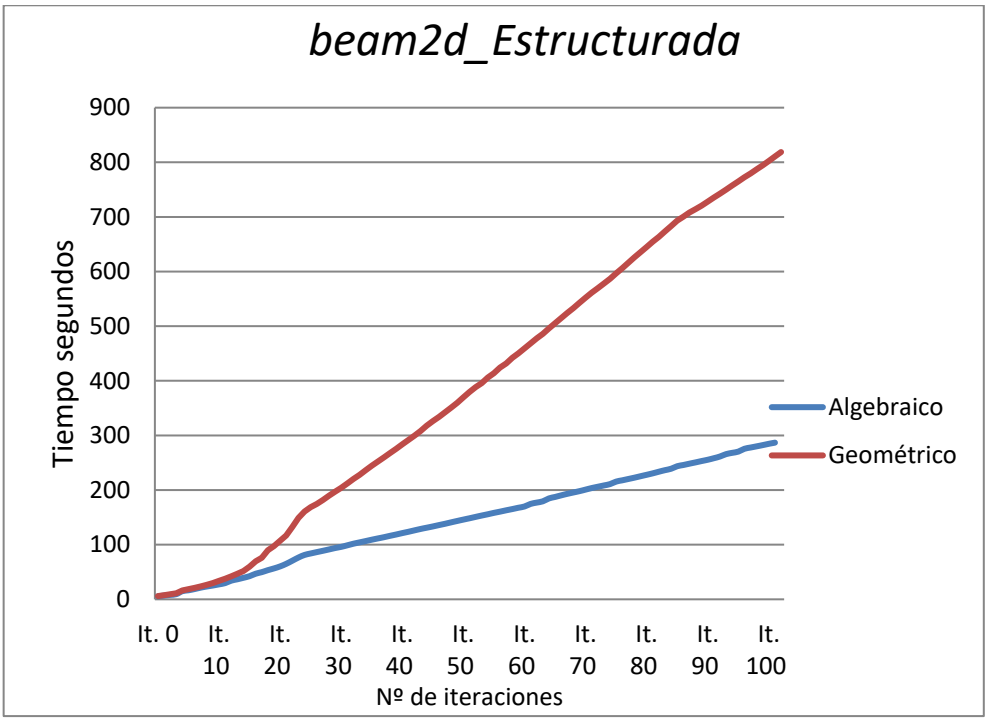


Gráfico 5.7 Comparativa métodos multigrid beam 2D_Estructurada

Malla desestructurada

Elementos: 679936

Incógnitas: 682498

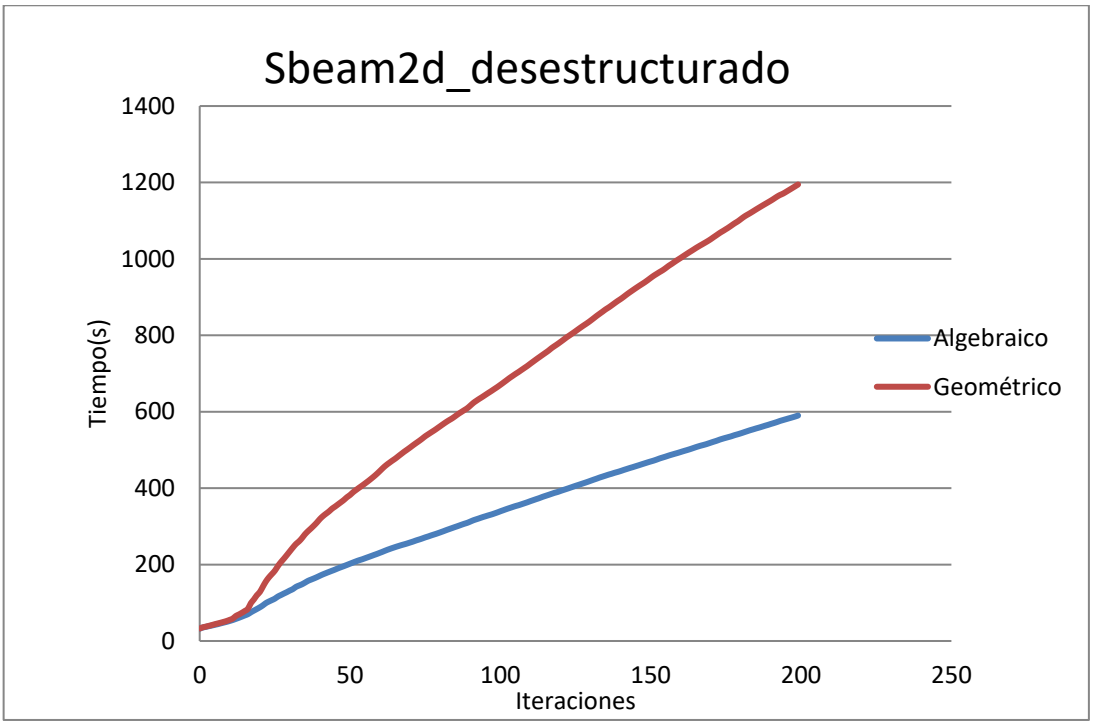


Gráfico 5.8 Comparativa métodos multigrid beam 2D_Desestructurada

En los gráficos 5.7 y 5.8 se ha representado los costes en cuanto a velocidad de procesamiento para cada iteración de un modelo 2d para una malla estructurada y desestructurada, respectivamente. Destaca como en ambos casos el multigrid algebraico es más eficiente. Este tiene un menor coste computacional por iteración, consiguiendo que se obtengan mayores velocidades de procesamiento para este modelo 2D.

Con una malla estructurada la diferencia en cuanto a tiempo de procesamiento entre el multigrid algebraico y el geométrico es de 500 segundos, mientras que con la malla desestructurada esa diferencia final es de unos 600 segundos. Siendo como se ha mencionado en ambos casos el multigrid algebraico más eficiente. Este resultado se puede decir que en un principio es inesperado, ya que, se sabe de forma empírica que el AMG funciona de forma más eficiente en mallas desestructuradas, dónde se tengan que realizar un gran número de iteraciones, debido a que este ofrece una gran velocidad de resolución por iteración. El multigrid geométrico es más eficiente en mallas estructuradas por regla general, dónde en número de iteraciones no sean tan altas. En el modelo estructurado se consigue una velocidad del multigrid algebraico respecto al geométrico de 2,66. Mientras, en el desestructurado de 2. Es decir, se está comportando mejor el AMG en el modelo discretizado con una malla desestructurada.

Esto incumple la "regla" empírica mencionada con anterioridad. La optimización topológica es un problema muy complejo y no existe una regla fija que diga con que método multigrid se va a alcanzar mayores velocidades. Si se pueden observar ciertas tendencias, como que, en ambos casos del modelo beam 2D, el multigrid algebraico es más eficiente. El tiempo computacional para resolver un problema depende del pre-post suavizamiento, el ciclo en V, el solver "directo" utilizado, el número de iteraciones, el tiempo necesario para realizar cada iteración y el número de condición del problema.

En optimización hay que tener muy en cuenta el condicionamiento del problema. Este depende de factores como la relación de aspecto, la malla utilizada (si esta es muy fina, el problema estará mal condicionado, los softwares comerciales suelen incluir herramientas de mejora de malla para mejorar el condicionamiento del problema), el radio del filtro etc… El multigrid algebraico ofrece más eficiencia en sistemas mal condicionados en comparación con el geométrico, donde el número de iteraciones que se tengan que realizar es alto, debido a que este ofrece una mayor velocidad por iteración en comparación con el geométrico.

La razón de que en el modelo utilizado con una malla estructurada (gráfico 5,7) sea más eficiente el algebraico, es que, este se condicionada mal. En el gráfico 5.7 se muestran como en las primeras 13 iteraciones el AMG y el GMG tienen velocidades muy parecidas. Pero a partir de la 13 el problema se condiciona mal, provocando un punto de inflexión, que se traduce en una pérdida de la velocidad en el geométrico, traduciéndose en un mal condicionamiento del sistema y otorgando una ventaja del algebraico respecto al geométrico.

De la comparativa de las dos gráficas se muestra cómo, aunque teóricamente el algebraico es más eficiente en mallas estructuradas, no es posible saber con certeza como se va a comportar un sistema. En la optimización no se está resolviendo un solo problema, se resuelve una gran multitud (uno por iteración) de problemas de elementos finitos, dónde dicho problema cambia en cada iteración y puede evolucionar hacia un sistema mejor o peor condicionado.

A continuación, se muestra el siguiente modelo, sbeam 2D-hole.

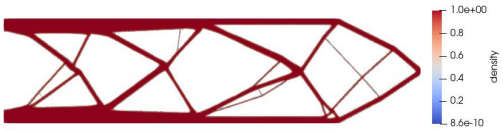
❖ Modelo 2: sbeam 2d con hueco.

Malla no estructurada

Elementos: 1523712

Incógnitas: 1528832

Los resultados con una malla no estructurada se muestran en el gráfico 5.9.



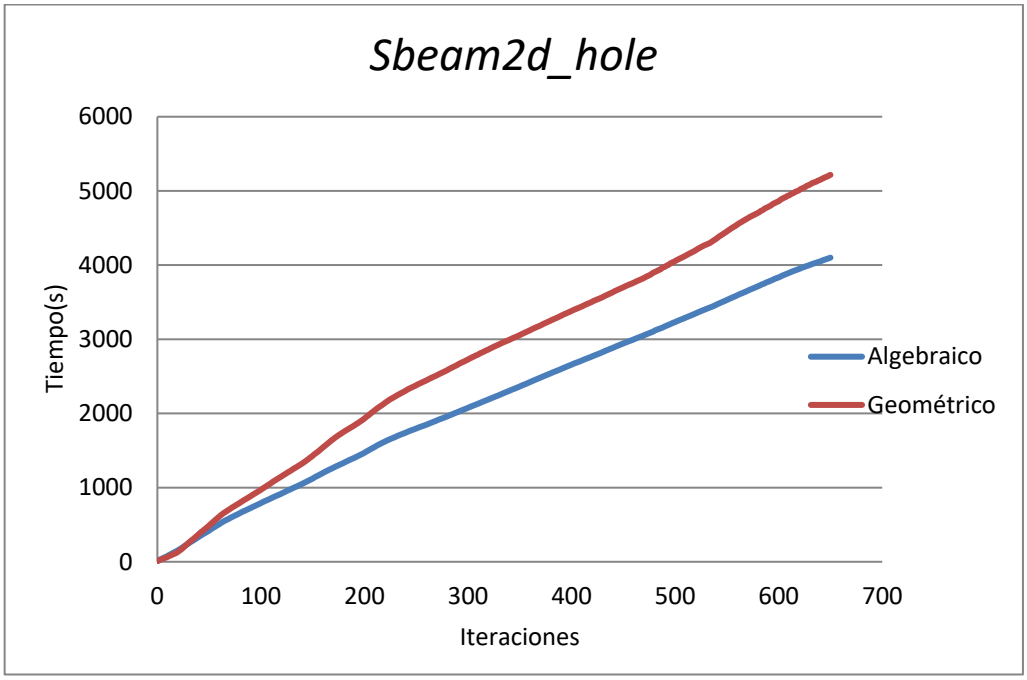


Gráfico 5.9 Comparativa métodos multigrid voladizo 2d con hueco

Se muestra como para este sólido 2D el algebraico vuelve a ser más eficiente. Destaca como en las primeras 100 iteraciones el incremento de eficiencia obtenido es poco apreciable. Pero el algebraico necesita menos tiempo por iteración, ganando eficiencia en comparación con el GMG cuantas más iteraciones se necesiten, consiguiendo que la diferencia en la solución total sea considerable.

Una vez discutidos los problemas 2D se va a estudiar el comportamiento de los métodos multigrid en sólidos en 3D.

❖ Modelo 3: sbeam 3d

Malla desestructurada

Elementos: 876544
Incógnitas: 473379
V015
Radio 0,125

En el gráfico 5.10 se muestra el tiempo de resolución con cada método multigrid para una malla desestructurada.

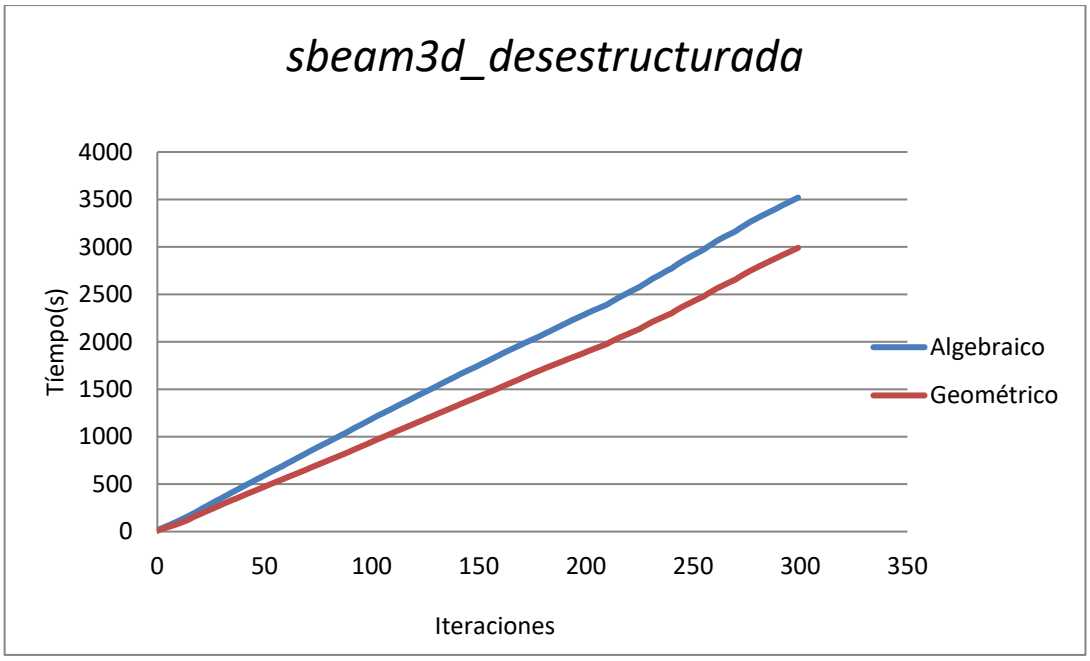
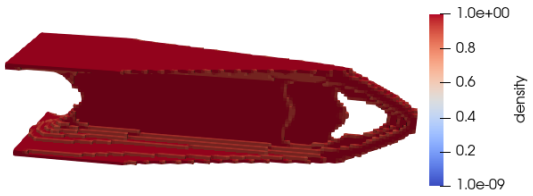


Gráfico 5.10 Comparativa métodos multigrid voladizo 3d, malla desestructurada

Malla estructurada cúbica

Elementos: 131072
Incógnitas: 421443
V=0,2
Radio 0,0625



En el gráfico 5.11 se muestra el tiempo de resolución con cada método multigrid para una malla estructurada cúbica.

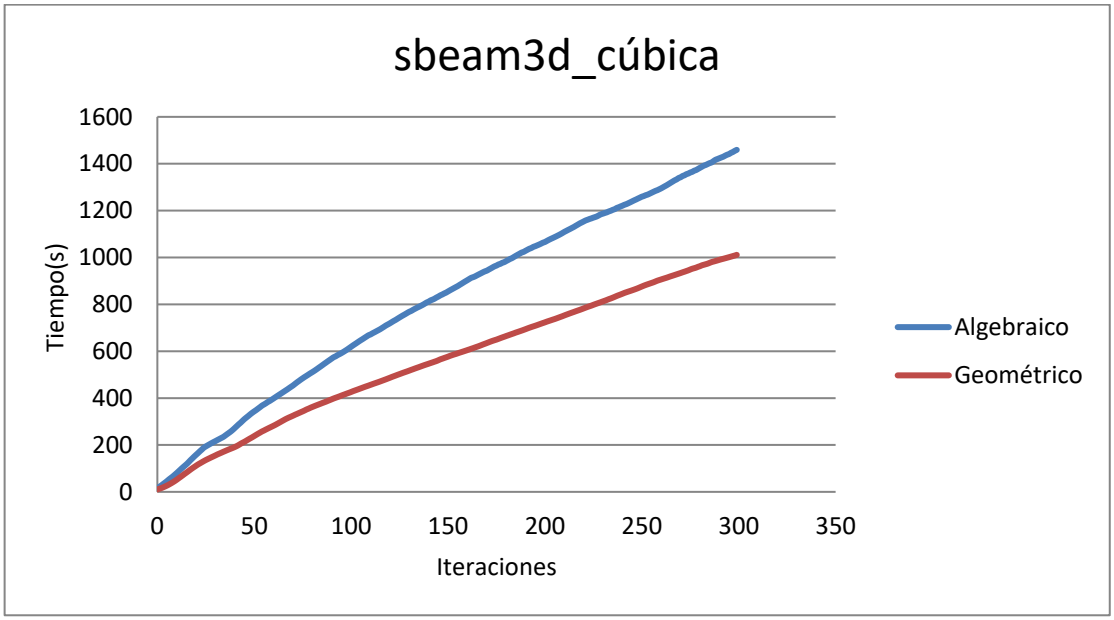


Gráfico 5.11 Comparativa métodos multigrid voladizo 3D, malla cúbica.

Destaca como el multigrid geométrico es más eficiente que el algebraico en los modelos 3D. Esto se debe a que, al realizar el ciclo en V en ·tres dimensiones, se necesita un menor número de saltos para hacer que la matriz de coeficiente reduzca su tamaño, debido a que este opera sobre la geometría. Mientras, el multigrid algebraico se abstrae de la geometría y opera solo sobre la matriz de rigidez, necesitando más saltos para llegar a la malla gruesa. La principal consecuencia de esto es que se obtenga una ventaja al utilizar el multigrid geométrico en sólidos 3D, reduciendo los tiempos de engrosamiento y reducción de la malla.

Al igual que en el caso anterior, hay que tener en cuenta el condicionamiento del problema. En el gráfico 5.11 se utiliza una malla estructurada cúbica. Dicha malla otorga una muy buena relación de aspecto. Esto mejora el condicionamiento del problema explicando la gran eficiencia que este consigue. En el gráfico 5.10 se utiliza una malla desestructurada. En este caso, el geométrico es más eficiente que el algebraico, aunque no con tanta claridad como en el caso anterior, posiblemente debido a que, la malla desestructurada está condicionando mal al problema, consiguiendo que la eficiencia de ambos métodos se asemeje.

A continuación, se muestra el modelo 4.

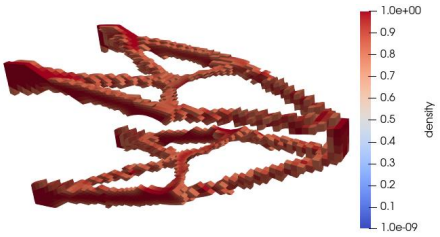
❖ Modelo 4: sbeam 3d-Hole

Malla estructurada cúbica

Elementos: 61440

Incógnitas: 201600

Radio 0,08



Los resultados se muestran en la gráfica 5.12

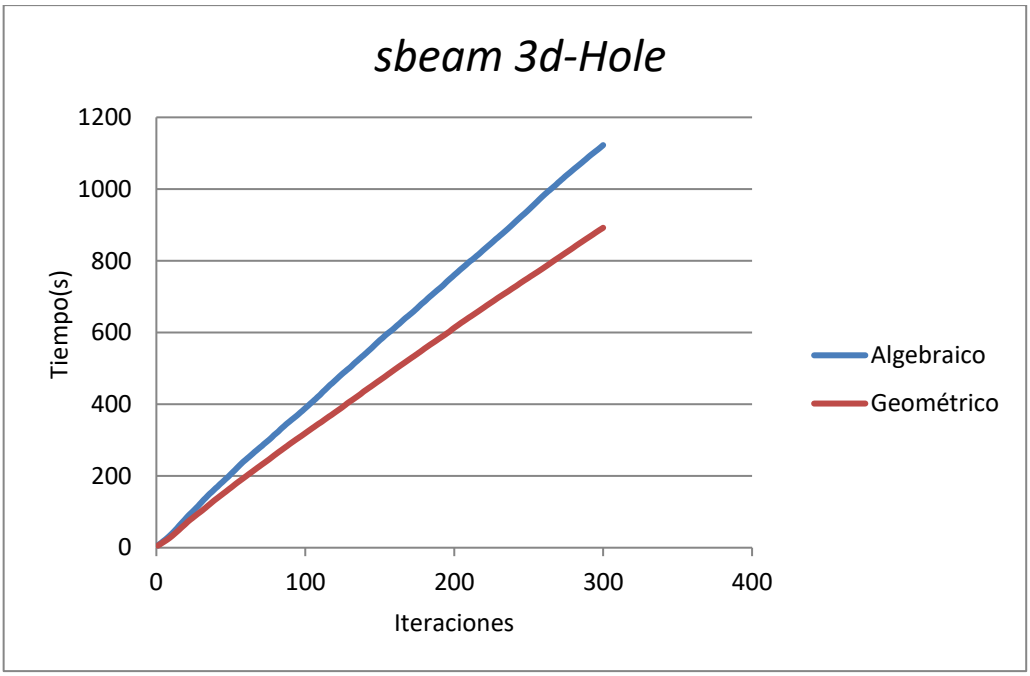


Gráfico 5.12 Comparativa métodos multigrid voladizo 3d con agujero

A continuación, se muestra el modelo 5

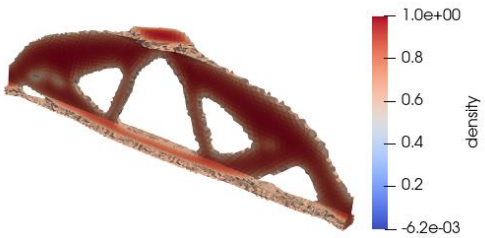
❖ Modelo 5: IPN

Elementos: 481280

Incógnitas: 274707

Radio 0,02

Los resultados son los siguientes:



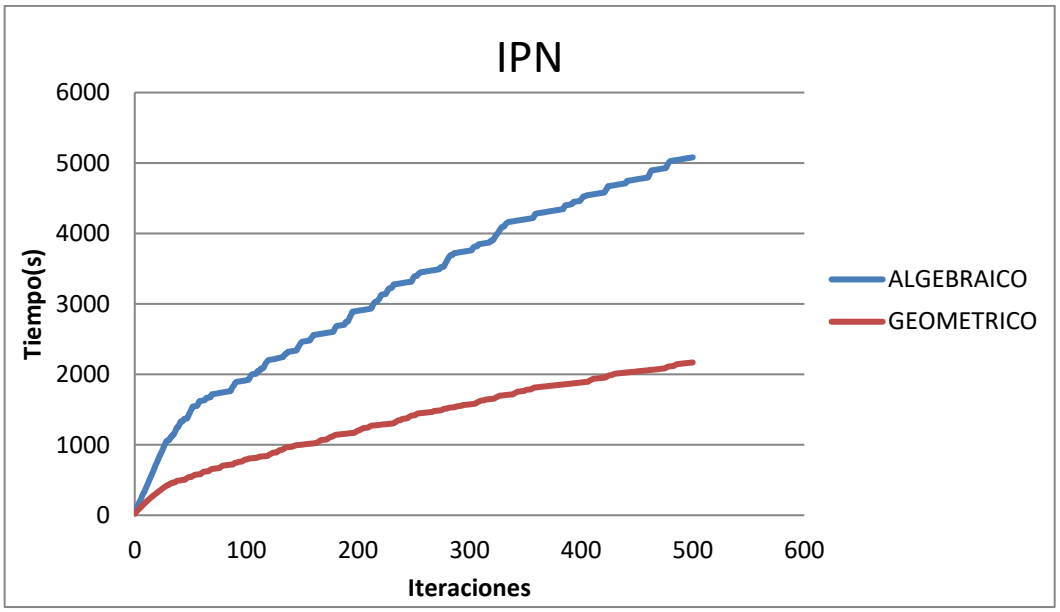


Gráfico 5.13 Comparativa métodos multigrid perfil IPE

Destaca que GMG vuelve a ser más eficiente en los modelos 3D. Esto se debe a que, como se ha explicado anteriormente el multigrid geométrico necesita menos saltos para realizar el ciclo en V, explicando de esta manera que este método sea más eficiente que el algebraico.

Que el sólido sea de 3D favorece la eficiencia del geométrico como se ha podido comprobar. Pero, este no es el único factor a tener en cuenta, influye el número de pre/post-smooth y el radio del filtro utilizado. Se han realizado más simulaciones para estudiar como este afecta al tiempo de solución necesario por ambos métodos multigrid.

Simulaciones con distintos radios

Las simulaciones han sido realizas con el modelo 3D, sbeam 3D, discretizado con una malla estructurada tetraédrica.

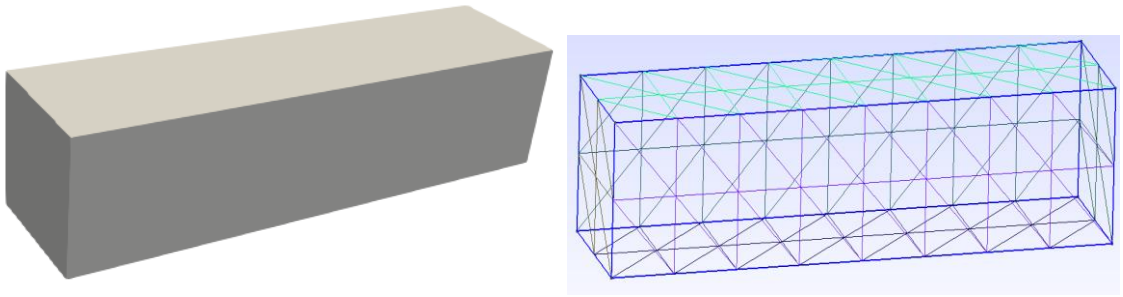


Figura 5.34 Sbeam 3D, malla estructurada tetraédrica

Elementos: 786432
Incógnitas: 421443
V012

El resultado de la optimización se muestra en la figura 5.35:

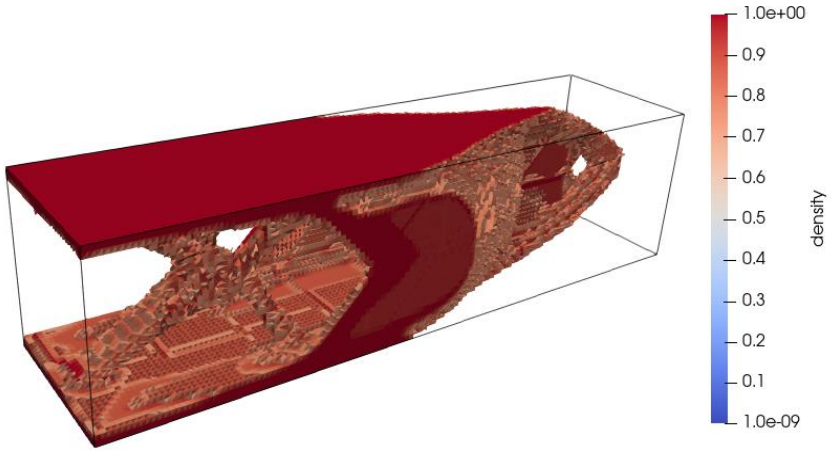


Figura 5.35Voladizo 3D, malla estructurada tetraédrica

La comparativa para distintos radios se muestra en las siguientes gráficas:

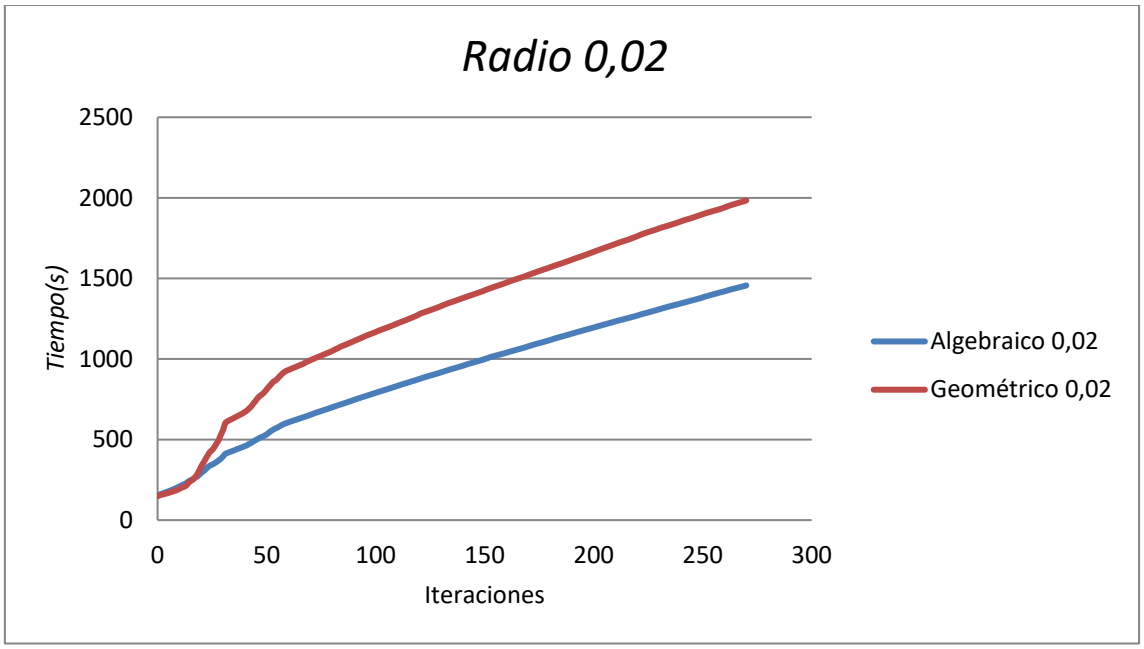


Gráfico 5.14 Comparativa multigrid con radio de filtro 0,02

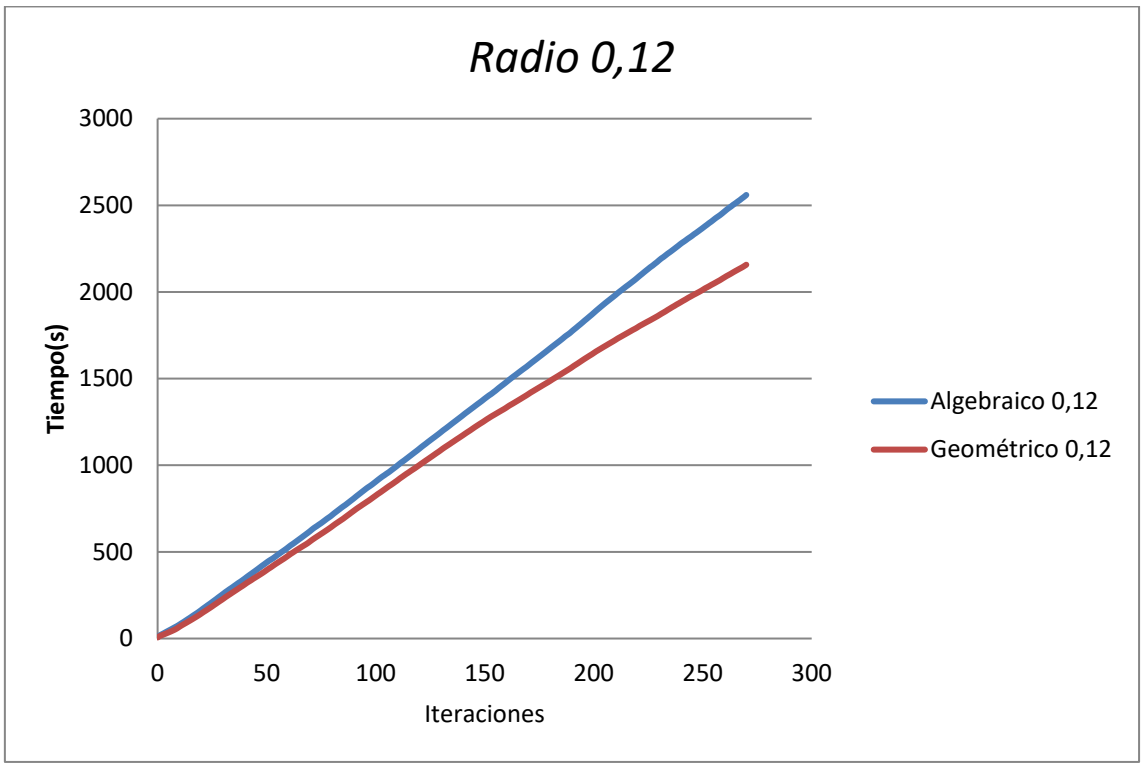


Gráfico 5.15 Comparativa multigrid con radio de filtro 0,12

Se muestra como al utilizar un radio de filtro grande, de 0,12, el geométrico resulta ser más eficiente que el algebraico. Pero, al utilizar un radio muy pequeño de 0,02 el algebraico gana en eficiencia. Esto se debe a que, utilizar un radio grande mejora el condicionamiento del problema y favorece que el geométrico tenga un menor coste computacional. Hay que tener en cuenta que, si se está utilizando una malla de gran calidad, muy fina, el utilizar un radio de filtro grande en el que se hace un promedio de los elementos no tiene sentido, lo que impide utilizar radios grandes que mejorarían el condicionamiento del problema.

En estas simulaciones se ha podido ver como no siempre el geométrico es más eficiente en 3D, este depende de cómo el sistema este condicionado. Resultando solo el multigrid geométrico más óptimo que el algebraico en unos casos muy concretos, dónde se simulan problemas 3D, bien condicionados, con un radio de filtro no muy pequeño.

A la hora de elegir un método multigrid para resolver un problema también hay que tener en cuenta la flexibilidad que otorga el AMG. Este al utilizar información de la matriz de coeficientes es fácilmente adaptable, otorgando una clara ventaja frente a un multigrid geométrico.

❖ **Para completar el estudio se han realizado simulaciones del modelo 4 con diferentes cores, con el fin de comprobar la escalabilidad de los dos métodos multigrid.**

Sbeam 3d-hole.Malla estructurada

Elementos: 61440

Incógnitas: 201600

En las gráficas 5.16 y 5.17 se han representado los resultados obtenidos para cada iteración con el multigrid algebraico y el multigrid geométrico respectivamente. Se han realizado las simulaciones con 1, 2, 4, 8 y 16 cores. Los resultados se muestran a continuación:

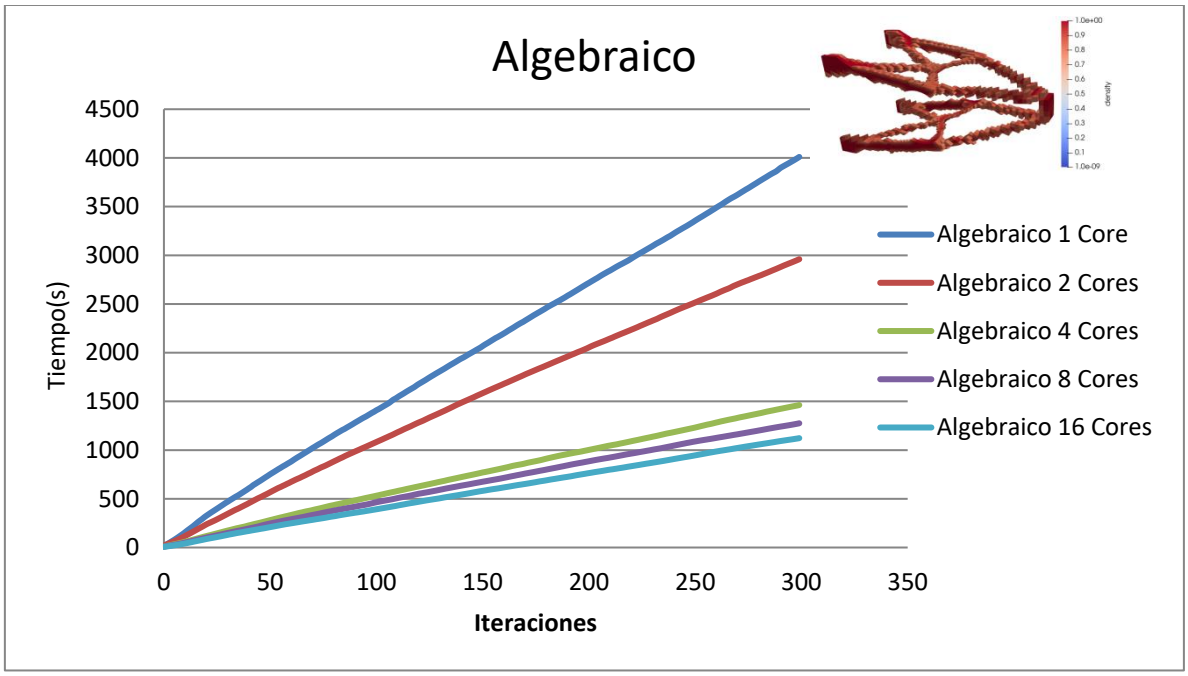


Gráfico 5.16 Comparativa con distintos procesadores multigrid algebraico

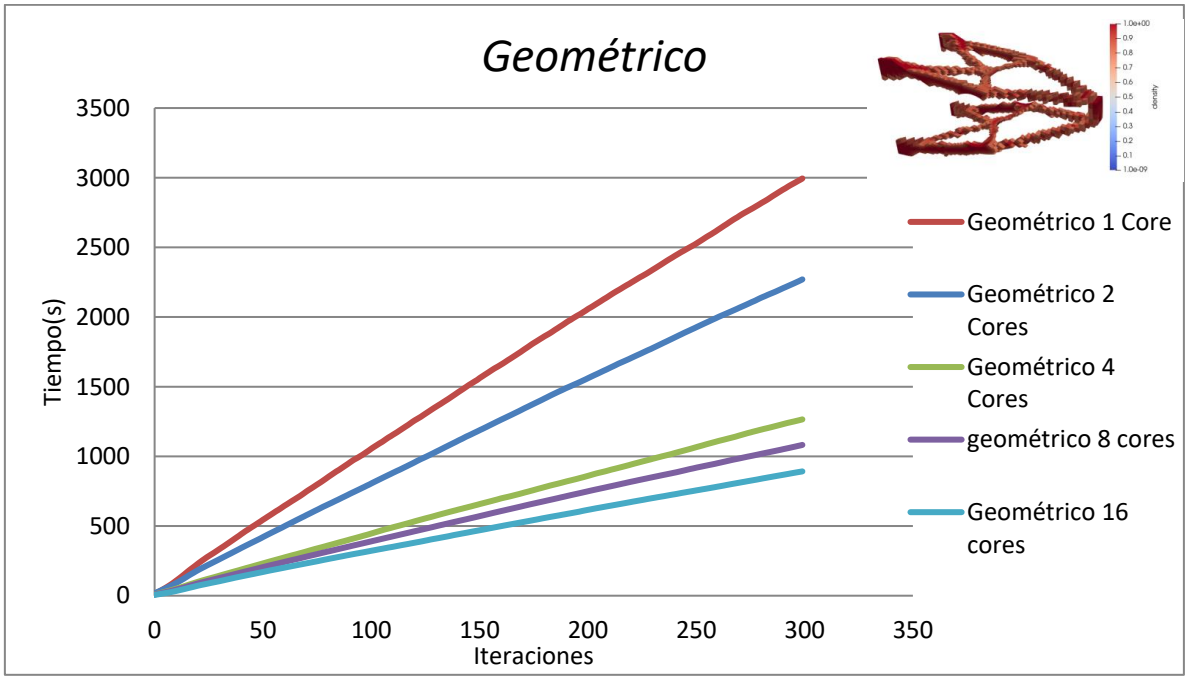


Gráfico 5.17 Comparativa con distintos procesadores multigrid geométrico

En estos gráficos se muestra la gran relevancia que tiene la computación en paralelo en la resolución de problemas complejos, en este caso de optimización topológica. Utilizar herramientas de computación paralela permite reducir drásticamente los tiempos necesarios en la computación como podemos ver en el gráfico 5.16, dónde el tiempo de computación necesario pasa de 4000 segundos con un core a 890 con 16 cores. Tanto las simulaciones realizadas con el multigrid algebraico como con el geométrico muestran un comportamiento parecido. En un inicio, hasta los 4 procesadores, el coste computacional obtenido se reduce de

forma drástica. A partir de los 4 cores se consigue alcanzar una mayor velocidad de procesamiento, pero en una menor medida. Esto se debe a que, al utilizar más procesadores estos tienen que realizar más tareas de comunicaciones reduciendo la eficiencia total que se obtiene. Para ver esto de forma más clara se ha representado el tiempo total obtenido y el speed-up con el multigrid algebraico y el geométrico, gráficas 5.18 y 5.19 respectivamente.

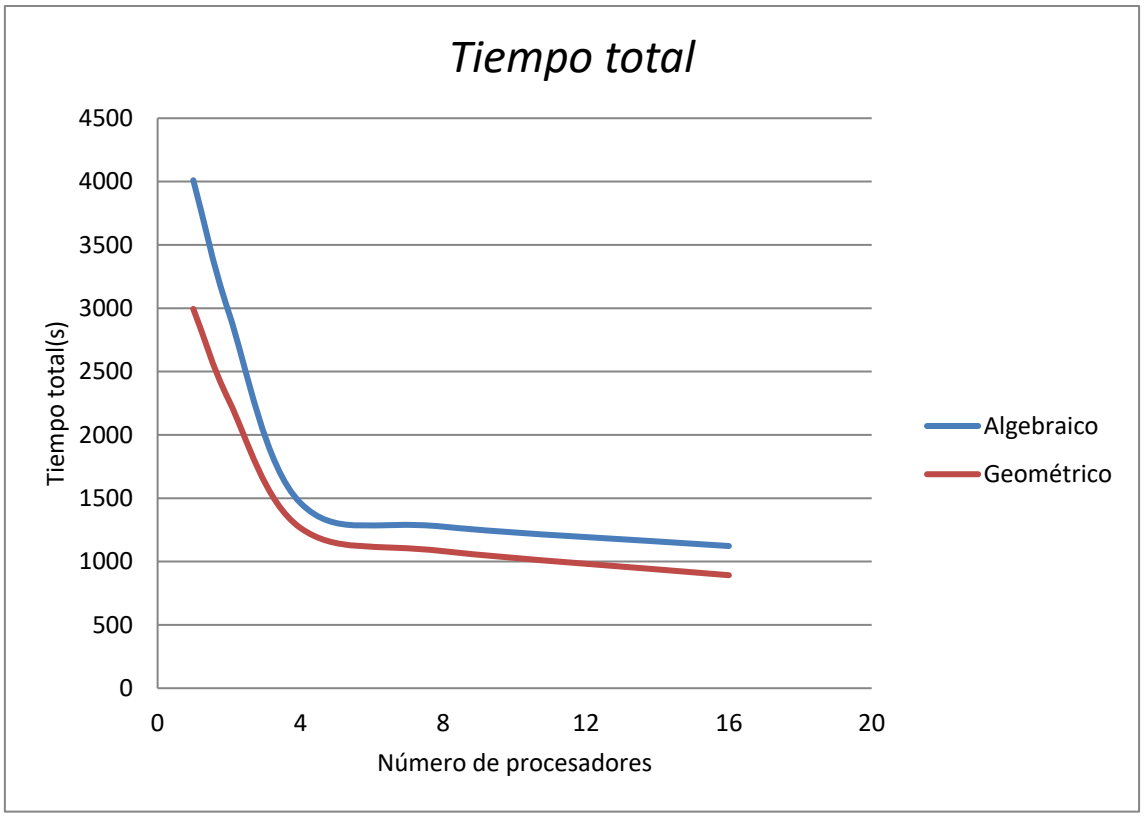


Gráfico 5.18 Tiempo total conambos métodos multigrid

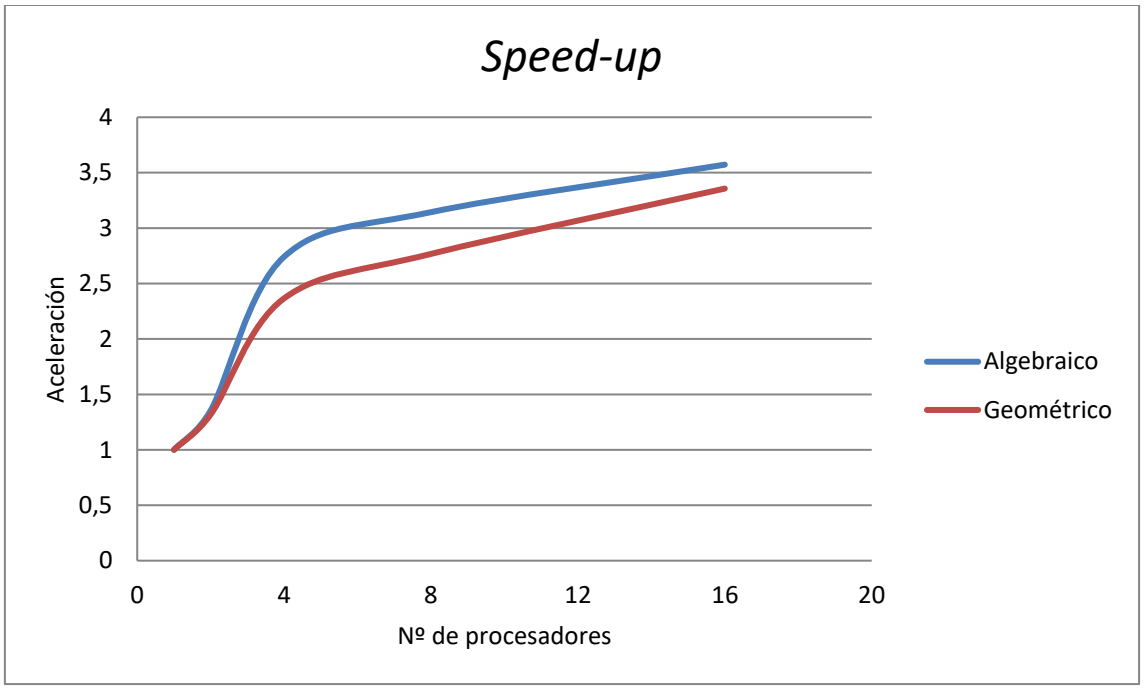


Gráfico 5.19 Speed-up

En el gráfico 5.18 se han representado los tiempos totales obtenidos con ambos métodos multigrid. Se puede ver como el geométrico es más eficiente con cada uno de los distintos números de los procesadores disponibles, pero ese rango de eficiencia va disminuyendo en la medida que se aumenta el número de procesadores, es decir, el algebraico muestra una mayor aceleración. Se puede ver, como se ha comentado anteriormente, que el coste computacional disminuye de manera muy significativa cuando se usan unos pocos cores, pero, a partir de 4 se reduce el nivel de eficiencia conseguido.

En el gráfico 5.19 se muestra el speed-up obtenido. Como se ha mencionado, aunque se obtienen mayores de velocidades de procesamiento con el multigrid geométrico, la aceleración obtenida al aumentar el número de cores disponibles es mayor con el algebraico, saturándose en ambos casos con 4 procesadores.

Resaltar que estos resultados no son directamente extrapolables, para sacar conclusiones habría que volver a realizar las simulaciones con más modelos y analizar los resultados, esta es una posible línea de investigación de trabajos futuros.

6 Conclusiones y trabajos futuros

6.1 Conclusiones

Se han realizado simulaciones con distintos modelos utilizando diferentes métodos multigrid para precondicionar un gradiente conjugado distribuido. Cada simulación ha sido realizada utilizando un multigrid geométrico y un multigrid algebraico. De dichas simulaciones se extraen las siguientes conclusiones:

- El multigrid algebraico es más eficiente en los modelos 2D. Debido a que, este necesita menor tiempo de resolución por iteración, siendo este especialmente óptimo en modelos 2D que necesiten un gran número de iteraciones.
- El multigrid geométrico es más eficiente sólo en unos casos muy concretos, en los que, el modelo sea 3D y con un radio de filtro grande, que mejore el condicionamiento del problema. La mayor eficiencia en sólidos 3D se debe a que este necesita menos saltos para realizar el ciclo en V. El multigrid algebraico en cambio, se abstrae de la geometría, operando sólo sobre la matriz de coeficientes, necesitando más saltos para realizar el ciclo en V y, por lo tanto, más tiempo de computación.
- El radio del filtro utilizado influye de manera muy significativa en el condicionamiento del problema, favoreciendo un aumento de eficiencia en el GMG conforme más grande es el radio. Un aspecto muy importante a tener en cuenta es que, si se utilizan mallas finas, que proporcionen un gran número de elementos, utilizar un filtro de gran radio no es conveniente, ya que, al realizar el promedio de los elementos incluidos dentro de dicho radio, se pierde el nivel de detalle que se quiere conseguir utilizando una discretización de gran precisión. Si el objetivo es conseguir resultados de alto nivel de precisión se requiere un radio pequeño, resultando ser el AMG más eficiente en esos casos.
- Se sabe de manera empírica que el multigrid algebraico es más eficiente en mallas desestructuradas, mientras que el geométrico en mallas estructuradas. Pero, no se puede saber de ante mano, al abordar un problema de optimización topológica cual es el método más eficiente, ya que, este depende de cómo el sistema este condicionado. Esto se debe a que la optimización es un problema muy complejo, dónde se está resolviendo una simulación por iteración y depende de un gran número de variables. No es posible saber con certeza, como va a evolucionar el problema, debido a que, este se actualiza en cada iteración y puede tener una tendencia a mejorar o empeorar el número de condición. Si mejora el condicionamiento, el multigrid geométrico tenderá a ser más eficiente. Si empeora lo será el algebraico. A pesar de no poder tener una certeza absoluta en cuanto a la eficiencia, al realizar las simulaciones se muestra que el GMG tiende a ser más eficientes en 3D y los AMG en 2D, como se ha mencionado anteriormente.
- Hay que tener también en cuenta la flexibilidad de cada método. El AMG al operar sólo con la información de la matriz de coeficientes es mucho más flexible. Esto unido a que el multigrid geométrico es sólo más eficiente en determinados casos explica que los softwares comerciales utilicen el multigrid algebraico para resolver los problemas de optimización.

6.2 Trabajos futuros

Existen diferentes líneas de investigación para continuar este estudio, pero a mi parecer, la más directa es un estudio de la escalabilidad de los métodos multigrid geométrico y algebraicos usando un clúster de computación científica. En este estudio se ha realizado la simulación con el modelo 4. El resultado ha sido que el multigrid algebraico proporcionaba una mayor aceleración conforme se aumentaban los procesadores disponibles. Este resultado podría ser algo puntual, por lo que, sin realizar más simulaciones no se podrían extraer conclusiones. El trabajo futuro a realizar consistiría en repetir las simulaciones de los modelos variando el número de procesadores disponibles con el fin de analizar la escalabilidad de cada método multigrid.

Bibliografía

[1] Brodtkorb, A. R., Hagen, T. R., & Sætra, M. L. (2013). Graphics processing unit (GPU) programming strategies and trends in GPU computing. Journal of Parallel and Distributed Computing, 73(1), 4-13.

[2] Wulf, W. A., & McKee, S. A. (1995). Hitting the memory wall: implications of the obvious. ACM SIGARCH computer architecture news, 23(1), 20-24.

[3] Bibing.us.es. (2021). Recuperado de: <http://bibing.us.es/proyectos/abreproy/4296/fichero/VolumenI%252F4.pdf>

[4] Cortés, G., (2021). Riunet.upv.es. Recuperado de: <https://riunet.upv.es/bitstream/handle/10251/11266/Tesis.pdf?sequence=1>

[5] Meza, C, (2021). Red.uao.edu.co. Recuperado de: <https://red.uao.edu.co/bitstream/handle/10614/4209/TME01179.pdf;jsessionid=997BDF1201720E5B520B770696B7A8F7?sequence=1>

[6] Amir, O., Aage, N. and Lazarov, B, (2013). On multigrid-CG for efficient topology optimization. Structural and Multidisciplinary Optimization, 49(5), pp.815-829.

[7] Help.solidworks.com. 2021. SIMP Method for Topology Optimization - 2019 - SOLIDWORKS Help. Recuperado de: https://help.solidworks.com/2019/English/SolidWorks/cworks/c_simp_method_topology.htm>

[8]Karypis,G.(2013).Metis a software package for partitioning unstructured graphs, partitioning meshes, and Computing fill-reducing orderings of sparse matrices.

[9] Falgout, Robert & Jones, Jim & Yang, Ulrike. (2006). The Design and Implementation of hypre, a Library of Parallel High Performance Preconditioners. 10.1007/3-540-31619-1_8.

[10] Computing.llnl.gov. 2021. HYPRE | Computing. [online]Disponible en: <https://computing.llnl.gov/projects/hypre-scalable-linear-solvers-multigrid-methods>

[11] Herrero-Pérez, David, and Pedro J. Martínez Castejón. (2021) "Multi-GPU acceleration of large-scale density-based topology optimization." Advances in Engineering Software 157: 103006.

[12 Ochoa, C., 2021. Repositorio.unal.edu.co. Disponible en: <https://repositorio.unal.edu.co/bitstream/handle/unal/64099/1030549087.2018.pdf?sequence=1&isAllowed=y>

[13] Cuellar Loyola, Nilton Alejandro & Pereira, Anderson & Menezes, Ivan. (2015). ROBUST TOPOLOGY OPTIMIZATION UNDER UNCERTAIN LOADS: A SPECTRAL STOCHASTIC APPROACH. 10.20906/CPS/CILAMCE2015-0882.

[14] Simonetti, Hélio & Almeida, Valério & Neves, Francisco & Del, Virgil & Del Duca Almeida, Virgil. (2021). Topological Optimization of three-dimensional Structures using the methods of moving asymptotes and optimality criteria.

[15] Grant, M., (2021). The Method of Moving Asymptotes - ppt video online download. [online] Slideplayer.com. Disponible en: <https://slideplayer.com/slide/8948423/> [Accessed 18 August 2021].

[16] Sundar, Hari, et al. (2012) "Parallel geometric-algebraic multigrid on unstructured forests of octrees." SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.

[17] Stüben, Klaus. "A review of algebraic multigrid." Numerical Analysis: Historical Developments in the 20th Century (2001): 331-359.

[18] Larrazabal Serrano, Germán Alberto. Técnicas algebraicas de precondicionamiento para la resolución de sistemas lineales. Universitat Politècnica de Catalunya, 2002.

[19] García-Cuevas González, Luis Miguel, et al. (2020). "Mecánica de fluidos computacional: reconstrucción de la solución para el cálculo de flujos y métodos multigrid."

[20] Watanabe, K., H. Igarashi, and T. Honma. (2005) "Comparison of geometric and algebraic multigrid methods in edge-based finite-element analysis." IEEE transactions on magnetics 41.5: 1672-1675.

[21] Notay, Yvan. (2009) "Aggregation-based algebraic multigrid." Electronic Transactions on Numerical Analysis.

[22]Gómez,Miguel(2019). Simulación paralela del método de elementos finitos en mecánica de sólidos

[23]Liu, Kai, and Andrés Tovar. (2014) "An efficient 3D topology optimization code written in Matlab." Structural and Multidisciplinary Optimization 50.6: 1175-1196.