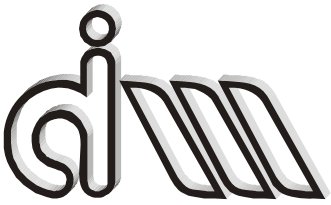
UNIVERSIDAD POLITÉCNICA DE VALENCIA

Departamento de Ingeniería Mecánica y de Materiales

Máster en Ingeniería Mecánica y Materiales



TESIS DE MÁSTER

DESARROLLO DE UN PROGRAMA DE OPTIMIZACIÓN DE FORMA DE COMPONENTES MECÁNICOS MEDIANTE OPTIMIZACIÓN TOPOLÓGICA ADAPTATIVA

Presentada por: D. Guillem Josep Cortés Carbonell

Dirigida por: Dr. D. Juan José Ródenas García

Valencia, Octubre de 2010

Resumen

La Optimización Topológica de componentes estucturales trata de determinar la ubicación y distribución óptima de material en una estructura para unas condiciones de restricción determinadas y atendiendo a una función objetivo determinada, en este caso la minimización de la energía de deformación.

En la presente Tesis de Máster se ha desarrollado un programa de Optimización Topológica Adaptativa a partir de un programa de Optimización Topológica inicial básico, que ha sido modificado para adaptarlo a un código de EF con mallados cartesianos independientes de la geometría, en el entorno de Matlab. El uso de este tipo de código de EF mejora el rendimiento computacional de los algoritmos de optimización topológica.

Se ha dotado al programa de Optimización Topológica de mayor versatilidad implementando ciertas aplicaciones como las de incluir zonas de material fijo en el componente (material que no formaría parte del proceso de optimización), desarrollo de una aplicación que permite calcular la cantidad de material necesaria (y su distribución) para no sobrepasar en ningún punto la tensión máxima admisible del material o la posibilidad de crear zonas en el dominio a fin de realizar tratamientos específicos de post-proceso.

Posteriormente se ha implementado la posibilidad de utilizar los algoritmos de h-adaptación de la malla del código de EF que resultan muy eficientes por estar basados en el uso de mallados cartesianos. Se ha conseguido una mayor resolución en la definición geométrica del componente y una disminución extra de la energía de deformación con una reducción del coste computacional.

Se han realizado estudios de los diferentes parámetros del programa y se han analizado diferentes ejemplos contrastados en cada etapa del trabajo para verificar la robustez y versatilidad del programa.

Palabras clave: Optimización Topológica, Elementos Finitos, Mallado independiente de la geometría, Adaptatividad.

Abstract

Topology Optimization of structural components tries to determine the location and optimal distribution of material in a structure under certain restrictions considering a given objective function, which is to minimize the strain energy in this case.

In this Master Thesis an Adaptative Topology Optimization program has been developed from an initial and basic Topology Optimization program, which has been modified in Matlab in order to be adapted to a FE code with cartesian meshes independent of the geometry. The computational efficiency of the Topology Optimization algorithms is improved by using this kind of FEM code.

The Topology Optimization program has gained versatility because of the implementation of certain applications like the inclusion of areas of fixed material in the component (material which will not be considered in the optimization process), development of an application which allows to calculate the amount of material (and its distribution) in order to maintain the stresses under a prescribed maximum admissible value and the possibility of creating areas in the domain in order to carry out specific post-process treatments.

Afterwards the possibility of using the h-adaptation algorithms of the mesh FE code was implemented. The algorithms used are very efficient due to the use of cartesian meshes. This process results then in a better geometrical definition of the component and provides both an extra strain energy decrease and a reduction of the computational cost.

The parameters of the program have been studied and several case studies have been analyzed at each stage of the work in order to verify the robustness and versatility of the program.

Key words: Topology Optimization, Finite Element Method, Cartesian Meshes Independent of the geometry, Adaptativity.

Resum

L'Optimització Topològica de components estructurals tracta de determinar la ubicació i distribució de material en una estructura per a unes condicions de restricció determinades i atenent a una funció objectiu determinada, en aquest cas la minimització de l'energia de deformació.

En la present Tesi de Màster s'ha desenvolupat un programa d'Optimització Topològica Adaptativa a partir d'un programa d'Optimització Topològica inicial bàsic, que ha estat modificat per a adaptar-lo a un codi d'EF amb mallats cartesians independents de la geometria, a l'entorn de Matlab. L'ús d'aquest tipus de codi d'EF millora el rendiment computacional dels algoritmes d'optimització topològica.

S'ha dotat al programa d'Optimització Topològica de major versatilitat implementant certes aplicacions com les d'incloure zones amb material fix al component (material que no formaria part del procés d'optimització), desenvolupament d'una aplicació que permet calcular la quantitat de material necessària (i la seua distribució) per a no sobrepassar en cap punt la tensió màxima admissible del material o la possibilitat de crear zones al domini per a poder realitzar tractaments específics de post-procés .

Posteriorment s'ha implementat la possibilitat d'utilitzar els algortimes de h-adaptació de la malla del codi d'EF que resulten molt eficients per estar basats en l'ús de mallats cartesians. S'ha aconseguit una major resolución en la definició geomètrica i una disminució extra de l'energia de deformació amb una reducció del cost computacional.

S'han realitzat estudis dels diferents paràmetres del programa i s'han analitzat diferents exemples constrastats en cada etapa del treball per a verificar la robustesa i versatilitat del programa.

Paraules clau: Optimització Topològica, Elements Finits, Mallat independent de la geometria, Adaptativitat.

Agradecimientos

En primer lugar me gustaría agradecer la entrega y dedicación que me ha prestado Juanjo, director de mi tesis de máster, durante todo este año que he pasado en el departamento. Me ha enseñado a trabajar dentro de un grupo de investigación y para mí ha resultado un momento muy importante en el paso de la etapa formativa a la profesional.

También me gustaría dar las gracias al grupo de personas que forman el departamento, especialmente a los compañeros en el aula de becarios, por la complicidad y el apoyo que me han ofrecido y que ha servido para llevar este trabajo con especial motivación.

Por otra parte, agradecer a todos mis amigos por estar ahí en los buenos y no tan buenos momentos y por aconsejarme tantas veces en esta etapa tan importante para mi futuro profesional.

Finalmente, a mi familia, mis padres y mi hermano, por haberme ayudado tanto en cada momento y por haberme apoyado siempre incondicionalmente.

Valencia, Octubre de 2010

Índice

1 INTRODUCCIÓN

El propósito de este trabajo es el de disponer de una herramienta de Optimización Topológica de gran rendimiento computacional con mallados independientes de la geometría. Para ello se ha integrado el módulo de optimización topológica en el software de Elementos Finitos disponible en el Área de Ingeniería Mecánica (AIM) del Departamento de Ingeniería Mecánica y de Materiales (DIMM). Este software usa técnicas que permiten la utilización de mallas de fácil construcción que son independientes de la geometría (Generalized Finite Element Method, GFEM). Se hace uso además de técnicas de adaptatividad de la malla. A continuación se introducen brevemente las ideas principales de estos ámbitos.

La optimización intenta dar respuesta a un tipo general de problemas donde se desea elegir la mejor solución entre un conjunto de elementos. El uso de técnicas de optimización para componentes industriales conduce a la obtención de piezas de menor peso y mejores prestaciones. Desde el punto de vista industrial, la disminución del material utilizado en una pieza, suele llevar asociada una reducción de costes que es muy importante en sectores donde se fabrican grandes series, tales como el sector automovilístico.

Un tipo de optimización de componentes es la Optimización Topológica, un proceso de determinación de la ubicación y distribución de material en una estructura para una determinada función objetivo y condiciones de restricción. Se trata de obtener la forma que debe tener una pieza (o el valor de ciertos parámetros) para minimizar (o maximizar) una cantidad física como por ejemplo la energía de deformación, tensión máxima alcanzada, etc. mientras el equilibrio y otras restricciones y variables de diseño son satisfechas.

La Optimización Topológica requiere discretizar el espacio de diseño en pequeños elementos, a los cuales se les asigna *material* o *no materia*" para construir así la forma final del componente. La discretización en elementos que usa el Método de los Elementos Finitos (MEF) hace que esta resulte una herramienta muy apropiada para tratar el problema

El MEF se ha consolidado en las últimas décadas como uno de los métodos numéricos más eficientes utilizados en la industria para resolver gran variedad de problemas en ingeniería, desde el diseño estructural y mecánico, la transferencia de calor, el modelado biológico, etc. El MEF se ha convertido en una herramienta

esencial de trabajo en industrias como las del automóvil, aeronáutica, aeroespacial, naval, ingeniería civil, etc. En este tipo de industrias, la obtención mediante el MEF del campo de tensiones, es de suma importancia en los procesos de diseño de componentes mecánicos en los que resulta necesario predecir de manera fiable la aparición de fallos estructurales.

En el MEF es necesario generar una malla de elementos finitos adaptada al espacio de diseño, lo que suele implicar la utilización de las técnicas tradicionales de mallado que pueden llevar involucrado un coste computacional importante. Por otro lado las técnicas de mallado tradicionales generan elementos de geometrías muy diversas que no son las más adecuadas en optimización topológica. La utilización de mallados cartesianos independientes de la geometría (GFEM) permite reducir el coste computacional del proceso puesto que, primero, se pueden definir espacios de trabajo de geometría compleja sin necesidad de acudir a los generadores de malla habitualmente utilizados, y segundo, porque estos mallados son muy adecuados en optimización topológica ya que todos los elementos de la malla son iguales, lo que permite reutilizar gran cantidad de resultados de cálculos anteriores.

Una de las técnicas que han hecho el MEF más eficiente es la de adaptatividad de la malla. Se basa en el uso de elementos de diferente tamaño según la zona del componente que se está discretizando. El origen de esta técnica se debe a la necesidad de aumentar la precisión de la solución en zonas concretas (zonas de mayor error en la solución) sin tener que aumentar los grados de libertad del problema excesivamente. En este trabajo se recoge la idea de la adaptatividad de la malla para mejorar el rendimiento del algoritmo de optimización utilizando mallas adaptadas en los lugares adecuados.

1.1 Objetivos

El objetivo general de esta tesis de master es el de dotar de un módulo de Optimización Topológica a un programa de cálculo de componentes estructurales 2D con mallados cartesianos independientes de la geometría utilizando el Método de los Elementos Finitos Generalizado (GFEM), el cual está disponible en el DIMM. En la segunda parte del trabajo se hace uso además de la técnica de adaptatividad para obtener resultados más satisfactorios con un reducido coste computacional

En general, se ha buscado dar versatilidad y robustez al programa integrando determinadas funciones. Para ello se han perseguido los siguientes objetivos parciales:

- Adaptar un algoritmo de optimización topológica disponible en la bibliografía [1] al entorno de cálculo del programa de GFEM, adaptando a su vez la estructura de datos dentro del programa de GFEM.
- Mejorar el algoritmo de optimización original para conseguir un mayor rendimiento computacional.
- Realizar un completo estudio para garantizar que las adaptaciones realizadas proporcionan, cuando la malla de cálculo coincide con la geometría de trabajo, los mismos resultados que el algoritmo de optimización original.
- Realizar un estudio, para el caso de mallas independientes de la geometría de trabajo, relativo a la influencia de los diferentes parámetros de los que depende la optimización topológica para verificar que el código presenta el comportamiento esperado.
- Añadir la posibilidad de definir zonas dentro de la geometría de trabajo donde tiene que existir necesariamente material debido, por ejemplo, a restricciones de fabricación.
- El algoritmo original de optimización tiene como objetivo obtener la distribución de material en el componente para un volumen de material dado que maximiza la rigidez del componente. A partir de este algoritmo, se generará un algoritmo que permita obtener la distribución óptima de material que, con el mínimo peso de material, proporciona tensiones en el componente por debajo de un determinado valor límite especificado por el analista.
- Comprobar la robustez del software mediante su aplicación a un conjunto de problemas de interés industrial.
- Aumentar la efectividad del programa implementado la técnica de adaptatividad de la malla al proceso de optimización topológica de forma que se obtengan geometrías con contornos mejor definidos y se consiga disminuir más la energía de deformación (función objetivo).
- Diseñar una interfaz gráfica para el uso directo del programa como módulo dentro del software de GFEM disponible. Así mismo, facilitar un manual de

usuario que permita acceder y manejar este programa de forma sencilla y entendible.

- Elaborar un manual de programador que explique detalladamente la estructura del programa desglosado en sub-funciones y por otra parte la estructura de datos empleada.

2 ANTECEDENTES

2.1 Optimización

La optimización de componentes da como solución la forma que debe tener una pieza para minimizar (o maximizar) una cantidad física como por ejemplo la energía de deformación, tensión máxima alcanzada, etc. mientras el equilibrio y otras restricciones y variables de diseño son satisfechas.

Este problema puede ser abordado desde diferentes puntos de vista. Los tres más importantes son: optimización de tamaño o dimensionamiento (sizing optimization), optimización de forma (shape optimization) y optimización topológica (topology optimization).

En un problema típico de optimización de tamaño, el objetivo es encontrar la distribución del espesor óptima de un problema lineal elástico, o bien los miembros más óptimos en una estructura articulada. La variable de diseño es el espesor de las barras y la variable a minimizar por ejemplo el desplazamiento relativo. En la siguiente figura (viga biapoyada con carga centrada) se muestra como mediante la optimización de tamaño se pueden calcular aquellas barras que trabajan más en la estructura.

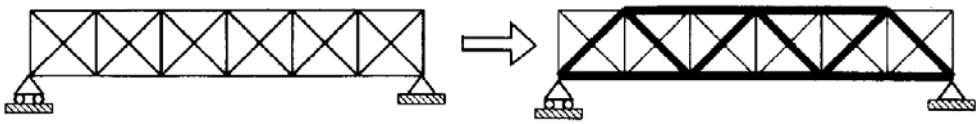


Figura 2.1 Optimización de tamaño

La principal característica de este tipo de problemas es que el dominio de diseño es conocido a priori y se mantiene fijo a lo largo del proceso de optimización.

Por otra parte, la optimización de forma tiene como objetivo encontrar la forma óptima del componente. En este caso el dominio de diseño cambia en el proceso iterativo ya que es precisamente la variable de diseño. Se trata básicamente de definir que forma adopta el perímetro de la geometría de trabajo para encontrar la forma óptima. Es por eso que en este tipo de optimización hay que definir a priori paramétricamente la geometría que se va a desarrollar, es decir, si se desea un

componente optimizado mediante agujeros, estos tienen que estar definidos a priori. En la siguiente figura se muestra un ejemplo.

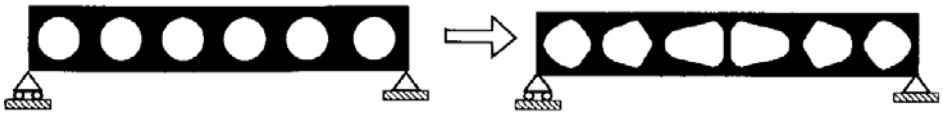


Figura 2.2 Optimización de forma

Por último, la optimización topológica de estructuras sólidas determina ciertas características como el número de agujeros junto con su ubicación y forma, en definitiva la conectividad del dominio. En esta metodología el dominio de diseño es constante en todo momento. Permite crear agujeros partiendo de una geometría continua. Estos agujeros se interpretan como zonas donde la rigidez del material es nula (realmente casi nula para evitar singularidades en el proceso de cálculo). A continuación se muestra como sería el problema inicial y el resultado obtenido mediante optimización topológica.

Figura 2.3 Optimización topológica

Si comparamos el diseño inicial de los tres métodos explicados, se observa que en los dos primeros se requiere de un diseño más avanzado, donde ya se requiere un conocimiento previo del comportamiento de la estructura. En cambio, la optimización topológica trabaja con geometrías de trabajo muy simples, sin necesidad de un diseño previo por parte del usuario.

2.2 El método de los elementos finitos

El proceso de optimización topológica es un proceso iterativo en el cual se va calculando la distribución óptima del material. Cada distribución de material calculada requiere ser evaluada de forma que se deben calcular los desplazamientos del componente. Para poder calcular dichos desplazamientos se recurre al método de los elementos finitos, el cual se expone a continuación.

2.2.1 FEM

El método de los elementos finitos (FEM en inglés) es un método numérico general para la aproximación de soluciones de ecuaciones diferenciales en derivadas parciales muy utilizado en diversos problemas de ingeniería y física.

El MEF está pensado para ser usado en computadoras y permite resolver ecuaciones diferenciales asociadas a un problema físico sobre geometrías complicadas, en el que las ecuaciones constitutivas y ecuaciones de evolución temporal del problema a considerar sean conocidas de antemano.

El cuerpo, estructura o dominio (medio continuo) sobre el que están definidas las ecuaciones diferenciales en forma débil o integral que caracterizan el comportamiento físico del problema, queda dividido en un número elevado de subdominios no-intersectantes entre sí denominados «elementos finitos».

Los cálculos se realizan sobre una malla de puntos (llamados nodos), que sirven a su vez de base para discretización del dominio en elementos finitos. De acuerdo con estas relaciones de adyacencia o conectividad se relaciona el valor de un conjunto de variables incógnitas definidas en cada nodo y denominadas grados de libertad. El conjunto de relaciones entre el valor de una determinada variable entre los nodos se puede escribir en forma de sistema de ecuaciones lineales (o linealizadas). La matriz de dicho sistema de ecuaciones se llama matriz de rigidez del sistema. El número de ecuaciones de dicho sistema es proporcional al número de nodos.

Típicamente el método de los elementos finitos se programa en problemas de mecánica de sólidos deformables para calcular el campo de desplazamientos y, posteriormente, a través de relaciones cinemáticas y constitutivas las deformaciones y tensiones respectivamente.

Una característica básica del método consiste en interpolar, dentro de cada elemento finito, el valor de las funciones incógnita en función de sus correspondientes valores nodales. En general, independientemente del tipo de elemento que estemos considerando, la interpolación de elementos finitos para una componente dada siempre se podrá escribir como:

$$u(x,y,z)=\sum_{i=1}^{n} N_i(x,y,z)u_i = \mathbf{N}(x,y,z)\ \mathbf{u}^e \tag{2.1}$$

Siendo *n* el número de nodos del elemento. Si el sistema es discreto, es posible obtener las ecuaciones de comportamiento de cada elemento, y matricialmente obtener las ecuaciones de comportamiento globales del sistema. En elementos finitos, una vez obtenidas las ecuaciones de comportamiento de elemento, en este caso no exactas debido a las aproximaciones consideradas, el proceso posterior será idéntico al caso de sistemas discretos.

Se considerará la formulación más usual del método de los elementos finitos, considerando como funcional el principio de la energía potencial total y en consecuencia se obtendrá una formulación en desplazamientos, ya que las variables independientes son los desplazamientos.

El Método de los Elementos Finitos se basa en la interpolación de una función (solución aproximada), en base al valor de dicha función en puntos conocidos. Para encontrar la solución aproximada se divide el dominio Ω en un conjunto de elementos (p.e. en el caso bidimensional triángulos o cuadriláteros, y en el caso tridimensional tetraedros o hexaedros), en los cuales las funciones incógnitas se sustituyen por las funciones interpoladas.

Aplicando el principio de energía potencial total estacionaria con respecto a los desplazamientos nodales, se obtiene un sistema algebraico de ecuaciones, a partir del cual es posible obtener los coeficientes de interpolación de los desplazamientos. Así consideraremos la expresión de la energía potencial total

$$\Pi_p=\int_V \left(\frac{1}{2}\boldsymbol{\varepsilon}^T\mathbf{D}\boldsymbol{\varepsilon}-\boldsymbol{\varepsilon}^T\mathbf{D}\boldsymbol{\varepsilon}_0+\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}_0\right)dV \quad -\int_V \mathbf{u}^T\mathbf{b}dV-\int_S \mathbf{u}^T\mathbf{t}dS-\mathbf{U}^T\mathbf{P} \tag{2.2}$$

en donde:

$\mathbf{u} = [\ u\ v\ w\ ]^T$: campo de desplazamientos.

$\boldsymbol{\varepsilon} = \{\ \varepsilon_x\ \varepsilon_y\ \varepsilon_z\ \gamma_{xy}\ \gamma_{yz}\ \gamma_{zx}\ \}$: campo de deformaciones.

$\mathbf{D}$: matriz de propiedades de material.

$\boldsymbol{\varepsilon}_0$, $\boldsymbol{\sigma}_0$: deformaciones y tensiones iniciales.

$\mathbf{b} = [b_x\ b_y\ b_z]^T$	: fuerzas volumétricas.
$\mathbf{t} = [t_x\ t_y\ t_z]^T$	: fuerzas superficiales.
$\mathbf{U}$	: desplazamientos nodales del sistema.
$\mathbf{P}$	: cargas puntuales aplicadas en los nodos.
S,V	:contorno y dominio de definición del problema.

Considerando la ec. (2.1) y después de ciertos cálculos [2], la expresión de la energía potencial total del sistema puede expresarse como:

$$\Pi_p = \frac{1}{2}\sum_{e=1}^{ne} \mathbf{u}^{eT}\mathbf{k}^e\mathbf{u}^e - \sum_{e=1}^{ne} \mathbf{u}^{eT}\mathbf{f}^e - \mathbf{U}^T\mathbf{P} \tag{2.3}$$

Donde los símbolos de sumatorio indican que incluimos las contribuciones de todos los elementos finitos.

Para completar el desarrollo, se debe determinar las ecuaciones algebraicas para calcular los desplazamientos nodales. Cualquier desplazamiento nodal del vector de elemento $\mathbf{u}^e$ también aparece en el vector global $\mathbf{U}$. De esta forma se puede expandir conceptualmente la matriz de rigidez $\mathbf{k}^e$ y el vector de fuerzas equivalentes $\mathbf{f}^e$ de cada elemento al tamaño global ($\mathbf{K}^e$ y $\mathbf{F}^e$). De esta forma, la ecuación (2.3) queda:

$$\Pi_p = \frac{1}{2}\mathbf{U}^T\mathbf{K}\mathbf{U} - \mathbf{U}^T\mathbf{F} \tag{2.4}$$

donde:

$$\mathbf{K} = \sum_{e=1}^{ne} \mathbf{K}^e \qquad \text{y} \qquad \mathbf{F} = \mathbf{P} + \sum_{e=1}^{ne} \mathbf{F}^e \tag{2.5}$$

Los sumatorios pueden entenderse como ensamblado de matrices de elemento por la adición de coeficientes que se solapan con respecto a la numeración de los grados de libertad (gdl) globales. Ahora Π_p es una función de los desplazamientos nodales $\mathbf{U}$. Haciendo Π_p estacionario con respecto a pequeños cambios en U_i, utilizando convenientemente las reglas de diferenciación, se puede escribir:

$$\left\{\frac{\partial \Pi_p}{\partial \mathbf{U}}\right\}=\mathbf{0} \qquad \rightarrow \qquad \mathbf{K}\cdot\mathbf{U}=\mathbf{F} \tag{2.6}$$

La última ecuación es una ecuación de equilibrio nodal (equilibrio de fuerzas en los nodos), donde **K** es una matriz simétrica y $K_{ij}=\partial^2\Pi_p/\partial U_i\partial U_j$. Esta ecuación matricial es un conjunto de ecuaciones algebraicas que puede utilizarse para obtener U.

2.2.2 Generalized Finite Element Method (GFEM)

En el presente trabajo se expone la técnica GFEM para la resolución de problemas de contorno. Las ideas fundamentales del GFEM son, por un lado, que el dominio a analizar es independiente de la malla y, por el otro, que se pueden incluir funciones de enriquecimiento local de la aproximación para incluir características conocidas de la solución. En la implementación sólo se ha considerado el primero de ellos.

En primer lugar se debe definir y comparar el método GFEM y el FEM. En general una malla FEM se usa para la construcción de la aproximación y para la integración numérica de las funciones. Dicha malla se crea por la división del dominio en triángulos o cuadriláteros curvilíneos no solapados, los cuales se les denomina elementos y deben satisfacer condiciones de conexión entre ellos y distorsión.

Como diferencia, en GFEM se usan dos mallas. Una es llamada malla de aproximación, usada para la construcción de la aproximación del problema y la otra corresponde a la malla de integración, la cual es construida a partir de cada elemento de la primera por separado, y es destinada a la evaluación numérica de todas las integrales.

Para la malla de aproximación, el único requisito indispensable es que cubra completamente el dominio del problema, mientras que la malla de integración es necesario que defina correctamente el dominio del problema, utilizando para ello un refinamiento especial de cada elemento de la malla de aproximación que lo necesite.

Cabe considerar que existen varios tipos de GFEM según T. Strouboulis *et. al*, [3], en relación a la malla de aproximación y la geometría del dominio. En todos

ellos se usa una malla clásica de elementos finitos, pero con diferente relación entre el contorno de la geometría y del dominio de la malla.

En el tipo de mallados utilizados en el GFEM del DIMM, el dominio de la malla de aproximación cubre el dominio original y no tienen fronteras comunes con la geometría. En la Figura 2.4 se muestra un ejemplo de un dominio complejo mallado donde se observa una malla de este tipo, dónde la malla de integración se ha construido por un refinamiento del elemento inicial en subdominios geométricamente semejantes. En el caso de la presente implementación, dicho refinamiento se realiza mediante la triangulación Delaunay.

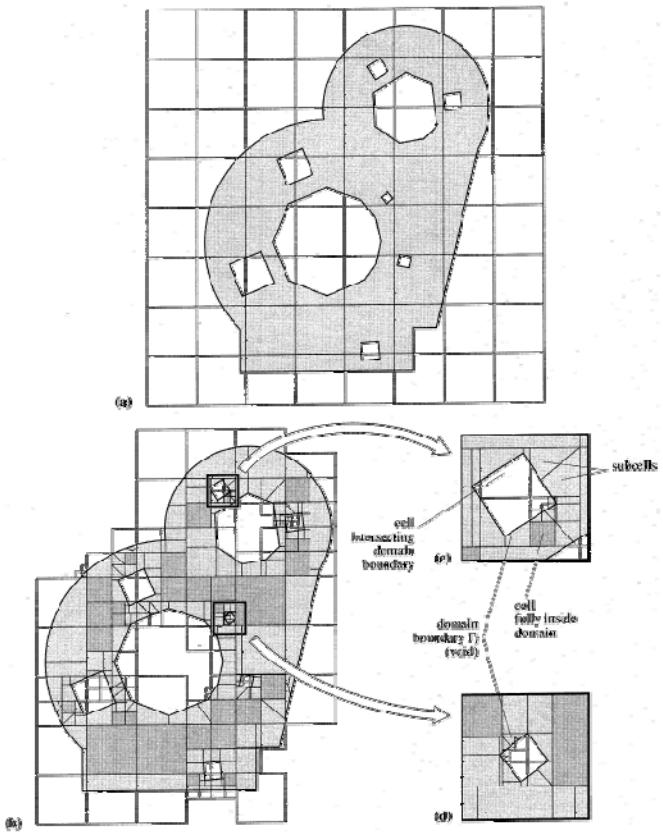


Figura 2.4 (a) Malla de aproximación obtenida empleando refinamiento uniforme de cuadriláteros regulares. (b) Malla de integración correspondiente. (c) y (d) Detalles de la malla de integración.

Cabe añadir que el software de GFEM del Departamento de Ingeniería Mecánica y de Materiales fue desarrollado inicialmente por Ing. Justo Martí Pellicer [4] en su proyecto final de carrera y posteriormente mejorado por Ing. Enrique Nadal Soriano en su trabajo de tesis de máster [5].

2.2.3 Adaptatividad

Las técnicas adaptativas son procedimientos que permiten mejorar el modelo de elementos finitos para disminuir el error de discretización, adaptándose a la solución del problema tras analizar las zonas donde el modelo proporciona mayor error. Se basa en el uso de elementos de diferente tamaño según la zona del componente que se está discretizando, de modo que se consigue aumentar la precisión en las zonas con elementos más pequeños sin necesidad de aumentar los grados de libertad del problema excesivamente. Se pueden definir tres tipos de adaptatividad:

- *h*-adaptatividad: se utilizan elementos de menor tamaño en las zonas de mayor error.

- *p*-adaptatividad: se utilizan elementos asociados a polinomios de mayor grado de interpolación en las zonas de mayor error.

- *hp*-adaptatividad: se combina el tamaño y el grado de interpolación del polinomio para conseguir mayor número de grados de libertad en la zona con mayor error.

En el software de GFEM desarrollado en el CITV se ha implementado la técnica de *h*-adaptatividad, que permite refinar la malla en ciertas zonas.

En este proyecto se aprovechará esta técnica para adaptarla al proceso de optimización topológica, de forma que el criterio de refinamiento no estará basado en la disminución del error si no que será un criterio que proporcione una solución con mayor resolución geométrica. Esto se explica más adelante en la sección 4.

2.3 Optimización topológica

La optimización topológica ha resultado ser una herramienta práctica de diseño. Después de muchos años funcionando con éxito en la industria automovilística, otros terrenos dentro de la industria se han beneficiado de su uso, como los procesos de diseño en aeronáutica e industrias de productos de consumo.

La última década ha visto un enorme progreso en la aplicación de optimización estructural. Muchos algoritmos basados en elementos finitos han madurado y han sido implementados en software que se utilizan diariamente para la resolución de problemas prácticos.

El proceso de diseño sigue un proceso iterativo donde la simulación computacional del comportamiento estructural es la base para tomar decisiones en el diseño. Es muy común que las modificaciones de diseño se deriven usando una aproximación de prueba y error. Sin embargo, el uso de la tecnología de optimización convierte el proceso de diseño en un proceso conducido directamente por análisis computacional.

Hasta principios de los 90, el uso de la optimización estructural estaba limitada a mejorar diseños con trazados predeterminados usando optimización de forma. La intuición y experiencia del diseñador jugaba entonces el papel principal cuando se definía el trazado inicial del diseño. Ha sido probado que puede conseguirse una sustancial mejora si se altera el concepto inicial disponiendo una estructura de cavidades dentro de la estructura.

2.3.1 Formulación del problema y parametrización del diseño.

El problema que va a ser definido a continuación combina diferentes características de problemas tradicionales de optimización de diseño estructural. El propósito de la optimización topológica es encontrar la distribución de la estructura en una región específica. Las únicas variables conocidas en el problema son las cargas aplicadas, las condiciones de apoyo (restricciones de desplazamiento), el volumen final de la estructura a calcular y posiblemente algunas restricciones de diseño adicionales como ciertas zonas dónde se precisa que exista/no exista material. En este tipo de problema, la forma y conectividad de la estructura no son conocidas a priori.

La topología de la estructura no se representa mediante funciones paramétricas estándar, si no por un conjunto de funciones definidas en el dominio fijo de diseño. Estas funciones representan una parametrización del tensor de rigidez en el continuo, y es la apropiada elección de esta parametrización la que proporciona una adecuada formulación para la optimización topológica.

Minimización de la energía de deformación

A continuación de describe la formulación general para el problema de optimización topológica en términos de distribución del material. El objetivo es minimizar la energía de deformación (o maximizar la rigidez global) bajo ciertas restricciones.

$$\begin{aligned} &\min_{u\in U,D} \mathbf{F}^{\mathbf{T}}\mathbf{U} \\ &\textit{sujeto a :} \quad \mathbf{K(D)U} = \mathbf{F} \\ &\qquad \mathbf{D} \in \mathbf{D}_{ad} \end{aligned} \tag{2.7}$$

Aquí, la ecuación de equilibrio está escrita en su forma variacional, con **U** denotando del espacio de campos de desplazamiento cinemáticamente admisibles; **F** es el vector de fuerzas.

En el problema (2.7), $\mathbf{D}_{ad}$ denota el conjunto de tensores de rigidez admisibles para el problema de diseño. En el caso del diseño topológico, $\mathbf{D}_{ad}$ podría ser el conjunto de tensores que contemplan las propiedades de material para un material isotrópico dado en el (desconocido) conjunto V^{mat} (subconjunto óptimo de puntos materiales) y cero en cualquier otro sitio.

Cuando se resuelven problemas de la forma (2.7) mediante medios computacionales, generalmente se utiliza una aproximación mediante la dicretización del problema usando elementos finitos. Es importante observar que hay dos campos de interés en (2.8), a decir el campo de desplazamientos $\mathbf{U}$ y la rigidez **D**. Si se utilizan las mismas mallas de elementos finitos para ambos campos, y se discretiza **D** como una constante en cada elemento, se puede escribir la forma discretizada de (2.8) como:

$$\begin{aligned} &\min_{u\in U,D} \mathbf{F}^{\mathbf{T}}\mathbf{U} \\ &\textit{sujeto a :} \quad \mathbf{K(D}_e\mathbf{)U} = \mathbf{F} \\ &\qquad \mathbf{D}_e \in \mathbf{D}_{ad} \end{aligned} \tag{2.8}$$

Donde **U** y **F** son los vectores de desplazamientos y fuerzas respectivamente. La matriz de rigidez **K** depende de de la rigidez $\mathbf{D}_e$ en el elemento *e*, nombrados como $e=1,\ldots,N$, y se puede escribir **K** como:

$$\mathbf{K} = \sum_{e=1}^{N} \mathbf{K}_e(\mathbf{D}_e)\,, \tag{2.9}$$

Donde $\mathbf{K}_e$ es la matriz de rigidez del elemento.

Parametrización del diseño

Cuando se diseña sobre la topología de la estructura, el interés es determinar la ubicación óptima de un material isótropo dado en el espacio, es decir, determinar qué puntos del espacio deben ser material y qué puntos deben permanecer vacíos (sin material). Para entenderlo mejor, se puede pensar en una representación geométrica de una estructura similar a una imagen en blanco y negro (negro material, blanco no material). En la forma discretizada de la geometría esto corresponde a una imagen en blanco y negro de la geometría con pixeles representados por la discretización de los elementos finitos.

Si se restringe la extensión espacial al dominio de referencia *V,* se trata entonces de determinar el subconjunto óptimo V^{mat} de puntos materiales. Para el problema de optimización descrito anteriormente, esta aproximación implica que el conjunto $\mathbf{D}_{ad}$ de tensores de rigidez admisibles sea el de los tensores:

$$\mathbf{D} = t_{V^{mat}}\mathbf{D}^{\mathbf{0}}, \quad t_{V^{mat}} = \begin{cases} 1\ si\ \mathbf{x} \in V^{mat}, \\ 0\ si\ \mathbf{x} \in V / V^{mat} \end{cases}$$

$$\int_V t_{V^{mat}}\, d\Omega = Vol(V^{mat}) \le VF \tag{2.10}$$

Aquí, la última desigualdad expresa un límite, *VF*, sobre la cantidad de material a nuestra disposición, de forma que la mínima energía de deformación es para un volumen limitado (fijo). El tensor $\mathbf{D}^{\mathbf{0}}$ es el tensor de rigidez para un material isótropo dado.

Destacar que esta definición de $\mathbf{D}_{ad}$ significa que se ha formulado un problema de diseño con variables distribuidas y discretizadas en el espacio (problema de 0s y 1s).

La aproximación más utilizada para resolver este problema es la de remplazar las variables de integración por variables continuas y después introducir algún tipo

de penalización que reconduzca la solución a valores discretos entre 0 y 1. Es entonces cuando el problema de diseño para un dominio fijo es formulado como un problema de dimensionamiento (u optimización de tamaño) mediante la modificación de la matriz de rigidez, de forma que ésta dependa continuamente de una función que es interpretada como una densidad del material. Esta función resulta ser la variable de diseño. El requisito es que la optimización de como resultado diseños con regiones mayormente de material y no material. Esto significa que los valores intermedios de esta función de densidad "artifical" sean penalizados de una manera análoga a otros problemas de optimización continua con aproximación a 0s y 1s.

Una posibilidad que ha resultado muy popular y eficiente es el llamado método SIMP (Solid Isoptropic Material with Penalization), modelo de rigidez proporcional y penalizada:

$$\mathbf{D}(\mathbf{x}) = \rho(\mathbf{x})^p \mathbf{D}^0, \quad p > 1,$$
$$\int_V \rho(\mathbf{x}) dV \le VF; \; 0 \le \rho(\mathbf{x}) \le 1, \quad \mathbf{x} \in V \tag{2.11}$$

Aquí la "densidad" $\rho(\mathbf{x})$ es la función de diseño y $\mathbf{D}^0$ representa las propiedades del material para un material isotrópico dado. Se refiere a ρ como densidad de material por el hecho de que el volumen de la estructura se evalúa como $\int_V \rho(\mathbf{x}) dV$

La densidad interpola entre las propiedades de material 0 y $\mathbf{D}^0$:

$$\mathbf{D}(\rho = 0) = 0, \quad \mathbf{D}(\rho = 1) = \mathbf{D}^0 \tag{2.12}$$

queriendo decir que si un diseño final tiene densidad cero y uno en todos los puntos, es un diseño formado por píxeles blancos y negros para los cuales la representación ha sido evaluada con un modelo físico correcto. En SIMP se utiliza el parámetro de penalización p como $p>1$ de modo que las densidades intermedias son desfavorables en el sentido de que la rigidez obtenida es pequeña comparada con el coste (volumen) del material. En otras palabras, escogiendo un valor de p mayor que uno, se hace "poco rentable" tener densidades intermedias en el diseño óptimo. De este modo la penalización se consigue sin utilizar ningún esquema de penalización explícito. Para problemas donde la restricción de volumen está presente, algunos estudios demuestran que la optimización sí que funciona

realmente para aquellos diseños con p lo suficientemente grande (para obtener diseños de "0s y 1s" más realísticos se suele utilizar $p \geq 3$. Además, se ha probado que para los problemas de minimización de la energía de deformación en la forma discretizada, existen soluciones óptimas de la forma "0s y 1s" si se utiliza una p lo bastante grande. El estudio de la influencia del parámetro p en la solución final se realiza en este trabajo en la sección 3.4.2.3 (Verificación del comportamiento del programa, parámetro *Penal*)

El modelo SIMP, esquema de interpolación, convierte el problema integral en un problema de dimensionamiento (optimización de tamaño) que típicamente resulta en un diseño de "0s y 1s" a todos efectos prácticos. Esto tiene otro problema asociado que el SIMP no resuelve: la carencia de soluciones únicas en el problema de distribución del material. Concretamente los resultados computacionales obtenidos son sensibles al nivel de refinamiento de la malla de elementos finitos. Como ya se ha comentado, el esquema de interpolación no resuelve directamente este problema. Este asunto es tratado más tarde en la sección *Independencia de la malla* (apartado 3.4.3)

2.3.2 Métodos de Resolución

El uso de un esquema de interpolación como el SIMP permite convertir el problema de optimización topológica en un problema de dimensionamiento en un dominio fijo. Comparado con muchos problemas de dimensionamiento tradicionales como estructuras de barras, estructuras de cables, etc., este problema difiere en el número de variables de diseño que normalmente es muy elevado (el número de parámetros de diseño y el número de variables de análisis son del mismo orden de magnitud). Por lo tanto, la eficiencia del procedimiento de optimización es crucial y normalmente se tienen que adoptar ajustes en el proceso para intercambiar número de restricciones por número de variables de diseño. El problema que se está tratando es un ejemplo de esto. Se pueden dar muchas variables ya que el problema de optimización tiene solamente una restricción a parte de las restricciones-caja que dan los límites superior y inferior en la variable densidad.

Condiciones de optimalidad

A continuación se describen las condiciones necesarias de optimalidad para la densidad ρ del problema del mínimo de la energía de deformación que utiliza el esquema de interpolación SIMP.

Siguiendo métodos con criterio de optimización estándar usados en optimización estructural, el problema simple de carga puede ser utilizado para generar de una forma muy eficiente esquemas computacionales de actualización para resolver problemas como el que se está tratando. La clave está en idear métodos iterativos que, para un diseño ya iniciado y con sus desplazamientos asociados, actualicen las variables de diseño en cada punto (o mejor dicho en cada elemento de la malla de elementos finitos) independientemente de la actualización en otros puntos, y que se basen en las condiciones necesarias de optimización. Se muestra a continuación a modo de recordatorio el problema (2.7) escrito para el caso de la interpolación SIMP. En el marco continuo esto es:

$$\begin{aligned} &\min_{u\in U,\rho} \ \mathbf{F}^{\mathbf{T}}\mathbf{U} \\ &\textit{sujeto a :} \quad \mathbf{K}(\mathbf{D})\cdot\mathbf{U} = \mathbf{F} \\ &\qquad \mathbf{D}(\mathbf{x}) = \rho(\mathbf{x})^{p}\mathbf{D}^{0}, \\ &\int_{V} \rho(\mathbf{x})dV \le VF; \quad 0 < \rho_{min} \le \rho \le 1. \end{aligned} \tag{2.13}$$

Observar que aquí se ha introducido un límite inferior ρ_{min} sobre la densidad para prevenir cualquier tipo de singularidad en el problema de equilibrio. Normalmente $\rho_{min} = 10^{-3}$.

Para introducir las restricciones de la ecuación anterior se utiliza el método de los multiplicadores de Lagrange [1] a través de la utilización de 3 de estos multiplicadores: Λ relativo a la restricción de volumen, $\lambda^{-}(\mathbf{x})$ relativo al límite inferior de ρ y $\lambda^{+}(\mathbf{x})$ relativo a su límite superior. Bajo el supuesto que $\rho \ge \rho_{\min} > 0$ (de forma que el campo de desplazamientos es único), las condiciones de optimalidad con respecto a las variaciones del campo de desplazamientos para ρ resultan:

$$\boldsymbol{\varepsilon}^{T}\frac{\partial \mathbf{D}}{\partial \rho}\boldsymbol{\varepsilon} = \Lambda + \lambda^{+} - \lambda^{-} \tag{2.14}$$

Para densidades intermedias ($\rho_{min} < \rho < 1$), la condición de la ecuación (2.14) se puede escribir como:

$$p \cdot \rho(\mathbf{x})^{p-1} \boldsymbol{\varepsilon}^T \mathbf{D}^0 \boldsymbol{\varepsilon} = \Lambda, \tag{2.15}$$

que expresa que la densidad de energía de deformación (lado izquierdo de la ecuación) es constante e igual a Λ para todos los elementos intermedios. Como se esperan áreas con alta energía por ser demasiado bajas en rigidez, se idea el siguiente esquema de actualización para las densidades:

$$\rho_{k+1} = \begin{cases} max\{(1-\varsigma)\rho_k, \rho_{min}\} & \text{si} \quad \rho_k B_K^{\eta} \le max\{(1-\varsigma)\rho_k, \rho_{min}\}, \\ min\{(1+\varsigma)\rho_k, 1\} & \text{si} \quad min\{(1+\varsigma)\rho_k, 1\} \le \rho_K B_K^{\eta}, \\ \rho_K B_K^{\eta} & \text{en cualquier otro caso} \end{cases} \tag{2.16}$$

Aquí ρ_K denota el valor de la variable densidad en la iteración *K*, y viene dado por la expresión:

$$B_K = \Lambda_K^{-1} p \rho(x)^{p-1} \varepsilon^T(u_K) \mathbf{D^0} \varepsilon(u_K), \tag{2.17}$$

Donde $\mathbf{u}_K$ es el campo de desplazamientos en la iteración *K*, determinado en la ecuación de equilibrio y dependiente de ρ_K. Observar que un óptimo (local) es encontrado si $B_K = 1$ para densidades ($\rho_{\min} < \rho < 1$). El esquema de actualización añade material en aquellas áreas donde la energía de deformación específica es mayor que Λ (es decir, cuando $B_K > 1$) y quita material si la energía es menor que este valor; esto únicamente ocurre si la actualización no viola los límites de ρ. Integrando la ecuación (2.17) se observa que Λ es proporcional (según el factor ρ) a la media de la densidad de la energía de deformación de la parte de la estructura que resulta con los valores intermedios de densidad.

El parámetro η, ec. (2.16), es un factor de amortiguamiento y ς un límite de movimiento. Ambos controlan cambios que ocurren en cada paso de la iteración y se pueden ajustar para cambiar la eficiencia del método. Observar que la actualización depende del valor del multiplicador de Lagrange Λ, y por tanto Λ debe ser ajustado en un bucle interno para satisfacer la restricción de la fracción de volumen. El volumen de los valores actualizados de las densidades es una función continua decreciente del multiplicador de Lagrange. Además, el volumen es estrictamente decreciente en los intervalos de trabajo. Esto significa que se puede determinar el valor de Λ mediante un método de bisección o un método de Newton. Los valores de η y ζ se escogen de forma experimental de modo que

obtengan resultados que converjan de forma rápida y estable. Los valores típicos utilizados para η y ζ son 0.5 y 0.2 respectivamente.

El tipo de algoritmo descrito ha sido usado numerosas veces en diseños de topología estructural con buenos resultados y está probado que es un método efectivo para problemas de gran escala. La efectividad del algoritmo viene del hecho de que cada variable de diseño (densidades) es actualizada independientemente de la actualización del resto de variables, excepto para el reescalado que tiene la función de satisfacer la restricción de volumen.

2.3.3 Programa de Optimización Topológica básico

2.3.3.1 Funcionamiento

El programa de optimización topológica que se ha elegido para partir de base en este proyecto es un código de MATLAB creado por el "Department of Solid Mechanics" de la "Technical University of Denmark" en Dinamarca [6]. El grupo TopOpt está formado por el departamento "Mechanical Engineering" y "Mathematics" de la misma Universidad con el propósito de promover extensiones teóricas y aplicaciones prácticas en el ámbito de la optimización topológica.

En esta sección se hace una amplia explicación del funcionamiento del programa, ya que este algoritmo será el utilizado para desarrollar el programa de optimización topológica que se integrará con el software de GFEM del CITV.

Este algoritmo está basado en el método de resolución SIMP descrito en los puntos 2.3.1 y 2.3.2. La variable densidad (o peso) denotada con la letra "ρ" hasta el momento y que hace referencia a la cantidad de material que se le asigna a cada elemento tomando valores entre 0 y 1, se va denotar a partir de ahora con la letra "x", ya que es la utilizada en diversas fuentes bibliográficas [1]. De la misma manera el multiplicador de Lagrange denotado como Λ pasa a denotarse λ.

El programa implementa un código de optimización topológica para la minimización de la energía de deformación en estructuras cargadas estáticamente. Las 99 líneas del código están divididas en 36 líneas para el programa principal, 12 para el criterio de optimización, 16 para el filtro de independencia de malla, y 35 para el código de elementos finitos. Realmente, excluyendo las líneas comentadas, solo se requieren 49 líneas para resolver el problema de la optimización topológica incluyendo el análisis mediante el MEF.

Este código básico utiliza algunas simplificaciones para poder resolver de una forma muy sencilla un problema de optimización topológica:

- El dominio de diseño (geometría de trabajo) es rectangular.
- Los elementos finitos son todos cuadrados del mismo tamaño.

El problema de optimización se basa en una ley potencial, donde el objetivo es minimizar la energía de deformación. Este problema se describe en la ec. (2.13), y se reescribir como:

$$\text{Mnimizar:} \quad \left.\begin{array}{l} c(x) = \mathbf{U}^{\mathbf{T}}\mathbf{K}\mathbf{U} = \sum_{e=1}^{N} (x_e)^{p}\, \mathbf{u_e}^{\mathbf{T}}\mathbf{k_0}\mathbf{u}_e \\ \textit{sujeto a } \dfrac{V(x)}{V_0} = f \\ \mathbf{K}\mathbf{U} = \mathbf{F} \\ 0 < x_{min} \le x \le 1 \end{array}\right\} \qquad (2.18)$$

Donde **U** y **F** son los vectores de desplazamiento global y fuerza respectivamente, **K** es la matriz global de rigidez, $\mathbf{u_e}$ y $\mathbf{k_e}$ son los vectores de desplazamiento y rigidez del elemento respectivamente, x es el vector de las variables diseño (densidades o pesos), $x_{\min}$ es un valor mínimo de peso (no es cero para evitar singularidades en la matriz de rigidez), N es el número total de elementos utilizado para discretizar el dominio, p es la potencia de penalización (normalmente con valor 3), $V(x)$ y V_0 son el volumen de material y el volumen del dominio de diseño respectivamente, y f es la fracción de volumen. La fracción de volumen, recordar que es la cantidad de material que se desea que quede en el componente, expresado en tanto por 1 (respecto el volumen total de la geometría de trabajo). De esta forma, si se introduce como dato inicial una fracción de volumen de 0.5 significa que el la solución final (figura obtenida) deberá ocupar un 50 % del área de la geometría de trabajo.

Se ha de recordar también que la variable de diseño peso está asociada a cada elemento y representa una especie de densidad de material. Los valores de los pesos están comprendidos entre 0 y 1, de forma que si un elemento tiene un peso asignado de 0 significa que no hay material en esa zona y si tiene un 1 significa que sí hay material. Un peso de 0.5 significaría que sí existe material pero con una densidad la mitad que la del elemento con peso 1.

Para conseguir la minimización de la energía de deformación, se sigue un proceso iterativo donde se van actualizando los valores de los pesos en los elementos. La estructura del algoritmo se presenta a continuación:

1) Diseño inicial. Se realiza una distribución homogénea del material en la geometría de trabajo asignando a todos los elementos el valor de la fracción de volumen *f*. La parte iterativa del proceso es entonces:
2) Se calcula la matriz de rigidez global teniendo en cuenta la actual distribución de pesos. Se entiende que si un elemento tiene peso 0 (es decir, no representa material) no tiene rigidez asociada.
3) Para una distribución de densidades dada, se calcula mediante elementos finitos el campo de desplazamientos para un cierto caso de cargas.
4) Se calculan las derivadas de la energía de deformación respecto la variable de diseño (peso) y se aplica un filtro a modo de alisado de los pesos.
5) Se calculan los nuevos pesos (actualización), según el método iterativo que se explico anteriormente (se trata de un bucle interno para encontrar el valor del multiplicador de Lagrange λ para la restricción de volumen).
6) Se vuelve al paso 2.

De esta forma se va siguiendo el proceso, que va iterando hasta un punto en el que el valor de los pesos en los elementos no cambia significativamente y el bucle termina. El proceso iterativo descrito se representa a continuación:

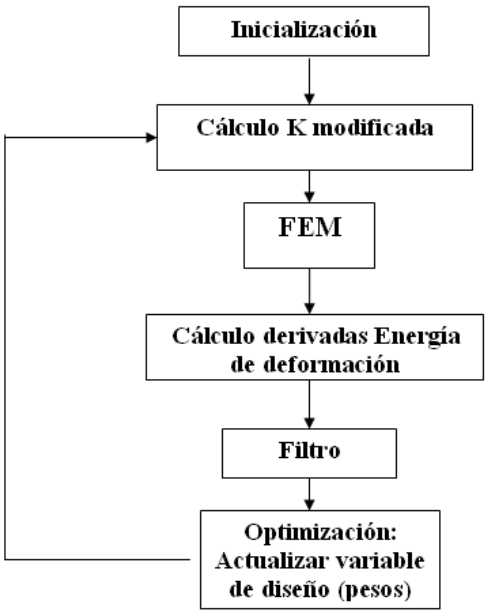


Figura 2.5 Proceso del algoritmo de optimización topológica

El problema de optimización (actualización de la variable de diseño pesos) se podría resolver utilizando diferentes aproximaciones como el *Optimality Criteria* (OC), *Sequential Linear Programming* (SLP) o el *Method of Moving Aymptotes* (MMA). Por simplicidad y manejabilidad este programa utiliza el estándar OC-method.

El método OC funciona en este caso siguiendo el esquema de Bendsoe (1995) de actualización heurística para las variables de diseño (pesos) [1]. La determinación de los pesos en la nueva iteración se describió en la sección 2.3.2 con la ec. (2.16). Se reescribe a continuación con la nueva notación de la variable peso (x en vez de ρ):

$$x_e^{new} = \begin{cases} \max\{x_{\min}, x_e - m\} & si \quad x_e B_e^{\eta} \le \max\{x_{\min}, x_e - m\},, \\ x_e B_e^{\eta} & si \quad \max\{x_{\min}, x_e - m\} < x_e B_e^{\eta} < \min\{1, x_e + m\} \\ \min\{1, x_e + m\} & si \quad \min\{1, x_e + m\} \le x_e B_e^{\eta} \end{cases} \tag{2.19}$$

Es decir, el nuevo valor de peso intentará tomar el valor $x_e B_e^{\eta}$ siempre que este valor esté comprendido en un rango de valores $[\max(x_{\min}, x_e - m), \min(1, x_e + m)]$. Esto es:

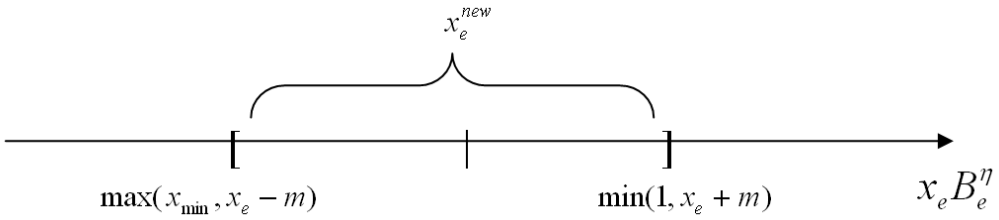


Figura 2.6 Rango de valores de x_e^{new}

Donde *m* (move) es un límite positivo, η (=1/2) es un coeficiente numérico de amortiguamiento y B_e se define de la condición de optimización según:

$$B_e = \frac{-\dfrac{\partial c}{\partial x_e}}{\lambda \dfrac{\partial V}{\partial x_e}} \tag{2.20}$$

donde λ es un multiplicador de Lagrange.

Es importante entender conceptualmente la expresión (2.20). El nuevo valor de peso, como ya se ha explicado, intentará tomar el valor $x_e B_e^{\eta}$, que depende del peso anterior (x_e) y de B_e. B_e varía:

1) de forma proporcional con la variación de la energía de deformación respecto al peso.
2) de forma inversamente proporcional a la variación del volumen total de material respecto el peso.

La variación de la energía de deformación respecto el peso será siempre negativa ya que si aumenta la cantidad de material en un elemento, la energía de deformación siempre disminuye (ya que los desplazamientos son menores). Se incluye por tanto un signo negativo en la ecuación para hacer este valor positivo. Si al aumentar un poco el peso, la energía de deformación disminuye mucho (una $-\partial c/\partial x_e$ grande), lógicamente interesará introducir más material en este elemento, por eso esta derivada afecta proporcionalmente a B_e .

Por otra parte, si aumentar un poco el peso del elemento implica un gran aumento del volumen total del material ($\partial V/\partial x_e$ grande), evidentemente no interesa introducir más material. Por eso esta derivada afecta de una forma inversamente proporcional a B_e .

El valor $\lambda\,\partial V/\partial x_e$ se determinará mediante un algoritmo de bisección de forma iterativa. Se debe prestar atención para no confundir este proceso iterativo con el bucle explicado en la Figura 2.5. Es decir, en dicho bucle se entra cada vez en el proceso de optimización para obtener una nueva distribución de los pesos en los elementos, y para obtener cada vez esta nueva distribución se requiere de otro proceso iterativo para poder calcular el valor $\lambda\,\partial V/\partial x_e$.

El algoritmo de bisección (proceso iterativo) consiste en lo siguiente: se toman unos límites iniciales l_1 (inferior) y l_2 (superior), y l_{mid} que es el valor intermedio entre l_1 y l_2 ; $\lambda\,\partial V/\partial x_e$ toma el valor de l_{mid} . Para este $\lambda\,\partial V/\partial x_e$ se calcula B_e y por tanto los nuevos valores x_e^{new} . Con esta distribución de pesos se comprueba si se está cumpliendo la condición de la fracción de volumen, y en función de esto se modifican los límites l_1 o l_2 de la siguiente forma:

- Si la distribución de pesos representa un volumen mayor a la fracción de volumen especificada:

  Esto significa que los pesos tendrán que tomar valores más pequeños, es decir, B_e tendrá que disminuir, y por tanto $\lambda\,\partial V/\partial x_e$ debe ser mayor. Para que $\lambda\,\partial V/\partial x_e$ sea mayor, se tendrá que aumentar el valor l_{mid} .

Para conseguir esto, el límite inferior l_1 toma el valor del antiguo l_{mid}, y el nuevo l_{mid} es ahora la media entre el nuevo l_1 y l_2

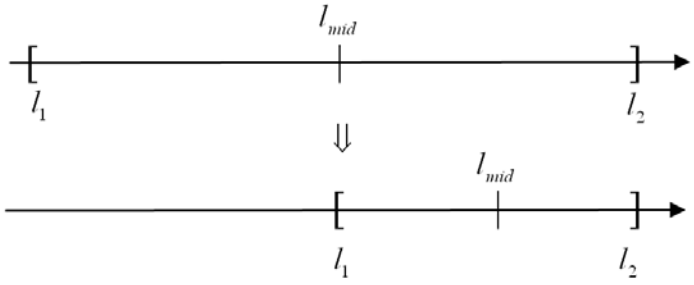


Figura 2.7 Nuevo valor de l_{mid} si los pesos representan un volumen mayor a la fracción de volumen

- Si la distribución de pesos representa un volumen menor a la fracción de volumen especificada:

Esto significa que los pesos tendrán que tomar valores más grandes, es decir, B_e tendrá que aumentar, y por tanto $\lambda \partial V / \partial x_e$ debe ser menor. Para que $\lambda \partial V / \partial x_e$ sea menor, se tendrá que disminuir el valor l_{mid}. Para conseguir esto, el límite superior l_2 toma el valor del antiguo l_{mid}, y el nuevo l_{mid} es ahora la media entre el nuevo l_2 y l_1.

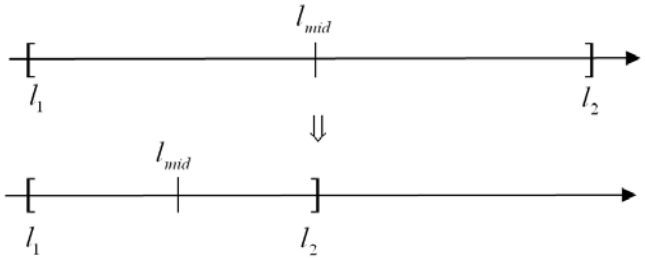


Figura 2.8 Nuevo valor de l_{mid} si los pesos representan un volumen mayor a la fracción de volumen

Esto proceso se repite iterativamente hasta que los valores l_1 y l_2 están muy cercanos (se utiliza una cierta tolerancia), de modo que la distribución de pesos cumple (prácticamente) la condición de la fracción de volumen.

Para el cálculo de B_e se precisa además de $\lambda \partial V / \partial x_e$, de la derivada de la energía de deformación (función objetivo) respecto el peso ($\partial c / \partial x_e$):

La derivada de la función objetivo se calcula como:

$$\frac{\partial c}{\partial x_e} = -p(x_e)^{p-1} \mathbf{u_e}^{\mathrm{T}} \mathbf{k_0} \mathbf{u}_e \tag{2.21}$$

Para garantizar la existencia de soluciones en el problema de la optimización topológica, se tiene que introducir algún tipo de restricción en el diseño resultante. En este programa se usa una técnica de filtrado. Para ello, las derivadas de la energía de deformación en cada elemento son modificadas de la siguiente forma:

$$\frac{\overline{\partial c}}{\partial x_e} = \frac{1}{x_e \sum_{f=1}^{N} \hat{H}_f} \sum_{f=1}^{N} \hat{H}_f x_f \frac{\partial c}{\partial x_f} \tag{2.22}$$

El operador de convolución (factor peso) $\hat{H}_f$ se define como:

$$\begin{aligned} &\hat{H}_f = r_{\min} - dist(e, f), \\ &\{ f \in N \mid dist(e, f) \le r_{\min} \}, \quad e = 1, ..., N, \end{aligned} \tag{2.23}$$

donde el operador *dist*(*e*,*f*) se define como la distancia entre el centro del elemento *e* y el centro del elemento *f*. El operador de convolución $\hat{H}_f$ es cero fuera del filtro de área, y decae linealmente con la distancia al elemento *f*. En vez de las derivadas originales, las derivadas modificadas son las que se usan en el "*Optimality Criteria update"*.

En términos prácticos, este filtro básicamente produce un alisado de las energías de deformación en los elementos, teniendo en cuenta el valor de las energías de deformación de los elementos vecinos. La variable $r_{\min}$ es el radio de cobertura que coge aquellos elementos alrededor del elemento *e* que se tendrán en cuenta para el alisado. El factor $\hat{H}_f$ da más importancia, obviamente, a los elementos más cercanos. Con esta técnica se evita que salgan soluciones muy pixeladas, dando una

mayor sensación de continuidad en la solución. Esto se muestra en secciones posteriores.

2.3.3.2 Implementación en Matlab

Programa Principal

El programa principal empieza distribuyendo el material uniformemente en el dominio de diseño. Después de unas cuantas inicializaciones, se llama a la subrutina de elementos finitos que devuelve el vector de desplazamientos **U**. Después se calculan las energías de deformación y sus derivadas para cada elemento. Las energías de deformación son modificadas con el filtro de malla-independiente explicado anteriormente, para más adelante llamar al *Optimality Criteria optimizer*. Después de cada iteración, se dibuja la distribución de densidades (pesos) en la geometría de trabajo. En este programa básico, el proceso de optimización se da por acabado cuando una variable `change` que representa el máximo cambio de densidad en una iteración, es menor que un 1 por cien. Si no ocurre, el bucle se vuelve a repetir. Se muestra a continuación el código de el programa principal .

```
        function top(nelx,nely,volfrac,penal,rmin)
        x(1:nely,1:nelx) = volfrac;
         loop = 0;
         change = 1.;
         while change > 0.01
             loop = loop + 1;
             xold = x;
             [U]=FE(nelx,nely,x,penal);
             [KE] = lk;
             c = 0.;
             for ely = 1:nely
                 for elx = 1:nelx
                     n1 = (nely+1)*(elx-1)+ely;
                     n2 = (nely+1)* elx +ely;
                     Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2;…
        2*n2+1;2*n2+2; 2*n1+1;2*n1+2],1);
                     c = c + x(ely,elx)^penal*Ue'*KE*Ue;
                     dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
                 end
             end
             [dc] = check(nelx,nely,rmin,x,dc);
             [x] = OC(nelx,nely,x,volfrac,dc);
             change = max(max(abs(x-xold)));
             disp(['It.:     '     sprintf('%4i',loop)     '     Obj.:     '
sprintf('%10.4f',c)…
                 ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
                 ' ch.: ' sprintf('%6.3f',change )])

             colormap(gray);  imagesc(-x);  axis  equal;  axis  tight;  axis
off;pause(1e-6);
         end
             figure;
             colormap(gray);  imagesc(-x);  axis  equal;  axis  tight;  axis
off;pause(1e-6);
```

Código 1. Programa principal

Filtering of sensitivities

La función `check` es la encargada de realizar el filtro de las derivadas de la energía de deformación. Como ya se explico anteriormente, con esto se consigue una solución menos pixelada y más continua. El programa original realiza esta función mediante 4 bucles `for` anidados, dos para pasar por todos los elementos (recorriendo ambos sentidos de la malla) y dos para recorrer, dentro de cada elemento, sus elementos vecinos. El resto queda explicado en la sección anterior. El código es:

```
function [dcn]=check(nelx,nely,rmin,x,dc)
dcn=zeros(nely,nelx);
for i = 1:nelx
    for j = 1:nely
        sum=0.0;
        for k = max(i-round(rmin),1):min(i+round(rmin),nelx)
            for l = max(j-round(rmin),1):min(j+round(rmin), nely)
                fac = rmin-sqrt((i-k)^2+(j-l)^2);
                sum = sum+max(0,fac);
                dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
            end
        end
        dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
    end
end
end
```

Código 2. Filtering of Sensitivities

Optimality Criteria optimizer

Este es el corazón del proceso de optimización. Aquí es donde realmente se determina cual es la forma óptima a obtener. Para calcular el peso nuevo que adquiere cada elemento, se implementa la ecuación (2.19) anteriormente descrita. En definitiva, x_e^{new} busca el valor $x_e B_e^{\eta}$, cumpliendo la condición de estar dentro de unos limites. Donde B_e es:

$$B_e = \frac{-\dfrac{\partial c}{\partial x_e}}{\lambda \dfrac{\partial V}{\partial x_e}} \tag{2.24}$$

El valor del multiplicador de Lagrange por la derivada del volumen respecto el peso $\lambda \partial V / \partial x_e$) que satisface la restricción de la fracción de volumen se puede encontrar con un algoritmo de bisección como ya se explico anteriormente. El algoritmo se inicializa con unos límites superior e inferior y después, el intervalo se parte repetidamente a la mitad hasta que su tamaño es inferior al límite de convergencia. La función utilizada para ello se muestra a continuación.

```
function [xnew]=OC(nelx,nely,x,volfrac,dc)
 l1 = 0; l2 = 100000; move = 0.2;
       while (l2-l1 > 1e-4)

       lmid = 0.5*(l2+l1);
       xnew = max(0.001,max(x-move,min(1.,…
min(x+move,x.*sqrt(-dc./lmid)))));

               if sum(sum(xnew)) - volfrac*nelx*nely > 0;
               l1 = lmid;
       else
               l2 = lmid;
       end
end
```

Código 3. *Optimality Criteria optimizer*

Código de Elementos Finitos

El código de elementos finitos utilizado es muy básico. La geometría de trabajo es siempre rectangular de manera de que se utilizan mallas regulares donde todos los elementos son exactamente iguales. Se utiliza por tanto la misma matriz de rigidez para todos los elementos, lo que hace que el código de EF sea muy eficaz.

El código entero de este programa básico de optimización topológica se encuentra en los Anexos.

Ejemplo de aplicación

Para una primera toma de contacto con el programa básico de optimización topológica, se ha analizado un caso muy simple, la mitad simétrica de una viga biapoyada con carga centrada. En la literatura inglesa se le llama como MBB (Messerschmitt-Bölkow-Blohm), pero se referirá a esta como viga biapoayada. La carga se aplica verticalmente en el lado superior izquierdo (para el caso simétrico). Se impone una condición de simetría en el lado izquierdo y la estructura se soporta en la esquina derecha inferior.

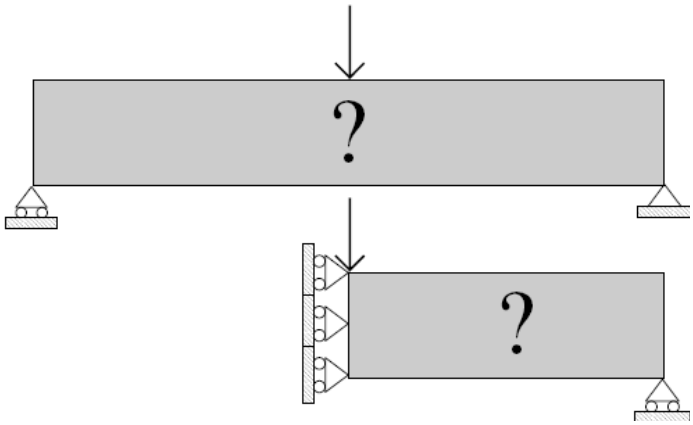


Figura 2.9 Condiciones de contorno viga biapoayada

Estas fuerzas y restricciones se introducen en el programa original accediendo manualmente a los elementos correspondientes. Uno de los objetivos de la tesis de máster es poder utilizar la interfaz gráfica disponible en el software de GFEM para poder definir interactivamente las fuerzas y restricciones.

A continuación se muestra el resultado del caso expuesto (Figura 2.10), introduciendo como parámetros: fracción de volumen = 0.5, parámetro penal = 3 y $r_{\min}$ =1.5. Se ha introducido una malla de 60x20 para su resolución:

top(60,20,0.5,3.0,1.5)

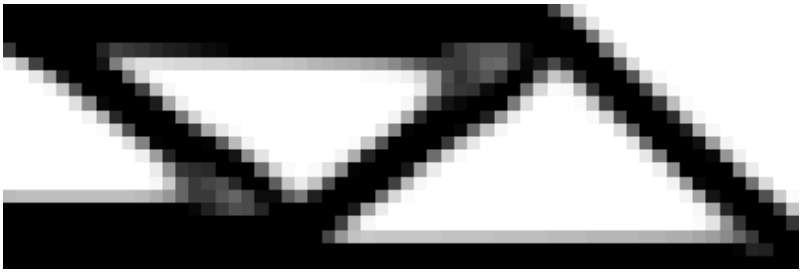


Figura 2.10 Solución viga biapoyada

Para obtener la figura óptima final, el programa ha utilizado 94 iteraciones.

3 PROGRAMA OPTIMIZACIÓN TOPOLÓGICA EN GFEM

La incorporación del programa de optimización topológica presentado anteriormente en el software de GFEM permitirá dotar a éste de una herramienta de optimización que será más potente y versátil que el programa original ya que podrá hacer uso de todas las aplicaciones que dispone este software y por tanto poder desarrollar de una forma más rápida y eficaz el programa de optimización. Las ventajas que proporcionará esta integración son:

- Introducción de datos mediante una interfaz gráfica.
- Generación de una geometría cualquiera mediante la interfaz gráfica.
- Posibilidad de definir una malla independientemente de la geometría de trabajo.
- Posibilidad de seleccionar el nivel de refinamiento de la malla.
- Posibilidad de calcular las tensiones internas del componente.

La idea general es cambiar la estructura de datos del código del programa básico de optimización para poder hacer uso del software de GFEM disponible. Se tendrán que hacer algunas modificaciones en el código para poder hacer uso de geometrías cualesquiera y poder realizar el cálculo de tensiones. Se pretende además mejorar la eficiencia del programa para reducir el coste computacional.

Una vez integrado el programa, se procede a comprobar si el proceso funciona correctamente, de modo que se comparan los resultados obtenidos con los resultados del programa básico reproduciendo el mismo caso (el de la viga viga biapoyada) para ambos códigos.

3.1 Integración del programa en el software de GFEM

En los siguientes apartados se explica detalladamente como se integra el programa de optimización topológica original [1] en el software de GFEM.

Primeramente se consideran geometrías coincidentes con los bordes de elemento, es decir, los elementos no son intersectados (apartado 3.1.1 a .3.1.3). Más adelante (apartado 3.1.4) se incluyen las modificaciones necesarias para considerar geometrías cualesquiera que corten elementos de la malla.

3.1.1 Integración

Tal y cómo se explico en los antecedentes (software disponible), la aplicación, después de haber creado todas las mallas y haber intersectado la geometría con los elementos, calcula cuales son los elementos activos para la malla en uso, es decir, aquellos elementos que están dentro o sobre la geometría. Estos elementos activos son los que necesita la subrutina `h_principal(ActiveEls)` del código de GFEM para calcular los desplazamientos de los nodos mediante el FEM. Aquí es donde se hace una de las modificaciones más substanciales del código de GFEM (no en cuanto a complejidad, sino en términos conceptuales). La matriz de rigidez de cada elemento (y por tanto la matriz de rigidez global) es modificada tal que:

$$K_e^{modificada} = (x_e)^{penal} K_e^{original} \tag{3.1}$$

de forma que la matriz de rigidez de cada elemento depende directamente del peso asociado. Si el optimizador decide que un elemento debe estar excluido (sin material), el peso es el mínimo, 0.01 (no es 0 para evitar singularidades en la matriz de rigidez global), consecuentemente no contribuye en términos prácticos a la rigidez del sistema.

A partir de ahí, la estructura seguida es básicamente la misma que la utilizada en el programa de optimización básico (Figura 2.5 mostrada anteriormente), aunque para cada uno de los pasos se realizan las modificaciones necesarias, tal y como se explica a continuación:

• **Inicialización**: Se inicializan todos los elementos con un peso igual a la fracción de volumen.

• **Cálculo K modificada**: Tal y como se ha descrito en el párrafo anterior.

• **FEM** (subrutina `h_principal`) Devuelve los desplazamientos en los nodos.

• **Cálculo derivadas Energía de deformación** (subrutina `opt_2`) Aquí se produce un cambio en la estructura de la variable que contiene esta información. En el programa base, esta variable era una matriz con tantas filas y columnas como

las filas y columnas de los elementos que formaban la malla de la geometría rectangular. En cambio, para poder darle un carácter más general y no limitarse a geometrías rectangulares, esta variable se convierte en un vector de tamaño igual al número total de elementos. Además, todas las variables que utiliza el programa de GFEM disponible están estructuradas así. Sólo se calcularán aquellos elementos que estén activos.

Se define un bucle que pasa por cada uno de estos elementos, donde se calcula la derivada de la energía de deformación de cada elemento utilizando la ecuación (2.21) que se reproduce aquí por comodidad.

$$\frac{\partial c}{\partial x_e} = -p(x_e)^{p-1}\mathbf{u_e^T k_0 u}_e \tag{3.2}$$

• **Filtro (**subrutina `check`). Aquí se precisa recoger las derivadas de energía de deformación de los elementos vecinos (tal y como se explico más arriba). El parámetro $r_{\min}$ determina el tamaño del "vecindario". Esta sub-función recibe una modificación substancial ya que la original era computacionalmente muy costosa, y se explica detalladamente en el apartado 3.1.1.1.

• **Optimización** (subrutina `OC`). En este caso ocurre lo mismo que lo explicado en la subrutina de cálculo de derivadas de energías de deformación, referente a la estructura de la variable. Se puede decir que esta subrutina es el corazón de la optimización. Esta subrutina recibe algunos cambios importantes para poder utilizar geometrías que no coinciden con la malla. Estas se explican en una sección posterior.

3.1.1.1 Implementación del Filtro

En primer lugar se debe aclarar el significado de r_{min}. Este parámetro es el radio de la circunferencia que define el área del vecindario de un elemento (Figura 3.1). r_{min} es por tanto una distancia. Dependiendo de ciertas circunstancias, tal y como se explica en la sección 3.4.3, este parámetro interesa que esté expresado como número de lados de elemento o como distancia absoluta. Para lo que sigue de trabajo y en caso de que no se indique lo contrario, r_{min} estará expresado como número de lados de elemento. Es decir, si r_{min} vale 2 significa que es igual a la longitud de dos lados de elemento.

Esta sub-rutina resultaba muy simple en el programa inicial, debido a la estructura matricial de la variable que guarda dichas derivadas, topológicamente similar a la estructura de la malla de elementos (tamaño de la matriz igual al tamaño de la malla). El problema es que se utilizan cuatro bucles `for` anidados para esta tarea y eso hace que el proceso tenga un coste computacional muy elevado, sobretodo para problemas con muchos elementos. Debido a la nueva estructura de datos de las variables y con el objetivo de mejorar la eficiencia de esta subrutina se ha modificado el código tal y como se describe a continuación.

Los pesos están estructurados ahora según un vector. Para esta sub-rutina se reestructura este vector (una sola vez) en forma de matriz de tamaño igual al tamaño de la malla. Se utiliza un único bucle que recorre los elementos activos. Para cada elemento se determina su ubicación calculando su columna *j* y su fila *i* en la malla. Después se calculan los elementos vecinos como un cuadrado centrado en dicho elemento y de lado dos veces el parámetro r_{min}. Para ello se calculan las columnas y filas iniciales y finales. Se usa un control que evita no seleccionar elementos más allá de la frontera de la malla. En la figura 16 se muestra un pequeño ejemplo para un valor $r_{\min}$ =3.

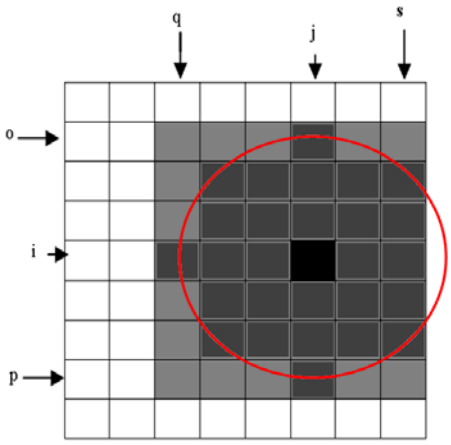


Figura 3.1 Ejemplo del filtro para $r_{\min}$ =3

Obsérvese que la columna *s* está tan solo a dos elementos de distancia del elemento en cuestión porque hace tope con la frontera de la malla.

Aquellos elementos vecinos que puedan estar fuera de la geometría de trabajo (recordar que la geometría de trabajo y la malla no son coincidentes) no son ningún

problema ya que de todo este "vecindario" se cogerán únicamente los elementos que estén activos (dentro de la geometría de trabajo). De los elementos vecinos seleccionados (todo el recuadro gris), al final solo se tienen en cuenta aquellos cuyo centro cae dentro del círculo rojo, gracias al funcionamiento del operador de convolución $\hat{H}_f$, explicado en la sección 2.3.1 , ec. (2.23).

Esta modificación del filtro puede parecer más complicada en cuanto a número de líneas empleadas en el código, pero se ha reducido substancialmente el tiempo de cálculo ya que se ha reducido de cuatro a uno los bucles necesarios. La nueva subrutina se denomina `check3`.

Para probar la mejora de la nueva implementación se ha hecho uso de la herramienta "profiler" de MATLAB que permite analizar los tiempos empleados en el proceso. Para ello se ha reproducido el mismo caso en el nuevo código y el programa original, interviniendo 2945 elementos. Los parámetros utilizados son los siguientes:

Volfrac=0.5; *Pena*l=3; $r_{\min}$=3; *NM*=4;

(donde *Volfrac* es la fracción de volumen, *Pena*l el parámetro de penalización, $r_{\min}$ el radio para el filtro y *NM* el nivel de malla empleado[1].)

Se ha medido el tiempo en recorrer una vez el filtro de derivadas de energía de deformación en ambos códigos.

[1] Nota: El nivel de malla es el sistema utilizado por el software de GFEM para determinar el nivel de refinamiento de la malla. La malla (rectangular) se subdivide inicialmente en un número de elementos que representan el nivel 0. El refinamiento uniforme en 4 elementos de cada uno de los elementos de nivel *n* genera el nivel *n+1*

Profile Summary

Generated 18-Nov-2009 12:48:02 using cpu time.

Programa original:

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
top4>check	1	0.480 s	0.480 s	

Código nuevo:

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
check3	1	0.303 s	0.303 s	

Tabla 3-1 Profile Summary 1

Se observa que se ha reducido el coste computacional en un 37 % para esta subrutina. Teniendo en cuenta que es una de las subrutinas que más tiempo invierte, este recorte de tiempo hace el proceso notablemente más eficiente. Además, cuando el número de elementos aumenta y el parámetro r_{min} es mayor, esta diferencia se pronuncia todavía más. Se ha reproducido para probarlo el mismo caso pero con una malla más fina, ahora actuan 11780 elementos. Los parámetros utilizados son:

Volfrac=0.5; *Penal*=3; r_{min} =6; *NM*=5;

Con esto se obtiene:

Profile Summary

Generated 18-Nov-2009 12:39:19 using cpu time.

Programa original

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
top4>check	1	6.367 s	6.367 s	

Código nuevo:

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
check3	1	1.415 s	1.415 s	

Tabla 3-2 Profile Summary 2

En esta ocasión se ha disminuido el tiempo de proceso en un 78 %.

Esta nueva subrutina *check3* es muy eficiente, pero el hecho de crear una matriz del tamaño de la malla para su funcionamiento hace que se limite el tamaño mínimo de elemento que se puede utilizar debido a problemas de capacidad. Debido a esto y a recientes avances en las líneas de investigación del departamento que permiten el uso de un número muy elevado de elementos en la malla, se hace necesario la implementación de una alternativa a esta subrutina.

De este modo nace la subrutina *check4,* que recurre al uso de una matriz virtual. Así pues, la matriz realmente no se crea, solo se accede a ella virtualmente. Es decir, el esquema de búsqueda esta basado en la estructura de una matriz. En este caso la obtención de los vecinos es un poco más costosa ya que de la forma anterior únicamente se tenía que acceder a la matriz mientras que ahora se deben calcular los vecinos. De forma general se ha probado que la rutina check4 tarde el doble (1.97 veces más) que la check3, pero como ventaja queda solucionado el problema de la capacidad de memoria.

Así pues, se podrá utilizar la subrutina *check3* de forma general, y la subrutina *check4* cuando se tenga un número muy elevado de elementos

3.1.2 Criterio de Convergencia

El programa básico inicial determinaba la convergencia de la solución tal y como se explica a continuación. Se calculaba en cada iteración la diferencia de pesos entre la nueva distribución de pesos y la anterior para cada elemento. De todas estas diferencias se escoge el valor máximo que se le denomina `Change` y se compara con un determinado valor llamado `MinChange`. Cuando `Change` es menor que `MinChange` se da por finalizado el proceso.

Este criterio si bien funciona en la mayoría de los casos, puede fallar cuando la distribución de pesos varía entre dos opciones muy similares pero no progresa.

Para solucionar esto y con el objetivo de determinar con más precisión el momento en el que el proceso converge, se consideran además otros criterios.

En primer lugar, además de considerar el cambio máximo de pesos, se considerará también el cambio medio. Es decir, se suman las diferencias de pesos entre la nueva distribución y la anterior y se divide entre el número de elementos. Este valor coge el nombre de `TotalRelChange` (Cambio total relativo).

Por otra parte, considerando que el programa tiene como objetivo minimizar la energía de deformación, se calcula la varibale "`SESlope`" (pendiente de la energía de deformación) que representa la variación de la energía de deformación en tanto por cien.

Por tanto, para cada iteración se calculan los parámetros. `Change`, `TotalRelChange` , y `SESlope` que miden la variabilidad del proceso. Con estos se define el criterio de convergencia comparándolos con los valores límite que son `MinChange`, `MinTotalRelChange` y `MinSESlope`, tal y como se explica a continuación. El proceso se da por finalizado si se cumple alguna de las tres siguientes condiciones:

- `Change` < `MinChange`
- -(Media de `SESlope` 3 últimas iteraciones) < `MinSESlope`
  y `Change` < `MinChange2`
- (Media de `TotalRelChange` 3 últimas iteraciones) < `MinTotalRelChange`
  y -(Media de `SESlope` 3 últimas iteraciones) < `MinSESlope2`
  y `Change` < `MinChange3`

La idea de trabajar con los valores de las tres últimas iteraciones se debe a que una iteración puede progresar muy poco sin que haya convergido todavía el proceso. A partir de datos experimentales se ha determinado que los valores límite más recomendables son los que se proponen a continuación:

`MinChange` = 0.02

`MinSESlope` = 0.0001

`MinChange2` = 0.07

`MinTotalRElChange` = 0.001

`MinSESlope2` = 0.0005

`MinChange3` = 0.075

Los valores anteriores se pueden modificar si se pretende acelerar el proceso de convergencia a costa de obtener un solución un poco menos precisa. Puede sorprender que los valores de `MinSESlope` son muy bajos, pero en muchas ocasiones la energía de deformación disminuye muy poco porque lo único que cambia es una pequeña zona del componente.

Los parámetros de variabilidad se calculan en la rutina `Variability`, mientras que las comprobaciones para la convergencia en `CheckEndTOLoop`.

3.1.3 Comprobación con geometrías coincidentes con contornos de elemento

Llegado a este punto, el programa básico de optimización topológica original queda incorporado al software de GFEM del departamento. Todavía faltan las modificaciones pertinentes para poder utilizar geometrías no coincidentes con la malla, pero se ha preferido comprobar primero el buen funcionamiento del software para geometrías que sí coinciden con la malla de elementos finitos. De esta forma se podrá comparar el resultado con la solución obtenida a través del programa básico original.

Para ello se ha analizado el caso de la viga biapoyada con una geometría de trabajo idéntica y mismo número de elementos en la malla. Se han utilizado 96x32 elementos (3072 elementos en total) con una fracción de volumen de 0.5, un parámetro de penalización de 3 y un $r_{\min}$ de 3. El resultado obtenido en ambos casos se muestra a continuación:

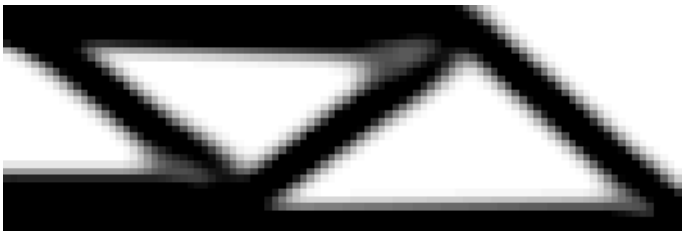


Figura 3.2 Resultado obtenido con el programa de optimización topológica original

Figura 3.3 Resultado obtenido con el nuevo código (software disponible con el programa de opt. top. base integrado)

Tal y como se observa, ambas figuras resultan a simple vista prácticamente la misma. La pequeña diferencia en la solución final proviene de la diferencia en el cálculo de los desplazamientos en nodos. El caso es que en la resolución mediante elementos finitos utilizada por el software disponible, los desplazamientos en contornos de Dirichlet se imponen, en forma débil, utilizando el método Mortar (basado en el método de los multiplicadores de Langrange) que permite que los contornos donde se aplican estas restricciones no coincidan con bordes de elemento. El campo de desplazamientos en nodos calculado de esta forma difiere un poco del método convencional, mayormente en zonas cercanas a los contornos de Dirichlet (alrededor de un 1 %). Aún así, esa diferencia es lo bastante pequeña como para despreciarla.

3.1.4 Uso de geometrías de trabajo cualesquiera (no coincidentes con contornos de elemento) y elementos de diferente tamaño

Hasta este punto, la geometría había sido meramente rectangular y coincidente con la malla debido a la limitación del programa base. A continuación se pretende

conseguir que la geometría pueda ser independiente de la malla para poder aprovechar al máximo las prestaciones del GFEM y su interfaz gráfica disponible.

Para ello, es esencial que el programa pueda utilizar elementos que no estén enteros dentro de la geometría. Afortunadamente, el software disponible ya realiza los cálculos oportunos para intersectar la geometría con estos elementos y mediante una técnica de triangulación calcular las matrices de rigidez correspondientes para cada elemento.

El hecho de que un elemento esté intersectado por un contorno de la geometría de trabajo implica que sólo un parte del área de este está activa. Esto es equivalente a tener un elemento de menor tamaño en la malla. Dado que en la segunda parte de la tésis se aborda el tema de la adaptativdad de la malla, donde aparecen elementos de diferente tamaño, las modificaciones que se adoptan en esta sección se harán teniendo en cuenta que más adelante se van a utilizar elementos de diferente tamaño, ya que la esencia del problema es la misma que la del problema de elementos parcialmente dentro de la geometría.

La presencia de elementos de diferente tamaño repercute en la expresión de la actualización de la variable peso:

$$x_e^{new} = x_e^{old} \left(\frac{-\dfrac{\partial c}{\partial x_e}}{\lambda \dfrac{\partial V}{\partial x_e}} \right)^{\eta} \tag{3.3}$$

El volumen (volumen de material) que aporta un elemento se define como

$$\text{Volumen aportado} = x_e \cdot V_e$$

siendo V_e el volumen del elemento. De modo que si un elemento tiene un peso de l, quiere decir que aporta un volumen de valor V_e. Por otra parte, el volumen total ocupado por el componente es la suma de los volúmenes aportados por todos los elementos:

$$V = \sum_{e=1}^{N} x_e \cdot V_e \tag{3.4}$$

De esta forma es fácil entender que:

$$\frac{\partial V}{\partial x_e} = V_e \tag{3.5}$$

Obsérvese que en 2D, cuando se habla de volumen se refiere al área. De modo que el volumen del elemento (V_e) es el área del elemento (A_e).El cálculo del nuevo peso de un elemento queda por tanto como:

$$x_e^{new} = x_e^{old}\left(\frac{-\frac{\partial c}{\partial x_e}}{\lambda\frac{\partial V}{\partial x_e}}\right)^{\eta} = x_e^{old}\left(\frac{-\frac{\partial c}{\partial x_e}}{\lambda \cdot V_e}\right)^{\eta} \overset{(en\,2D)}{=} x_e^{old}\left(\frac{-\frac{\partial c}{\partial x_e}}{\lambda \cdot A_e}\right)^{\eta} \tag{3.6}$$

Tal y como estaba el programa hasta el momento (geometría coincidente con bordes de elemento) el área de todos los elementos (A_e) era la misma. Es decir, el valor $\lambda\partial V/\partial x_e$ era constante para todos los elementos. Por tanto en el proceso iterativo para encontrar todos los nuevos pesos, el valor l_{mid} correspondía a $\lambda\partial V/\partial x_e$.

$$x_e^{new} = x_e^{old}\left(\frac{-\frac{\partial c}{\partial x_e}}{\lambda\frac{\partial V}{\partial x_e}}\right)^{\eta} = x_e^{old}\left(\frac{-\frac{\partial c}{\partial x_e}}{l_{mid}}\right)^{\eta} \tag{3.7}$$

Recordar que el valor de l_{mid} se debe encontrar mediante el algoritmo de bisección para satisfacer la condición de que el volumen del componente sea igual al especificado (fracción de volumen), (véase sección 2.3.3.1).

Debido a la introducción de elementos de diferente tamaño el valor $\partial V/\partial x_e$ varía (toma el valor de A_e) y por tanto el valor a encontrar mediante el algoritmo de bisección ya no es $\lambda\partial V/\partial x_e$ si no únicamente λ (multiplicador de Lagrange). Es decir, $l_{mid} = \lambda$. Esto queda tal que:

$$x_e^{new} = x_e^{old}\left(\frac{-\frac{\partial c}{\partial x_e}}{\lambda\frac{\partial V}{\partial x_e}}\right)^{\eta} = x_e^{old}\left(\frac{-\frac{\partial c}{\partial x_e}}{l_{mid}\cdot V_e}\right)^{\eta} \overset{(en\,2D)}{=} x_e^{old}\left(\frac{-\frac{\partial c}{\partial x_e}}{l_{mid}\cdot A_e}\right)^{\eta} \tag{3.8}$$

Obsérvese que esta nueva configuración es válida tanto para mallas con elementos de diferente tamaño y/o elementos parcialmente dentro de la geometría como para mallas con elementos regulares (se trata de un caso específico). Aunque pueda parecer en primera instancia que las expresiones (3.7) y puedan dar valores de pesos (x_e^{new}) diferentes para una misma malla regular, hay que destacar que esto no ocurre porque como se ha explicado, el valor de l_{mid} cambiaría para satisfacer la condición de fracción de volumen. A fin de cuentas se puede entender el valor l_{mid} como un escalado de los valores de los pesos en todos los elementos para que se cumpla el valor de la fracción de volumen especificada por el usuario.

De este modo, tenemos

$$x_e^{new} = x_e^{old}\left(\frac{-\frac{\partial c}{\partial x_e}}{\lambda\frac{\partial V}{\partial x_e}}\right)^{\eta} = x_e^{old}\left(\frac{-\frac{\partial c}{\partial x_e}}{l_{mid}\cdot A_e}\right)^{\eta} \tag{3.9}$$

donde:

$$\frac{\partial c}{\partial x_e} = -p(x_e)^{p-1}\mathbf{u_e^T k_0 u}_e \tag{3.10}$$

Esta manera de calcular los pesos y energía de deformación presenta sin embargo un problema cuando se trata con elementos de diferente tamaño. Aunque la matriz de rigidez no dependa del tamaño del elemento en problemas de dos dimensiones (es la misma si la forma se mantiene), la energía de deformación varía proporcionalmente y de forma directa con el área. De modo que si tenemos un elemento cuadrado de área A con una energía de deformación E y lo subdividimos

en cuatro cuadrados de igual tamaño, cada uno de área A/4, la energía de deformación de cada uno de estos es de E/4.

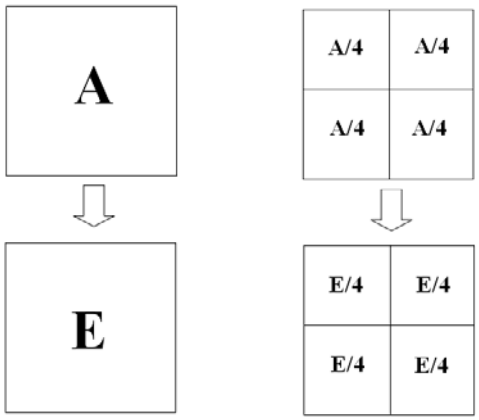


Figura 3.4 Efecto de la subdivisión sobre la energía de deformación

Esto se debe a que la diferencia entre el valor de los desplazamientos en nodos en uno de los sub-elementos mostrados es menor (una cuarta parte) que la del elemento entero, y esto se refleja en el producto $\mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e$

Este hecho presenta un problema especialmente en el filtro de la derivada de la energía de deformación ya que en el alisado de un elemento que está rodeado por elementos de tamaños diferentes habría que escalar cada uno de los valores de la energía de deformación en función del tamaño de éste. Para evitar esto, se ha planteado reescribir las ecuaciones (3.9) y (3.10) de la siguiente manera:

$$x_e^{new} = x_e^{old} \left(\frac{-\dfrac{\partial c}{\partial x_e}}{l_{mid}} \right)^{\eta} \tag{3.11}$$

donde:

$$\frac{\partial c}{\partial x_e} = \frac{-p(x_e)^{p-1}\mathbf{u_e}^{\mathbf{T}}\mathbf{k_0}\mathbf{u}_e}{A_e} \tag{3.12}$$

De modo que se pasa de dividir entre el valor del área del elemento dentro del cálculo de la derivada de la energía de deformación. De esta forma se evita el problema mencionado anteriormente y además se trabaja con energía de deformación por unidad de área (E_0) lo que permite que ésta se distribuya uniformemente:

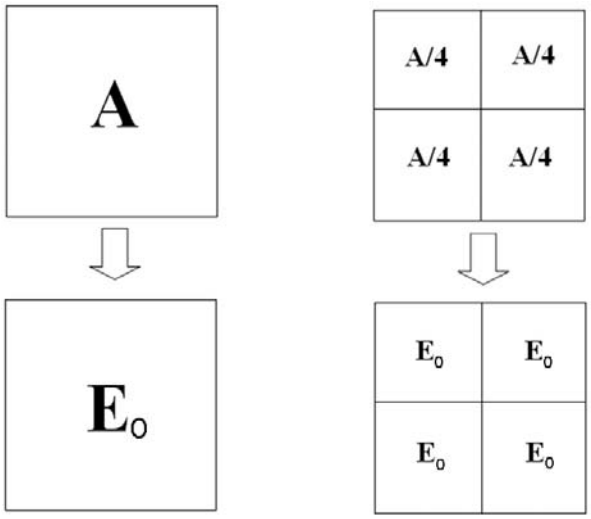


Figura 3.5 Efecto de la subdivisión sobre la energía de deformación tras el cambio

Se ha creado una función llamada `ElementArea` que calcula las áreas de trabajo de cada elemento. Se tiene que realizar además otra modificación en el *Optimality Criteria* (implementado con la subrutina `OC`). Recuérdese que el algoritmo realiza el proceso de iteración hasta encontrar la solución para la cual la suma de los pesos de todos los elementos es igual a la fracción de volumen por el número de elementos. La condición era tal que:

$$if \quad \sum_{i=1}^{N} x_i - f \cdot N \tag{3.13}$$

(donde f representa la fracción de volumen y N el número total de elementos)

Ahora, con la modificación de normalizar la energía de deformación respecto el área, esta condición queda tal que:

$$if \quad \sum_{i=1}^{N} x_i \cdot A_{e,i} - f \cdot A_{total} \tag{3.14}$$

De forma que el peso de cada elemento se ve afectado por el área del elemento, sólo aquella que está dentro de la geometría para elementos intersectados.

Estas modificaciones son suficientes para poder crear cualquier geometría de trabajo y que ésta sea independiente de la malla, ya que ahora se usa energía de deformación por unidad de área (y sus derivadas) para calcular la variable de diseño (pesos). Llegado a este punto, por tanto, ya se puede hacer uso de la interfaz gráfica del software disponible.

Para probar el correcto funcionamiento de la aplicación con geometrías no coincidentes con la malla, se ha reproducido el caso de la viga biapoyada tal y como se muestra a continuación:

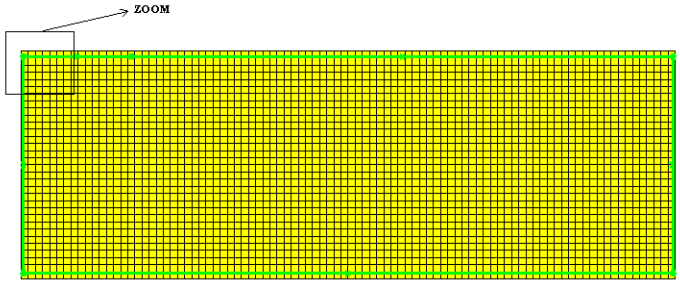


Figura 3.6 Caso de la viga biapoyada con geometría no coincidente

Figura 3.7 Ampliación zoom Figura 3.6

Se observa que la geometría no coincide con la malla de modo que los elementos intersectados no están enteros dentro de la geometría de trabajo. La

solución obtenida con las modificaciones indicadas anteriormente ha sido como se esperaba la misma que la obtenida en la Figura 3.3 donde se utilizaban geometrías coincidentes. El resultado se muestra a continuación:

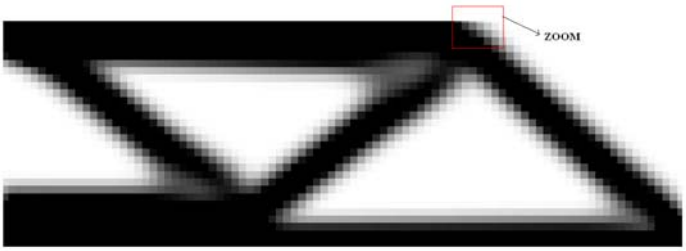


Figura 3.8 Solución para el caso de la Figura 3.6

Figura 3.9 Zoom Figura 3.8

Se concluye por tanto que la aplicación para geometrías no coincidentes con la malla funciona correctamente. Si se observa detenidamente en la frontera de la geometría de trabajo (mirar el zoom mostrado en la Figura 3.9), se ven claramente los elementos intersectados. El software disponible utiliza una función para dibujar valores constantes en la geometría de trabajo. En este caso se ha hecho uso de esta función para dibujar los pesos de los elementos. Cabe aclarar aquí, que debido al mayor coste computacional que requiere esta función para representar la figura obtenida, se va a hacer uso en la mayor parte de los ejemplos de la tesis de máster del modo de representación utilizado en el programa básico de optimización topológica. En este caso los elementos intersectados se representan enteros, pero hay que tener claro que es solo a modo de representación, porque realmente estos elementos solo aportan la porción que está dentro de la geometría de trabajo.

Como ya se dijo anteriormente, la modificación establecida en esta sección sirve tanto para el uso de geometrías no coincidentes con la malla como para el uso de elementos de diferente tamaño. La comprobación de esta última aplicación se hace más adelante en la sección 4 cuando se estudia la adaptatividad de la malla.

3.1.5 Geometrías fijas

Como ya se explico, la optimización topológica juega con una gran ventaja frente a la optimización de forma (shape optimization): las geometrías que se generan no dependen de un diseño inicial, y se pueden formar tantos agujeros como se precisen. Por decirlo de alguna forma, hay una menor limitación para crear nuevas ideas. Pero en determinadas ocasiones, se precisan limitaciones para cumplir ciertos requisitos del diseño, y a priori la optimización topológica no cuenta con estas a menos que se haga algo al respecto. Es el caso de zonas en la geometría de trabajo donde debe existir necesariamente material. En ocasiones se precisa material en ciertas zonas por razones constructivas.

Para solucionar este problema se ha propuesto crear, mediante la interfaz gráfica, geometrías adicionales que definan zonas sobre la geometría de trabajo, donde el material tenga que existir necesariamente. La idea es asignar pesos de valor máximo (1) y siempre constante a los elementos que estén dentro de esta zona.

Estas geometrías adicionales reciben el nombre de geometrías fijas, y estarán definidas por contornos independientes. Para ello se crea una variable nueva dentro de `Geo` llamada `Geo.Contour(i).Fixed`, de modo que si el contorno *i* adquiere valor `Geo.Contour(i).Fixed=1` se tratará de un contorno que define un área interior de geometría fija, y si adquiere valor 0 se tratará de una geometría de trabajo común.

Esto se implementa en la interfaz gráfica en *Pre-procesor* → *Points And Curves*. Se ha añadido a la ventana la parte de definición de contornos, quedando tal y como se ve en la Figura 3.10

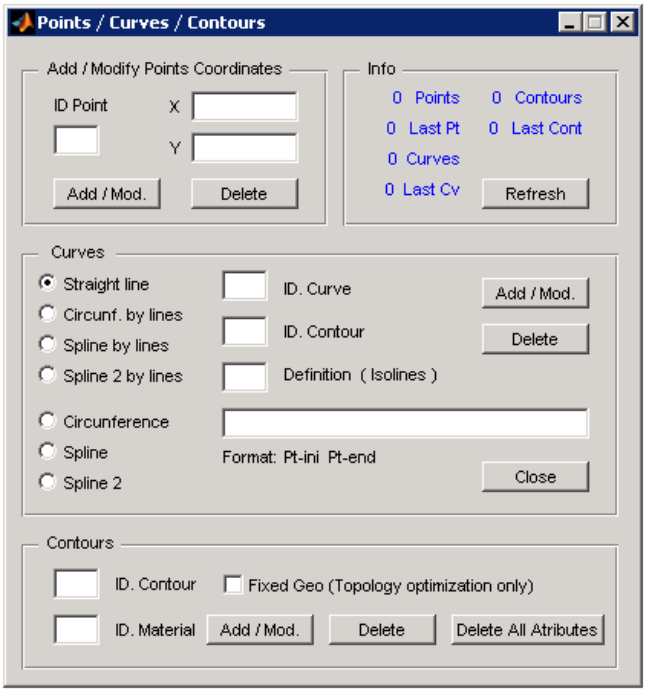


Figura 3.10 Ventana para definición de puntos/curvas/contornos de la interfaz gráfica

En la casilla ID.Contour se introduce el número de contorno (antes definido en Curves) y se selecciona "Fixed Geo" si se desea geometría fija. En caso contrario se tratará como geometría de trabajo.

Se ha añadido aquí también la opción de definir diferentes materiales dentro de la geometría de trabajo, ya que el procedimiento sería muy similar. Es decir, se crea un contorno y se le asigna un material. El programa seleccionará los elementos del contorno y les asignará las propiedades de dicho material. Aunque en esta tesis de máster no se vaya a tratar este asunto, parecía oportuno implementar esta opción para futuros trabajos en la línea de investigación.

El procedimiento a seguir es el siguiente:

- Después de la inicialización de parámetros, numeración de elementos y nodos, cáculo de elementos vecinos, etc., se comprueba si existen contornos de geometrías fijas.

-Si es así:

- Se guardan con otros nombres las variables que van a ser tratadas.

- Se intersecta dicho contorno (la geometría fija) con la malla (función `Prueba_intersect`), y se obtienen aquellos elementos que están dentro del contorno. A estos se les llama `FixedEls`.

- Se recuperan las variables tratadas, para que tomen el valor que tenían antes de realizar esta operación.

- A continuación se calculan (bien haya geometrías fijas o no) los elementos Activos intersectando los contornos de la geometría de trabajo con la malla. Estos reciben el nombre de `ActiveEls`.

- Ahora se trata de encontrar aquellos elementos que pertenezcan a la geometría fija y además sean `ActiveEls` (`FandAEls`, Fixed and Active Elements). Eso se consigue con una nueva subrutina llamada `findFandAEls`:

```
FandAEls=findFandAEls(FixedEls,ActiveEls);
```

- Finalmente se les asigna a estos elementos un peso de valor 1 (para que exista material) durante todo el proceso de optimización.

Cuando tengamos geometrías fijas, el optimizador (subrutina `OC`) trabajará únicamente con los elementos que estén activos y no estén fijos, que se han etiquetado como `Param.NonFixedElm`. La fracción de volumen estará referida a la zona de trabajo que no sea geometría fija. Cabe esperar que el proceso funcione correctamente ya que el algoritmo utilizado trabaja de forma que la actualización del peso de cada elemento en cada iteración es independiente de los pesos de los demás elementos. De esta forma queda solucionado el problema de restricción de zonas para existencia necesaria de material.

A modo de ejemplo para comprobar el correcto funcionamiento, se ha analizado otra vez el caso de la viga biapoyada añadiendo dos zonas donde tiene que existir necesariamente material:

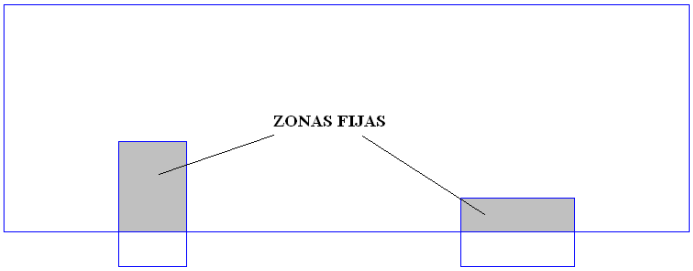


Figura 3.11 Definición del problema para geometría fijas

La solución obtenida ha sido:

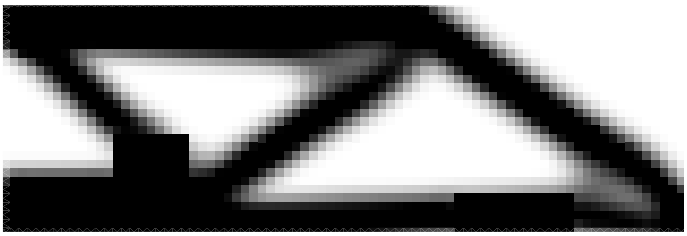


Figura 3.12 Solución para el caso de la Fig.23

Se concluye por tanto que la aplicación para utilizar geometrías fijas donde tenga que existir necesariamente material funciona correctamente.

3.1.6 Fracciones de volumen pequeñas

Se han analizado diferentes ejemplos con el software que se ha obtenido tras las modificaciones explicadas y se ha observado que la aplicación no funciona correctamente para fracciones de volumen pequeñas (del orden de 0.1 y menores). Tras una fase de investigación para encontrar el fallo se ha observado que el problema reside en la subrutina del optimizador (`OC`) del programa original debido a la naturaleza del algoritmo.

Recuérdese que en la actualización de los pesos x_e^{new} trata de buscar el valor $x_e B_e^{\eta}$ (siempre que este valor esté dentro de un rango), donde

$$B_e = \frac{-\dfrac{\partial c}{\partial x_e}}{\lambda \dfrac{\partial V}{\partial x_e}} \tag{3.15}$$

Recuérdese también que el optimizador trabaja con un algoritmo de bisección para encontrar el valor de $\lambda \partial V / \partial x_e$, de modo que se tienen un límite inferior l_1 y uno superior l_2 que van tomando valores según el volumen (para una configuración dada de pesos) sea mayor o menor que la fracción de volumen, reduciéndose el rango progresivamente hasta llegar a un cierto valor que es $\lambda \partial V / \partial x_e$.

Si la fracción de volumen es pequeña, el hecho de que el peso de un elemento cambie de valor, afecta de forma más pronunciada al volumen total del componente. Es decir, si la fracción de volumen disminuye la derivada del volumen total respecto al peso de un elemento ($\partial V / \partial x_e$) aumenta. Esto se ha probado experimentalmente y además se ha observado que esa relación es exponencial: a medida que la fracción de volumen disminuye, $\partial V / \partial x_e$ aumenta exponencialmente. Esto repercute en que el valor $\lambda \partial V / \partial x_e$ aumente muchísimo. El problema que se había encontrado residía entonces, en que el valor original del límite superior estaba tomando como valor una cifra demasiado pequeña, menor al valor $\lambda \partial V / \partial x_e$. En el programa de optimización topológica original [1] ese valor era de 100.000, pero para fracciones de volumen del orden de 0.05 se puede llegar hasta valores del orden de 10^{15}. La solución ha consistido sencillamente en elevar el valor original del límite superior para que este suceso no ocurra más. Se ha elevado a una cifra muy alta (10^{30}) ya que se observa que no influye mucho en el tiempo requerido para el proceso.

A raíz de este estudio también se ha observado que a medida que el tamaño de elemento disminuye, el valor $\partial V / \partial x_e$ también lo hace y de modo lineal. Esto es comprensible, ya que un elemento la mitad de pequeño, tiene un peso asociado que influye la mitad en el volumen total del componente.

Además, durante las pruebas realizadas con valores de $\lambda \partial V / \partial x_e$ altos para comprobar el buen funcionamiento de la aplicación, se ha observado que para valores del orden de 10^{11}, el código presenta un error. Para salir del bucle del método de bisección, se utiliza una tolerancia de 10^{-4}, es decir, el proceso finaliza

cuando la diferencia entre el límite superior e inferior es de 10^{-4}. El problema es que para valores del orden de 10^{11}, esa diferencia requiere el uso de al menos 15 cifras, que es el máximo de cifras que utiliza MATLAB. Por tanto para valores por encima de 10^{11}, la aplicación no es capaz de calcular el $\lambda \partial V / \partial x_e$. Para solucionar el problema se ha modificado la condición para terminar la iteración, de modo que la diferencia entre el límite superior e inferior no sea en valor absoluto si no en valor relativo (en porcentaje). De este modo, la condición del bucle `while` de la subrutina `OC` queda como:

$$while \quad (100 * \frac{l_2 - l_1}{l_1} > 10^{-7}) \tag{3.16}$$

Se ha decidido normalizar respecto l_1 porque l_2 toma valores muy grandes inicialmente (que haría cumplir la condición). Para evitar singularidades se ha tenido que modificar el valor inicial de l_1 de 0 a un valor cercano a 0 (10^{-50}).

Con estas modificaciones, ya se pueden utilizar fracciones de volumen tan pequeñas como requiera la aplicación.

3.1.7 Cálculo de tensiones

El software disponible de GFEM tiene la posibilidad de calcular las tensiones internas del componente para un campo de desplazamientos dado. Recientemente el algoritmo implementado para ello ha sido mejorado por D. Enrique Nadal Soriano como parte de su tesis de máster en el *Máster de Ingeniería Mecánica y de Materiales* [5]. Este nuevo algoritmo toma las tensiones del MEF previamente calculadas en cada elemento (denominadas $\boldsymbol{\sigma}_e^h$) y aplica un procedimiento de reconstrucción para obtener un campo $\boldsymbol{\sigma}^*$ de gran precisión que tiene un grado muy alto de cumplimiento de las ecuaciones de equilibrio y compatibilidad.

El proceso de cálculo de tensiones es perfectamente aplicable a la optimización topológica, ya que cuando acaba el proceso de cálculo de la distribución de densidades en el componente se dispone del campo de desplazamientos en nodos para esa distribución y cargas específicas.

Recordar que las tensiones $\boldsymbol{\sigma}$ en un elemento se calculan entonces a partir de los desplazamientos de sus nodos $\mathbf{u}_e$. Estos desplazamientos son conocidos a partir

del vector global de desplazamientos nodales **U**. De la teoría de la elasticidad, la relación entre tensiones y deformaciones para comportamiento lineal es

$$\boldsymbol{\sigma}=\mathbf{D}(\boldsymbol{\varepsilon}-\boldsymbol{\varepsilon}_0)+\boldsymbol{\sigma}_0 \tag{3.17}$$

en la que las deformaciones mecánicas $\boldsymbol{\varepsilon} = \mathrm{B}\cdot\mathrm{u}_e$ están producidas por los desplazamientos de los nodos. La matriz B se obtiene de la aplicación del operador L (que contiene primeras derivadas) sobre la matriz de funciones de forma (que dependen de las coordenadas), con lo que deberá evaluarse en los diferentes puntos del elemento en donde se deseen calcular las tensiones. Suponiendo $\boldsymbol{\varepsilon_0} = \boldsymbol{\sigma_0} = \mathbf{0}$ (por simplificar), la tensión en un elemento queda:

$$\boldsymbol{\sigma}_e^h = \mathbf{D}\cdot\mathbf{B}\cdot\mathbf{u}_e \tag{3.18}$$

Ahora bien, en el caso que nos ocupa, esta expresión se tendrá que ver modificada de alguna forma, porque la existencia de las densidades (pesos) en los elementos implica que este pueda existir (como material) o no (o existir con una densidad intermedia). Para averiguar como abordar el problema, se expone el siguiente planteamiento:

$$\mathbf{2}\Pi_e = \mathbf{U}_e{}^{\mathbf{T}}\mathbf{K}\mathbf{U}_e\mathbf{;} \tag{3.19}$$

Donde Π_e se refiere a la energía de deformación del elemento e.

Tal y como se explico en secciones anteriores, la energía de deformación con la presencia de las densidades de elemento se calcula como:

$$\mathbf{2}\Pi_e\mathbf{'} = x_e^{penal}\cdot\mathbf{U}_e{}^{\mathbf{T}}\mathbf{K}\mathbf{U}_e = x_e^{penal}\cdot\mathbf{2}\Pi_e \tag{3.20}$$

Siendo $\Pi_e\mathbf{'}$ la energía de deformación con existencia de densidades de elemento.

Por otra parte, la energía de deformación se puede calcular como:

$$\mathbf{2}\Pi_e = \int\boldsymbol{\sigma}\cdot\boldsymbol{\varepsilon}\cdot dV = \int\boldsymbol{\sigma}^{\mathbf{T}}\cdot\mathbf{D}^{-\mathbf{1}}\cdot\boldsymbol{\sigma}\cdot dV \tag{3.21}$$

Y la energía de deformación considerando la existencia de densidades de elemento:

$$\mathbf{2}\Pi_e\mathbf{'} = \int\boldsymbol{\sigma}\mathbf{'}\cdot\boldsymbol{\varepsilon}\cdot dV = \int\boldsymbol{\sigma}\mathbf{'}^{\mathbf{T}}\cdot\mathbf{D}^{-\mathbf{1}}\cdot\boldsymbol{\sigma}\mathbf{'}\cdot dV \tag{3.22}$$

siendo $\boldsymbol{\sigma}\mathbf{'}$ la tensión considerando la existencia de densidades de elemento.

Si consideramos que $\boldsymbol{\sigma}'$ se puede calcular como

$$\boldsymbol{\sigma}' = x_e^m \cdot \boldsymbol{\sigma} \tag{3.23}$$

Entonces:

$$\begin{aligned} 2\Pi_e' &= \int \boldsymbol{\sigma}'^{\mathbf{T}} \cdot \mathbf{D}^{-1} \cdot \boldsymbol{\sigma}' \cdot dV = \int x_e^m \boldsymbol{\sigma}^{\mathbf{T}} \cdot \mathbf{D}^{-1} \cdot x^m \boldsymbol{\sigma} \cdot dV = \\ &= x_e^{2m} \cdot \int \boldsymbol{\sigma}^{\mathbf{T}} \cdot \mathbf{D}^{-1} \cdot \boldsymbol{\sigma} \cdot dV = x_e^{2m} \cdot 2\Pi_e \end{aligned} \tag{3.24}$$

Combinando las ecuaciones (3.20) y (3.24) se obtiene:

$$x_e^{penal} = x_e^{2m} \quad \Rightarrow \quad m = \frac{penal}{2} \tag{3.25}$$

Por tanto:

$$\boldsymbol{\sigma}' = x_e^{\frac{penal}{2}} \cdot \boldsymbol{\sigma} \tag{3.26}$$

De esta forma, se tendrá que modificar el cálculo de tensiones de forma que:

$$\boldsymbol{\sigma}_{\mathbf{e}} = x_e^{\frac{penal}{2}} \cdot \mathbf{D} \cdot \mathbf{B} \cdot \mathbf{u}_e \tag{3.27}$$

En el siguiente apartado se utiliza el cálculo de tensiones para desarrollar una aplicación que permite, mediante un proceso iterativo, evaluar la cantidad y distribución de material óptima para conseguir componentes en los que no se supere la tensión límite de fallo.

3.2 Optimización de componentes bajo criterio de fallo

Hasta el momento, el programa creado recibe como variable de entrada la fracción de volumen, que se mantiene constante durante todo el análisis. La razón es que el algoritmo matemático utilizado funciona únicamente con este parámetro como valor de entrada. Pero parece paradójico que una aplicación de optimización que tenga como uso más directo la disminución de material en componentes mecánicos, reciba como parámetro inicial la cantidad de material deseada. Disponer únicamente de este tipo de optimización puede no resultar práctico en muchas ocasiones.

Por eso, en este apartado se desarrolla una aplicación que sea más útil a nivel práctico en el diseño de componentes mecánicos. Esta aplicación será capaz de

calcular la cantidad de material necesaria (fracción de volumen) y la distribución de ésta (optimización) para cumplir un criterio de fallo. El critero será que las tensiones máximas del componente no superaren una tensión última introducida por el usuario. Se tiene la posibilidad además de introducir un coeficiente de seguridad para que las tensiones máximas estén por debajo de la tensión última en un cierto porcentaje, de forma que la tensión máxima admisible en el componente sea la tensión última de fallo dividida entre el coeficiente de seguridad:

$$S_{max\,adm} = \frac{S_u}{C.S.} \tag{3.28}$$

Donde $S_{max\,adm}$ es la tensión máxima admisible, S_u es la tensión última de fallo y $C.S$ es el coeficiente de seguridad.

La aplicación funcionará de forma que la fracción de volumen (que pasará a ser variable de diseño en vez de parámetro inicial) se irá modificando de la siguiente forma:

- Si la máxima tensión en el componente es mayor a la tensión máxima admisible: se incrementa la fracción de volumen.
- Si la máxima tensión en el componente es menor a la tensión máxima admisible: se reduce la fracción de volumen.

3.2.1 Funcionamiento del algoritmo

La idea básica es crear un algoritmo que permita encontrar la fracción de volumen para la cual la tensión máxima encontrada en el componente sea igual a la tensión máxima admisible.

Para ello, se ha tenido que programar previamente una subrutina que sea capaz de encontrar la tensión máxima en un componente dado. Es decir, para una fracción de volumen dada se realiza todo el proceso de optimización (obteniendo así el componente final), se calculan las tensiones del componente y finalmente se localiza la tensión máxima en este.

La esencia del algoritmo será calcular para una fracción de volumen dada (y su correspondiente componente optimizado) la tensión máxima encontrada y en función de esto aumentar o disminuir la fracción de volumen según convenga. Este

proceso se repite de forma iterativa hasta encontrar la fracción de volumen buscada.

Hay que tener en cuenta que cada una de estas iteraciones corresponde a un análisis de optimización, con lo cual es muy importante construir un algoritmo que sea capaz de converger a la fracción de volumen deseada en las mínimas iteraciones posibles. El algoritmo que se propone se expone a continuación:

• Iteración 1

$$\begin{cases} si \quad \sigma_{max1} > S_{max\,adm} \Rightarrow & \text{Imposible: hay que disminuir carga o aumentar } S_{max\,adm} \\ si \quad \sigma_{max1} < S_{max\,adm} \Rightarrow & \text{Sigue el proceso} \end{cases}$$

• Iteración 2

Mediante un ajuste lineal $\Rightarrow$ $Volfrac_3$

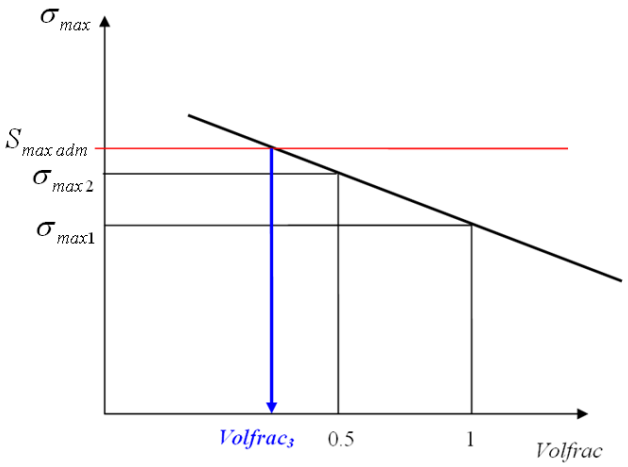


Figura 3.13 Representación gráfica del ajuste lineal para obtener $Volfrac_3$

• <u>Iteración i</u> ($i \geq 3$)

Mediante un ajuste cuadrático $\Rightarrow$ $Volfrac_{i+1}$

En la siguiente figura se muestra gráficamente el proceso seguido en la iteración 3 para cacular ***Volfrac₄***

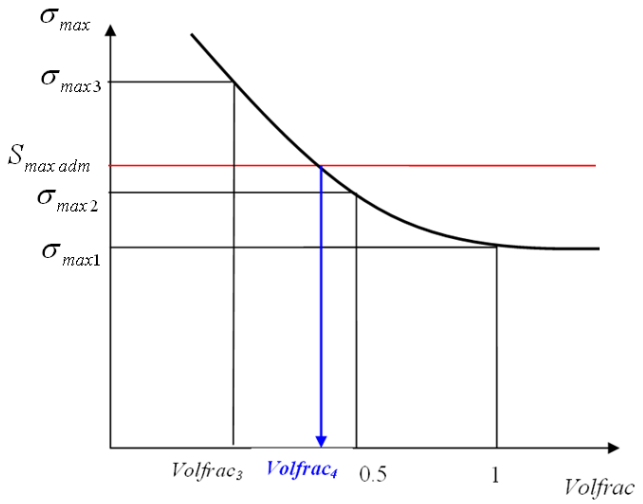


Figura 3.14 Representación gráfica del ajuste cuadrático para obtener $Volfrac_4$

De esta forma, el proceso va convergiendo a la fracción de volumen que cumple que:

$$\sigma_{max\ i} = S_{max\ adm} \tag{3.29}$$

El proceso finalizará cuando la diferencia entre la tensión máxima en el componente y la tensión máxima admisible esté por debajo de una cierta tolerancia

$$\frac{\sigma_{max} - S_{max\ adm}}{S_{max\ adm}} \times 100 < Tolerancia \tag{3.30}$$

La tolerancia tomará valores del orden del 1 o 2 %.

3.3 Dos técnicas de optimización

Con el algoritmo expuesto, se puede decir que el programa tiene ahora dos posibles vertientes que se explican a continuación.

Optimización A: Minimización de las tensiones

Este es el proceso de optimización que se ha venido tratando durante todo el trabajo. Es decir, para una cantidad de material determinada (fracción de volumen

fija) se minimiza la energía de deformación. O lo que es lo mismo, distribuir una cantidad fija de material de la forma más óptima posible para que las tensiones máximas en el componente sean lo más pequeñas posibles.

Optimización B: Minimización de la cantidad de material

En este proceso se hace uso de la aplicación presentada anteriormente. El dato de entrada es la tensión máxima admisible en el componente y el resultado es la pieza optimizada, es decir, la cantidad de material necesaria y su distribución (forma de la pieza).

Se puede ver claramente aquí la doble vertiente que presenta el programa. Bien se le puede dar como dato de entrada la cantidad de material y calcular la forma óptima para encontrar las mínimas tensiones posibles. O bien se le puede asignar como dato de entrada la tensión máxima admisible y calcular la cantidad de material necesaria. Hay que tener presente sin embargo, que el segundo tipo de optimización hace uso del primer tipo.

3.4 Verificación de la aplicación informática

Hasta el momento solo se ha explicado el proceso que se ha seguido para integrar el programa de optimización topológica original en el software de GFEM. Dado que la metodología utilizado por el GFEM no es una técnica convencional ni es por tanto la técnica utilizada incialmente en la optimización topológica se requiere de una exhausta verificación para comprobar que la aplicación funciona correctamente. Para ello se procede como sigue:

- En primer lugar se analiza un ejemplo diferente al de la viga biapoyada. Se estudiará concretamente el caso de un gancho, ya que permite utilizar una geometría relativamente compleja (no rectangular) y ofrece resultados interesantes.
- En segundo lugar se realiza un estudio de la influencia de los parámetros iniciales de diseño en la solución final. Se pretende por un lado verificar que los resultados obtenidos coinciden con los que se predicen en la bibliografía, y por otro lado entender mejor el comportamiento de la aplicación.

3.4.1 Verificación mediante un ejemplo de aplicación. El gancho

Para probar el buen funcionamiento de la integración del programa básico original en el software de GFEM, se ha analizado el caso de otro componente con una geometría no rectangular. Se ha elegido un gancho como ejemplo por el juego de formas que puede aportar la solución final. Este ejemplo será utilizado posteriormente, además de la viga biapoyada, para el estudio de la influencia de los parámetros iniciales en la geometría final.

La carga se describe como una presión aplicada en la base del gancho que es cero en los extremos y máxima en el punto medio.

Antes de presentar la geometría de trabajo empleada para el gancho en cuestión, es interesante comentar lo que ocurre cuando la geometría de trabajo es la siguiente:

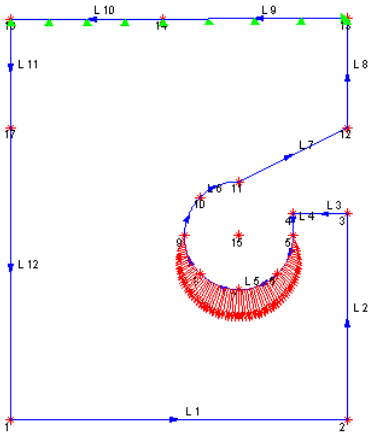


Figura 3.15 Geometría de trabajo A

Figura 3.16 Solución a la geometría

Tal y como se observa, el resultado es una estructura que tiene dos apoyos en la parte superior, a diferencia de un gancho convencional que suele tener uno. El caso es que el programa busca la opción más óptima sin ningún tipo de limites, creando otro apoyo exterior que lógicamente reduce la cantidad de material empleada (o disminuye el esfuerzo para una misma cantidad de material) ya que le da más inercia al componente. La falta de límites en la optimización topológica es una de las ventajas más importantes, pero tal y como se observa en este caso puede convertirse en un inconveniente. Por este motivo se tienen que añadir ciertas

limitaciones cuando las prestaciones requeridas deban cumplir alguna especificación.

En este caso en concreto, se necesita que el gancho tenga un solo apoyo. Para ello, se ha creado la siguiente geometría, que impide el apoyo exterior (Figura 3.17)

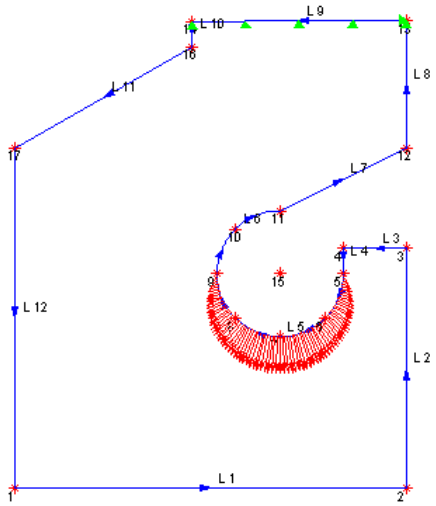


Figura 3.17 Geometría de trabajo

Con esta geometría y los siguientes parámetros iniciales, se ha obtenido el siguiente gancho (se muestran algunas de las iteraciones para alcanzar la geometría final)

Volfrac=0.3; *Penal*=3; r_{min} =1.5; *NM*=5 (9216 elementos)

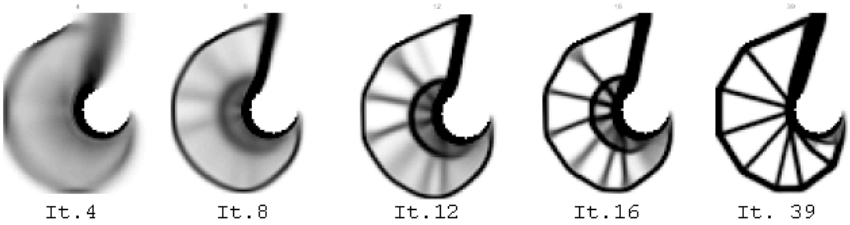
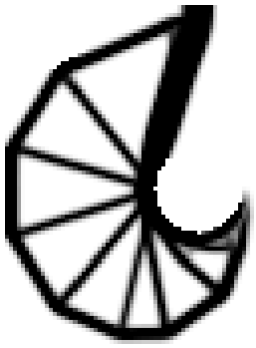


Figura 3.18 Evolución de la solución.



It. 79

Figura 3.19 Solución final

Esta es la geometría óptima según el optimizador con los parámetros iniciales anteriormente indicados. Como se verá en el siguiente apartado, según se varían los parámetros, esta geometría va a cambiar.

Como ya se comentó anteriormente, existe la posibilidad de representar la solución final con la geometría de trabajo. Recordar que aunque en las figuras anteriormente mostradas aparezcan elementos completos en la intersección de la malla con la geometría, realmente solo está trabajando la parte del elemento que está dentro de la geometría, y que se estaba dibujando de esta forma para agilizar la representación. Se muestra a continuación la solución anterior con la geometría de trabajo. La diferencia en este caso reside únicamente en la representación del arco donde se aplica la carga, que ahora aparece correctamente recortado por la geometría de trabajo.

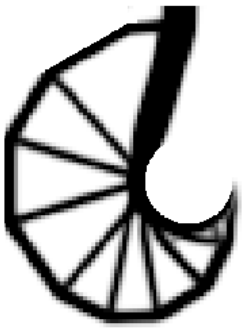


Figura 3.20 Soluicón Figura 3.19 dibujada con la geometría de trabajo

3.4.2 Verificación del comportamiento del programa. Efecto de los distintos parámetros.

A continuación se realiza un estudio exhaustivo de la influencia que tienen los parámetros iniciales de diseño en la solución final. Se pretende concluir que el comportamiento de la aplicación es coherente con la teoría de la bibliografía y además entender más a fondo el comportamiento del programa para poder interpretar mejor los resultados, obtener conclusiones y determinar posibles líneas futuras de investigación.

Hay siete parámetros que influyen en la solución final (geometría óptima) que son:

- Geometría de trabajo.
- Tamaño de elemento.
- Forma de los elementos.
- Parámetro Penal (penalización)
- Parámetro Volfrac (fracción de volumen)
- Parámetro rmin (radio de influencia para el filtro)
- Error de cálculo en los desplazamientos mediante el FEM.

La forma de los elementos influye porque la geometría a representar está formada por pequeños elementos con dicha forma. En esta tesis de máster se utiliza por simplicidad elementos cuadriláteros ya que el uso de otras formas queda fuera del ámbito de estudio en este trabajo.

Por otra parte, el error de cálculo en los desplazamientos influye lógicamente en el campo de desplazamientos. Un campo de desplazamientos diferente da lugar a una distribución de pesos diferente. Para medir cómo de influyente puede ser este factor, más adelante se hará un estudio empleando elementos cuadriláteros lineales y cuadráticos, ya que introducen un error en el cálculo de desplazamientos diferente. Para todos los estudios que se presentan a continuación se utilizan elementos cuadriláteros lineales.

3.4.2.1 Geometría de trabajo

Este factor es evidentemente muy influyente, ya que la geometría final obtenida tiene que estar dentro de la geometría de trabajo. Cuando se propone este factor como parámetro influyente y de interés relevante es porque en ciertos casos,

pequeñas variaciones en la geometría de trabajo pueden afectar de forma importante a la geometría final. Esto no ocurre siempre, pero un buen ejemplo de ello es el gancho mostrado anteriormente.

Para los dos siguientes análisis se va a utilizar:

Volfrac=0.3; *Penal*=3; r_{min} =1.5; *NM*=4;

La geometría a utilizar será la mostrada anteriormente (Figura 3.17), solo que para cada caso las líneas L1 y L12 van a tomar coordenadas diferentes:

Caso1: L1→ Y=-13 L12→ X=-17 punto1= (-17,-13) (menos espacio de trabajo)

Caso2: L1→ Y=-17 L12→ X=-21 punto1= (-21,-17) (más espacio de trabajo)

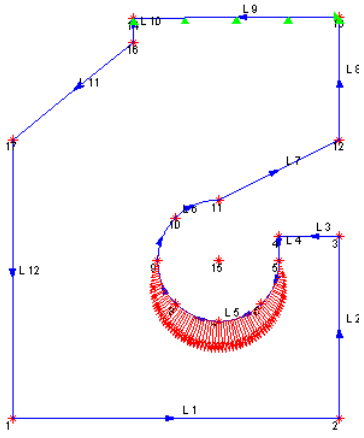


Figura 3.21 Caso1

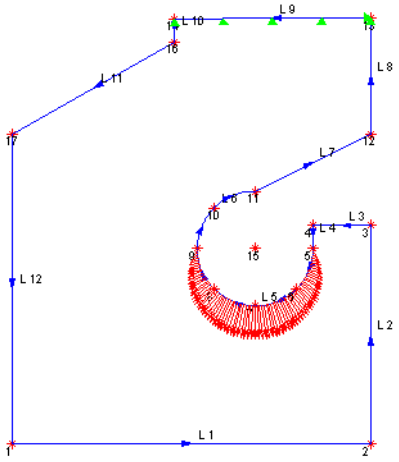


Figura 3.22 Caso2

Tal y como se aprecia, la diferencia entre ambas geometrías es poca. Pero el resultado difiere bastante:

Figura 3.23 Solución Caso1

Figura 3.24 Solución Caso2

En el caso 1, el optimizador ha optado por crear un arco exterior que le da inercia al componente. Este arco ha llegado justo ha tocar el lado L12 por la izquierda y el L1 por abajo. En cambio, cuando se ha aumentado la geometría de trabajo para dejar crecer ese arco (que parecía no tener bastante espacio), el gancho se ha "encogido" desapareciendo el arco exterior. En la siguiente figura se muestra la evolución de la forma en el caso 2:

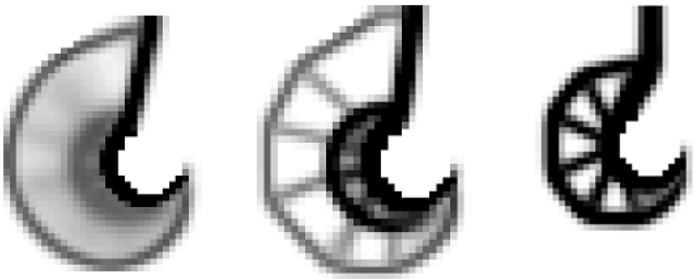


Figura 3.25 Evolución solución Caso2

Se observa que inicialmente el optimizador intenta crear dicho arco, pero posteriormente este desaparece. El optimizador propone como mejor solución una figura con arco exterior mientras la geometría lo permita, tan exterior como sea posible. Evidentemente, cuanto más exterior sea el arco más largo es y por tanto más delgado debe ser su espesor para un mismo volumen. Pero resulta que el espesor de una barra que puede ser representado viene limitado por el tamaño de elemento: cuanto más pequeño sea el tamaño de elemento las barras a representar podrán ser más delgadas. De este modo, el espesor del arco exterior en el caso 2 es

demasiado pequeño para poder ser representado con el tamaño de elemento dado, y este acaba desapareciendo. Sin embargo, en el caso1 el espesor es más grande y sí puede ser representado, quedando por tanto el arco hasta la forma final.

Esta situación no suele ser habitual, y aquí se ha representado el caso límite para mostrar qué podría ocurrir en casos muy concretos. Si bien generalmente la geometría final no va a variar tanto, sí se llega a la conclusión de que ciertas barras en la estructura aparecerán o no dependiendo del tamaño de elemento empleado. Esto se demuestra con más claridad en la siguiente sección donde se analiza el efecto del tamaño de elemento utilizado.

3.4.2.2 Tamaño de elemento

A continuación se compara el caso2 de la anterior sección, con el caso3 que utiliza exactamente la misma geometría de trabajo pero un nivel de malla más fino. Es decir, el tamaño de elemento se va a dividir por cuatro (área).

Volfrac=0.3; *Penal*=3; r_{min} =1.5

Figura 3.26 Caso 2 (*NM*=4)

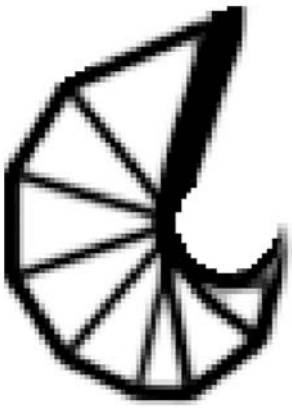


Figura 3.27 Caso 3 (*NM*=5)

Cómo era de esperar, la barra del arco exterior que el optimizador fue incapaz de dibujar en el caso2, sí se ha podido dibujar en el caso3 porque el tamaño de elemento al ser más pequeño ha permitido dibujar un espesor de la barra lo suficientemente delgado.

En el caso del gancho, la aparición del arco exterior implica un cambio importante en la geometría final. A continuación se muestra el caso de la viga biapoyada variando el nivel de malla.

Volfrac=0.5; *Penal*=3; r_{min} =1.5

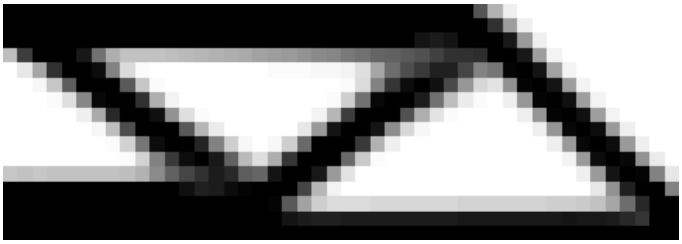


Figura 3.28 Caso1. NM=3

Figura 3.29 Caso2. NM=4

Figura 3.30 Caso3. NM=5

En este caso, si bien la estructura principal de la solución se mantiene, se observa que la aparición de barras más delgadas ocurre a medida que disminuye el tamaño de elemento.

En definitiva, se concluye que el tamaño de elemento va a influir mucho en la solución final. Con elementos más pequeños se obtendrán geometrías más complejas, formadas generalmente por un mayor número de barras, ya que estas podrán ser más delgadas. Se dice por tanto que la solución es dependiente del nivel de mallado. Sabiendo por tanto que existe una relación importante entre el tamaño de elemento y el espesor mínimo de las barras, dicha propiedad puede ser utilizada

para controlar justamente ese espesor mínimo o dicho de otra manera, controlar la complejidad de la geometría a obtener.

Pero esta característica puede presentar algún inconveniente si lo que se requiere es una geometría más bien simple (barras de mayor sección), ya que sólo se podría obtener con mallas más bastas, y por tanto con resultados poco afinados. Por ejemplo, si se desea obtener una geometría como la de la viga biapoyada del caso 1 (arriba), solo se podría conseguir con una malla de nivel 3 (en este caso concreto). Esto da lugar a una definición muy basta y con zonas donde no queda clara la geometría, por ejemplo, en la unión entre barras que se muestra en la Figura 3.31.

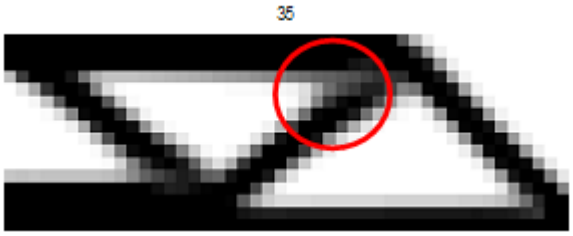


Figura 3.31 Zona de poca definición

Una solución a este problema sería utilizar adaptatividad de la malla. Es decir, para conseguir una geometría con menos barras (como la del caso 1) y con una buena definición, se podría crear inicialmente la geometría con la malla más basta y posteriormente aplicar adaptatividad de la malla en aquellas regiones donde se requiriese una mayor definición. Básicamente el criterio de adaptatividad sería el de subdividir elementos en aquellas zonas donde existe una variación importante de la variable de diseño (pesos).

Por otra parte, en la bibliografía se propone otra solución. Se trata de utilizar el parámetro r_{min} para conseguir que la solución sea independiente del nivel de malla. El parámetro $r_{\min}$ es el encargado de conseguir una solución más alisada, de forma que si el r_{min} aumenta se consigue que la derivada de la energía de deformación dependa más del valor de ésta en sus vecinos. Aumentando el r_{min} a medida que se aumenta el nivel de malla se puede conseguir que la solución sea la misa independientemente del nivel de mallado. Esta parte se explicará de forma más exhaustiva posteriormente, una vez quede explicado el funcionamiento del parámetro r_{min}.

3.4.2.3 Parámetro Penal (penalización)

Tal y como se explico en los antecedentes referentes a la optimización topológica, para conseguir que la solución esté bien definida por elementos que salen (valor peso 0) y entran (valor peso 1), se introduce un parámetro de penalización para reducir la eficiencia de los elementos con densidades (pesos) intermedios. Esto se consigue con una ley potencial de la siguiente manera:

$$\mathbf{K}'_i(x_i) = x_i^{\,p}\mathbf{K}_i \tag{3.31}$$

Las matrices $\mathbf{K}'_i$ y $\mathbf{K}_i$ representan la matriz penalizada y la matriz real del elemento *i*, respectivamente. La potencia *p* es el factor de penalización que siempre debe ser mayor que uno. De esta forma, si se tiene un factor de penalización alto, los pesos con valores intermedios (entre 0 y 1) se elevan a potencias altas y por tanto sus matrices de rigidez se ven altamente afectadas. Los elementos con pesos de valor 1 en cambio, aunque estén elevados a potencias altas, el valor que multiplica a la matriz de rigidez siguen siendo 1. Este método recibe el nombre de "Density Method" o "Single Isotropic Material with Penalty (SIMP) Method".

En todos los ejemplos anteriores se ha utilizado como penalización un valor de 3, el recomendado en la bibliografía [1] ya que suele dar soluciones más razonables. En esta sección se hace un estudio del efecto que produce variar este parámetro. Para ello se ha analizado nuevamente el caso del gancho dejando los otros parámetros fijos y variando la penalización con los valores 5 (Caso1) 3 (Caso2) y 1.5(Caso3). Se muestra además las evoluciones de estos.

Volfrac=0.3; r_{min} =1.5; *NM*=5;

Geometría de trabajo: la misma que la del Caso2 en la sección "Geometría de trabajo".

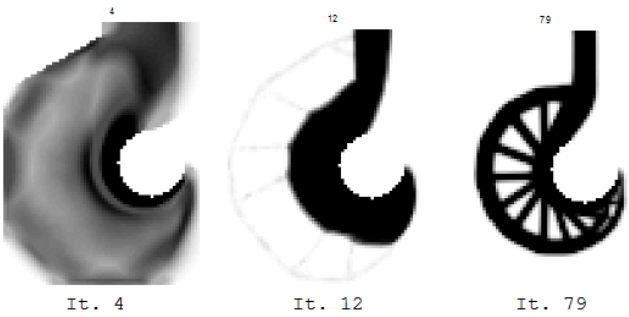


Figura 3.32 Evolución Caso1 (Penal=5)

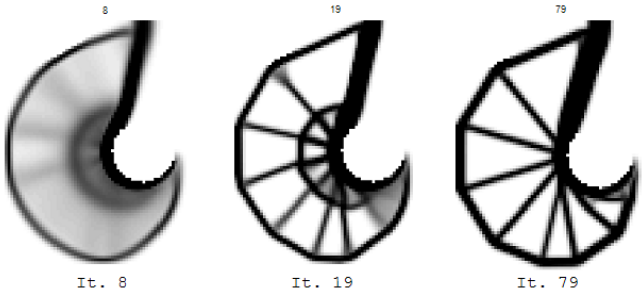


Figura 3.33 Caso2 (Penal=3, estándar)

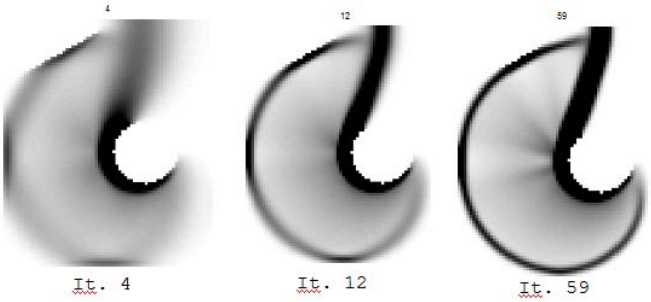


Figura 3.34 Caso3 (Penal=1.5)

Como se puede observar, en el caso1, donde se ha aumentado el penal a 5, se produce de nuevo el fenómeno de la desaparición del arco exterior. En esta ocasión el motivo es diferente. Los pesos intermedios contribuyen muy poco a la rigidez del componente al estar usando un factor de penalización alto. El arco exterior requiere de un número elevado de pesos intermedios para ser representado y por tanto el optimizador opta por compactar más el material (elementos casi exclusivamente con pesos 0 ó 1) quedando la figura tal y como se muestra. Cabe decir que se está trabajando en el caso límite y con una pequeña variación de algún parámetro la geometría cambia radicalmente al aparecer o desaparecer dicho arco.

En cambio, en el caso3, donde se ha disminuido el penal de 3 a 1.5, los elementos intermedios contribuyen más a la rigidez y la geometría final resulta poco compacta, con muchos elementos grises. Aunque pueda parecer que los resultados obtenidos con una penalización tan baja no tengan ningún sentido, tal vez podrían servir en ciertas ocasiones. Por ejemplo, el resultado del caso3 se podría interpretar como dos barras (interior y exterior) unidas por una especie de membrana de material con menor espesor, pudiendo ser un resultado válido igualmente.

El caso2 es una solución intermedia que permite dibujar el arco exterior sin difuminar tanto el material como en el caso3. Normalmente se utiliza esta configuración (penal=3) ya que suele dar mejores resultados.

En todo caso, el factor de penalización da a la aplicación más versatilidad en el sentido de que el usuario puede escoger, por decirlo de alguna forma, el nivel de compactación del material.

3.4.2.4 Fracción de volumen

La fracción de volumen (o parámetro *Volfrac*) es la proporción del área de trabajo que va a disponer de material y toma valores entre 0 y 1. A continuación se muestra una variación de este factor con los ejemplos del gancho y la viga biapoyada.

Gancho

r_{min} =1.5; *Penal*=3; *NM*=5;

Geometría de trabajo: la misma que la del Caso2 en la sección "Geometría de trabajo".

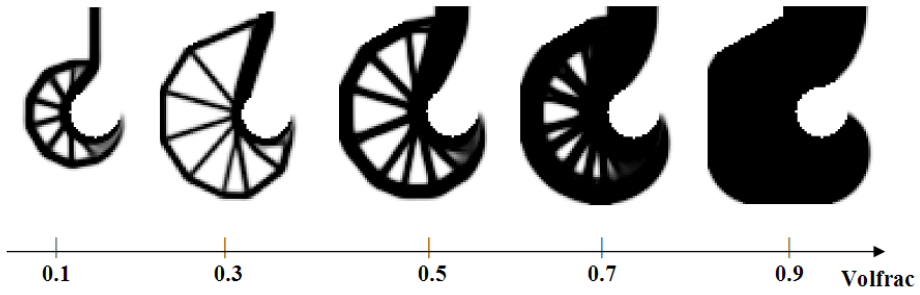


Figura 3.35 Variación de Volfrac. Gancho.

Viga biapoyada

r_{min} =1.5; *Penal*=3; *NM*=4;

Geometría de trabajo: la misma que en los casos anteriores.

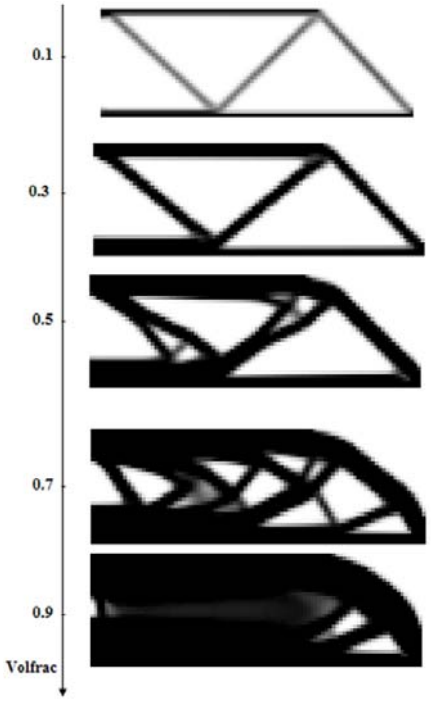


Figura 3.36 Variación de Volfrac. Viga biapoyada.

Se observa claramente que la cantidad de material aumenta según se incrementa la fracción de volumen. Además, la forma de la geometría puede incluso variar un poco, como es el caso de aumentar el *Volfrac* de 0.1 a 0.3 en el gancho, o el caso de aumentar de 0.3 a 0.5 en la viga biapoyada, donde las barras se bifurcan.

Es obvio, que para mayores fracciones de volumen, la geometría final es más robusta, y va a soportar mayores cargas.

3.4.2.5 Parámetro r_{min}

El parámetro r_{min} es el radio de influencia (respecto al tamaño de elemento) que se toma en el filtro para seleccionar aquellos elementos vecinos de cada elemento que influirán en el valor de la derivada de la energía de deformación de este.

Como ya se explicó en secciones anteriores, el filtro es una especie de alisado de la variable de diseño para dar lugar a soluciones más razonables (menos

pixeladas). El valor r_{min} recomendado en la bibliografía es 1.5 [1]. A continuación se muestra la variación de r_{min} en el caso del gancho.

Penal=3; *NM*=5; *Volfrac*=0.3

Geometría de trabajo: la misma que la del Caso2 en la sección "Geometría de trabajo".

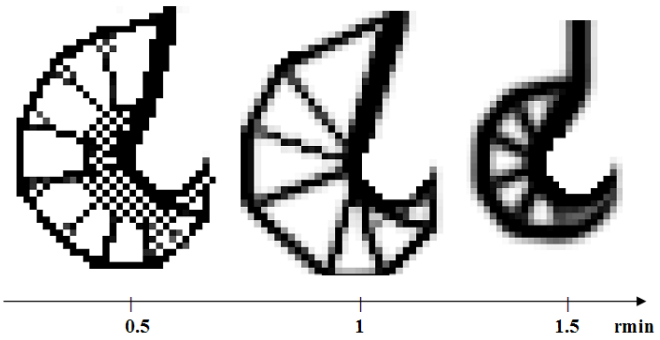


Figura 3.37 Variación r_{min} Gancho.

En el caso de r_{min} 0.5 se tienen en cuenta únicamente los 4 elementos vecinos más próximos (arriba, abajo, derecha e izquierda), dando lugar a una figura muy pixelada (efecto *checkerboard* [1]), donde el ancho de las barras puede estar formado por un menor número de elementos. Ese es el motivo de que bajando el r_{min} el arco exterior sí aparezca, a diferencia del caso estándar que es con r_{min} 1.5. Cabe recordar que esto ocurre porque de nuevo se está trabajando justo en los valores de r_{min} que provocan dichos cambios. Puede observarse que el espesor de la barra exterior es más o menos de 1 elemento. El motivo es que los elementos de pesos intermedios no aparecen, ya que al no estar "alisado" el material la transición de la zona de material a la de no material puede ser directa, es decir se pasa de elementos de peso 1 a elementos de peso 0.

En el caso de r_{min} igual a 1 las barras aumentan su espesor, pero la figura sigue estando muy pixelada. Es por este motivo que generalmente se usa r_{min} con valor 1.5, ya que salen figuras más razonables. Sí lo que se requiere es que puedan aparecer barras de menor espesor, es preferible subir el nivel de malla o incluso bajar un poco el nivel de penalización antes que disminuir el r_{min}.

3.4.2.6 *Tamaño de elemento VS Parámetro Penal*

Se ha demostrado que el tamaño de elemento y la variación del factor de penalización influyen de forma importante en la geometría óptima final. Por eso parece interesante elaborar una especie de matriz variando ambos parámetros. Se modifica el nivel de malla de 4 a 6 (mayor nivel de malla, menor tamaño de elemento) y el parámetro *Penal* toma los valores 2, 3 y 5.

Volfrac=0.3; r_{min} =1.5;

Geometría de trabajo: la misma que la del Caso2 en la sección "Geometría de trabajo".

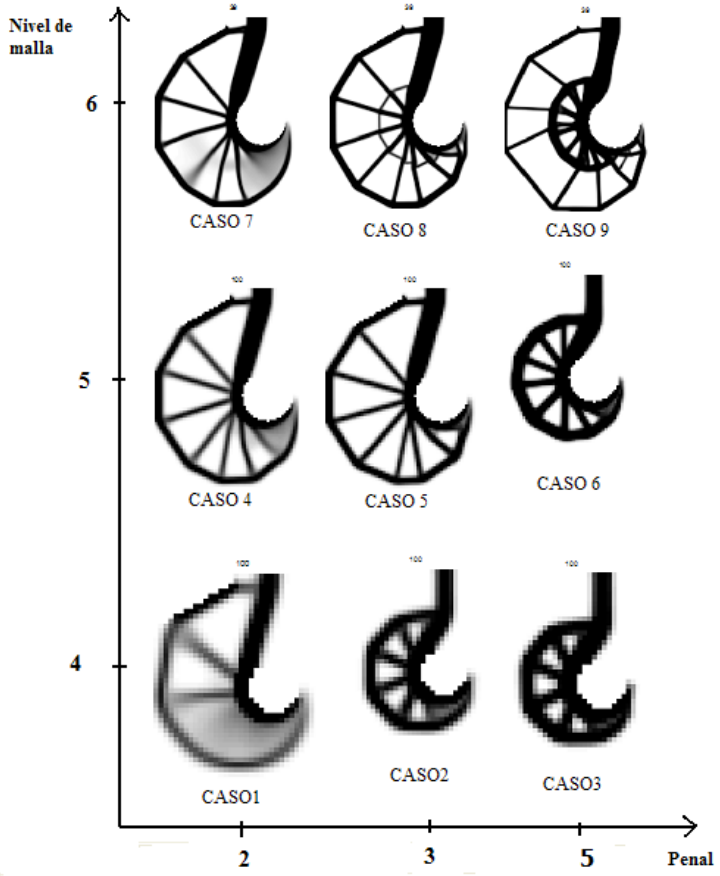


Figura 3.38 Matriz Penal VS Nivel de malla

En primer lugar, parece evidente que el uso de factores de penalización bajos dará lugar a soluciones menos coherentes en el sentido de que el material queda muy disperso (casos 1, 4 y 7) con zonas muy difuminadas aún utilizando tamaños de elemento muy pequeños.

En segundo lugar, se observa como ya se dijo, que utilizar un factor de penalización mayor implica una compactación del material. Por ejemplo, en la transición del caso 8 al 9 la estructura principal del componente se mantiene, pero al aumentar penal de 3 a 5 el arco exterior disminuye de espesor, mientras que el arco interior aumenta. Este cambio responde a dicha compactación.

Además, queda probado que reducir el tamaño de elemento implica una disminución del espesor mínimo de barras a poder representar. Observar por ejemplo lo que ocurre al aumentar el nivel de malla de 5 a 6 con un penal de 3 (caso 5 y 8): la idea principal de la geometría es la misma, pero las barras del arco interior que tienen un espesor pequeño solo pueden aparecer si el tamaño de elemento es menor (NM=6). Ocurre algo parecido en la transición del caso 6 al 9, donde el arco exterior en este caso es de espesor pequeño y solo aparece con menor tamaño de elemento.

El hecho de poder representar barras más delgadas da lugar a geometrías más complejas (observar caso 8 y 9) que en todo caso tendrán una manufacturabilidad más complicada. La aparición de agujeros por propia decisión de la aplicación es uno de los puntos fuertes de la optimización topológica. Pero la fabricación de piezas que se obtienen con el uso de mallas muy finas resulta todavía hoy en día un "handicap".

3.4.2.7 Error de cálculo en los desplazamientos.

Para poder calcular las derivadas de la energía de deformación hace falta calcular previamente los desplazamientos en nodos para una determinada distribución de los pesos (inicialmente se le asigna a todos los elementos peso igual a la fracción de volumen). Evidentemente, si el campo de desplazamientos varía, las derivadas de la energía de deformación también lo harán y consecuentemente la nueva distribución de los pesos, quedando una geometría final distinta.

Ahora bien, aún incrementando un poco el error en cálculo de los desplazamientos, no cabe esperar que la solución final cambie mucho, ya que si bien los valores de los desplazamientos puedan variar, el comportamiento del componente (deformaciones) será muy parecido, y por tanto no se prevé que cambie mucho la solución final. Para probar esta hipótesis, se procede a analizar el

caso del gancho para elementos cuadriláteros lineales y cuadráticos, ya que el error cometido mediante el MEF es diferente (menor con elementos cuadráticos).

Volfrac=0.3; *Penal*=3; r_{min} =1.5; *NM*=4;

Geometría de trabajo: la misma que la del Caso2 en la sección "Geometría de trabajo".

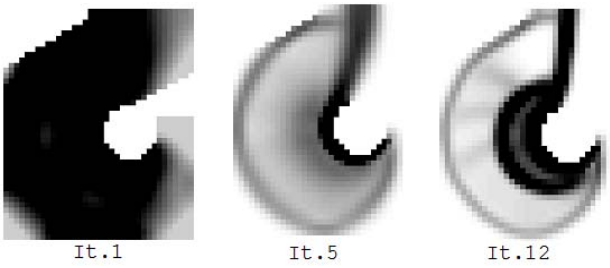


Figura 3.39 Elementos lineales

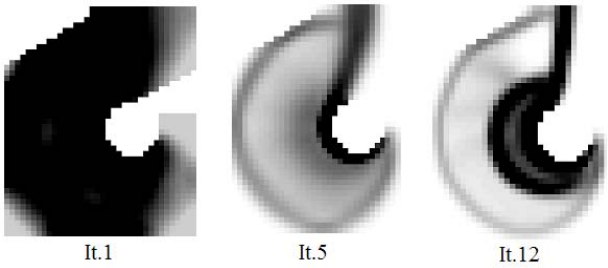


Figura 3.40 Elementos cuadráticos

Como se puede observar, cada una de las iteraciones del proceso en ambos casos son prácticamente iguales. De hecho a simple vista es casi imposible encontrar ninguna diferencia. Queda probado por tanto que el método que se está utilizando es poco sensible a pequeñas variaciones del error de cálculo en los desplazamientos.

El estudio de parámetros realizado ha sido contrastado con la bibliografia y los resultados obtenidos son completamente coherentes con esta. De modo que se concluye que la aplicación de optimización topológica queda completamente integrada al software de GFEM y que su funcionamiento es correcto.

3.4.3 Independencia de la solución respecto el tamaño de elemento

Tal y como se mostró en la anterior sección (apartado de tamaño de elemento), el nivel de refinamiento de la malla influye de forma drástica en la solución obtenida. La teoría establece que para una misma fracción de volumen, un mayor número de agujeros en la estructura mejoran el comportamiento de la pieza. El caso es que tal y como se va incrementando el nivel de malla, las barras a representar pueden ser más delgadas y por tanto pueden aparecer más agujeros. En el caso extremo la solución sería una rejilla formada por barras muy pequeñas y numerosos agujeros. Este comportamiento se ve claramente en el caso de la viga biapoyada (Figura 3.41)

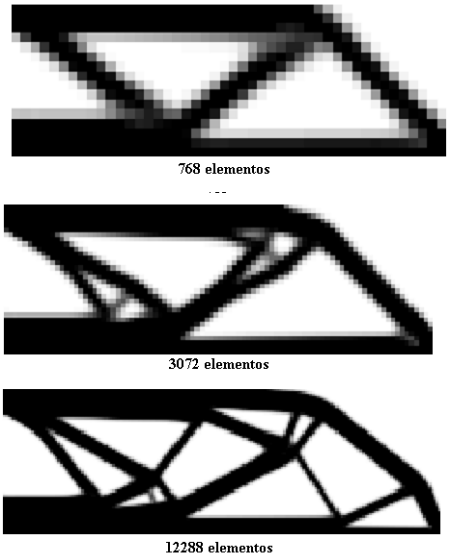


Figura 3.41 a) b) y c) Evolución de la viga biapoyada según se incrementa el nivel de refinamiento

Se dice entonces que la solución depende del tamaño de elemento. El inconveniente de este comportamiento es que a priori no se puede conseguir una solución como la de la Figura 3.41a) con una malla como la de la Figura 3.41c), y por tanto la solución de la Figura 3.41a) siempre tiene una baja resolución. Esto se puede solucionar a través del parámetro r_{min} (tal y como se comentó más arriba), según se explica a continuación.

El parámetro r_{min} representa el radio que define el área de influencia (círculo) centrada en un cierto elemento para el filtro (alisado) de las derivada de energía de deformación. Los elementos que se tienen en cuenta en el alisado de dicho elemento son los que quedan dentro del círculo formado. El r_{min} es por tanto una distancia. Hasta el momento la unidad de medida de esta distancia ha sido la longitud del lado de elemento, de modo que si $r_{min} = 3$ el círculo formado mide *6 x long. lado de elemento* de diámetro. Otra forma de medir r_{min} es cómo distancia absoluta. Supongamos por ejemplo que la altura de la estructura mide 50 metros, y la luz 200. Si r_{min} valiese 7 *m* se tomarían como elementos los que quedaran dentro de un círculo de diámetro 14 *m* centrado en el elemento, independientemente del tamaño del elemento.

Una vez definido el r_{min} como valor absoluto se trata únicamente de mantener esa distancia de influencia constante en los diferentes niveles de malla para obtener así la misma solución independientemente del tamaño de elemento utilizado. Es decir, el radio de cobertura debe ser el mismo (en valor absoluto) independientemente del refinamiento. Evidentemente, si el nivel de malla es mayor (menor tamaño de elemento), se tendrán en cuenta más elementos en el vecindario del elemento a alisar, tantos como quepan dentro del radio de cobertura.

Se muestra a continuación el efecto de esta medida en el caso de la viga biapoyada. Las medidas de la geometría de trabajo son 60 de ancho por 20 de alto (unidades de longitud). Se va a tomar $r_{min} = 1.97$ unidades de longitud (que sería equivalente a tomar $r_{min} = 1.5$ (elementos) con un de nivel de malla 3 para el caso de la viga biapoyada), un 9.85 % de la altura. La solución obtenida se muestra a continuación:

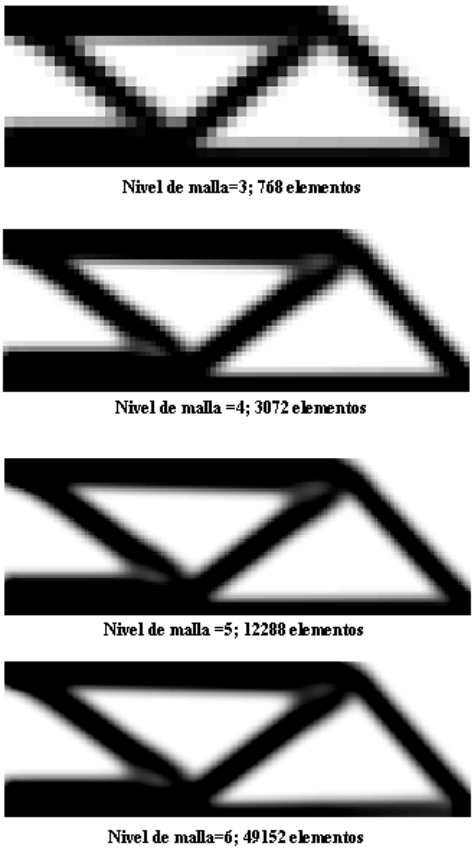


Figura 3.42 Evolución de la viga biapoyada según se incrementa el nivel de refinamiento y manteniendo el valor r_{min} constante en valor absoluto

Se observa que la solución obtenida es exactamente la misma en todos los casos. En esta ocasión, a medida que se disminuye el tamaño de elemento no aparecen barras más delgadas, sino que aparece la misma figura pero menos pixelada.

En el caso del gancho, se observaba anteriormente que para diferentes tamaños de elemento, la solución final variaba substancialmente. Se muestra aquí de nuevo la figura por comodidad (Figura 3.43).

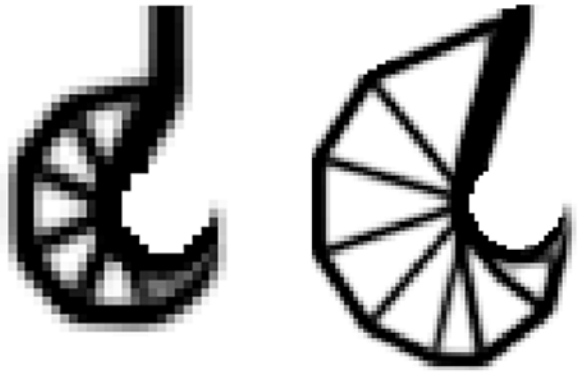


Figura 3.43 Evolución del gancho según se incrementa el nivel de refinamiento

De nuevo, si ahora se mantiene el r_{min} constante (en valor absoluto) se espera que la solución en mallas más finas sea la misma (no aparezca el arco exterior). Para probarlo se toma r_{min} = 1.325 (unidades de longitud) que equivalen al anterior r_{min} = 1.5 para el caso de NM=4. Se mantendrá el r_{min} constante para niveles de malla 5 y 6 (4 y 16 veces más elementos respectivamente). El resultado es el siguiente:

Figura 3.44 Evolución de la viga biapoyada según se incrementa el nivel de refinamiento y manteniendo el valor r_{min} constante en valor absoluto

Nuevamente se prueba que la medida de mantener constante el radio de cobertura que se utiliza en el filtrado de las derivadas de la energía de deformación (parámetro r_{min}) es un método eficaz para conseguir la independencia de la solución respecto el tamaño de elemento.

El programa constará por tanto de las dos posiblidades:

1) r_{min} distancia absoluta: las unidades del r_{min} serán las unidades de longitud utilizadas en el problema.

2) r_{min} distancia normalizada: las unidades del r_{min} serán la longitud del lado de elemento.

Estas opciones se implentan en la interfaz gráfica tal y como se muestra en la siguiente figura:

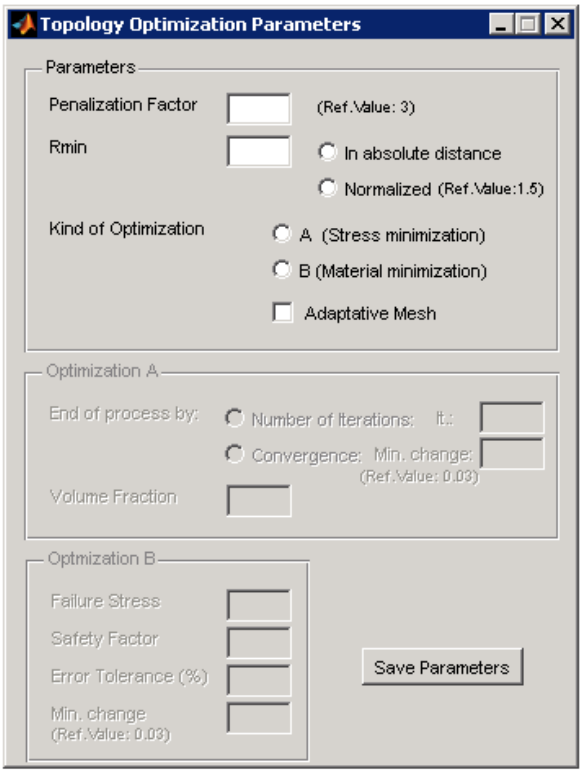


Figura 3.45 Interfaz gráfica

Cabe decir, sin embargo, que este método presenta algunos inconvenientes:

- En primer lugar, parece ser que refinando la malla se consigan soluciones más definidas. Pero cabe decir que las zonas intermedias de paso de material a no material continuan siendo las mismas. Obsérvese por ejemplo, en el caso del gancho (Figura 3.46)), las zonas marcadas con un círculo rojo. Es evidente que aumentando el nivel de refinamiento no se ha determinado de forma más precisa la transición de material a no material. Y eso se debe justamente al hecho de que el radio $r_{\min}$ utilizado para el caso de *NM*=6 supone tomar muchos elementos (aproximadamente tres veces más que con *NM*=5).

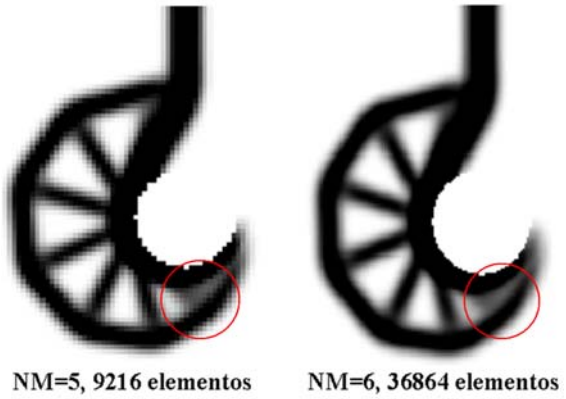


Figura 3.46 Detalles zonas intermedia de Fig. 47

- En segundo lugar, si bien el método parece eficaz, no es tan evidente que sea eficiente. Para obtener una alta definición con esta medida se precisa refinar toda la malla con el coste computacional que esto lleva asociado.

Además, la independencia de la solución respecto el nivel de refinamiento se tiene únicamente si dicha solución es representable con el nivel de malla más basto. Es decir, una solución que se obtiene con un nivel de malla basto siempre será representable en un nivel de malla más fino, pero no necesariamente al contrario. Téngase en cuenta que dado un nivel de malla el espesor de las barras que se puede representar tiene un mínimo.

Los inconvenientes descritos anteriormente serían solucionados si se utilizara (como ya se comentó anteriormente) el método de *h*-adaptatividad de la malla. Se trataría de incrementar el nivel de refinamiento únicamente en zonas de transición

de material. De esta forma, sí se definiría mejor esta transición y además se incrementaría la eficiencia de la aplicación al disminuir substancialmente el número de elementos empleados. Este tema se trata en la sección 4 *Programa de Optimización Topológica en GFEM con adaptatividad.*

3.4.4 Ejemplos de aplicación

Una vez integrado el programa de optimización original en el software de GFEM, con todas las modificaciones presentadas anteriormente y con el estudio de parámetros realizado, se procede a utilizar el software desarrollado en otros ejemplos a fin de comprobar su robustez.

Cabe recordar aquí que para cada caso, las soluciones que ofrece el programa variarían según se cambiasen los parámetros de diseño, tal y como se mostró en el estudio de parámetros (3.4.2). No tendría sentido presentar de nuevo la gama de soluciones que se pueden obtener para cada caso ya que el objetivo de esta sección no es otro que mostrar el uso que se puede hacer del programa creado en esta tesis de máster. Por ello, para cada caso se presentan las soluciones más representativas a modo de ejemplo.

Tal y como se explicó en el apartado 3.3, existen dos tipos de optimización según se desee optimizar la forma del componente para una cantidad de material dada (para minimizar las tensiones), o bien encontrar la cantidad de material necesaria y su distribución para una tensión máxima admisible. Para cada uno de los ejemplos que se muestran a continuación se va a calcular:

- Optimización A. (Minimización de las tensiones). Se obtendrá la forma óptima del componente para una fracción de volumen determinada.
- Cálculo de tensiones. Se calculará y representará la distribución de tensiones para demostrar que las modificaciones que se explicaron en el apartado 1.5.2.6 funcionan correctamente. Los análisis de tensiones son necesarios además para el tipo de optimización B.
- Optimización B (minimización de la cantidad de material). Se calcula la cantidad de material necesaria (fracción de volumen) y su distribución para no superar una tensión última de fallo (que es dato de entrada)

Para la representación de las tensiones, se ha optado, a modo de ejemplo, por calcular y representar la tensión de Von-Misses ya que en ingeniería estructural se usa esta tensión para determinar fallo en materiales dúctiles.

Para trabajar con el algoritmo que consigue el tipo de optimización B, en todos los casos se ha utilizado un módulo de Young de 2.1 $x10^{11}$, un coeficiente de Poisson 0.3 y una tolerancia del 2 % para aproximar la tensión máxima encontrada en el componente a la tensión máxima admisible.

3.4.4.1 Conducto sometido a presión interna

El problema de optimización que se ha resuelto corresponde a la optimización de la sección de un conducto sometido a una presión interna. En el modelo de elementos finitos se han empleado dos planos perpendiculares de simetría, de este modo, sólo es necesario modelar una cuarta parte de la sección.

La carga aplicada es una presión en la circunferencia interior y en dirección normal a esta, de valor $0.9x10^6$ Pa.

La geometría de trabajo utilizada ha sido la mostrada a continuación.

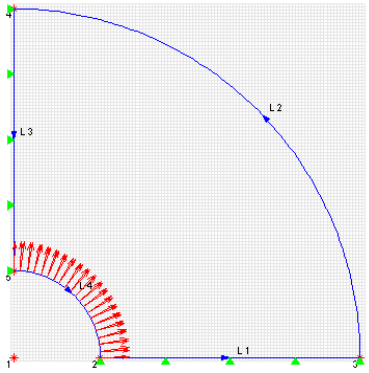


Figura 3.47 Geometría de trabajo del conducto sometido a presión interna

<u>Optimización A: Minimización de las tensiones</u>

La forma óptima de un conducto es aquella cuyos contornos interno y externo son arcos de circunferencia.

Para analizar este caso se ha utilizado:

Volfrac=0.5; *Penal*=3; r_{min} =1.5 (elementos); *NM*=6 (11092 elementos activos)

A continuación se representa el proceso de evolución y la solución final obtenida:

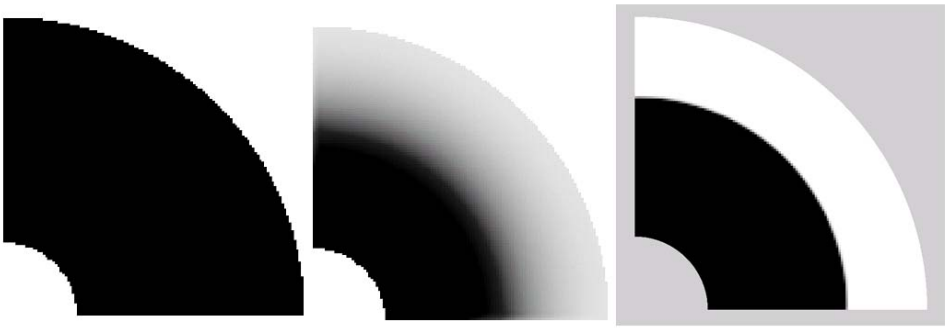


Figura 3.48 Evolución y solución final del conducto sometido a presión interna. (optimización A)

Es evidente que el programa ha calculado la forma óptima, ya que esta coincide exactamente con la forma que sostiene la teoría. Si se modifica la fracción de volumen (*Volfrac*) se obtiene obviamente la misma forma pero con mayor o menor espesor. Es interesante observar además la ausencia de agujeros en esta figura. Aunque el programa opte generalmente por formar una geometría formada por barras y agujeros, esta vez no lo ha hecho por la simple razón de que la forma más óptima es la que se ha obtenido. Esto pone de manifiesto que el programa no está diseñado exclusivamente para formar barras y agujeros, si no que se rige por un proceso de optimización general sin ningún tipo de restricción.

Para este caso en concreto, parece irrelevante el nivel de refinamiento adoptado, ya que al tratarse de una geometría sencilla y no disponer de agujeros, la solución final obtenida es la misma.

La primera imagen mostrada en la Figura 3.48 corresponde a la inicialización de los pesos, es decir, todos los pesos toman el valor de la fracción de volumen (en este caso 0.5). El hecho de que se represente la figura con todos los elementos en negro (valor que corresponde habitualmente al peso de valor 1) es porque la gama de colores funciona de forma que el blanco se asocia al mínimo peso (0.001) y el negro al máximo peso en la figura. Como inicialmente todos los valores son 0.5, este corresponde al máximo y consecuentemente la figura sale completamente negra.

Cálculo de tensiones

El conducto se ha modelado en un estado en deformación plana, es decir, se ha analizado una sección lejos de los extremos de un conducto largo de forma que no aparezcan efectos del borde del conducto relacionados a un estado de tensión plana. La solución se muestra a continuación.

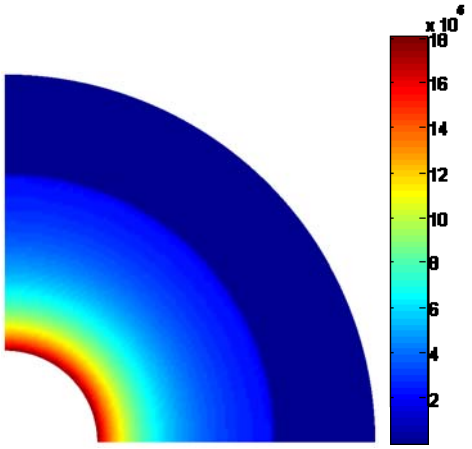


Figura 3.49 Distribución de tensiones de Von Misses. Conducto sometido a presión interna

La máxima tensión se da en la cara interna del conducto y tiene un valor de unos 1,8 MPa.

Optimización B: Minimización de la cantidad de material

Los datos de entrada (geometría de trabajo, cargas aplicadas, parámetros iniciales de diseño, etc.) son los mismos empleados en ejemplos anteriores. La tensión última del material será de 1.9 MPa con un coeficiente de seguridad de 1. Es decir, la tensión máxima admisible será de 1.9 MPa. La solución utilizando la aplicación ha sido la mostrada a continuación:

Figura 3.50 Solución para el tipo de optimización B. Conducto sometido a presión interna

Que corresponde a una fracción de volumen de 0.33928. La tensión máxima alcanzada ha sido de 1.892 MPa y se han requerido 5 análisis de optimización para alcanzar la convergencia del proceso. En la siguiente tabla se presentan los valores de fracción de volumen y máxima tensión para cada iteración del algoritmo.

Tabla 3-3 Evolución del proceso (optimización B). Conducto sometido a presión interna

Iteración	Fracción de Volumen	Máxima tensión alcanzada (MPa)
1	1	1.683
2	0.5	1.790
3	0.2	2.117
4	0.377	1.860
5	0.339	1.892

3.4.4.2 Viga biapoyada

Este el caso que se ha venido analizando durante toda la tesis de máster porque era el ejemplo que utilizaba inicialmente el programa básico de optimización. La carga aplicada tiene la forma mostrada en la Figura 3.47, con un máximo de 60 MPa.

Optimización A: Minimización de las tensiones

Se han utilizado los mismos datos de la sección 3.1.4 (Figura 3.8), que daban la siguiente solución:

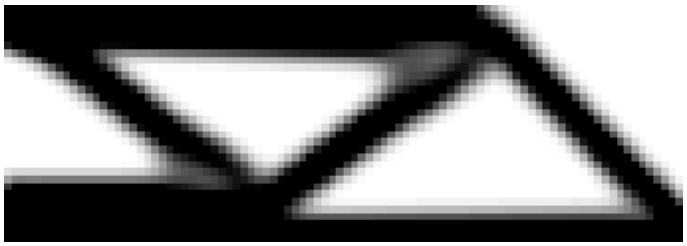


Figura 3.51 Solución para el tipo de optimización A. viga biapoyada

Cálculo de tensiones

Para poder realizar el análisis en tensiones de la viga biapoyada se tienen que tener en cuenta primero un par de consideraciones referentes a las singularidades que aparecen en el análisis debido tanto a la existencia de fuerzas puntuales como a restricciones de desplazamiento. Estos dos problemas se dan frecuentemente cuando se trabaja con elementos finitos y el usuario tiene que ser capaz de interpretar los resultados debidos a estas singularidades y solucionarlos en la medida de lo posible.

Tal y como se mostró en secciones anteriores la viga biapoyada tiene un fuerza puntual vertical en la esquina superior izquierda, y una restricción de desplazamiento en *X* en la esquina inferior derecha.

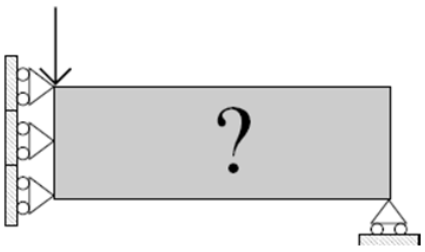


Figura 3.52 Condiciones de contorno viga biapoyada

Una fuerza puntual es teóricamente una carga que se aplica en un área de valor cero, lo cual implica unas tensiones en ese punto infinitas. Para poder solucionar esto, se ha modificado la fuerza puntual por una presión distribuida a lo largo de una pequeña línea tal y como se muestra a continuación:

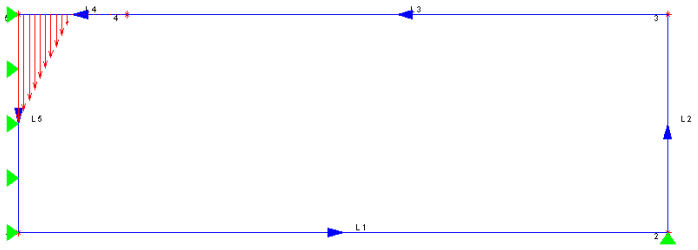


Figura 3.53 Modificación de la fuerza por una presión aplicada en una linea

De esta forma se evita la singularidad de la fuerza puntual. En cambio, la singularidad que aparece debido a la restricción de desplazamiento en la esquina inferior derecha no se puede evitar.

Con estos datos, las tensiones internas se muestran a continuación:

Figura 3.54 Distribución de tensiones en la viga biapoyada

Como se puede observar, las tensiones calculadas en el apoyo derecho inferior son extremadamente altas ($7x10^8$) debido a la singularidad. Por eso, tiene que haber un conocimiento previo por parte del usuario para interpretar el resultado y no tener en cuenta estas tensiones tan altas en el momento de diseñar. A continuación se realiza un reescalado de la gama de colores para poder representar de una forma más adecuada las tensiones:

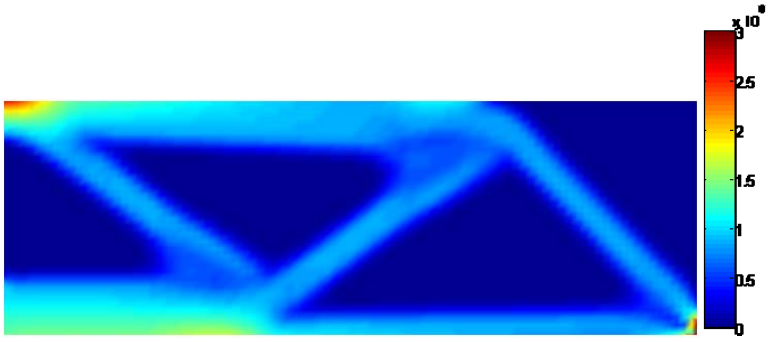


Figura 3.55 Reescalado de la gama de colores de las tensiones en la Figura 3.54

Ahora sí se observa con claridad la distribución de tensiones. Las tensiones máximas se obtienen justo donde está aplicada la presión, con un valor de 240 MPa. Obsérvese además, como la correcta distribución del material consigue que las tensiones en el resto de la estructura se repartan de forma homogénea.

Es interesante también mostrar a modo de ejemplo, la posibilidad de representar los desplazamientos en el componente, tal y como se muestra en la siguiente figura.

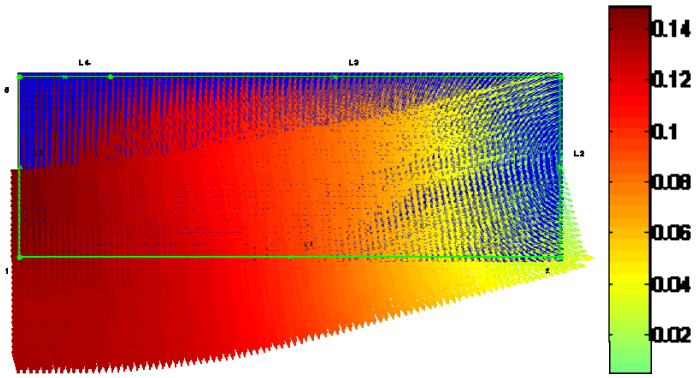


Figura 3.56 Desplazamientos en la viga biapoyada

Se observa que los resultados obtenidos son coherentes, ya que los máximos desplazamientos se producen justo en la parte central de la viga inferior, donde el momento flector es máximo. Se puede ver como justo en esta zona los desplazamientos son completamente verticales (debido a la simetría de la figura) y como a medida que nos acercamos al extremo, donde se apoya la estructura, los desplazamientos en *x* aumentan.

Optimización B: Minimización de la cantidad de material

En este caso se trata de encontrar la cantidad de material necesaria y su distribución para que las tensiones máximas alcanzadas no superen una cierta tensión última de fallo. El problema es que para ello se requiere, después de cada análisis de optimización, encontrar la máxima tensión en el componente, y se sabe que esta se va encontrar en la singularidad mencionada anteriormente. Cómo esta tensión no se debería tener en cuenta realmente para este análisis, se ha creado un tipo de geometría, que se define en la interfaz gráfica, que determina zonas donde no se tendrán en cuenta las tensiones para esta aplicación. El proceso que se ha seguido para crear este tipo de geometrías ha sido similar al de las geometrías fijas (apartado 1.5.2.4). Estas geometrías serán útiles generalmente cuando existan singularidades en el problema de definición.

Se ha definido por tanto una geometría auxiliar, tal y como se observa en la Figura 3.57, donde no se tendrán en cuenta las tensiones para encontrar esa tensión máxima en el componente.

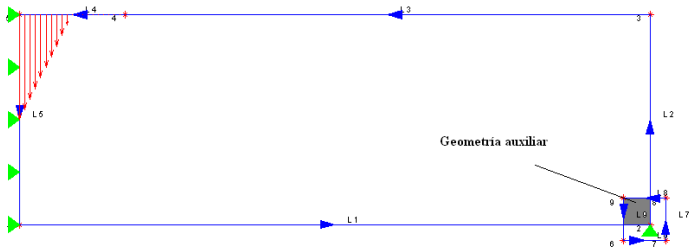


Figura 3.57 Geometría de trabajo con zona donde no se tendrán en cuenta las tensiones

Una vez solucionado el problema de la singularidad se procede a realizar el tipo de optimización B. Se toma como tensión última de fallo 400 MPa y un coeficiente de seguridad de 1.33. Es decir, la tensión máxima admisible será de 300 MPa. La solución se muestra a continuación.

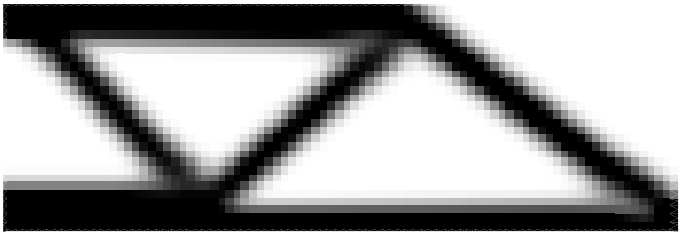


Figura 3.58 Solución para el tipo de optimización B. Viga biapoyada

Que corresponde a una fracción de volumen de 0.39885. La máxima tensión encontrada ha sido de 302.15 MPa que dista un 0.7 % de la tensión máxima admisible (recordar que se está utilizando una tolerancia del 2% para que la solución converja). Han sido necesarios 5 análisis de optimización (del tipo A) para obtener la solución final. En la siguiente tabla se muestra la evolución del proceso.

Tabla 3-4 Evolución del proceso (optimización B). Viga biapoyada.

Iteración	**Fracción de Volumen**	**Máxima tensión alcanzada (MPa)**
1	1	149.10
2	0.5	255.10
3	0.285	465.71
4	0.443	268.96
5	0.399	302.15

3.4.4.3 Gancho

Para este caso, que ya se ha analizado en numerosas ocasiones anteriormente, se utiliza la geometría de la Figura 3.17 (sección 3.4.1), con una presión distribuida tal y como se observa en la geometría que tiene un máximo de 50 MPa.

Optimización A: Minimización de las tensiones

Se utilizarán los mismos parámetros que en la sección 3.4.1, que ofrecían la solución que se muestra a continuación.

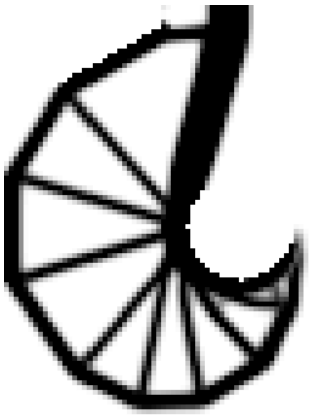


Figura 3.59 Solución para el tipo de optimización A. Gancho

Cálculo de tensiones

Se analiza el caso de tensión plana.

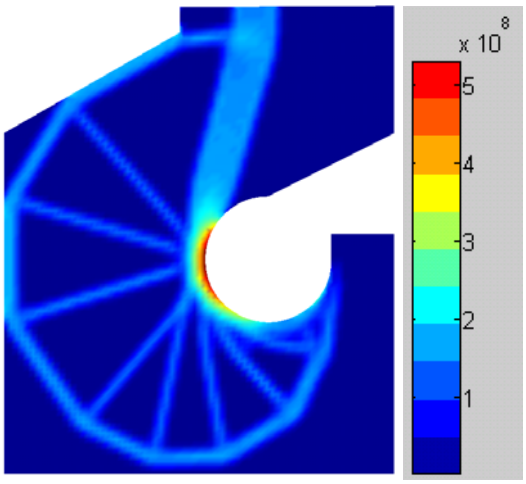


Figura 3.60 Distribución de tensiones de Von Misses. Gancho.

La máxima tensión obtenida se encuentra en la zona de aplicación de la carga y es de 525 MPa.

Optimización B: Minimización de la cantidad de material

Se toma un valor para la tensión última de fallo en este caso de 500 MPa con un coeficiente de seguridad de 1.25. La tensión máxima admisible en el componente será por tanto de 400 MPa. La solución ha sido la que se muestra a continuación.

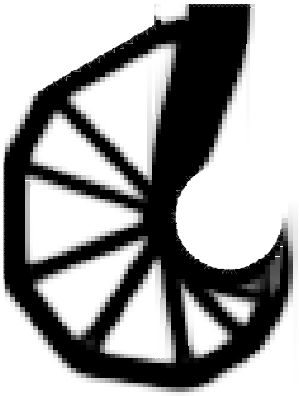


Figura 3.61 Solución para el tipo de optimización B. Gancho

Que corresponde a una fracción de volumen de 0.42027. La tensión máxima alcanzada ha sido de 399.08 MPa. Se han utilizado 5 pasos de optimización para poder encontrar dicha fracción de volumen.

Tabla 3-5 Evolución del proceso (optimización B). Gancho.

Iteración	Fracción de Volumen	Máxima tensión alcanzada (MPa)
1	1	266.63
2	0.5	363.49
3	0.3115	493.95
4	0.4391	388.89
5	0.42027	399.08

3.4.4.4 Viga en voladizo

El siguiente caso que se analiza es el de una viga en voladizo. En esta ocasión la geometría de trabajo es rectangular. Se imponen restricciones de desplazamiento en *X* e *Y* en el empotramiento (lado izquierdo del rectángulo) y se aplica una carga vertical hacia abajo en el centro del lado derecho de valor 50 MPa. Esta fuerza está aplicada en forma de tangencial para evitar singularidades.

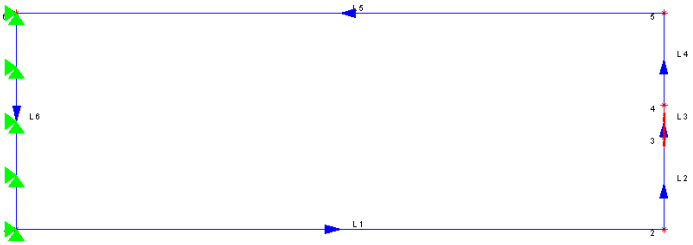


Figura 3.62 Geometría de trabajo de la viga en voladizo

<u>Optimización A: Minimización de las tensiones</u>

En un primer análisis se utiliza una fracción de volumen de 0.5, con un total de 10384 elementos. Se muestra a continuación parte del proceso de optimización hasta obtener la solución final.

Volfrac=0.5; *Penal*=3; r_{min} =1.5 (elementos); *NM*=6 (10384 elementos activos)

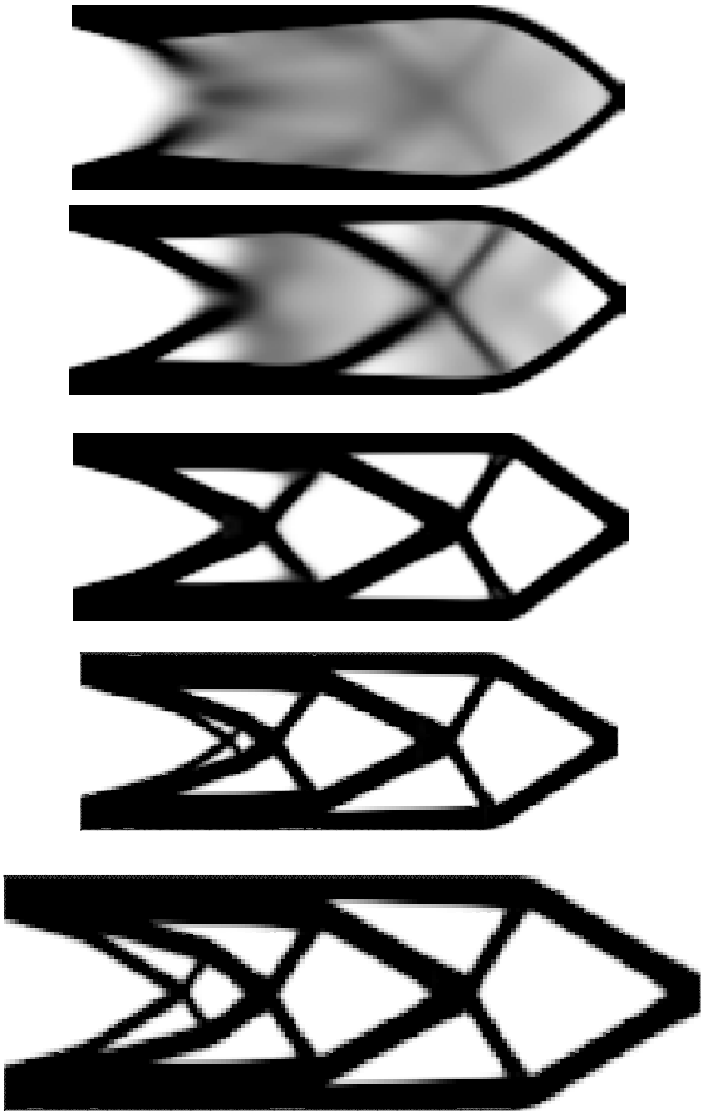


Figura 3.63 Evolución y solución para el tipo de optimización A. Viga en voladizo

Aquí es interesante observar la estructura de barras que ofrece la solución del programa (truss-like structures en la bibliografía inglesa). Esta coincide otra vez con la solución que se encuentra en fuentes bibliográficas para el caso de la viga en voladizo.

Cálculo de tensiones

Se ha utilizado el caso de tensión plana. La distribución en tensiones se muestra a continuación

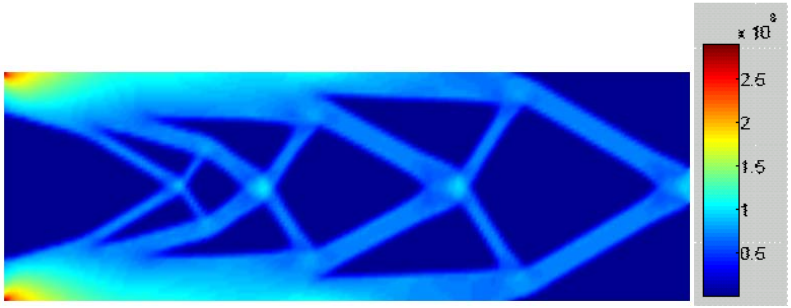


Figura 3.64 Distribución de tensiones de Von Misses. Viga en voladizo.

Las máximas tensiones se encuentran en el empotramiento, pero hay que tener en cuenta que aquí existe una singularidad. En el resto de la figura se puede observar que la distribución del material es tal que las tensiones se reparten de forma homogénea, demostrando así la optimalidad de la solución. La optimización de componentes es al fin y al cabo la obtención de formas que ofrezcan una distribución de las tensiones lo más homogénea posible.

Se observa además, que a medida que se avanza hacia el empotramiento las tensiones aumentan debido al incremento del momento flector. El programa responde a este fenómeno incrementado el espesor de las barras situadas en la parte superior e inferior

Optimización B: Minimización de la cantidad de material

En este caso ocurre lo mismo que con la viga biapoyada, es decir, debido a la existencia de singularidades que producen el elevado incremento de las tensiones (hecho que no ocurre en la realidad, dado que los empotramientos nunca son perfectos), se debe recurrir a la definición de una geometría auxiliar para no tener en cuenta estas tensiones. En la siguiente figura se muestra la geometría de trabajo con esta nueva geometría auxiliar.

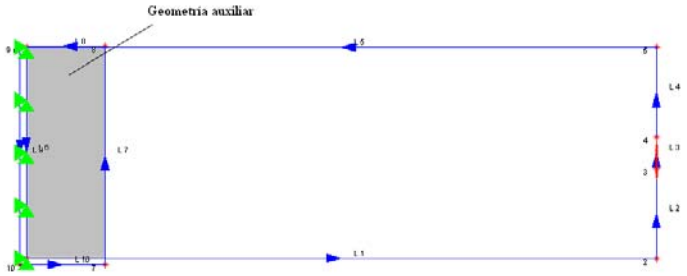


Figura 3.65 Geometría de trabajo y geometría auxiliar para definir una zona donde no se tengan en cuenta las tensiones

Una vez desestimadas las tensiones debidas a la singularidad se procede a realizar el proceso de optimización de tipo B. Se ha tomado como tensión última de fallo de 200 MPa y un coeficiente de seguridad de 1.33. Se tiene por tanto una tensión máxima adimisible de 150.37 MPa. El resultado se muestra a continuación:

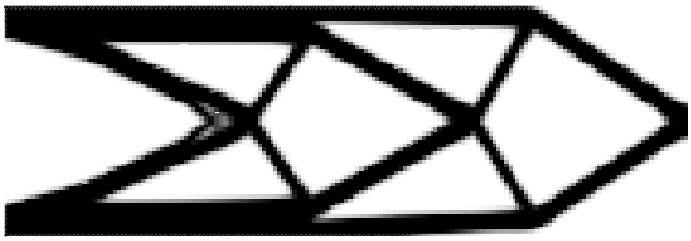


Figura 3.66 Solución para el tipo de optimización B. Viga en voladizo

Que corresponde a una fracción de volumen de 0.37673. La tensión máxima encontrada ha sido de 153.13 MPa, y han sido necesarios 6 análisis de optimización (de tipo A) para que la solución convergiese. La evolución del proceso se muestra en la siguiente tabla.

Tabla 3-6 Evolución del proceso (optimización B). Viga en voladizo.

Iteración	Fracción de Volumen	Máxima tensión alcanzada (MPa)
1	1	117.10
2	0.5	125.07
3	0.2	340.52
4	0.44933	128.74
5	0.38912	144.35
6	0.37673	153.13

3.4.4.5 Volante de inercia

Un volante de inercia es un elemento totalmente pasivo, que únicamente aporta al sistema una inercia adicional de modo que le permite almacenar energía cinética. Este volante continúa su movimiento por inercia cuando cesa el par motor que lo propulsa. De esta forma, el volante de inercia se opone a las aceleraciones bruscas en un movimiento rotativo. Así se consiguen reducir las fluctuaciones de velocidad angular. Es decir, se utiliza el volante para suavizar el flujo de energía entre una fuente de potencia y su carga. En la actualidad numerosas líneas de investigación están abiertas a la búsqueda de nuevas aplicaciones de los volantes, como por ejemplo la absorción de la energía de frenado de un vehículo, de modo que se reutilice posteriormente en su aceleración (KERS).

Se pretende en este caso determinar la geometría óptima de una volante de inercia. La geometría de trabajo está formada por una corona circular. La circunferencia que define el agujero del medio tiene restringidos los desplazamientos en X e Y. En la circunferencia exterior se aplica una carga tangencial en el sentido contrario a las agujas del reloj.

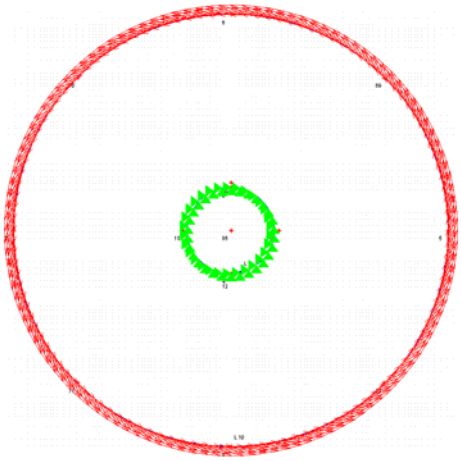


Figura 3.67 Geometría de trabajo

Este caso es interesante además porque se precisa definir geometrías fijas para generar zonas donde tiene que existir el material necesariamente. Concretamente se requiere material en una corana exterior y otra interior tal y como muestra la Figura 3.68.

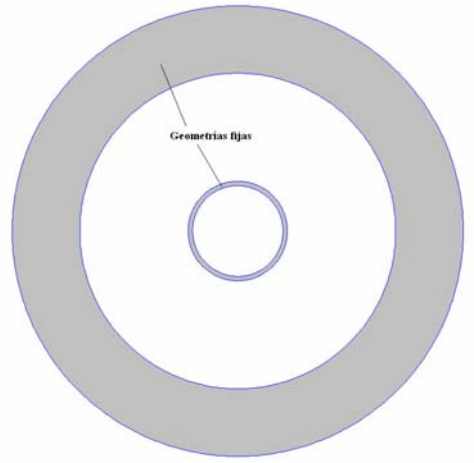


Figura 3.68 Geometría de trabajo con geometrías fijas.

La carga empleada en este caso es una presión tangencial en la circunferencia exterior de valor 10 MPa

Optimización A: Minimización de las tensiones

Una vez definidas la geometría de trabajo, las restricciones de desplazamientos, cargas y geometrías fijas, se procede a arrancar el programa de optimización topológica. La solución para un nivel de malla 5 (eso siginifica 3092 elementos activos para este caso en concreto) y una fracción de volumen de 0.5 se muestra a continuación junto con su solución.

Volfrac[1]=0.5; *Penal*=3; r_{min} =1.5 (elementos); *NM*=5 (3092elementos activos)

Nota: Recuérdese que la fracción de volumen cuando existen geometrías fijas se refiere a la geometría de trabajo que no contiene geometrías fijas.

Figura 3.69 Evolución y solución para el tipo de optimización A. Volante de inercia. 3092 elementos activos

Tal y como se comentó en secciones anteriores, para obtener las figuras del proceso iterativo no se utiliza la subrutina que permite representar correctamente la geometría de trabajo, ya que esta es computacionalmente más costosa. Esta representación se utiliza únicamente para obtener la figura final. Obsérvese que en la tercera imagen de la Figura 3.69, sí que se definen bien los contornos de la geometría de trabajo.

[1] Recuérdese que la fracción de volumen cuando existen geometrías fijas se refiere a la parte de la geometría de trabajo que no contiene geometrías fijas

La figura obtenida parece muy razonable y de hecho es muy parecida a la figura que se obtiene en las referencias [7] y que se muestra a continuación.

Figura 3.70 Volante de inercia

La única diferencia es que la solución obtenida de la figura 70, esta girada 45º. Si rotamos la imagen justamente 45º, se observa que está coincide con la imagen de la Figura 3.70.

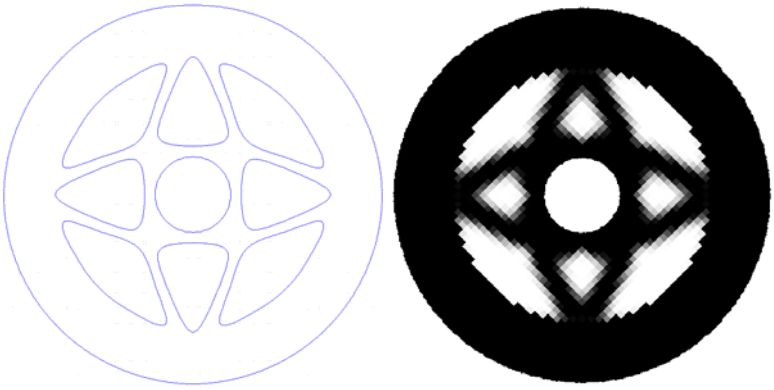


Figura 3.71 Comparación de la solución obtenida con otros métodos de optimización y el método del programa utilizado (sol. rotada 45º)

Si se requiere de una mayor complejidad de la estructura, con barras de menor espesor, se debe aumentar el nivel de refinamiento. En las siguientes figuras 73 y 74, se muestra la evolución y solución para un nivel de refinamiento de 6 y 7 respectivamente (12368 y 49472 elementos activos). Se ha mantenido la fracción volumen a 0.5 y el resto de parámetros al mismo valor.

Volfrac=0.5; *Penal*=3; r_{min} =1.5 (elementos); *NM*=6 (12368 elementos activos)

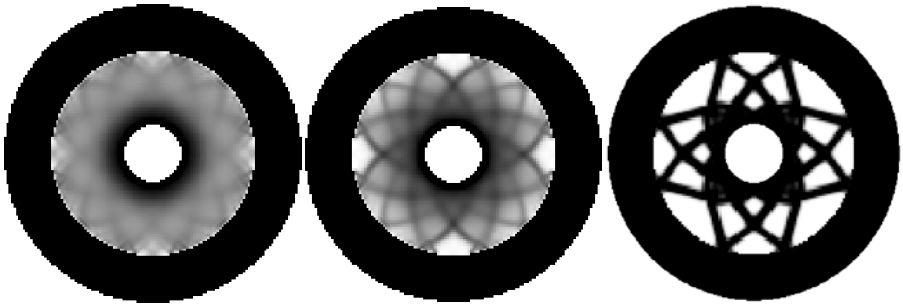


Figura 3.72 Evolución y solución para el tipo de optimización A. Volante de inercia. 12368 elementos activos

Volfrac=0.5; *Penal*=3; r_{min} =1.5 (elementos); *NM*=7 (49472 elementos activos)

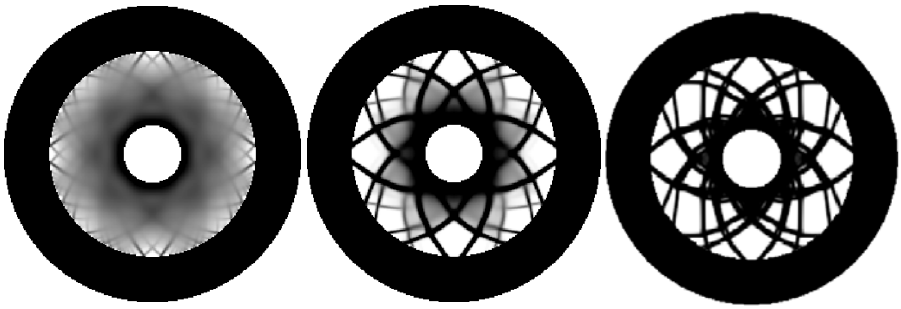


Figura 3.73 Evolución y solución para el tipo de optimización A. Volante de inercia. 49472 elementos activos

Puede ser interesante además, incrementar la fracción de volumen para ver como evoluciona la distribución de material. Se ha utilizado el nivel de malla 6 y se ha incrementado la fracción de volumen a 0.75.

Volfrac=0.75; *Penal*=3; r_{min} =1.5 (elementos); *NM*=6 (12368 elementos activos)

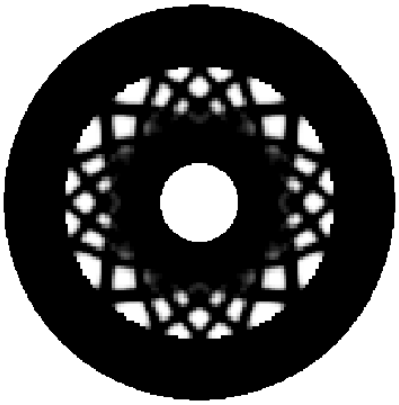


Figura 3.74 Solución para el tipo de optmización A. Volante de inercia. Fracción de volumen de 0.75.

Cálculo de tensiones

Para la carga tangencial de 10 MPa en la circunferencia exterior se obtiene la siguiente distribución de tensiones:

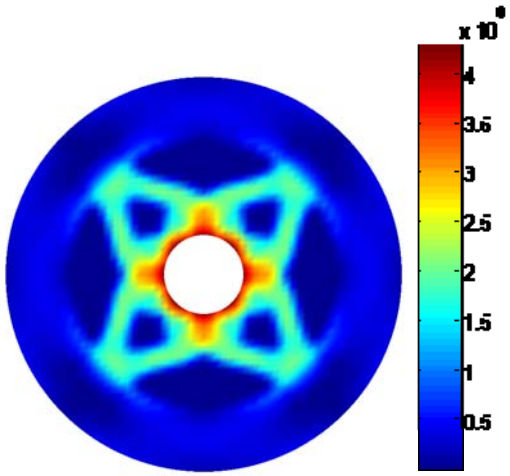


Figura 3.75 Distribución de tensiones de Von Misses. Volante de Inercia.

La máxima tensión del componente se encuentra en la circunferencia interior, justo donde están aplicadas las restricciones de desplazamiento, con un valor de 430 MPa. Tiene sentido que sea en esta zona porque es ahí donde las deformaciones son máximas.

Optimización B: Minimización de la cantidad de material

Se ha tomado como tensión última un valor de 500 MPa con un coeficiente de seguridad de 1.1, lo que implica que la tensión máxima admisible es de 454.54 MPa. El resulto se muestra en la siguiente figura.

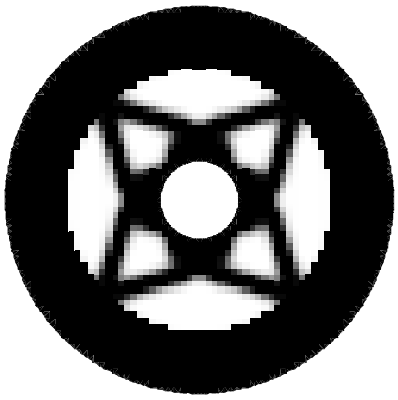


Figura 3.76 Solución para el tipo de optimización B. Volante de inercia.

Que corresponde a una fracción de volumen de 0.41087. La máxima tensión en el componente se da en la circunferencia interna y tiene un valor de 448.38 MPa. Se han analizado 6 procesos de optimización de tipo A hasta encontrar la solución óptima. En la siguiente tabla se muestra la evolución del proceso

Tabla 3-7 Evolución del proceso (optimización B). Volante de incercia.

Iteración	Fracción de Volumen	Máxima tensión alcanzada (MPa)
1	1	411.80
2	0.5	422.90
3	0.2	889.54
4	0.4695	429.85
5	0.4274	443.35
6	0.4109	448.38

3.4.4.6 Prótesis de cadera

Se realiza a continuación el análisis de optimización topológica para el caso de una prótesis de cadera. Cabe destacar que se trata de una simplificación de un caso real y que el objetivo del estudio no es el de conseguir un resultado para una inmediata aplicación industrial sino obtener un modelo aproximado que permita corroborar el buen funcionamiento del programa. La geometría de trabajo utilizada se muestra a continuación junto con las condiciones de contorno.

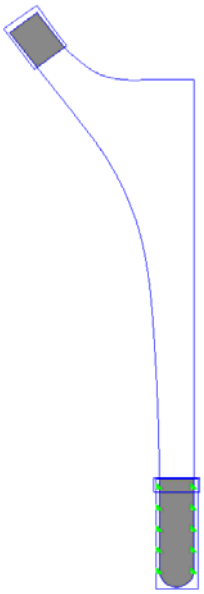


Figura 3.77 Geometría y condiciones de contorno

Las zonas grises han sido definidas para que NO se tengan en cuenta en el análisis de optimización topológica, ya que por razones de funcionalidad interesa que exista material en cualquier caso.

Las restricciones de desplazamiento se han puesto en la parte inferior, de manera que la prótesis quede empotrada (simulando la adherencia al fémur).

Las cargas se han definido en el cabezal de la prótesis de forma normal y tangencial a esta, tal y como se muestra a continuación, con un valor total de 16 MPa (una fuerza de 130 Kg)

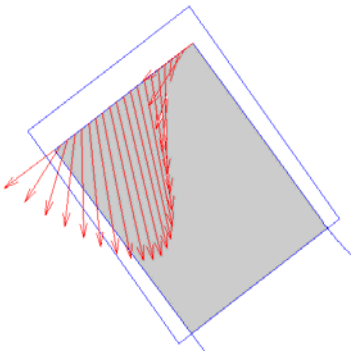


Figura 3.78 Definición de las cargas

En un primer análisis, se realiza la optimización con una fracción de volumen fija (0.6 en este caso) para que el programa distribuya el material óptimamente de manera que se minimicen las tensiones dentro del componente.

Optimización A: Minimización de las tensiones

Se muestra a continuación la evolución (y algunas iteraciones intermedias) del proceso para una fracción de volumen de 0.6.

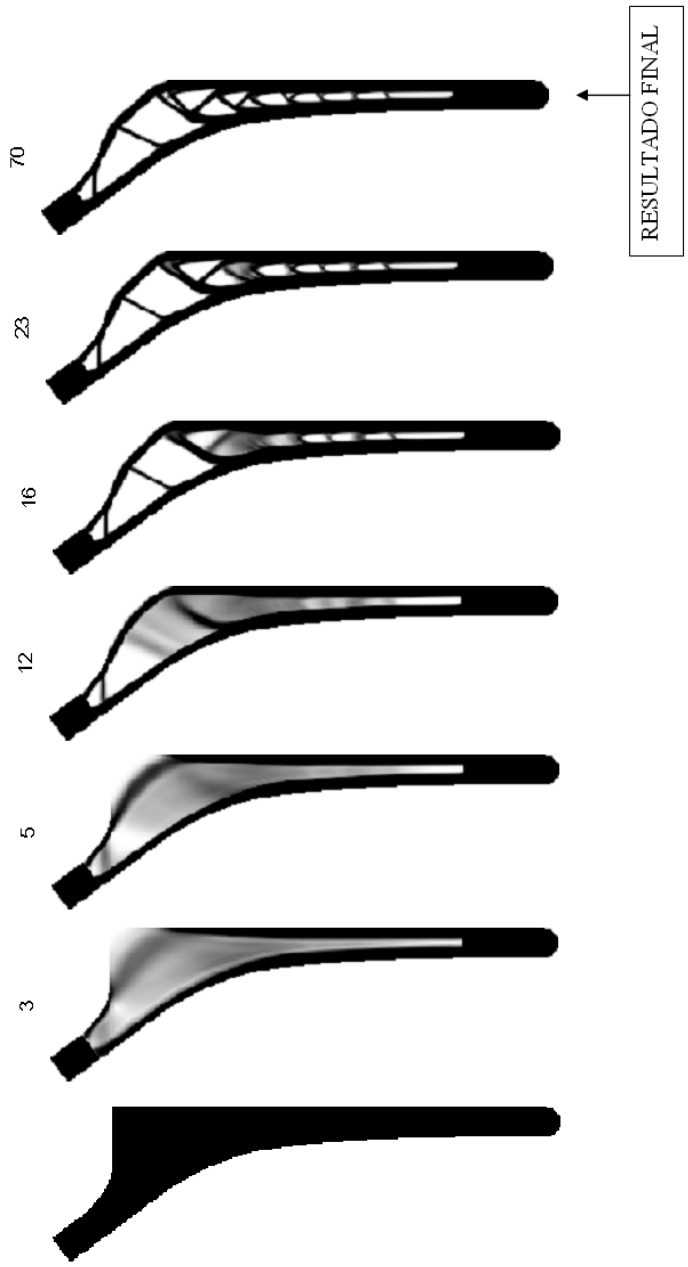


Figura 3.79 Pasos intermedio y solución final Prótesis de Cadera

Cálculo de tensiones

Con esta distribución de masa final, se pueden calcular las tensiones en el componente:

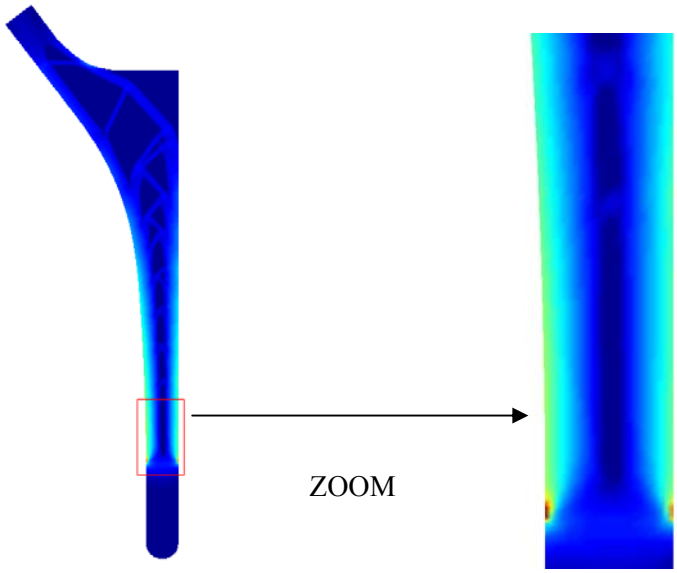
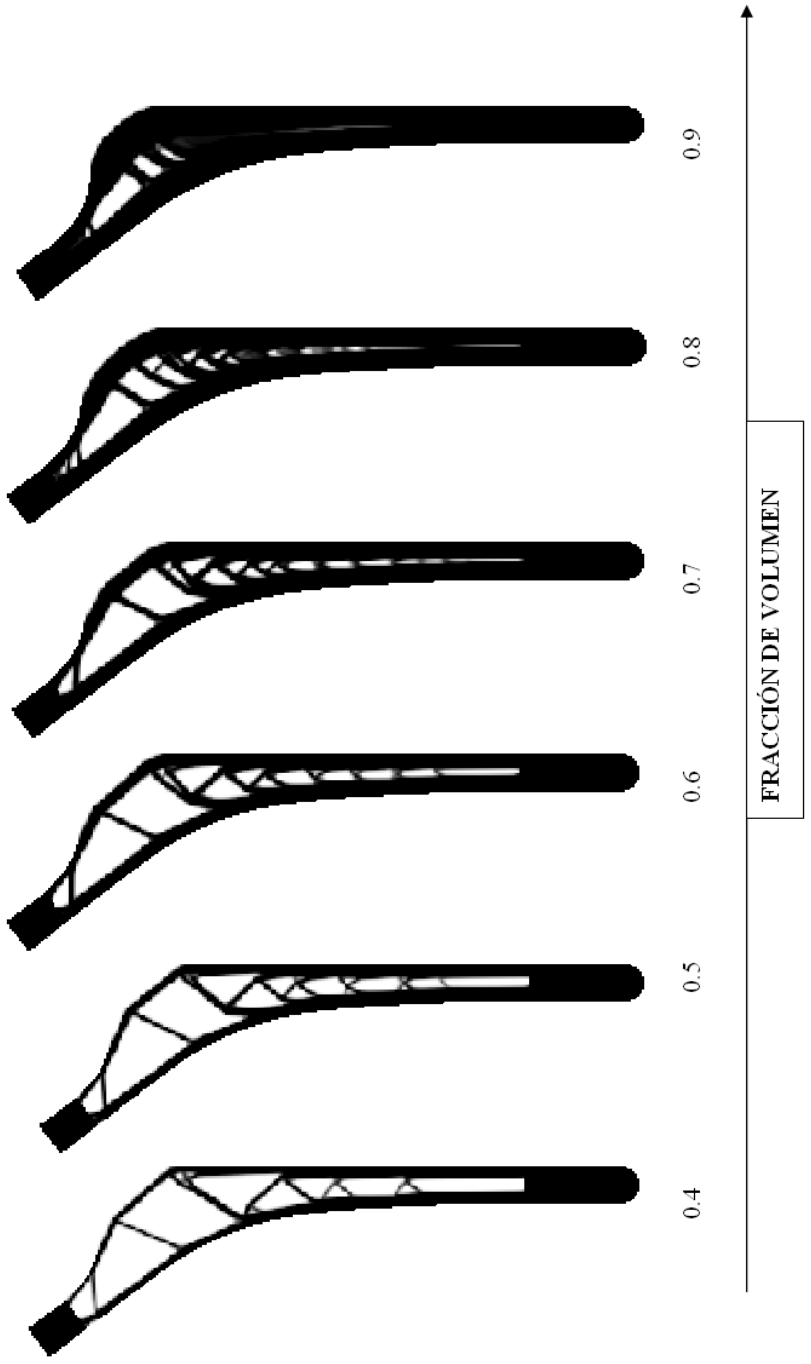


Figura 3.80 Cálculo de tensiones

Debido a las restricciones de desplazamiento impuestas, aparecen singularidades que provocan tensiones muy altas, tal y como se puede apreciar en la figura. Estas tensiones no son reales y por eso se ha creado una zona (justo en esta parte) para que no se tengan en cuenta tales tensiones. De esta forma, exceptuando las tensiones debidas a singularidades, la máxima tensión encontrada en el componente es de 255.51 MPa.

De la misma forma que se ha analizado el caso para una fracción de volumen de 0.6, se puede realizar el análisis para otras fracciones de volumen. Se muestra a continuación los resultados obtenidos para fracciones de volumen desde 0.4 a 0.9


0.9
0.8
0.7
0.6
0.5
0.4
FRACCIÓN DE VOLUMEN

Optimización B: Minimización de la cantidad de material

Por otra parte, y haciendo uso de la optimización topológica, se puede calcular la cantidad de material necesaria (fracción de volumen) y su distribución para que no se sobrepase en ningún punto del componente una cierta tensión última introducida por el usuario. Para conseguirlo, el programa utiliza un algoritmo matemático que analiza varios casos con diferentes fracciones de volumen hasta que converge a la fracción de volumen buscada.

Para poner un ejemplo, se introduce como tensión última 310 MPa, con un factor de seguridad de 1.2. Es decir, la máxima tensión admisible en el material será de 258.3.

Se realiza el proceso y se obtienen los siguientes análisis.

Tabla 3-8 Evolución análisis

Iteración	Fracción de Volumen	Máxima tensión alcanzada (MPa)
1	1	249.73
2	0.5	259.26
3	0.6188	250.06
4	0.5246	253.61
5	0.5089	258.07

El resultado es el que se muestra a continuación, con una tensión máxima de 258,07 MPa (no se tienen en cuenta las tensiones debidas a las singularidades)

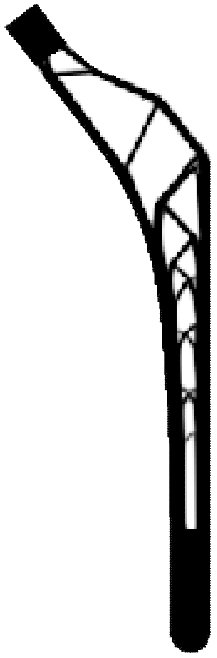


Figura 3.81 Solución para tensión máxima admisible 258.3 MPa

3.4.4.7 Cuadro de bicicleta

A continuación se realiza el análisis de optimización topológica para el caso de un cuadro de bicicleta. La modelización del problema (geometría de trabajo, cargas, condiciones de contorno, etc.) se ha hecho de forma aproximada siendo el principal interés del estudio la respuesta del programa en un caso más real.

La geometría de trabajo y cargas aplicadas son las que se muestran a continuación

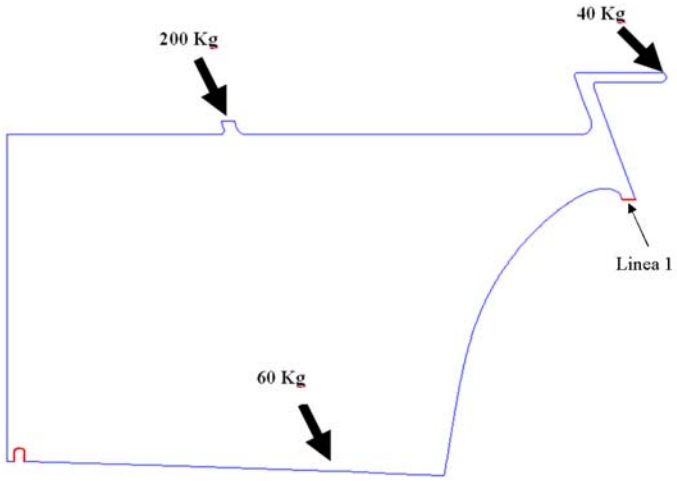


Figura 3.82 Definición de geometría y cargas

Las condiciones de contorno son:

- Restricción de desplazamiento en *y* en la línea horizontal 1.
- Restricción de desplazamiento en *x* e *y* en la hendidura donde acopla la rueda trasera.

El manillar de la bicicleta se ha añadido a la geometría para poder crear un momento en el cuadro de forma más real.

Optimización A: Minimización de las tensiones

Se ha utilizado una fracción de volumen de 0.3 para el análisis. Se muestra a continuación la evolución y solución final

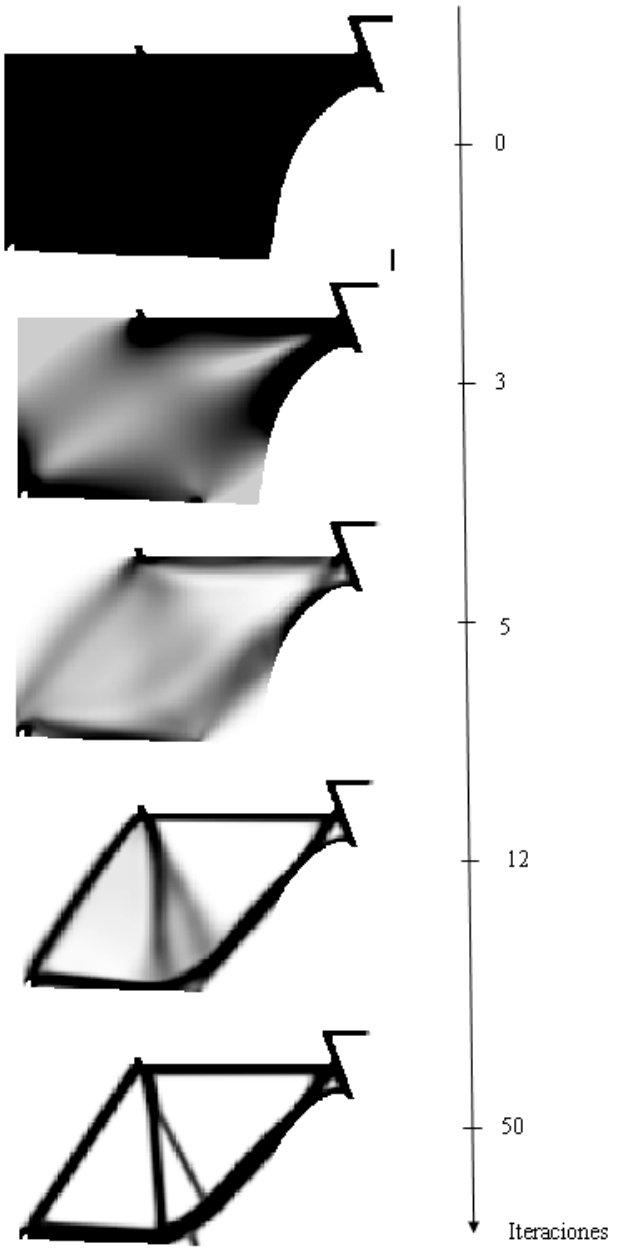


Figura 3.83 Iteraciones intermedias para la bicicleta

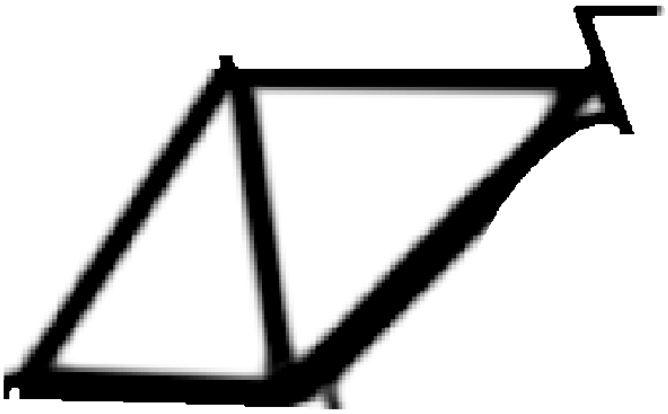


Figura 3.84 Solución final (70 iteraciones)

4 PROGRAMA OPTIMIZACIÓN TOPOLÓGICA EN GFEM CON ADAPTATIVIDAD.

Hasta este momento se ha dotado el software de GFEM del departamento de un módulo de optimización topológica a partir de un programa básico. Se han hecho las modificaciones necesarias para poder utilizar geometrías cualesquiera que intersectan elementos de la malla. Se han desarrollado ciertas aplicaciones y funciones que han otorgado versatilidad al módulo.

En esta sección se pretende ir más allá y aprovechar la posibilidad de utilizar adaptatividad de la malla ya que el software de GFEM del departamento dispone de esta función. Esta se utiliza para refinar la malla en ciertas zonas de modo que el error se minimice. Pero en esta sección se aprovechará la aplicación para obtener un mayor definición y resolución de las soluciones obtenidas. Tal y como se explico en la sección 3.4.3 (*Independencia de la solución respecto el tamaño de elemento*), se pueden conseguir componentes con espesores de barra grandes (solución de una malla basta) con mallas más finas utilizando el método de mantener el r_{min} en valor absoluto constante. El inconveniente de este método, como ya se explico, es que el coste computacional es muy elevado ya que se debe refinar toda la malla. Además, el resultado obtenido resulta ser una solución menos pixelada pero con el mismo grado de definición, ya que la zona de transición de material a no material se mantiene igual.

Con la técnica de la adaptatividad se pretende partir de una malla basta y refinar en aquellas zonas de transición de material a no material, es decir zonas con pesos de valor intermedio, entre 0 y 1 (excluidos).

El principal *handicap* de la adaptativdad es el uso de elementos de diferente tamaño. Esto implica que se hagan un serie de modificaciones en ciertas funciones del programa que se explicación en la siguiente sección.

4.1 Integración del programa en el software de GFEM con adaptatividad

4.1.1 Derivada de energía de deformación por unidad de área

En la expresión de la actualización de los pesos

$$x_e^{new} = x_e^{old}\left(\frac{-\dfrac{\partial c}{\partial x_e}}{\lambda\dfrac{\partial V}{\partial x_e}}\right)^{\eta} \tag{4.1}$$

aparece la expresión $\partial V/\partial x_e$. Tal y como se explico en la sección 3.1.4 (Uso de geometrías de trabajo cualesquiera) esta expresión es constante para todos los elementos cuando se utilizan mallas regulares, ya que $\partial V/\partial x_e$ tomo el valor A_e. Pero no ocurre esto cuando se utilizan elementos de diferente tamaño. Ya se explicó en la sección 3.1.4 cómo se procedía y finalmente se llegaba a la conclusión de que se utilizan las derivadas de energía de deformación por unidad de área. De este modo el programa funciona correctamente para elementos de diferente tamaño.

4.1.2 Modificaicones en la rutina *check* (filtro)

El programa de optimización topológica básico disponía de un filtro muy simple pero muy costoso computacionalmente, que ya se modificó según se explica en la sección 3.1.1.1, creando dos nuevas funciones llamadas *check3* y *check4*. Estas nuevas subrutinas son muy eficientes para mallas regulares pero quedan obsoletas cuando se utiliza adaptatividad, ya que el sistema de búsqueda de vecinos se ejecutaba mediante la creación de una matriz que representaba la malla regular.

El nuevo sistema de búsqueda de vecinos, que se denominará *check5,* se basa en escoger de todos los elementos activos, únicamente aquellos cuyos centros esten dentro de unos intérvalos para las coordenas *x* e *y*. Estos intérvalos estarán centrados en el elemento que se esté alisando en cada momento y su límites dependerán del valor del r_{min}.

Esta rutina si bien es un poco más costosa que las presentadas anteriormente, es la única solución para el uso de elementos de diferente tamaño.

Con esto quedan seleccionados los vecinos que forman un recuadro alrededor del elemento en proceso de alisado. Tal y como se explicó en la sección 3.1.1 (*Integración del programa en el software de GFEM sin adaptatividad*) es el factor de convolución $\hat{H}_f$ es que se encarga de que entre los vecinos, sólo intervengan en el alisado los que caigan dentro del círculo con radio r_{min}. Con este funcionamiento en mente, la adaptatividad presenta otro problema, que se muestra a continuación.

Figura 4.1 a) y b) Selección de elementes mediante el filtro

Para el caso de la Figura 4.1 a) donde se procede al alisado del elemento marcado en verde, si se tiene un radio r_{min} que forma el círculo marcado en rojo, se puede dar la situación representada, donde las áreas marcadas en gris no intervienen en el alisado debido a que los centros de dichos elementos no caen dentro del área de alisado, aún estando tan cerca del elemento a alisar.

Se puede dar el caso contrario, Figura 4.1 b), donde se aumenta un poco el r_{min} y se tiene en cuenta todo el elemento, incluyendo mucha área que cae fuera del campo de influencia debido a las diferencias de tamaño entre elementos.

Para solucionar este problema se ha seguido la estrategia de seleccionar inicialmente los vecinos en un recuadro alrededor del elemento a alisar, tal y como se había comentado, y posteriormente bajar la información de los elementos de tamaño superior hasta el nivel de malla en el que se encuentra el elemento a alisar. Con esta modificación, siguiendo el ejemplo de la Figura 4.1 b) los elementos más grandes que el elemento a alisar intervendrían según se muestra a continuación:

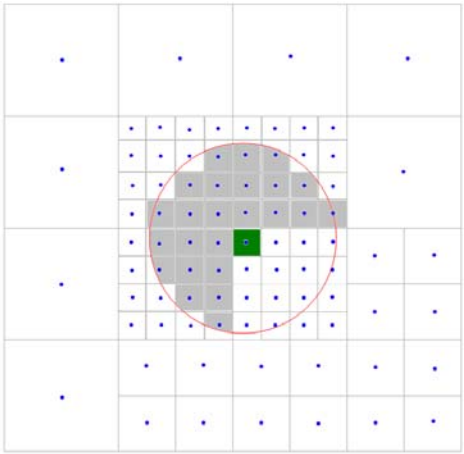


Figura 4.2 Selección de elementos mediante el filtro (con modificación)

De forma que aunque la zona de influencia no se adapta perfectamente a un círculo (nunca lo puede hacer tal y como está implementado el programa), se evitan grandes aportaciones o ausencias de área de influencia, tal y como pasaba antes. Hay que destacar que aunque los elementos que intervienen no formen un círculo debido a la topología de estos, no tiene la mayor importancia porque los elementos más alejados del elemento a alisar tienen menor influencia que los más cercanos.

Hay que considerar ahora el caso opuesto, es decir, cuando el elemento a alisar es de gran tamaño y dentro de su área de alisado entran elementos de menor tamaño. La situación se muestra a continuación:

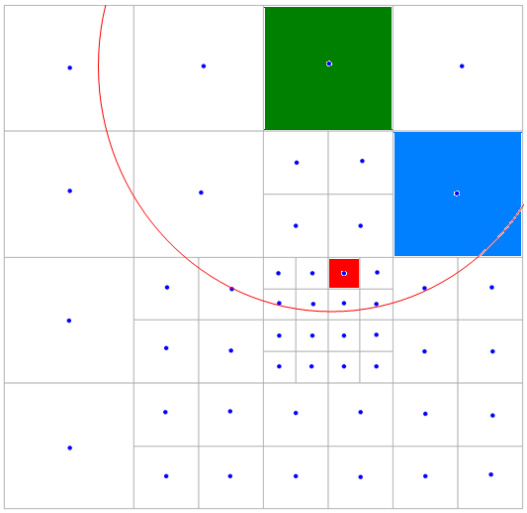


Figura 4.3 Selección elementos mediante filtro cuando el elemento a alisar es un elemento grande

El elemento a alisar sería el pintado en verde. En este caso no hay necesidad de bajar ni subir información entre niveles. El problema en este caso reside en que elementos a la misma distancia del elemento a alisar, como el azul y el rojo en la figura anterior (recordar que considera el centro de elemento) pero de diferente tamaño tendrían exactamente la misma influencia. A simple vista se entiende que el elemento azul debe influir más en el alisado que el rojo. Sin embargo, tal y como está formulada la expresión del alisado,

$$\frac{\overline{\partial c}}{\partial x_e} = \frac{1}{x_e \sum_{f=1}^{N} \hat{H}_f} \sum_{f=1}^{N} \hat{H}_f x_f \frac{\partial c}{\partial x_f} \tag{4.2}$$

ambos tienen la misma influencia ya que sólo cuenta la distancia entre estos y el elemento a alisar. Una solución residiría en bajar la información de todos los elementos al nivel del elemento más pequeño, alisar como si fuese una malla regular y después subir la información. Pero esto conllevaría un coste computacional considerable. Por tanto se propone una solución mucha más sencilla que pasa por ponderar además de por la distancia, por el área del elemento. De esta forma, la expresión del alisado queda como:

$$\overline{\frac{\partial c}{\partial x_e}} = \frac{1}{x_e \sum_{f=1}^{N} A_{full} \cdot \hat{H}_f} \sum_{f=1}^{N} \hat{H}_f \cdot A_{full,f} \cdot x_f \frac{\partial c}{\partial x_f} \tag{4.3}$$

donde A_{full} es el área del elemento (entero). De esta manera el elemento azul ponderará 16 veces más que el elemento rojo. O lo que es lo mismo, si el elemento azul estuviese subdividido en 16 sub-elementos ponderaría de la misma forma que si no lo estuviese.

Cabe aclarar que el filtro descrito hace referencia a la subrutina *check5,* que se utilizará únicamente cuando se implemente la función de adaptatividad de la malla, ya que es más costosa que *check3* y *check4,* que se utilizarán por tanto en el resto de casos.

4.1.3 r_{min}: distancia absoluta/ distancia normalizada

Hay que recordar que en el programa inicial de optimización topológica, el r_{min} tenia como unidad de longitud la longitud del lado de elemento. Después se explicó en la sección 3.4.3 cómo hacer independiente la solución respecto el nivel de malla utilizando siempre el mismo r_{min} independientemente del tamaño de elemento. Para ello resultaba más fácil medir el r_{min} en distancia absoluta (m por ejemplo) de forma que se podía mantener siempre un determinado valor sin tener en cuenta el nivel de malla utilizado.

Cuando se utiliza adaptativdad de la malla, se va a disponer de elementos de diferente tamaño, por tanto hay dos formas de asignar el valor de r_{min} a cada elemento:

1) Utilizar el mismo valor de r_{min} para todos los elementos independientemente del tamaño de estos. Se dice entonces que utilizamos r_{min} con distancia absoluta. La unidad de longitud para definirlo será la utilizada durante todo el problema (m por ejemplo).

2) Utilizar un r_{min} que dependa del tamaño de elemento. La unidad de medida será entonces la longitud del lado de elemento a alisar. Se denominará r_{min} con distancia normalizada.

Estas dos formas de abordar el proceso de alisado son fundamentales para definir cómo va a ser la solución del problema. En el caso de r_{min} con distancia

absoluta, la solución va estar marcada por los elementos más grandes. Es decir, tal y como se explico en la sección 3.4.2.5 (*Parámetro r_{min}),* el r_{min} debe ser como mínimo el valor de la longitud de lado de elemento. De no ser así el alisado no tiene efecto y la solución aparece sin elementos con pesos intermedios, solo elementos con pesos de 0s y 1s. Así pues, utilizando el r_{min} con distancia absoluta, este se tendrá que definir como mínimo como el lado de elemento del elemento de mayor tamaño en la malla. Esto implica que el r_{min} sea un valor muy grande para los elementos más pequeños de la malla y el efecto será que en las zonas muy refinadas el alisado tengo mucho efecto y por tanto saldran zonas menos pixeladas pero igual de poco definidas. Esto es lo mismo que ocurria cuando se explicó el apartado "*Indepencia de la solución con el tamaño de elemento*".

Con la segunda manera explicada, utilizando el r_{min} normalizado, lo que se consigue es que en las zonas más refinadas si se pueden representar detalles con más precisión y por tanto se define mejor la geometría, o lo que es lo mismo, se consigue una mejor resolución.

4.1.4 Criterio de refinamiento

El objetivo de utilizar el refinamiento adaptativo es el de poder definir mejor las zonas de transición de material a no material en el componente. Por tanto, la idea es refinar justo en esta zona para que los elementos siendo de menor tamaño puedan representar de una forma más detallada la solución en esta parte. Dichas zonas tienen la particularidad de que los elementos situados aquí tienen asignados pesos intermedios, es decir entre 0.001 y 1 (excluidos). Recuérdese que no se permitía alcanzar el valor de peso 0 por problemas de singularidad en la matriz de rigidez.

En el proceso de adaptatividad se tienen que elegir de alguna forma aquellos elementos que van a ser refinados. Por tanto lo más sencillo es escoger aquellos elementos que estén activos y que tengan su peso dentro de un determinado intervalo. Se escoge este intervalo como $]0.001,1[$ de forma que se refinarían todos los elementos intermedios.

4.1.5 Algoritmo de adaptatividad

El algoritmo de adaptatividad está formado básicamente por el bucle del proceso de adaptatividad que define el número de veces que se refina la malla hasta llegar a una solución que ya no progresa más. Para cada malla diferente se tiene un bucle de optimización topológica que da como solución una distribución de pesos determinada cuando éste converge. Se muestra este algoritmo en el siguiente esquema:

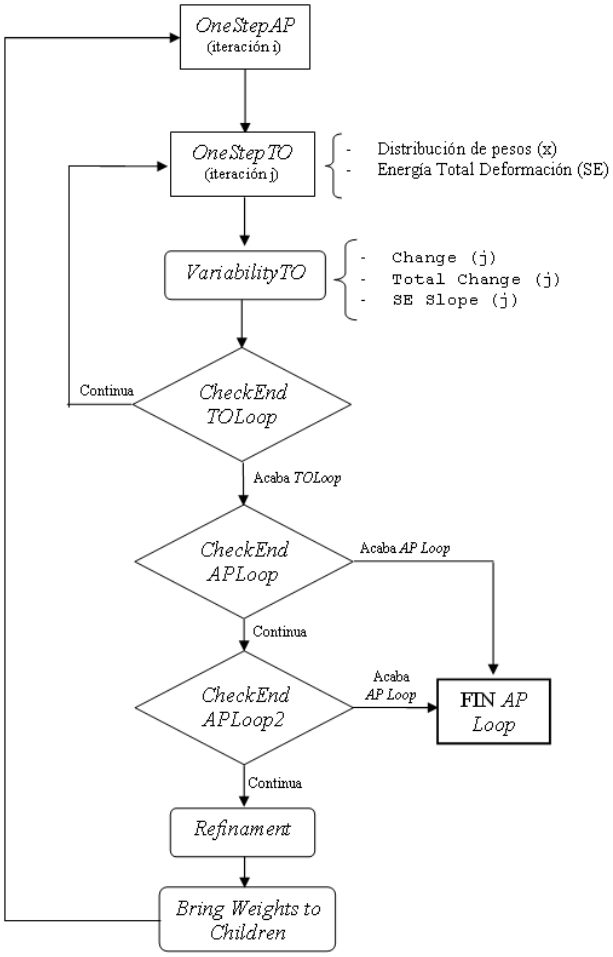


Figura 4.4 Algoritmo adaptatividad

Donde:

- *OneStepAP*: Una iteración del Proceso de Adaptatividad.
- *OneStepTO*: Una iteración del proceso de Optimización Topológica. Este paso comprende el cálculo de derivadas de energía de deformación, el filtro (o alisado) y la distribución de pesos.
- *Variability TO*: Calcula parámetros que miden la variabilidad del proceso de Optimización topológica:
    - *Change*: Máximo cambio en el peso de un elemento entre la distribución de pesos anterior y la actual.
    - *Total Change*: Es la media del cambio de peso en un elemento entre la distribución de pesos anterior y la actual.
    - *SE Slope*: Es el tanto por cien que varía la energía de deformación de todo el componente, que es SE (Strain Energy).
- *Check End TO Loop*: Determina, a raíz de los parámetros anteriores, si el proceso de Optimización Topológica ya ha convergido.
- *Check End AP Loop*: A raíz del valor de la energía de deformación encontrada en las dos últimas mallas determina si el proceso ya ha convergido o se debe seguir refinando.
- *Refinament*: Realiza el refinamiento de la malla según el criterio de refinamiento explicado anteriormente.
- *Bring Weights to Children*: Proyecta los pesos de los elementos refinados a sus respectivos hijos.
- *Check End AP Loop 2*: Al igual que "*Check End AP Loop*" determina si el proceso de adaptatividad de la malla ya ha convergido pero con otro criterio diferente. En este caso, si el refinamiento realizado subdivide una cantidad de elementos inferior a un valor determinado, entonces el proceso finaliza, ya que la solución en el siguiente paso sería demasiado similar a la actual.

Hay que aclarar que cada vez que empieza el proceso de optimización topológica con una malla diferente, lo hace con la distribución de pesos anterior. Esto se puede conseguir gracias a la proyección de los pesos de los elementos

refinados a sus hijos. De esta manera las iteraciones requeridas para converger a la solución en cada malla nueva son mínimas ya que la forma básica del componente ya está definida, solo se debe redefinir mejor las zonas refinadas.

4.1.6 Critetrio de convergencia algortimo adaptatividad

En la sección 3.1.2 se expuso el criterio de convergencia para el proceso de optimización topológica. Este se va a mantener igual. Por otra parte se precisa de un criterio de convergencia para el proceso de adaptatividad de la malla, para saber cuando se debe parar de refinar.

En este caso se comprueba la variabilidad del proceso en dos momentos diferentes del bucle. El primero de ellos después de obtener la solución del proceso de optimización topológica para una malla determinada, y se llama *Check End AP Loop* (ver esquema de la Figura 4.4) Se evalúa entonces la disminución en la energía de deformación que supone la nueva figura respecto la anterior. Si la energía ha disminuido bastante, el proceso continúa, pero si no ha disminuido mucho el proceso el bucle termina. Para evaluar esto se calcula la diferencia de energía de deformación asociada al componente de la actual malla y el de la malla anterior, que se denomina *SESlopeAP* (pendiente de la energía de deformación del proceso de adaptatividad) y se expresa en porcentaje. Si este valor está por debajo del valor *MinSESlopeAP* entonces se da por finalizado el proceso.

El otro momento de comprobación es después de realizar el refinamiento y obtener la nueva malla, la función que lo implementa se llama *Check End AP Loop2* (ver esquema Figura 4.4). Aquí se evalúa si la nueva malla obtenida es similar a la anterior, en cuyo caso la solución sería prácticamente la misma y por tanto el proceso terminaría. Si el número de elementos refinados (*APElsSub*) está por debajo de un valor determinado (*MinAPElsSub*) entonces se considera que el proceso ha convergido.

Los valores límites *MinSESlopeAP* y *MinAPElsSub* se han escogido como 0.1 y 5 respectivamente en base a pruebas experimentales.

4.1.7 Verificación con un ejemplo

Para verificar el buen funcionamiento de la aplicación de la adpatividad de la malla en optimización topológica se va analizar el caso de la viga en voladizo.

Sin adaptatividad la solución obtenida con una malla de nivel 4 era la siguiente:

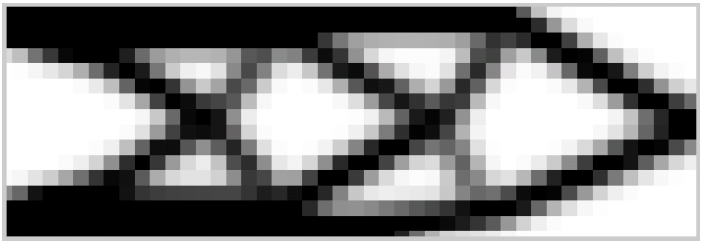


Figura 4.5 Solución viga en voladizo sin adaptatividad

Con esta distribución de pesos se obtiene una energía de deformación de $3.7726 \cdot 10^6$. Sin hacer uso de la adaptatividad, el método que se tenía antes para conseguir esta solución con mayor definición era el de *Independencia de la solución respecto el tamaño de elemento* (3.4.3). Para ello se requería refinar toda la malla y mantener el valor de r_{min} (en valor absoluto). Se muestra a continuación una malla de nivel 6 y la solución que se obtendría sin adaptatividad con este método.

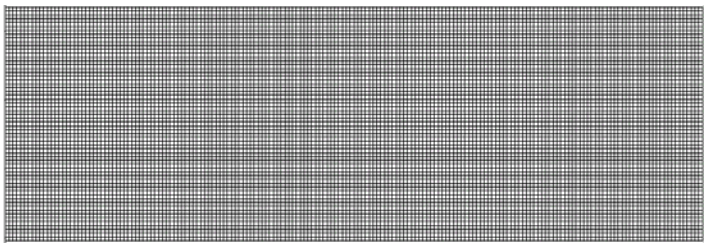


Figura 4.6 Malla regular de nivel 6 para la viga en voladizo

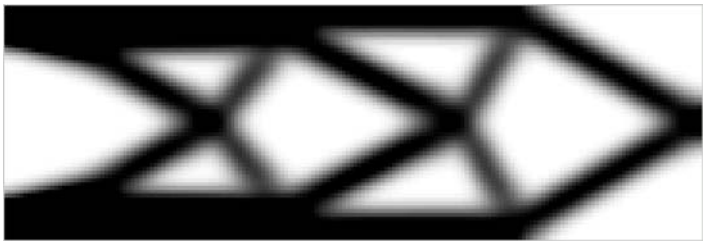


Figura 4.7 Solución malla Figura 4.6

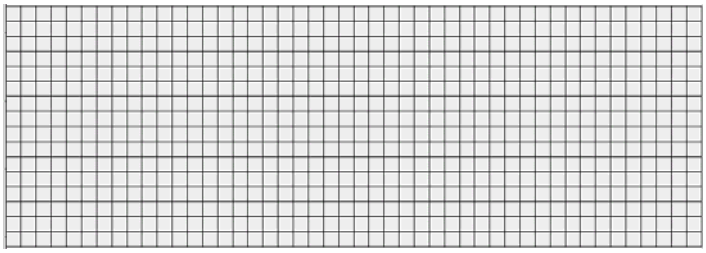
La energía de deformación asociada a esta solución es de $3.7452 \cdot 10^6$ que es únicamente un 0.72 % menor. El coste computacional requerido para obtener esta solución es muy elevado ya que hay que refinar toda la malla. Y además las zonas

de transiciónde material a no material si bien quedan menos pixeladas no quedan mejor definidas.

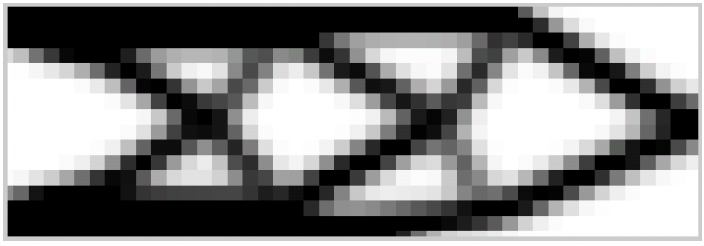
Utilizando la adaptatividad de la malla con el proceso de optimización topológica tal y como se ha explicado se obtiene el resultado que se muestra a continuación.

Se inicia el problema con una malla de nivel 4 y se van refinando elementos. Se ha escogido como mínimo tamaño de elemento el de nivel 7. El proceso se muestra a contincuación. Las siglas ER se refieren a Elementos Refinados en dicha malla.

Malla 1 (inicial)

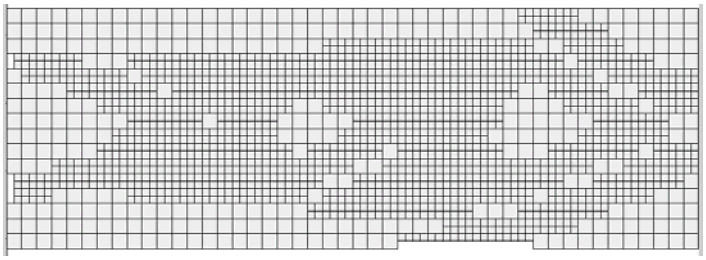


Solución malla 1



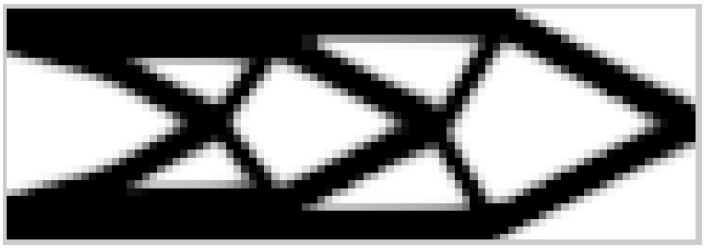
SE=3.77·10^6

Malla 2



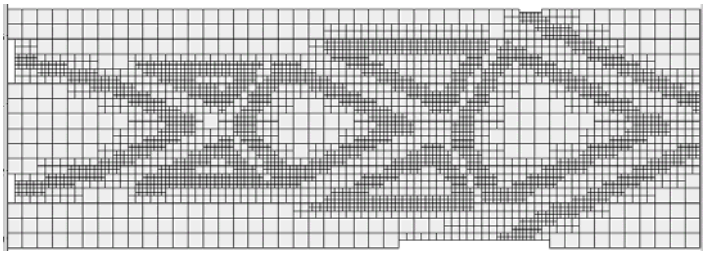
ER=389

Solución malla 2



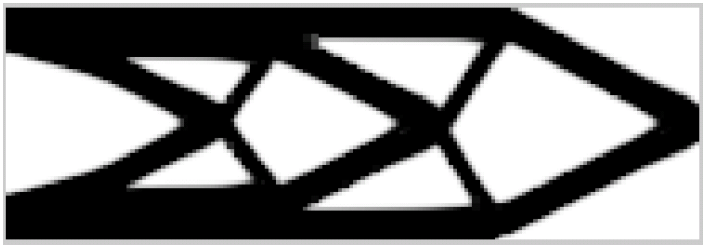
SE=3.32·10^6 ;
ΔSE=13.62%

Malla 3

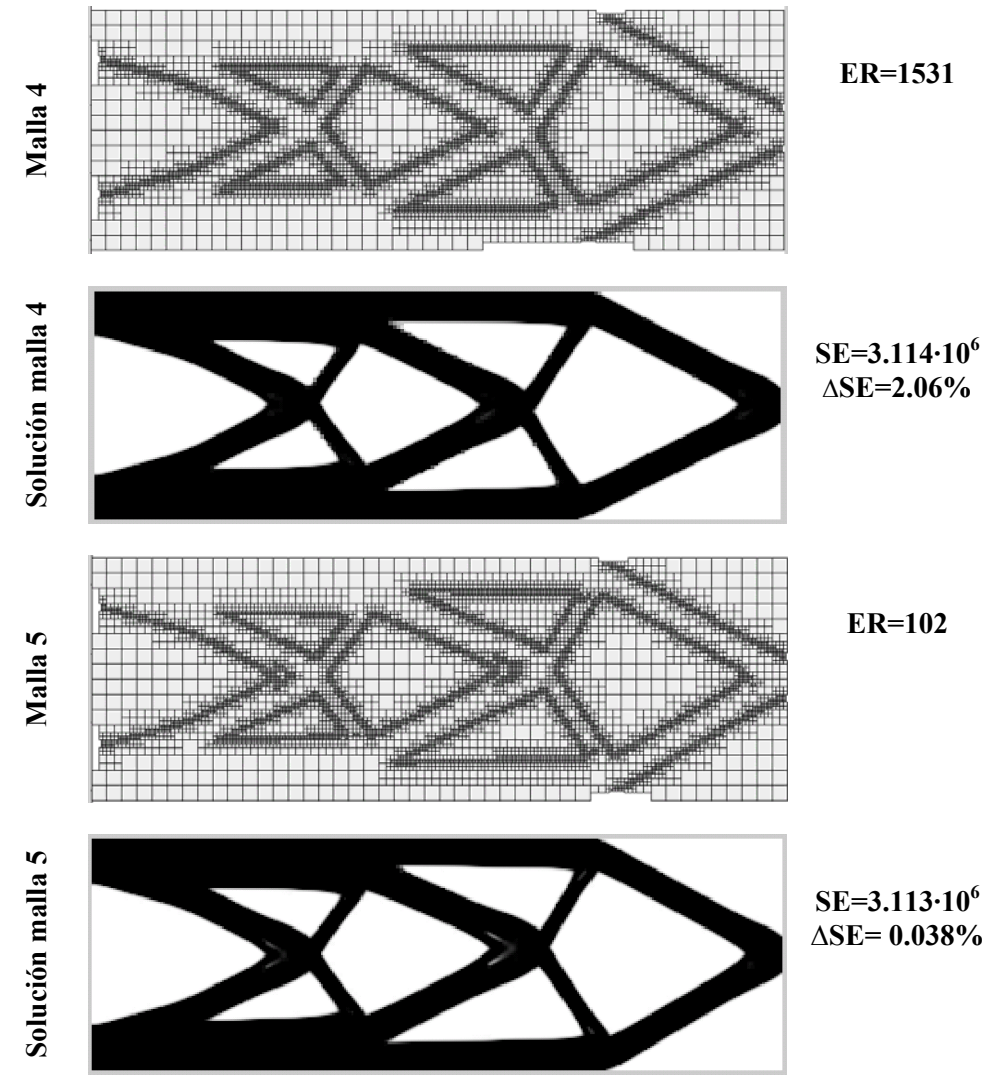


ER=720

Solución malla 3



SE= 3.18·10^6,
ΔSE = 4.45%



El proceso converge en este caso porque la disminución de la energía de deformación ya no es substancial, es menor al valor lítime (0.01). Se reduce la energía de deformación un total de un 17.5 % desde la primera iteración. La evolución se muestra a continuación.

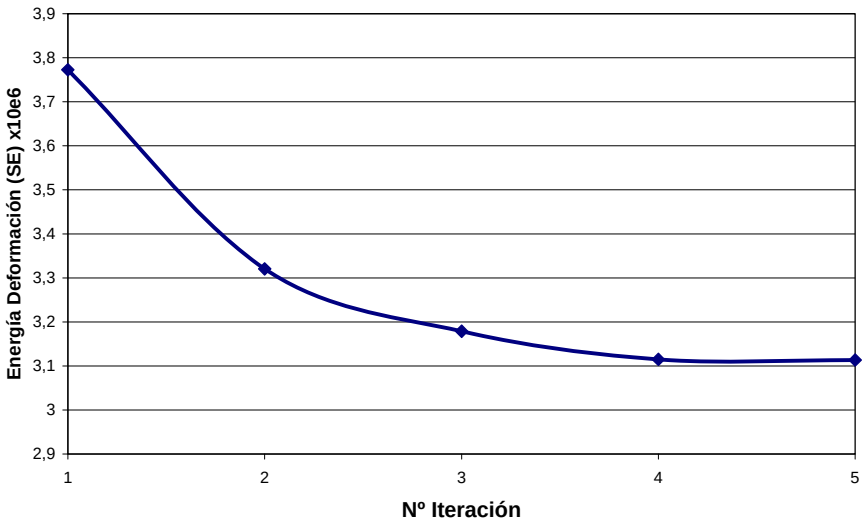


Figura 4.8 Evolución de la energía de deformación respecto los pasos iterativos

Con esto se produce una disminución total en la energía de deformación del 17.5 % y además se consigue una mejor resolución y definición de la geometría.

Cabe destacar además que, con la malla adaptada, no solo se produce una disminución de la energía de deformación respecto la malla sin adaptar, si no que también se tiene una energía de deformación menor que la solución de la malla regular con el nivel de malla más fino. La energía de deformación para la malla regular de nivel 6 era 3.74·106 (Figura 4.7), mientras que la energía de deformación para la malla adaptada desde el nivel 4 al nivel 6 es de 3.18·106, que resulta un 15 % menor.

4.2 Ejemplos aplicación

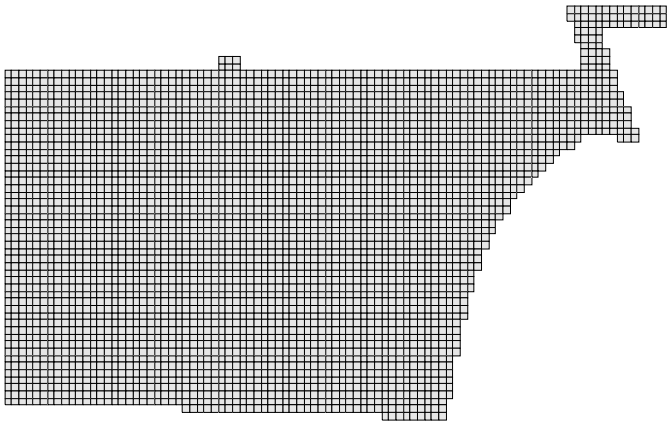
A continuación se exponen los mismos casos utilizados en el programa de optimización topológica de la sección 3.4.4, pero en este caso haciendo uso de la adaptatividad de la malla. Los resultados obtenidos van en la misma línea que el caso de la viga en voladizo presentado anteriormente.

Las goemtrías de trabajo y restricciones (desplazamientos y cargas) son las mismas que las utilizadas en el programa sin adaptatividad. En alguno de los casos

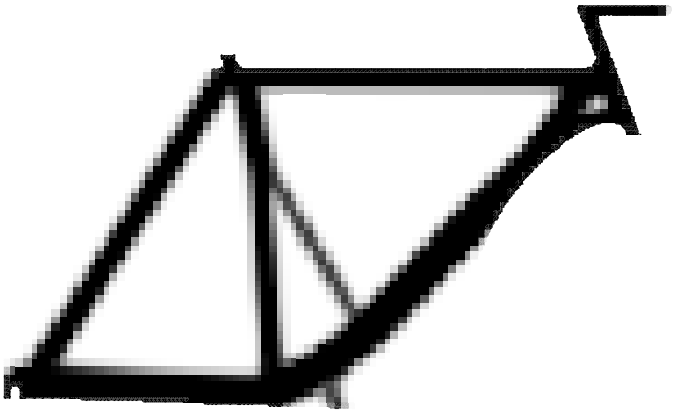
se muestra la evolución de las mallas en los pasos intermedios mientras que en otros se omiten estas y se presenta únicamente la solución final a modo de síntesis.

4.2.1 Cuadro de bicicleta

Malla 1

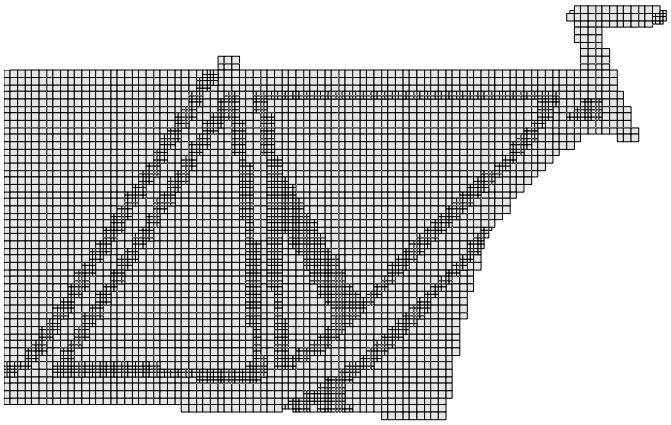


Solución Malla 1



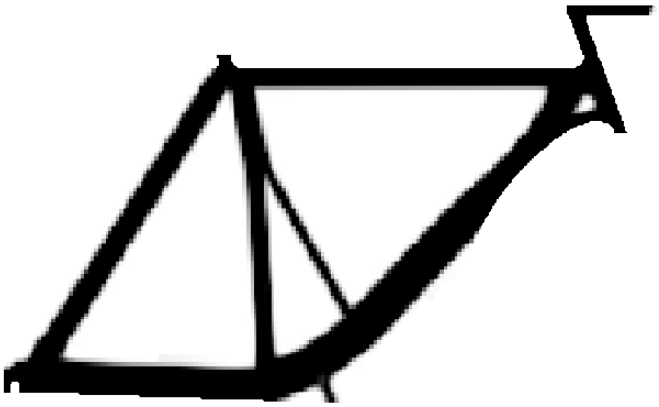
SE= 23.355

Malla 2



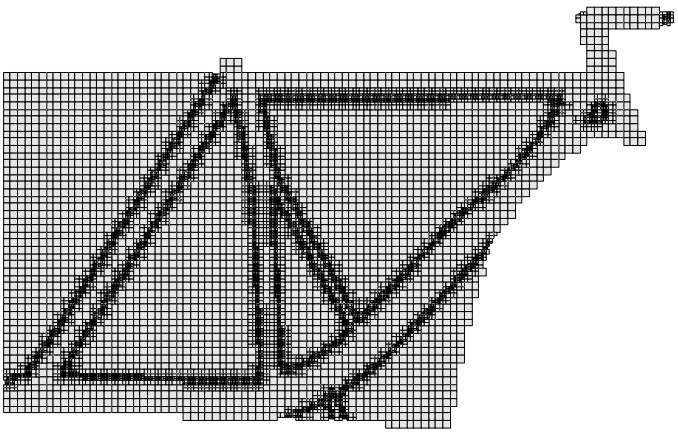
ER=703

Solución malla 2



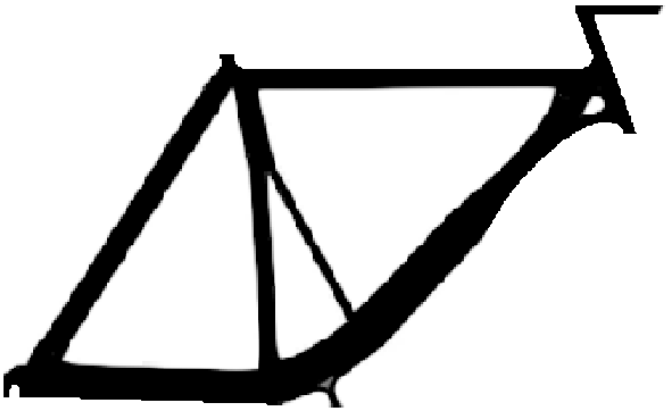
SE= 2.614
ΔSE =3.17%

Malla 3



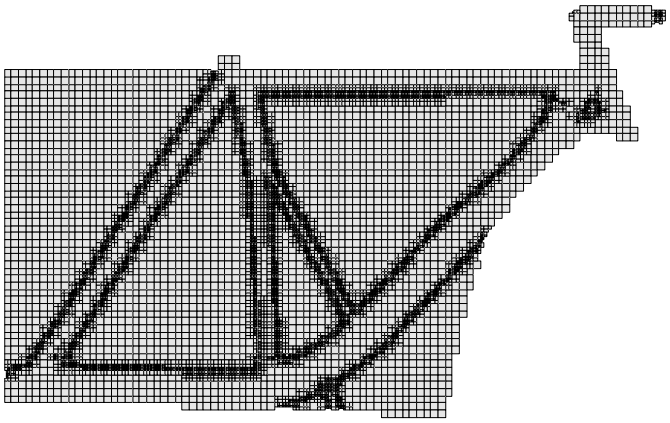
ER=1438

Solución malla 3



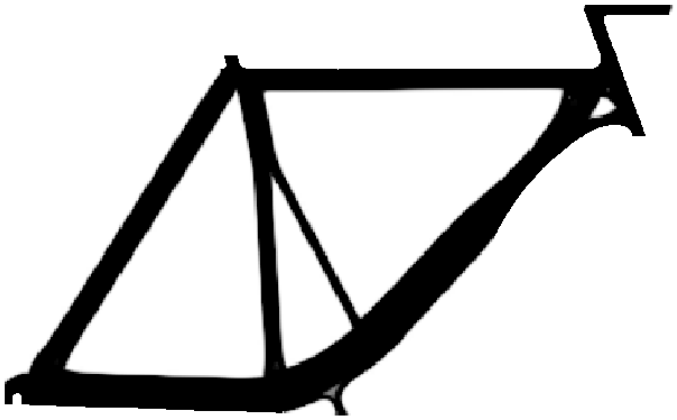
SE = 22.327
ΔSE=1.27 %

Malla 4



ER=62

Solución malla 4

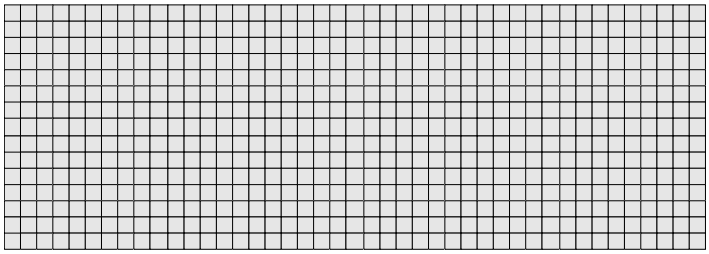


SE = 22.3247;
ΔSE=0.01 %

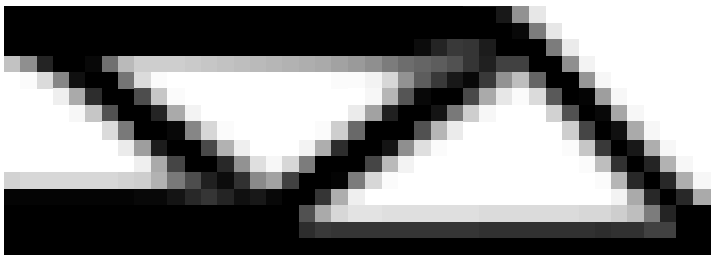
En este caso, la disminución total de la energía de deformación es de 4.4 %. Se observa además que la figura obtenida mediante la malla adaptada es exactamente la misma que la figura de la malla regular inicial pero con una mejor definición en las zonas de cambio de material a no material.

4.2.2 Viga Biapoyada

Malla 1:

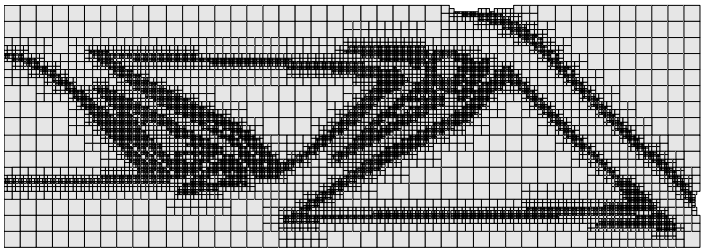


Solución malla 1:



SE = $2.81 \cdot 10^6$

Malla 8:



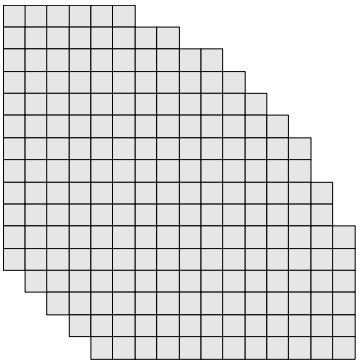
Solución malla 8:

SE = 2.427 ·10^{6}

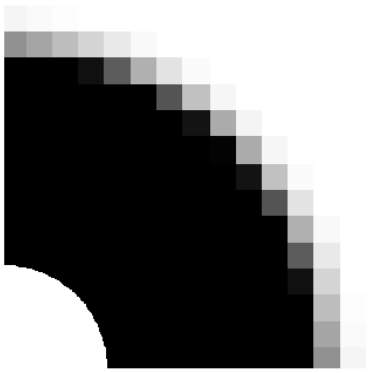
Se produce una reducción total de la energía de deformación del 13.6 %. En este caso, la solución final difiere bastante de la solución original. Este se debe a que, en este ejemplo concretamente, cuando se refina en la zona de paso de material a no material, el programa tiende a dibujar barras más finas en estas zonas, produciéndose así la bifurcación de las barras más gruesas originales.

4.2.3 Conducto sometido a presión interna

Malla 1

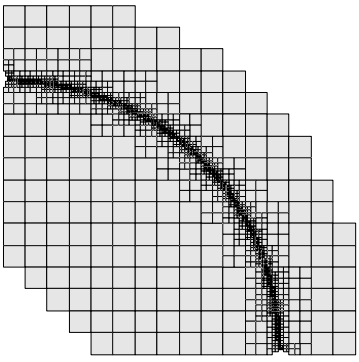


Solución malla 1



SE = 2.27 · 10^2

Malla 4



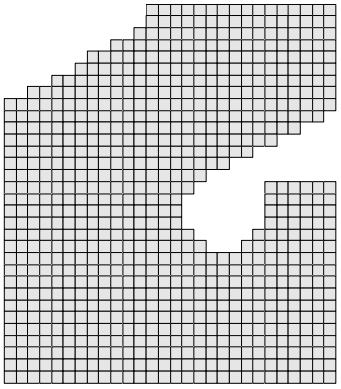
Solución malla 4

SE = 2.25 · 10^2

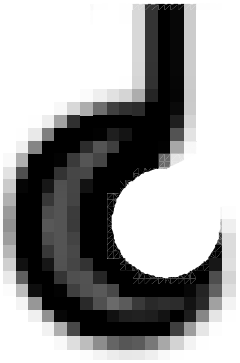
En este caso la disminución total de la energía de deformación es del 1 %. Por otra parte se aprecia como queda mejor definido el contorno exterior del tubo.

4.2.4 Gancho

Malla 1

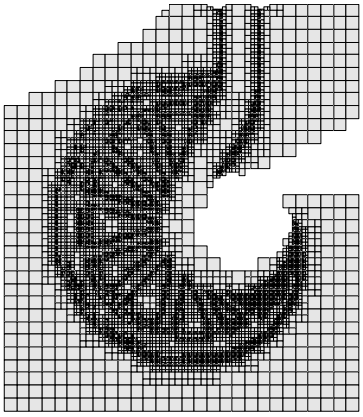


Solución malla 1



SE = 4.626 · 10^7

Malla 6



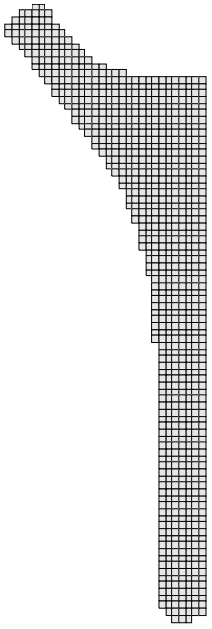
Solución malla 6

SE = 3.628 ·10^7

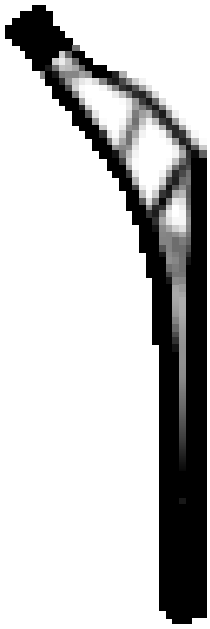
Reducción en la energía de deformación total de 21.6 %. Se han requerido 6 iteraciones.

4.2.5 Prótesis de Cadera

Malla 1

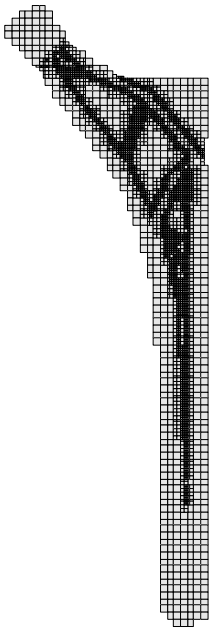


Solución malla 1



SE = 72.82 · 10^6

Malla 5



Solución malla 5

SE = 67.9 · 10^6

Reducción en la energía de deformación total 6.7 %. Se han requerido 5 iteraciones. Se observa que en la zona inferior de la prótesis, la forma del componente no quedaba bien definida con la malla inicial, y queda bien resuelta con el uso de la adaptatividad..

5 OTROS PROCEDIMIENTOS *h*-ADAPTATIVOS

5.1 Mejora de la solución mediante proyección de la información

Hasta este punto se ha conseguido primeramente desarrollar un módulo de Optimización Topológica en el software de GFEM a partir de un programa básico. Posteriormente se ha implementado la posibilidad de utilizar adaptatividad de la malla, con lo que se ha conseguido:

Definir mejor la solución obtenida inicialmente con una malla regular.

Disminuir el valor de la energía de deformación asociada al componente.

Tal y como se explicó, las mallas bastas dan lugar a soluciones con mayores espesores de barra, mientras que mallas finas originan soluciones más complejas con mayor número de barras y de menor espesor. Por ejemplo, el caso de la viga biapoyada:

Figura 5.1 a) y b) Soluciones para la viga biapoyada con malla basta y fina respectivamente

El uso de la adaptatividad que se ha hecho hasta el momento ha permitido mantener en gran medida la forma original de una solución obtenida mediante una malla regular, refinando principalmente en las zonas de transición de material a no material de manera que la solución queda mejor definida.

En esta sección se intenta dar respuesta al problema que aparece cuando se pretende conseguir un componente formado por un gran número de barras de poco espesor. Esto se consigue con mallas muy finas y por tanto conlleva un coste computacional muy elevado, ya que se tienen elementos de poco tamaño en toda la malla. Se plantea por tanto, la posibilidad de conseguir la solución que se obtendría

con una malla fina a partir de una malla más basta mediante el uso de la adaptatividad y la combinación de proyectar y subir información entre la malla basta y la fina. Es decir, conseguir la solución de la Figura 5.1 b) a partir de la malla basta.

La solución final debe ser una malla basta que esté refinada únicamente en las zonas donde aparezca material. Para lograr esto se recurre al principio basado en la obtención de una solución mejorada a través de un campo alisado de los pesos y desplazamientos, que se proyectaría en una malla más fina, seguido de un paso del proceso iterativo y posteriormente una subida de la información a la malla basta refinando en zonas donde los pesos varíen más. La idea es obtener una $\mathbf{x}_{i+1}$ de gran resolución a partir de la malla fina. Para que el cálculo de EF no se realice en la malla fina, se realiza en la malla no refinada y se proyecta en la malla fina. A continuación se explica el funcionamiento del proceso de forma más clara y detallada.

En primer lugar cabe recordar el bucle iterativo del proceso de optimización normal sin adaptatividad.

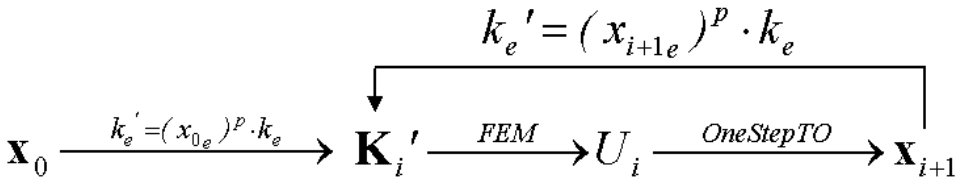


Figura 5.2 Esquema bucle iterativo proceso de optimización (sin proyección)

Es decir, a partir de unos determinados valores de desplazamientos en los nodos (U_i) se calcula la nueva distribución de los pesos (x_{i+1}). Y con estos pesos se calcularían los desplazamientos asociados a dicha distribución del material (U_{i+1}).

El esquema que representa el proceso propuesto se muestra a continuación.

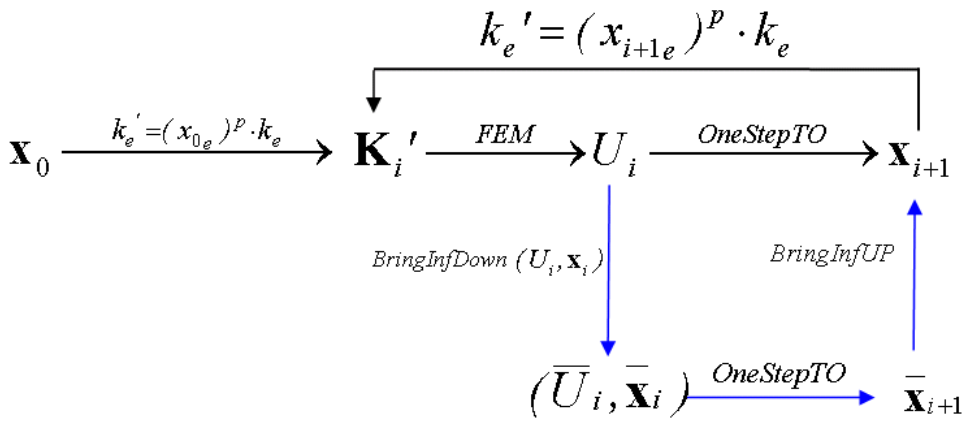


Figura 5.3 Esquema bucle iterativo proceso de optimización con proyección

- Primeramente se sigue el proceso normal durante un determinado número de iteraciones, obteniendo así una cierta distribución de material. La forma del componente en este paso es todavía difusa. Por ejemplo, para el caso de la viga bioapoyada, sería la siguiente:

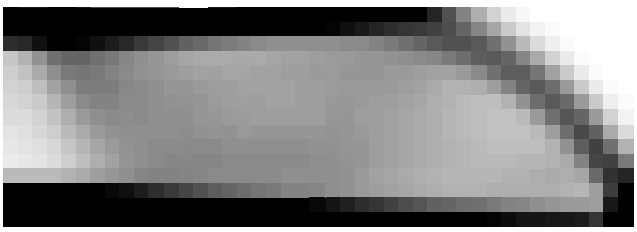


Figura 5.4 Distribución de material para el caso de la viga biapoyada en un paso intermedio

- Con esta distribución de material (x_i) en la malla regular, que da lugar a los desplazamientos (U_i), se proyectan estos dos campos hacia una malla más fina mediante la subrutina *BringInfDown*. Con esto se obtiene el campo de desplazamientos y pesos $(\overline{U}_i)$ $(\overline{x}_i)$ en la malla fina. Cabe recordar que los pesos están asociados a cada elemento. Por ello, primeramente se promedia el campo de pesos en nodos, después se proyectan los valores

nodales de desplazamientos y pesos a los nodos de la malla fina, y finalmente se calculan los valores de los pesos en los elementos de esta malla como media de los valores nodales. Siguiendo con el ejemplo mostrado anteriormente, la viga biapoyada quedaría en este paso como se muestra a continuación:

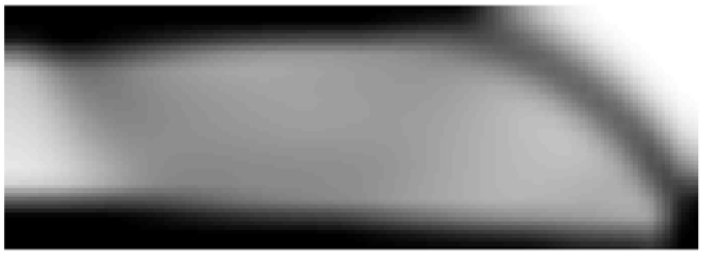


Figura 5.5 Proyección de los pesos representados en la figura Figura 5.4

- Después se trata de recalcular los nuevos pesos en la malla fina mediante un paso del proceso iterativo de optimización (*OneStepTO*). Con esto se obtiene $(\overline{x}_{i+1})$.

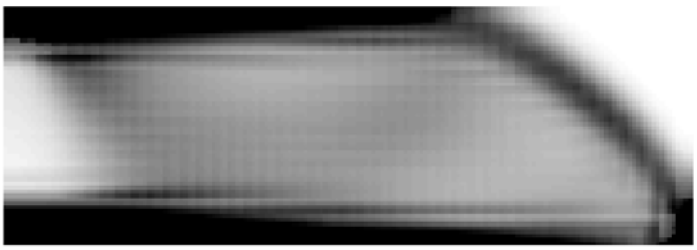


Figura 5.6 Distribución de pesos después de un paso del proceso iterativo

- Con esta nueva distribución de material se trata ahora de subir la información de los nuevos pesos a la malla basta, refinando en aquellas zonas donde existe una mayor gradiente en los pesos. El criterio seguido ha sido el de refinar un elemento en el caso de que la diferencia entre el mayor y menor peso de los hijos de un elemento sea mayor de un cierto valor. Se ha adoptada este valor como 0.08. Si el elemento se refina, los hijos toman los valores de la malla fina. Si no se refina el elemento padre toma el valor de la media de los hijos. Con esto se obtiene una segunda malla, que está adaptada, y la nueva distribución de pesos.

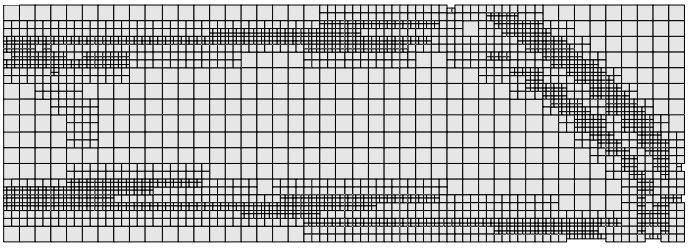


Figura 5.7 Malla adaptada según la variación de los pesos

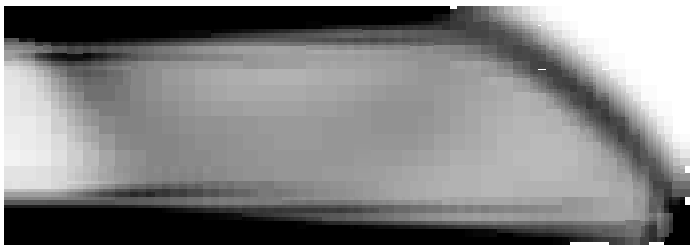


Figura 5.8 Distribución de pesos en la malla basta después de subir la información

- La idea sería continuar con el proceso de optimización normal un determinado número de iteraciones para después volver a repetir el proceso explicado.

Esta era la idea que se tenía para intentar conseguir la solución que daría una malla fina a partir de una malla basta inicial. Pero desafortunadamente, después de implementar el proceso en el programa de GFEM y analizar diferentes ejemplos, se ha llegado a la conclusión de que este método no puede llegar al objetivo mencionado. La solución final que se obtiene por ejemplo con el caso de la viga biapoyada se muestra a continuación:

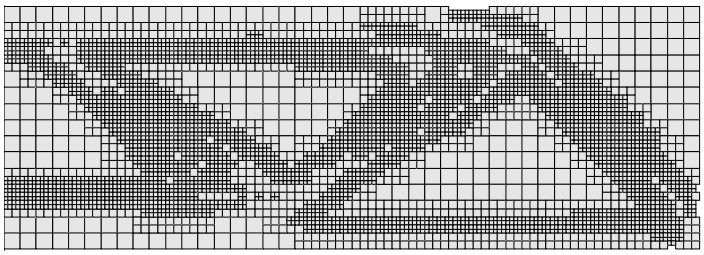


Figura 5.9 Malla adaptada obtenida al final del proceso

Figura 5.10 Solución final viga biapoyada

Que difiere del tipo de solución que se pretendía conseguir (Figura 5.11)

Figura 5.11 Solución que se pretendía conseguir

De la misma manera, se han analizado otros casos y se han obtenido el mismo tipo de resultados, por ejemplo con la viga en voladizo, cuyo resultado final del proceso se muestra a continuación:

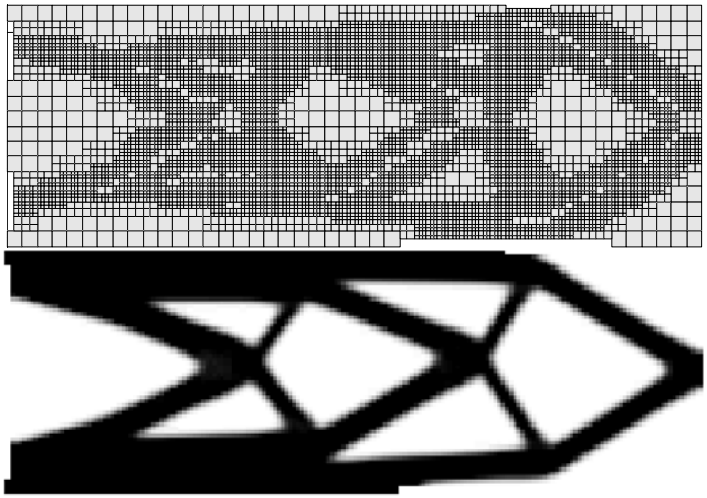


Figura 5.12 a) y b) Malla y solución para el caso de la viga en voladizo

Se concluye por tanto que la técnica propuesta no consigue los resultados esperados. Se aprecia una incoherencia en la distribución de los pesos cuando, después de haber proyectado los desplazamientos y pesos a la malla fina, se da un paso de optimización (*OneStepTO*), y la solución queda tal y como se muestra en la Figura 5.13.

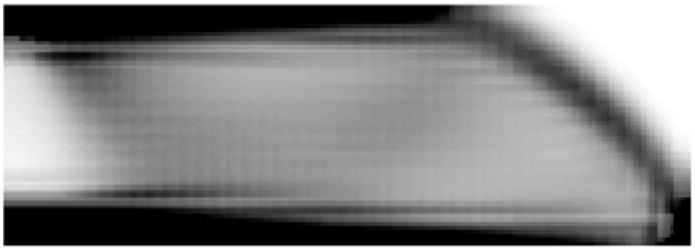


Figura 5.13 Distribución de pesos después de un paso del proceso iterativo (Figura 5.6)

Se observan discontinuidades en la distribución del material y zonas donde este queda distribuido siguiendo la estructura de la malla basta, es decir, se puede apreciar la forma de los elementos de la malla basta.

Esto debe probablemente a que existe una inconsistencia entre el campo de desplazamientos y pesos proyectados sobre la malla fina. Es decir, para una determinada distribución de material (x_i), hay asociada una distribución de

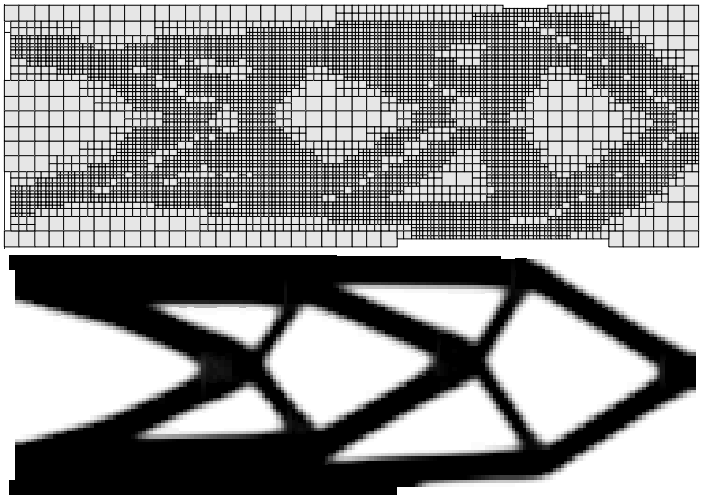


Figura 5.12 a) y b) Malla y solución para el caso de la viga en voladizo

Se concluye por tanto que la técnica propuesta no consigue los resultados esperados. Se aprecia una incoherencia en la distribución de los pesos cuando, después de haber proyectado los desplazamientos y pesos a la malla fina, se da un paso de optimización (*OneStepTO*), y la solución queda tal y como se muestra en la Figura 5.13.

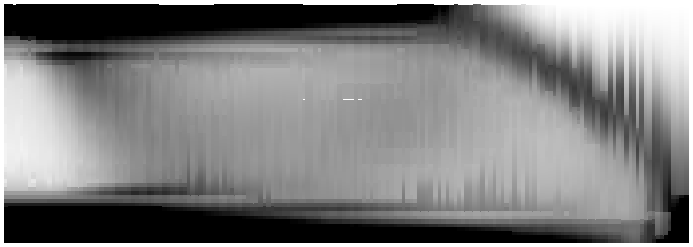


Figura 5.13 Distribución de pesos después de un paso del proceso iterativo (Figura 5.6)

Se observan discontinuidades en la distribución del material y zonas donde este queda distribuido siguiendo la estructura de la malla basta, es decir, se puede apreciar la forma de los elementos de la malla basta.

Esto debe probablemente a que existe una inconsistencia entre el campo de desplazamientos y pesos proyectados sobre la malla fina. Es decir, para una determinada distribución de material (x_i), hay asociada una distribución de

desplazamientos (U_i). Cuando estos campos se proyectan hacia la malla fina, obteniendo $(\overline{U}_i)$ $(\overline{x}_i)$, existe una inconsistencia porque el campo proyectado de los pesos daría lugar a un campo de desplazamientos que no es necesariamente $(\overline{U}_i)$, ya que la relación entre pesos y desplazamientos no es lineal. Para probar esto, se ha calculado el campo de desplazamientos asociado a $(\overline{x}_i)$ y se ha realizado un paso de optimización (*OneStepTO*). Se ha comprobado entonces que la distribución de pesos obtenida de esta forma sí es correcta, es decir, no aparecen las discontinuidades mencionadas. El problema de realizar el cálculo de desplazamientos en la malla fina es que requiere un gran coste computacional, que es justamente lo que se intentaba evitar.

5.2 Otras alternativas

Se proponen otras posibles soluciones para trabajos futuros. Con el fin de obtener la solución de la malla fina con menos coste computacional se propone como primera parte del proceso trabajar con una malla completamente refinada. De esta manera se obtiene una figura con barras tan finas como pueda dibujar la malla.

Una vez el proceso empieza a converger se trataría de reducir el número de grados de libertad en aquellas zonas que no evolucionan más. Si se analiza detenidamente un proceso de optimización topológica se observa que generalmente el componente converge en un porcentaje muy alto en la mitad del proceso como tarde. El resto del tiempo se dedica a matizar zonas puntuales. Esto se puede observar claramente en la Figura 3.18 y Figura 3.19 de la sección 3.4.1, donde se aprecia la evolución del proceso con el ejemplo del gancho. De las 79 iteraciones que dura el proceso, sólo han sido necesarias unas 30 para que el proceso convergiese de forma considerable.

La idea es por tanto eliminar aquellos grados de libertad asociados a los elementos que ya no evolucionan en la segunda parte del proceso. Se proponen dos formas de hacer esto.

1) Eliminar elementos que representen “no material” y que ya no presenten evolución en el proceso.
2) Desrefinar aquellos elementos que ya nos presenten evolución en el proceso. Esto serían tanto los elementos que representen material como lo que no, pero cumpliendo la condición de que ya no evolucionen.

Con la segunda posibilidad además de reducir el número de grados de libertad se deja abierta la opción de refinar los elementos de nuevo en caso de que esto sea necesario. Una tercera opción sería combinar las dos anteriores, de modo que se eliminaran los elementos que representen "no material" y se desrefinaran aquellos que representen material y ya no evolucionen más.

Con este método, la primera parte del proceso (convergencia del componente global) tardaría lo mismo pero en la segunda parte (optimización únicamente de ciertas zonas) se reduciría sustancialmente el coste computacional.

6 CONCLUSIONES Y FUTURAS LINEAS DE INVESTIGACIÓN

6.1 Conclusiones

En el desarrollo de esta tesis de master se ha conseguido dotar a un software de elementos finitos con mallados independientes de la geometría (GFEM) disponible en el CITV con un módulo de optimización topológica, a partir de un programa de optimización topológica básico.

Se han explicado las modificaciones en el código del software disponible que han sido necesarias para adaptar el algoritmo del programa original para considerar mallados independientes de la geometría. Posteriormente se ha verificado el correcto funcionamiento del programa a través de un estudio del efecto de los diferentes parámetros en la solución final y la realización de diversos ejemplos de aplicación. Los resultados obtenidos han sido coherentes probando así la robustez de la aplicación.

Finalmente se ha dotado al programa de la posibilidad de utilizar adaptatividad de la malla para aumentar la eficacia del proceso. Se consigue con esto definir mejor los contornos de las figuras sin necesidad de refinar toda la malla y disminuir la energía de deformación (función objetivo del algoritmo).

La integración del programa de optimización original en el software de GFEM ha permitido hacer uso de las herramientas existentes en este para crear un módulo de optimización topológica más versátil. Las ventajas que proporciona esta integración son:

- Introducción de datos mediante una interfaz gráfica.
- Generación de una geometría cualquiera mediante la interfaz gráfica.
- Posibilidad de definir una malla independientemente de la geometría de trabajo.
- Posibilidad de seleccionar el nivel de refinamiento de la malla.
- Posibilidad de calcular las tensiones internas del componente.
- Posibilidad de utilizar adaptatividad de la malla en el proceso.

Con el apoyo de estas herramientas se ha conseguido:

- La posibilidad de definir zonas dentro de la geometría de trabajo donde tiene que existir necesariamente material.
- El desarrollo de una aplicación que implementa un algoritmo capaz de encontrar la cantidad de material necesaria y su distribución para que no se sobrepase en ningún punto del componente una tensión máxima admisible.
- Obtener figuras mejor definidas, especialmente en la interfase material-no material (contorno) gracias al uso de adaptatividad de la malla. Se consigue con esto además una disminución extra de la energía de deformación.

Todas estas mejoras aplicadas al programa de optimización topológica inicial han resultado en un programa capaz de trabajar con cualquier tipo de geometría, versátil y robusto debido a las funciones implementadas. Siempre con la pauta de obtener el mínimo coste computacional posible. Los ejemplos que se han llevado cabo han servido para corroborar el buen funcionamiento del programa.

6.2 Futuras líneas de investigación

El programa queda abierto al futuro desarrollo del mismo mediante el perfeccionamiento y la implementación de nuevas funcionalidades. Así pues se pueden mencionar las siguientes líneas como punto de partida:

- Desarrollo de un programa de optimización topológica en 3D a partir del programa desarrollado en este trabajo que trabaja en dos dimensiones.
- Posibilidad de vincular un programa de optimización topológica en 3D con técnicas de fabricación aditiva del tipo laser cusing rápido (impresión en 3D). Se explica detalladamente dicha técnica y las opciones de utilizar el programa de optimización con esta en el siguiente apartado.
- Implementación de una técnica para poder extraer las geometrías obtenidas en formato CAD a modo de *post-proceso*. Esto permitiría el posible uso de las soluciones obtenidas bien en tareas de fabricación directamente, o bien en posteriores tratamientos de la geometría como por ejemplo un proceso

de optimización evolutiva para acabar de optimizar al máximo la geometría del componente.

6.2.1 Integración con técnicas de fabricación aditiva

Se entiende como Fabricación aditiva al conjunto de técnicas, tecnologías y métodos que permiten la fabricación rápida, flexible y competitiva de piezas (ya sean prototipos, modelos, moldes o productos finales) directamente a partir de información electrónica.

Al contrario que las máquinas de control numérico y las fresadoras, cuya naturaleza es sustractiva, la fabricación aditiva combina líquido, composite, o materiales en capas para crear piezas imposibles de producir por ningún otro método. La técnicas de fabricación aditiva consisten en la construcción por capas de una pieza que se realiza directamente a partir a partir de una tomografía de un modelo 3-D de CAD de la pieza mediante el aporte de material (p.e. polvos de plástico o de metal) que se sinteriza por medio de un aporte de energía externo. Las piezas obtenidas a partir de estos equipos son prototipos o piezas únicas plenamente funcionales. Este tipo de tecnologías están especialmente indicadas para aplicaciones tales como piezas para vehículos de competición, industria aeroespacial, satélites, implantes, prótesis a medida, calzado deportivo de élite, etc. En la figura siguiente se muestran algunos ejemplos de productos obtenidos mediante estas tecnologías.

Figura 6.1 Diversas piezas fabricadas mediante tecnologías de fabricación rápida

A continuación se muestra una clasificación básica de este tipo de tecnologías

- Tecnologías para obtener piezas de plástico

- o Estereolitografía (SLA)
- o Sinterizado selectivo láser (SLS)
- o Modelado por deposición de hilo fundido (FDM)
- o Fotopolimerización por proyección por máscara (DLP)

• Tecnologías para obtener piezas de metal

- o Laser Cusing. Fusión de polvo metálico por láser
- o EBM. Electron Beam Melting. Fusión de polvo metálico por chorro de electrones

En sus inicios, los años 80, estas tecnologías se llamaron tecnologías de "Rapid Prototyping" puesto que sus principales aplicaciones eran el desarrollo de prototipos estéticos y funcionales. El año 1987 la compañía 3D Systems lanza la tecnología de Estereolitografía (SLA). En 1988, la misma empresa 3D System desarrolló y comercializó la primera generación de resinas acrílicas. En 1990 la empresa alemana EOS vende sus primeros sistemas de estereolitografía. En los años 90 aparecen nuevas tecnologías de fabricación aditiva entre las que destaca la SLS, y en los años 2000 aparece la nueva generación de maquinas capaces de procesar más materiales entre los que destaca el metal y plásticos con características especiales . Esto permitió ampliar el número de aplicaciones de estas tecnologías, creando incluso utillajes rápidos, de este modo apareció el término "Rapid Tooling". Dada la enorme evolución en las tecnologías y en los materiales que son capaces de procesar las mismas fue posible obtener piezas finales directamente apareciendo el término "Rapid Manufacturing".

En la actualidad en Europa se esta adoptando el término de "Fabricación Aditiva" para llamar a estas tecnologías independientemente de su aplicación final, prototipo, utillaje o pieza final, dada la esencia de fabricación por capas de este tipo de tecnologías.

En el año 2000 habían instalados en Europa 349 tecnologías de Rapid Manufacturing y en el año 2007 ésta cifra se elevó a más de 1500 instalaciones8, lo que implica que invertir esfuerzos en estas tecnologías es viable ya que se presenta claramente una evolución ascendente.

Existen numerosos proveedores de estas tecnologías en España, empresas privadas y centros tecnológicos con el conocimiento y la capacidad de realizar servicios de Rapid Manufacturing. Entre ellos se encuentra el Instituto Tecnológico Metalmecánico (AIMME) de la Comunidad Valenciana que ha mostrado su interés en el desarrollo de herramientas de diseño eficientes orientadas a la obtención de

componentes con geometrías óptimas fabricados mediante estas tecnologías y que está dispuesto a participar activamente en dichos desarrollos aportando tanto medios técnicos como el know how relativo a la fabricación rápida.

6.2.1.1 Integración de tecnologías

La referencia [8] expone claramente que uno de los factores que ha limitado el uso de la optimización topológica ha sido la falta de tecnologías adecuadas para la fabricación de las estructuras optimizadas que proporciona este método. Muchas de las estructuras optimizadas que proporciona esta técnica no pueden ser ejecutadas mediante tecnologías de fabricación convencionales como las de fresado, torneado, electroerosión, etc., principalmente porque éstas son técnicas sustractivas con las que resulta imposible acceder a partes internas del componente dado que el material de los alrededores no puede ser eliminado. En dicha referencia, los autores muestran la potencialidad de una herramienta que integra la técnica de optimización topológica con una técnica de fabricación aditiva del tipo laser cusing en la producción de componentes para la industria aeroespacial.

La utilización de GFEM en optimización topológica resulta de gran interés dado que permite mantener el uso de mallas cartesianas, muy adecuadas para esta técnica de optimización, al tiempo que permite definir espacios de trabajo de geometría compleja sin necesidad de acudir a los generadores de malla habitualmente utilizados en el MEF. Se ha de mencionar que algunos autores10 ya han mostrado que para mejorar el rendimiento del algoritmo de optimización topológica resulta necesario utilizar mallas refinadas/adaptadas en los lugares adecuados. La Fig. 5 compara la solución obtenida en la optimización topológica de una viga utilizando mallados adaptados con los obtenidos con un mallado no adaptado con un tamaño de malla igual al mínimo usado en el mallado adaptado. Las diferencias entre las soluciones son de tan solo un 0.1% pero con un tiempo de cálculo 3 veces inferior en el caso de las mallas adaptadas

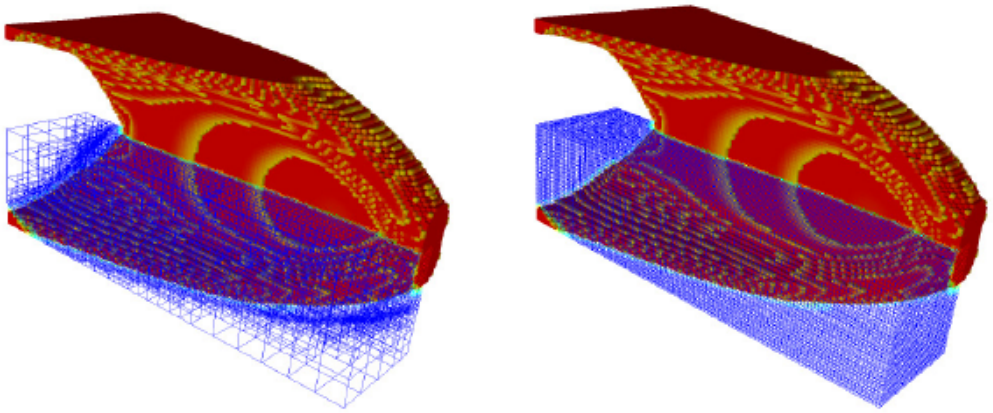


Figura 6.2 Comparación de soluciones obtenidas mediante mallados adaptados (izquierda) y mallados uniformes (derecha)

Una primera posibilidad para realizar la integración de estas técnicas de diseño y cálculo con las técnicas de fabricación rápida consiste en generar, a partir de la geometría obtenida por el algoritmo de optimización topológica, un fichero en formato *.STL que se ha convertido en el formato estándar de transmisión de datos para la industria de Rapid Manufacturing. Los ficheros STL son procesados por el software de las máquinas de Rapid Manufacturing que convierte la geometría 3D en las secciones 2D necesarias para la fabricación de la pieza.

Existe sin embargo una segunda alternativa que ha de ser estudiada. El uso de mallas cartesianas resulta muy adecuado para extraer información sobre las capas que serían posteriormente utilizadas para fabricar el componente, lo que facilitaría enormemente la integración directa de estas técnicas de diseño y cálculo con las técnicas de fabricación rápida sin la necesidad de obtener en ningún momento un modelo de CAD (en formato STL o cualquier otro).

7 Bibliografía

[1] M.P. Bendsoe, O. Sigmund. *Topology Optimization. Theory, Methods and Applications*. ISBN 3-540-42992-1 Springer, 2003.

[2] Zienkiewicz O., Taylor R, Zhu J., *The finite Element Method: its basics and fundamentals*. Sexta edición, Elsevier Butterworth Heinemann, 2000.

[3] Strouboulis, T., Copps, K., Babuška, I. *The generalizad finite elemetn method*. *Computer Methods in Applied Mechanics and Engineering*, 190(32-33):4081-4193, 2001

[4] Justo Martí Pellicer, *Implementación de un programa 2D para Resolución de problemas de elasticidad lineal mediante el Método de los Elementos Finitos utilizando mallados independientes de la geometría*. DIMM, 2008.

[5] Enrique Nadal Soriano, *Desarrollo de un programa de refinamiento h-adaptativo para el análisis mediante GFEM de problemas elásticos con mallados independientes de la geometría*. DIMM, 2009.

[6] O. Sigmund et.al., *www.topopt.dtu.dk*, TopOpt Group, Department of Solid Mechanics, Technical University of Denmark.

[7] M. Papadrakakis, B.H.V. Topping, *Innovative Computational Methods for Structural Mechanics*, Saxe-Coburg Publications, 1999.

[8] Ref. DD Gill, CJ Atwood, J Robbins, TE Voth, P Dewhurst, DG Taggart. Titanium Cholla: Lightweight, *High-Strength Structures for Aerospace Applications*. SANDIA REPORT. SAND2007-6775. 2007

8 ANEXOS

ANEXO 1. Código del programa de Optimización topológica inicial básico, disponible en la referencia 1.

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE
SIGMUND, OCTOBER 1999 %%%
function top(nelx,nely,volfrac,penal,rmin);
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
change = 1.;
% START ITERATION
while change > 0.01
    loop = loop + 1;
    xold = x;
    % FE-ANALYSIS
    [U]=FE(nelx,nely,x,penal);
    % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
    [KE] = lk;
    c = 0.;
    for ely = 1:nely
        for elx = 1:nelx
            n1 = (nely+1)*(elx-1)+ely;
            n2 = (nely+1)* elx +ely;
            Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;
                2*n2+2; 2*n1+1;2*n1+2],1);
            c = c + x(ely,elx)^penal*Ue'*KE*Ue;
            dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*
            Ue'*KE*Ue;
        end
    end
    % FILTERING OF SENSITIVITIES
    [dc] = check(nelx,nely,rmin,x,dc);
    % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
    [x] = OC(nelx,nely,x,volfrac,dc);
    % PRINT RESULTS
    change = max(max(abs(x-xold)));
    disp([' It.: ' sprintf('%4i',loop) ' Obj.: '
    sprintf('%10.4f',c) ...
```

```
    ' Vol.: ' sprintf('%6.3f',sum(sum(x))/
    (nelx*nely)) ...
        ' ch.: ' sprintf('%6.3f',change )])
    % PLOT DENSITIES
    colormap(gray); imagesc(-x); axis equal; axis
    tight; axis off;pause(1e-6);
end
%%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%%
function [xnew]=OC(nelx,nely,x,volfrac,dc)
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
    lmid = 0.5*(l2+l1);
    xnew = max(0.001,max(x-move,min(1.,min(x+move,x.
    *sqrt(-dc./lmid)))));
    if sum(sum(xnew)) - volfrac*nelx*nely > 0;
        l1 = lmid;
    else
        l2 = lmid;
    end
end
%%%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%%%
function [dcn]=check(nelx,nely,rmin,x,dc)
dcn=zeros(nely,nelx);
for i = 1:nelx
    for j = 1:nely
        sum=0.0;
        for k = max(i-round(rmin),1):
            min(i+round(rmin),nelx)
            for l = max(j-round(rmin),1):
                min(j+round(rmin), nely)
                fac = rmin-sqrt((i-k)^2+(j-l)^2);
                sum = sum+max(0,fac);
                dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)
                *dc(l,k);
            end
        end
        dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
    end
end
%%%%%%%%%% FE-ANALYSIS %%%%%%%%%%%%
function [U]=FE(nelx,nely,x,penal)
[KE] = lk;
```

```
K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*
(nely+1));
F = sparse(2*(nely+1)*(nelx+1),1); U =
sparse(2*(nely+1)*(nelx+1),1);
for ely = 1:nely
    for elx = 1:nelx
        n1 = (nely+1)*(elx-1)+ely;
        n2 = (nely+1)* elx +ely;
        edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1;
            2*n2+2;2*n1+1; 2*n1+2];
        K(edof,edof) = K(edof,edof) +
        x(ely,elx)^penal*KE;
    end
end
% DEFINE LOADSAND SUPPORTS(HALF MBB-BEAM)
F(2,1) = -1;
fixeddofs = union([1:2:2*(nely+1)],
[2*(nelx+1)*(nely+1)]);
alldofs = [1:2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);
% SOLVING
127
U(freedofs,:) = K(freedofs,freedofs) \
F(freedofs,:);
U(fixeddofs,:)= 0;
%%%%%%%%%% ELEMENT STIFFNESS MATRIX %%%%%%%%
function [KE]=lk
E = 1.;
nu = 0.3;
k=[ 1/2-nu/6 1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
    -1/4+nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8];
KE = E/(1-nu^2)*
[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
    k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
    k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
    k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
    k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
    k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
    k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
    k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
```

PROGRAMA DE OPTIMIZACIÓN TOPOLÓGICA MEDIANTE EF CON MALLADOS CARTESIANOS INDEPENDIENTES DE LA GEOMETRÍA

Manual del usuario

ÍNDICE

1 Introducción

El propósito de este Manual de Usuario es explicar el funcionamiento y manejo del módulo de Optimización Topológica integrado en el programa de Elementos Finitos con mallados cartesianos independientes de la geometría del Centro de Investigación de Tecnología de Vehículos (CITV) de la UPV. Este documento está asociado a la tesis de máster de Ing. Guillem Josep Cortés Carbonell con título *Desarrollo de un programa de Optimización de forma de componentes mecánicos mediante Optimización Topológica Adaptativa* de 2010 ,a la cual se hace referencia varias veces en este manual.

Este manual explica el funcionamiento del programa desde el punto de vista del usuario, que hace uso de este a través de una interfaz gráfica mediante el programa MATLAB. Existe además otro documento llamado “Manual del programador” donde se explica detalladamente el funcionamiento interno del programa, la estructura de variables utilizada y las funciones más importantes.

2 Ventana principal

El programa de optimización topológica es un módulo del software general de GFEM disponible en el CITV. Es decir, forma parte de este y por ello se accede desde la ventana principal.

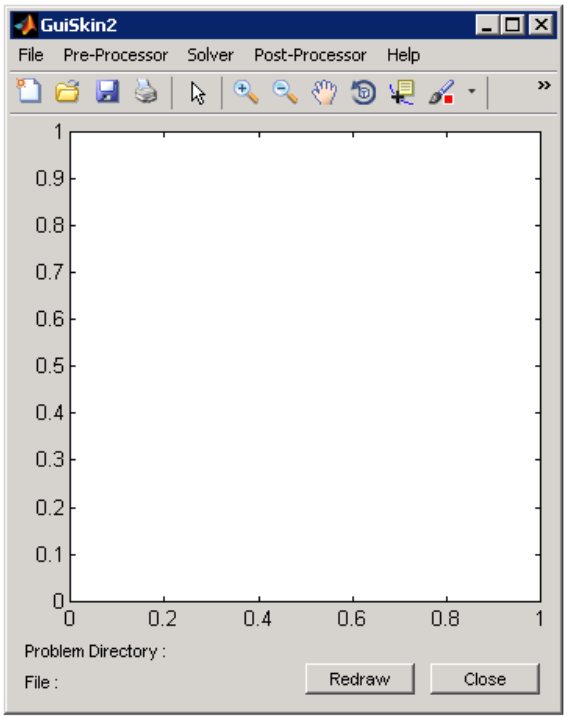


Figura 2.1 Ventana principal GuiSkin2

Mediante esta ventana se puede acceder a cada una de las partes del programa a través de la barra de menú de la parte superior. Queda constituida por:

Barra de menús: Situada en la parte superior, permite navegar a través de los diferentes submenús que dan acceso a cada una de las partes del programa mediante las cuales se podrán ir modelando, resolviendo y analizando los problemas.

- *Barra de herramientas*: Situada inmediatamente por debajo de la barra de menú, dispone de las herramientas estándar de las figuras de MATLAB con las principales funciones para el control de archivos, áreas gráficas, etc.
- *Área gráfica*: Situada en la parte central de la ventana y ocupando la mayor parte de la misma, facilita el preproceso mostrando la generación de la geometría, aplicación de restricciones y cargas.

- *Problem Directory*: En la parte inferior izquierda de la ventana, muestra el directorio de trabajo de la aplicación en el que se encuentra el fichero con la estructura de datos, archivos de cargas, restricciones, etc
- *File*: Debajo del *Problem Directory* aparece el nombre del fichero que almacena la estructura de datos con toda la información del problema actual.
- *Botón Redraw*: Situado en la parte inferior izquierda de la ventana, recalcula todos los parámetros geométricos, de restricción y de cargas aplicadas a una pieza regenerando el área gráfica con la nueva configuración.
- *Botón Close*: A la derecha de *Regenerate*, cierra la aplicación realizando una copia de seguridad en el *Current Directory* con el nombre del archivo de trabajo actual seguido de "_Bkp". En caso de no haberse definido un directorio de trabajo y un nombre de archivo, no se realizará ninguna copia. Esta definición se hace cargando un archivo ya existente o guardando los datos del problema que se esté realizando.

Mediante la barra de menús se puede acceder a las diferentes partes del programa, ésta dispone de las siguientes opciones:

- *File*. En este menú se encuentran las opciones clásicas para abrir un nuevo problema.
- *Pre-Processor*. Permite el acceso a las opciones de pre-proceso del problema como son la adquisición de coordenadas *x* e *y* de puntos a partir de un fichero de texto, definición de la geometría (*Points and Curves*), imposición de restricciones de desplazamiento en puntos y/o curvas (*Constrains*), aplicación de cargas de fuerzas puntuales, presiones en el contorno y fuerzas volumétricas (*Loads*), definición de materiales (*Material*) y espesor (*Thickness*).
- *Solver*. El menú *Solver* da acceso a las opciones de *Analysis Options*, *FEM* (Finite Element Method) y *Topology Optimization*. Dentro de este último sub-menú se accede al tipo de optimización *SIMP* (Solid Isotropic Material with Penalization), y con esto a las opciones *Parameters* y *Mesh* & *Solve*. La opción Parameters abre la ventana que controla todos los parámetros del proceso de optimización topológica. La opción *Mesh* & *Solve* sirve para controlar los parámetros de la malla y arrancar el proceso.

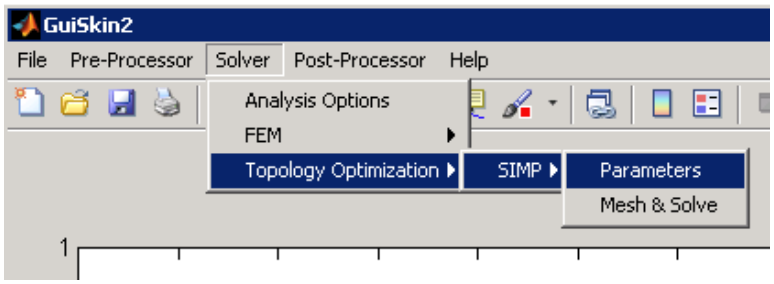


Figura 2.2 Submenú *Solver*.

- *Post-Processor*. Ofrece la posibilidad de mostrar el resultado de cálculos intermedios como el vector de fuerzas, desplazamientos y tensiones.
- *Help*. Menú de información acerca de la aplicación en el que encontramos la opción *About GuiSkin* que nos abre una ventana con información general sobre el programa y contacto con el responsable de la misma.

3 Guía Práctica

Para explicar de forma más clara el funcionamiento del programa se realizará un ejemplo a modo de tutorial explicando paso a paso el procedimiento a seguir. Se va a utilizar el caso de una viga en voladizo.

La geometría de trabajo es rectangular, de dimensiones 2 x 0.667 m. Se imponen restricciones de desplazamiento en X e Y en el empotramiento (lado izquierdo del rectángulo) y se aplica una carga vertical hacia abajo en el centro del lado derecho de valor 50 MPa. Esta fuerza está aplicada en forma de tangencial para evitar singularidades.

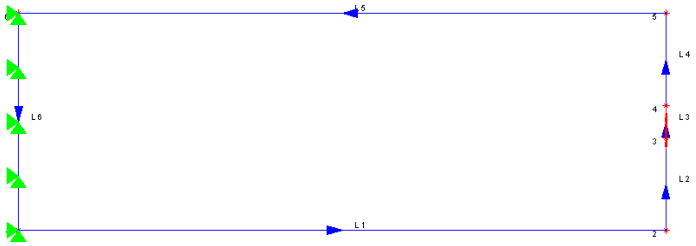


Figura 3.1 Geometría de trabajo de la viga en voladizo

Tanto la geometría como las restricciones de desplazamiento y aplicación de fuerzas se llevan a cabo desde el menú *Pre-Processor.*

Las propiedades de material se definen en *Pre-Processor* → *Material* y son:

- Módulo de Young: $2.1x10^{11}$
- Coef. Poisson: 0.3

que corresponden a las propiedades del acero.

El tipo de de problema es "Plane Stress" (tensión plana) con un espesor de 0.2 m. Esto se define en *Solver* → *Analysis Options*. Antes de continuar es recomendable guardar los cambios con *File* → *Save.*

Se va a resolver el caso de la viga en voladizo con los diferentes tipos de optimización (A y B) y finalmente se resolverá un caso de optimización A con adaptatividad de la malla.

El tipo de optimización A funciona con una fracción de volumen que es asignada por el usuario. El programa encuentra para esta cantidad de material cual es la distribución óptima de este.

Por otra parte, el tipo de optimización B funciona con una tensión máxima admisible que es asignada por el usuario. El programa encuentra cual es la cantidad de material (fracción de volumen) necesaria (y su distribución) para satisfacer esta tensión máxima en todos los puntos del componente.

3.1 Optimización tipo A

Para acceder a los parámetros de la optimización topológica, se entra por el menú *Solver* → *Topology Optimization* → *SIMP* → *Parameters* tal y como se indica en la Figura 2.2. La ventana que aparece se muestra a continuación:

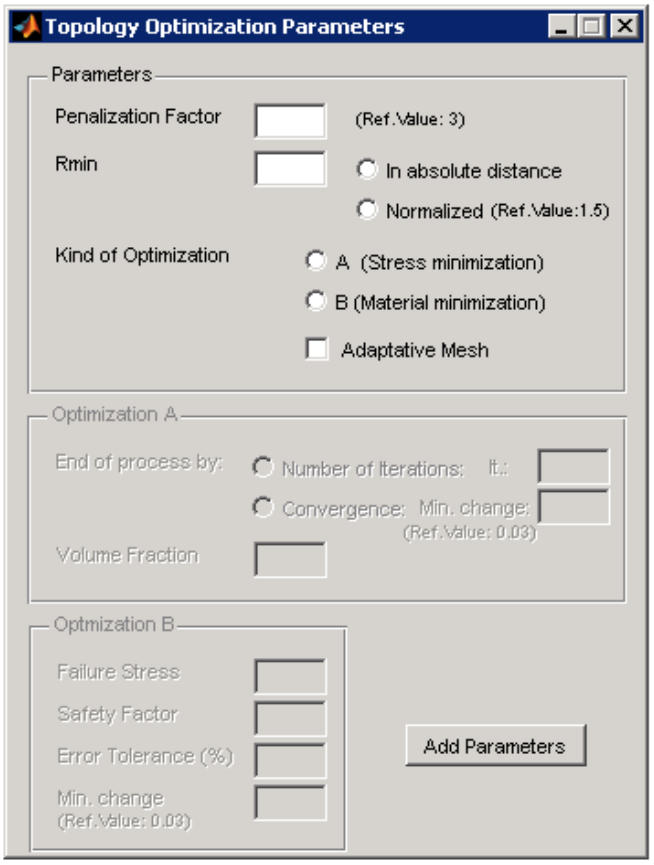


Figura 3.2 Ventana Parámetros Optimización Topológica

En primer lugar se definen los parámetros del proceso. Se escogerá como factor de penalización el valor de referencia (3). Recordar que este valor sirve para penalizar los pesos de valores intermedios (entre 0 y 1), de forma que cuanto mayor es este valor, menos pesos intermedios se observan y por tanto el material se distribuye de una forma más compacta.

Seguidamente se le asignará al *Rmin* un valor de 1.5 de tipo Normalized. Recordar que el *Rmin* es el radio de la circunferencia que se utiliza en cada elemento como área de influencia en el proceso de alisado de los pesos (filtro). Esto proceso sirve para que los pesos no salgan únicamente de valores 0 y 1. El *Rmin* se puede definir como *In absolute distance*, donde *Rmin* tomaría como dimensión la unidad de longitud utilizada en el problema, o como *Normalized*, donde *Rmin* toma como unidad la longitud del lado de elemento. Normalmente se utiliza *Rmin* = 1.5 *Normalized* (1.5 lados de elemento) ya que da generalmente soluciones de mejor calidad.

A continuación se selecciona el tipo de Optimización A (*Stress minimization*). Este tipo de optimización trabajo con fracción de volumen constante. Es decir, se define la cantidad de volumen en tanto por 1 sobre el total de la geometría de trabajo. Con esto queda asignada la cantidad de material a utilizar y el programa trata de obtener la forma más óptima con esta cantidad de material.

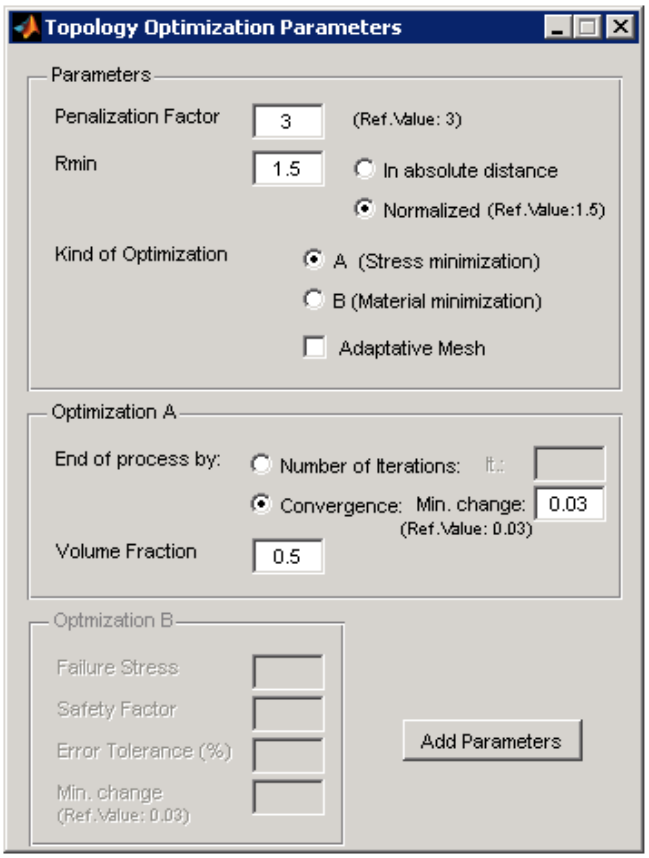


Figura 3.3 Definición parámetros Optimización Topológica A

En este punto hay que definir el criterio de finalización del proceso. La primera opción, *Number of Iterations*, da la posibilidad al usuario de definir manualmente el número exacto de iteraciones que tendrán lugar. Con la segunda opción el proceso termina por un criterio de convergencia, es decir, acaba cuando la solución ya no progresa más. En esta opción se tiene que definir el parámetro *Min. Change* (Cambio mínimo). De esta forma el proceso termina cuando el máximo cambio en el peso de todos los elementos está por debajo del valor mínimo asignado. Cabe decir, que realmente el criterio de convergencia no viene definido únicamente por este valor. También se tienen en cuenta ciertos cambios en el proceso como la evolución en la disminución de la energía de deformación. Esto se explica más

detalladamente en la sección 3.1.2 de la tesis de máster. A efectos prácticos, el valor más relevante para decidir el final del proceso es el *Min. Change*.

Se va a escoger el criterio de finalización como *Convergence* con un *Min. Change* de *0.03*, que es generalmente el valor de referencia. La fracción de volumen será de *0.5*, aunque se podría elegir otro valor a modo de prueba.

Una vez definidos los parámetros se añaden con *Add Parameters* y se cierra la ventana. Para que queden guardados los parámetros para próximas ocasiones hay que entrar en el menú *File* y guardar de nuevo.

A continuación se tienen que definir los parámetros referentes a la malla a través de *Solver* → *Topology Optimization* → *SIMP* → *Mesh & Solve*

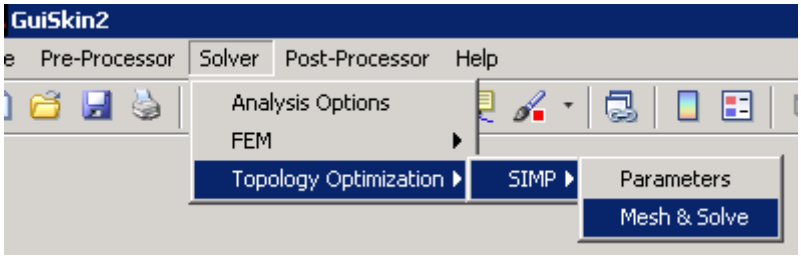


Figura 3.4 Entrada a *Mesh & Solve*

La ventana que nos aparece, con los datos que introduciremos se muestra a continuación:

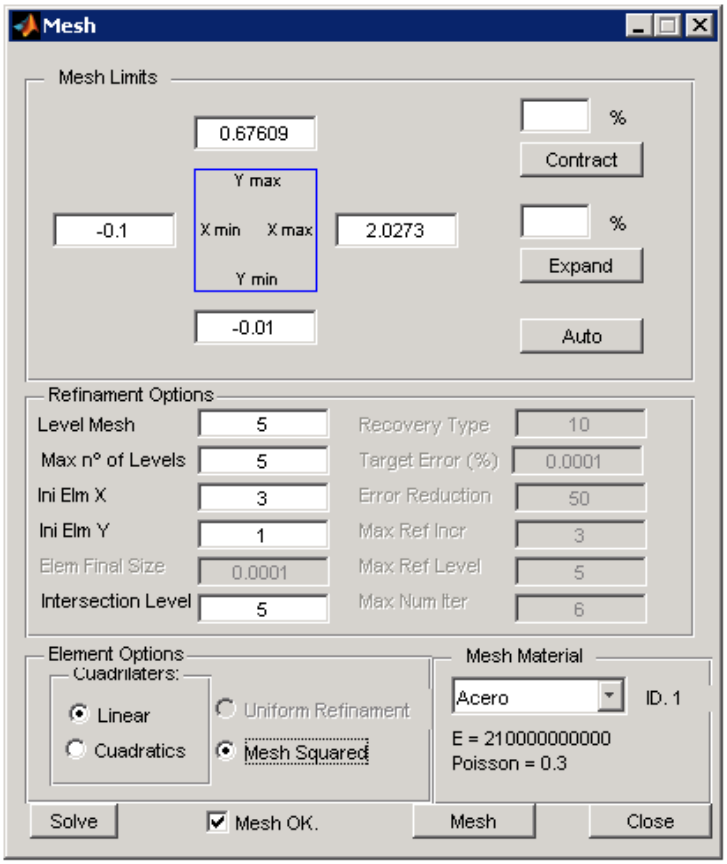


Figura 3.5 Ventana de parámetros del mallado

Primeramente se definen los límites de la malla. La opción más aconsejable es hacerlo de manera automática con el botón "Auto". Si se prefiere se puede hacer manualmente. Con la opción "Auto" el programa calcula directamente cual es la malla rectangular que se ajusta mejor a la geometría de trabajo diseñada.

Posteriormente se regulan los parámetros referentes al nivel de malla y refinamiento de esta. En *L*evel *mesh* se escoge como 5. Como se va a utilizar únicamente una malla ya que no se va a refinar esta, se definirá el máximo número de niveles (*Max nº of Levels*) como 5 y el *Intersection Level* también. Por otro lado se definen los elementos iniciales en *X* e *Y.* Como la geometría de trabajo tiene una relación de 3 a 1 se definirán los elementos iniciales como 3 en *X* y 1 en *Y*.

En el bloque *Element Options* se marcará la casilla *Mesh Squared*. Aunque el programa de optimización puede funcionar con elementos que no sean cuadrados, es recomendable que sea así ya que la cantidad de material que aporta un elemento en la dirección horizontal y vertical es la misma. Se elige la opción de elementos lineales. Aunque el programa puede funcionar también con elementos cuadráticos, ya se probó en su momento que a efectos prácticos el resultado del programa de optimización es esencialmente el mismo.

Finalmente se escoge de los materiales definidos el que se necesite para el presente análisis. En este caso se escoge el único que se ha definido, el acero.

Una vez definidos los parámetros se malla con la opción *Mesh*. Se activará en este momento la casilla de *Mesh OK* que nos indica que la malla ya esta realizada.

Para arrancar entonces el análisis solo queda pulsar el botón *Solve*.

Una vez empieza la resolución del problema, inicialmente pasan unos segundos donde el programa calcula la intersección de la geometría con la malla y otros parámetros. Después empieza el proceso iterativo de optimización. Las figuras que se van obteniendo aparecen en una nueva ventana. En la ventana principal de MATLAB aparece para cada iteración una serie de resultados que muestran la evolución del proceso. Por ejemplo, para la iteración 23:

```
It:23; MaxCh:0.0943; ACh:0.00882; SE:19587.79; SESlope%:-0.40
```

Esto es:

- `It:` número de iteración.
- `MaxCh`: Máximo cambio en el peso de todos los elementos.
- `ACh`: Significa “Average Change” y es el cambio calculado como media entre todos los elementos.
- `SE`: “Strain Energy”. Energía de deformación asociado al componente de dicha iteración.
- `SESlope`: Es el cambio de la energía de deformación en porcentaje.

Estos parámetros se utilizan para decidir cuando ha convergido el proceso. Los valores límite están asignados por defecto y a priori no son modificables por el usuario ya que es conveniente utilizar siempre los mismos. Si se quisieran cambiar estos, se puede hacer accediendo a la rutina *Situación* en las líneas 140-144.

Para el caso concreto de la viga en voladizo, después de 27 iteraciones, el proceso converge y se obtiene el siguiente resultado:

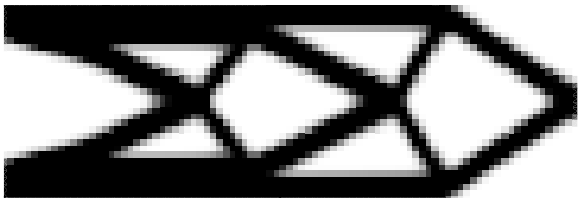


Figura 3.6 Resultado de la viga en voladizo para la Optimización A

Aparece además en la *Command Window* de MATLAB la máxima tensión que se alcanza en el componente:

```
Máxima tensión en el componente: 261108449.4884
```

Es decir, 261.1 MPa. Mediante el submenú *Post-Processor* → *Stress* se obtienen las tensiones en todo el componente.

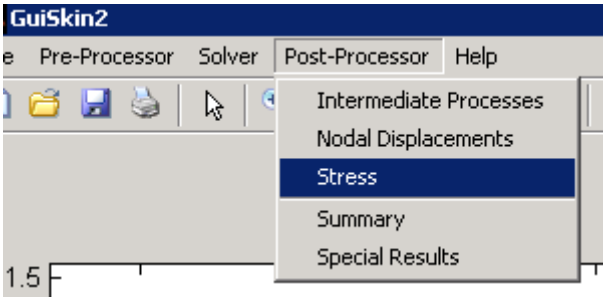


Figura 3.7 Entrada al graficado de tensiones

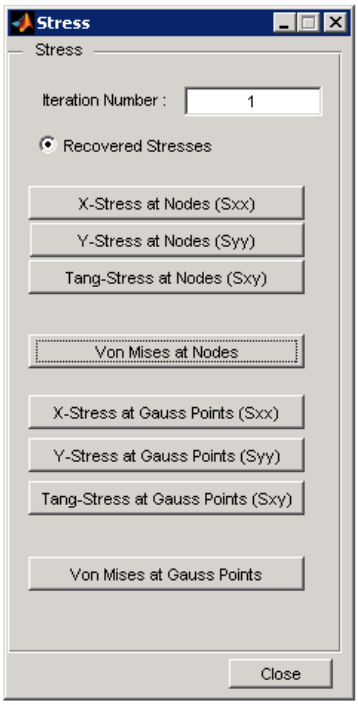


Figura 3.8 Ventana de parámetros de tensiones

La ventana que aparece permite obtener las tensiones para las diferentes mallas utilizadas. Como se ha utilizado una única malla, en la casilla *Iteration Number* pondremos 1. Se pueden obtener los diferentes tipos de tensiones tanto en nodos como en puntos de Gauss. Además se puede activar la opción de *Recovered Stresses* que nos dará un campo de tensiones de mayor precisión. Esto queda mejor explicado en la tesis de Máster de Ing. Enrique Nadal Soriano.[1]

Se pueden obtener por ejemplo las tensiones de Von Mises en nodos con la opción de *Recovered Stresses* activada. Esto nos daría el siguiente resultado:

[1] Enrique Nadal Soriano, *Desarrollo de un programa de refinamiento h-adaptativo para el análisis mediante GFEM de problemas elásticos con mallados independientes de la geometría*. DIMM, 2009.

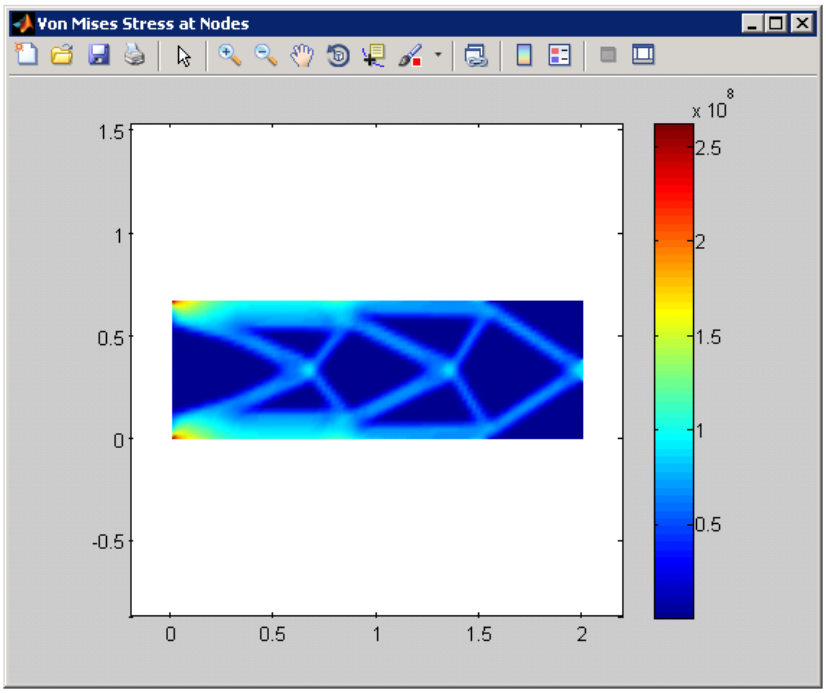


Figura 3.9 Tensiones de Von Mises en nodos con *Recovered Stresses*

Se puede observar en esta figura que hay un pico de tensiones en el empotramiento de la viga justo en la parte superior e inferior. Esto se debe a la existencia de una singularidad por un concentrador de tensiones. Si no se tienen en cuenta estos picos de tensión, las tensiones más altas en el resto del componente rondarían los 120 MPa.

Existe una aplicación en el programa para no tener en cuenta este tipo de tensiones, que es especialmente útil en el tipo de Optimización B, ya que este proceso trata de encontrar la fracción de volumen que cumpla que todos los puntos del componente estén por debajo de la tensión máxima admisible definida por el usuario. Se realiza un ejemplo de cómo utilizar esta aplicación dentro del ejemplo de optimización B.

3.1.1 Aplicación para crear geometrías fijas

Para crear una geometría de estas características se procede a crear una geometría diferente a la original, aunque en el mismo lugar de trabajo, asignándole otro identificador de contorno, por ejemplo el "2".

Para el ejemplo de la viga en voladizo se propone generar una pequeña área dentro de la viga donde tenga que ir necesariamente material. Una posibilidad sería un pequeño cuadrado, tal y como se muestra a continuación.

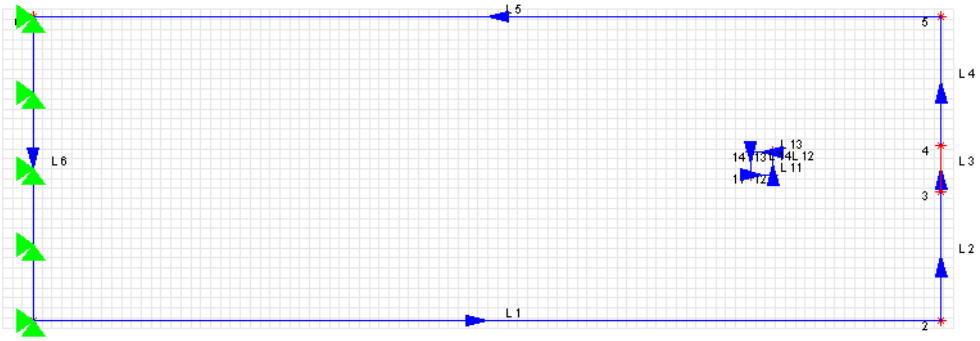


Figura 3.10 Viga en voladizo con geometría auxiliar para material fijo

Para asignarle a esta geometría la propiedad de material fijo, en el bloque de *Contours* dentro de la ventana *Points and Curves*, se le asigna al contorno 2, una ID. Material de 1 y se activa la casilla *Fixed Geo*.

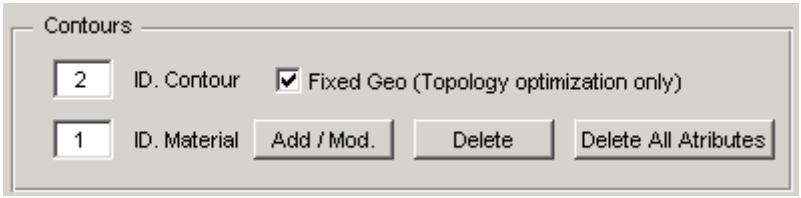


Figura 3.11 Bloque *Countours*, activación *Fixed Geo*

Una vez creada la geometría auxiliar se abre el sub-menú *Mesh & Solve* y se arranca el problema de nuevo con el botón *Solve*. La solución para el caso mostrado sería la siguiente.

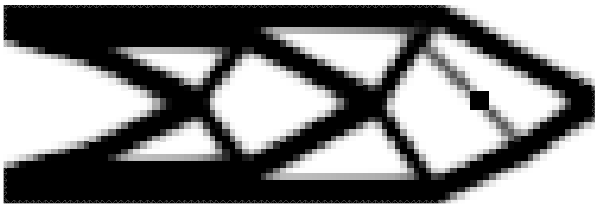


Figura 3.12 Solución con geometría fija.

3.2 Optimización tipo B

Este tipo de optimización funciona utilizando el tipo de optimización A de forma iterativa para calcular la cantidad de material necesaria (fracción de volumen) para que el componente resista una tensión máxima admisible introducida por el usuario. Para elegir este tipo de optimización se abre la ventana de *Topology Optimizaton Parameters* (Figura 3.3), y se activa la casilla de *Optimization B*. Se activa entonces el bloque correspondiente donde hay que introducir los valores de máxima tensión admisible (o tensión de fallo), coeficiente de seguridad, tolerancia en el error (indica el tanto por cien que puede variar la tensión máxima que alcanza el componente respecta la máxima admisible) y el cambio mínimo en los pesos para utilizarlo en el criterio de convergencia de cada uno de los procesos. Los valores a introducir se muestran en la Figura 3.13.

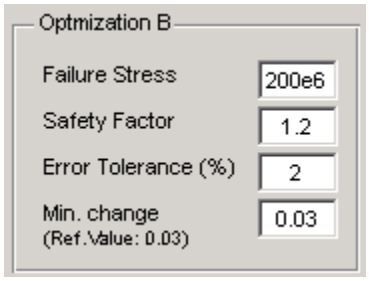


Figura 3.13 Parámetros Optimización B

Una vez introducidos se guardan con *Save Parameters*. Para este tipo de optimización es muy útil la aplicación para desestimar ciertas tensiones provocadas por singularidades, tal y como se comentó anteriormente. Se explica a continuación cómo hacer uso de esta aplicación.

3.2.1 Definición de zona en la que se desestiman tensiones

Para desestimar las tensiones que aparecen en ciertos lugares de la geometría se crean unas geometrías auxiliares con este fin. Para ello se entra en el mismo submenú que se utiliza para crear la geometría normal (*Pre-processor → Point and Curves*). En el caso de la viga en voladizo, la geometría utilizada se ha creado con un contorno que se habrá definido con el identificador "1". Ahora se trata de crear otra geometría diferente en el mismo lugar de trabajo con otro contorno, al que llamaremos "3". La nueva geometría tiene que abarcar aquella zona donde no queramos que se consideren las tensiones en los posteriores análisis. La geometría propuesta es un rectángulo a la parte izquierda de la viga, según se muestra a continuación.

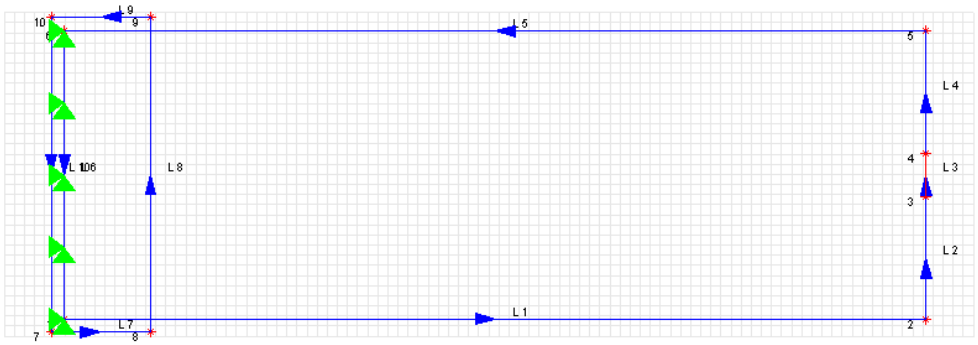


Figura 3.14 Geometría auxiliar para desestimar tensiones.

Hay que tener cuidado con la malla cuando se modifique o añada alguna geometría en el lugar de trabajo. Siempre hay que rehacer la malla con la opción *Auto* en la venta de *Mesh & Solve*.

En la ventana *Points and Curves*, aparece en la parte inferior un bloque dedicado a *Countours* (Figura 3.15). Aquí se define si el contorno es normal, asignándolo material "1" o si el contorno es de geometría auxiliar para desestimar tensiones, asignándole material 2. En nuestro caso, y continuando con el ejemplo anterior, asignaremos al contorno 3 el ID. Material 2 (Figura 3.15). La opción de *Fixed Geo* sirve para crear contornos auxiliares que permitan fijar material en una zona concreta, de forma que el programa mantiene el material, optimizando únicamente el resto de la geometría. Se ve la aplicación de esto en el siguiente apartado.

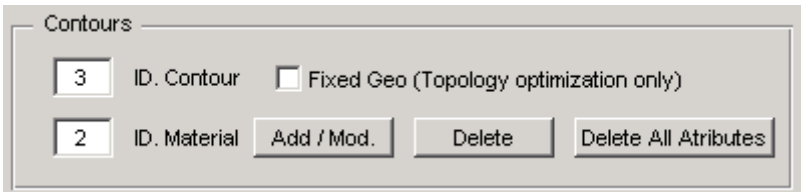


Figura 3.15 Bloque *Countours* en la ventana de *Points and Curves*

Una vez definida la geometría auxiliar, se arranca el proceso, como siempre pulsando el botón *Solve* dentro de la ventana *Mesh & Solve*.

Durante el proceso se va imprimiendo en pantalla (en el *Command Window*) para cada cálculo (proceso de optimización A) la fracción de volumen utilizada en dicha iteración, los parámetros de la evolución de dicho proceso y la tensión máxima alcanzada.

Para el caso en cuestión han hecho falta 7 iteraciones. La tensión máxima admisible del problema era de 166.67 MPa (200 MPa con coeficiente de seguridad de 1.2). La fracción de volumen calculada que cumple con esta condición es de 0.333, y la tensión máxima encontrada en el componente de 167 MPa. La solución obtenida se muestra a continuación:

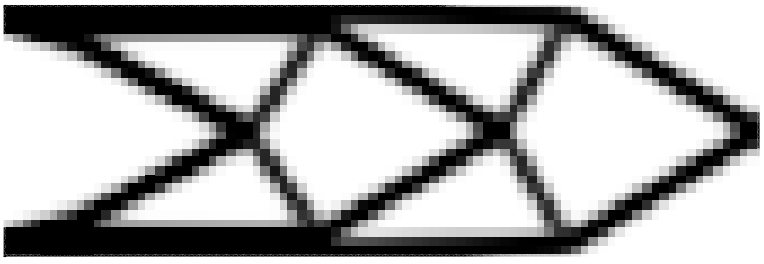


Figura 3.16 Resultado para la optimización tipo B

3.3 Optimización con adaptatividad de la malla

Si se activa la función de adaptatividad el programa trabaja con mallas refinadas. Se inicia el proceso con una malla regular y cuando este converge se refina la malla en aquellas zonas de transición de material a no material. Para activar esta función solo hay que seleccionar la opción *Adaptative Mesh* (Figura 3.3). La función de adaptatividad se puede activar tanto para el tipo de optimización A como B. Con el tipo de optimización B, primero se sigue el proceso iterativo hasta encontrar la fracción de volumen que cumple los requisitos de la tensión última de fallo. Los procesos de optimización topológica que se realizan para llegar hasta este punto funcionan sin adaptatividad. Una vez obtenido el componente optimizado para la fracción de volumen final, se inicia el proceso de adaptatividad de la malla.

Para el caso que vamos a analizar (viga en voladizo) escogeremos el tipo de optimización A. Los parámetros de la optimización serán los mismos que los definidos en la Figura 3.3.

Los parámetros del criterio de convergencia del proceso de adaptatividad de la malla no son a priori modificables por el usuario a través de la interfaz gráfica, porque son válidos para todos los casos en general. Si se quieren modificar se encuentran en `Situacion` en las líneas 148-152. Son los parámetros `OptTop.VrbParam.SESlopeAP y OptTop.VrbParam.ElsSubAP.`

En el proceso de refinamiento, cada elemento solo se puede refinar en un nivel cada vez.. El nivel máximo de refinamiento es el que se define en la ventana *Mesh & Solve* en las casillas *Max nº of Levels* y *Intersection Level* (Figura 3.5). El nivel inicial de la malla es de la casilla *Level Mesh*. Para el caso que nos ocupa se insertarán los valores que se muestran a continuación

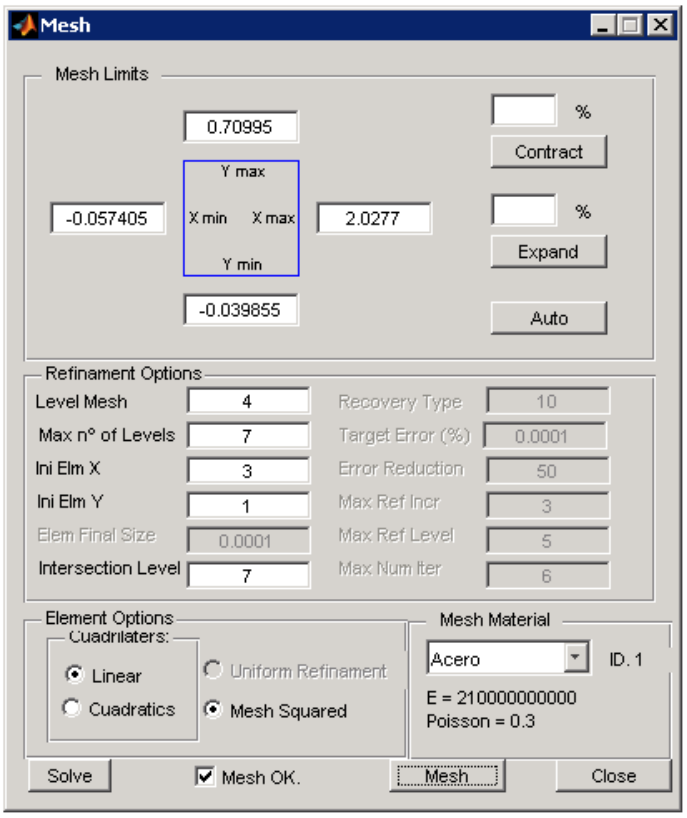


Figura 3.17 Parámetros de la malla para el proceso de adaptatividad

Una vez hecho se crean las mallas con el botón *Mesh* y se inicia el proceso con el botón *Solve*. A medida que transcurre el proceso se van mostrando las diferentes mallas que se utilizan y las respectivas soluciones de cada una de estas. La solución se muestra a continuación. Si se desea ver las imágenes de todo el proceso se pueden encontrar en la sección 4.1.7 de la tesina de máster.

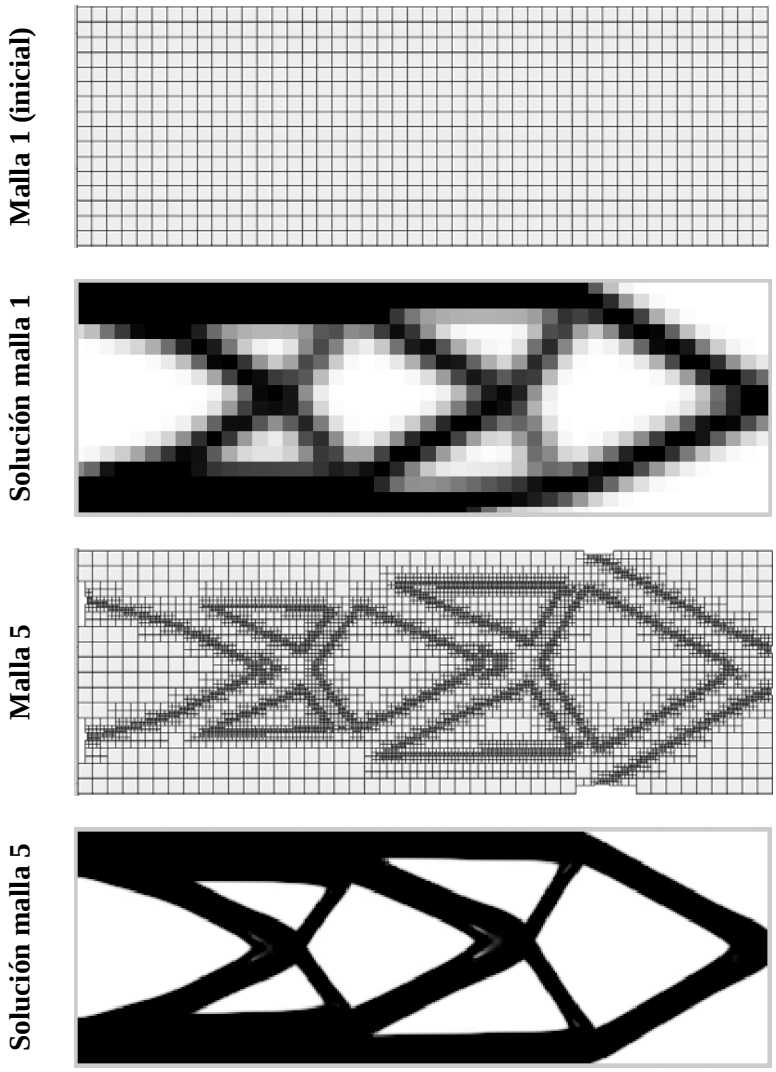


Figura 3.18 Solución proceso optimización con adaptatividad

PROGRAMA DE OPTIMIZACIÓN TOPOLÓGICA MEDIANTE EF CON MALLADOS CARTESIANOS INDEPENDIENTES DE LA GEOMETRÍA

Manual del programador

ÍNDICE

1 Introducción

El manual del programador tiene por objeto detallar la estructura y organización de de la aplicación como extensión del manual del usuario. Se explica el significado de las principales variables utilizadas y el funcionamiento de las principales funciones involucradas en el programa.

En este manual se hace referencia únicamente a la parte correspondiente del proceso de optimización topológica. Para información referente al resto del programa de GFEM se puede consultar el Manual del Programador presente en el PFC de Don. Justo Martí y Pellicer[1].

Los parámetros asociados al programa de optimización topológica se definen a partir de menú de la ventana principal del *GuiSkin2* a través de *Solver → Topology Optimization → SIMP → Parameters.* Las características de la ventana *Topology Optimization Parameters* que aparece entonces se guardan en un fichero tipo **.fig*, que se llama `TopOpt_Parameters.fig` y se guarda en *\Preproc\gui\subgui*. Por otra parte, el código vinculado a esta ventana se guarda en un archivo tipo **.m* con el mismo nombre que el anterior.

La edición de los archivos **.fig* debe hacerse imperativamente desde la aplicación GUIDE mientras que los de tipo **.m* pueden ser editados con el editor de MATLAB o con un editor de textos convencional, teniendo siempre presente que la modificación del texto del archivo **.m* puede causar daños irreparables sobre el resto de la aplicación.

En los siguientes apartados se presenta primeramente la estructura de variables utilizada, con una pequeña explicación de su función en el programa y el valor inicial, y posteriormente una explicación de las principales funciones y su papel en el programa.

[1] Justo Martí Pellicer, *Implementación de un programa 2D para Resolución de problemas de elasticidad lineal mediante el Método de los Elementos Finitos utilizando mallados independientes de la geometría*. DIMM, 2008

2 Estructura de variables

La estructura es inicializada con la subrutina *DataStruct.m* y los valores adoptados son los reflejados en las tablas de los puntos siguientes. La mayor parte de la información referente al proceso de optimización topológica se almacena en la variable global `OptTop`. Parte de esta información son parámetros del proceso que introduce el usuario a través de la interfaz gráfica. Si es este el caso, se almacena inicialmente en la variable global `FEMGui.OptTop`, y posteriormente se guarda el valor en la variable `OptTop`.

A continuación se exponen las variables involucradas en el proceso de optimización topológica. Primero se muestran las que se recogen desde la interfaz gráfica, y por tanto se guardan en primera instancia en `FEMGui.OptTop` y seguidamente se muestran todas las demás. Se incluye además una pequeña explicación de la estructura que mantiene cada variable y su significado.

Estructura	**Descripción**
`OptTop`	Raíz de la estructura
`.Penal`[2]	Variable que almacena el parámetro de penalización utilizado para calcular las matrices de rigidez modificadas.
`.rmin`	Variable que almacena la variable r_{min} que se utiliza en la sub-función `check` para realizar el filtro (alisado) de los pesos.
`.typermin`	Variable que indica el tipo de unidad utilizada para definir r_{rmin} . Con valor 1 r_{rmin} se mide con las unidades de longitud utilizadas en el problema (distancia absoluta). Con valor 2, la unidad de longitud utilizada es el lado de elemento (distancia normalizada)

[2] En adelante `.Propiedad` se utilizara para indicar `Estructura.Propiedad`, donde la Estructura esta definida anteriormente

Estructura	**Descripción**
`.StressAplicattion`	Variable lógica (0/1) que indica si está activada el tipo de optimización B. Si toma valor 0 significa que está activado el tipo de Optimización A. Si toma valor 1 está activado el tipo de Optimización B (*StressAplicattion*)
`.Adaptative_TO`	Variable lógica (0/1) que indica si la opción de adaptatividad de la malla está activada.
`.ChangeOn`	Variable lógica (0/1) que indica el criterio de final de proceso de optimización topológica utilizado. Si toma valor 0 el proceso termina según el número de iteraciones definido por el usuario. Si toma valor 1 el proceso termina cuando este converge según un criterio de convergencia más complejo que involucra a la variable `OptTop.Change.`
`.Iteraciones`	Variable que almacena el número de iteraciones que tendrán lugar en el proceso de optimización si es el criterio de final de proceso utilizado (`OptTop.ChangeOn`=0)
`.Change`	Variable que almacena el valor más determinante en el criterio de convergencia si `OptTop.ChangeOn`=1. Básicamente el proceso converge si el máximo cambio entre los pesos de todos los elementos en una actualización es inferior a `OptTop.Change`.
`.VolFrac`	Variable que almacena la fracción de volumen utilizada en el proceso de optimización si esta es de tipo A.
`.LastStress`	Variable que almacena la tensión última de fallo considerada para el tipo de Optimización B.
`.SC`	Variable que almacena el coeficiente de seguridad referente a la tensión última de fallo, que es considerado en el tipo de Optimización B.

Estructura	Descripción
`.tol`	Variable que almacena la tolerancia (o nivel de error) que se considera en el tipo de Optimización B. Se refiere a la diferencia porcentual máxima permitida entre la tensión máxima encontrada en el componente y la tensión máxima admisible (tensión última de fallo/Coef. Seguridad)

Las variables mostradas en la tabla anterior son parámetros que se recogen desde la interfaz gráfica. A continuación se muestran el resto de variables utilizadas en el proceso de optimización topológica. Son variables que se calculan a partir de estas últimas o parámetros que no son modificables a priori por el usuario.

Estructura	Descripción
`.NivelMalla`	Variable que almacena el nivel de malla utilizado en el proceso de optimización topológica. Es igual a la variable Param.MeshSelectedLevels. Para el caso de optimización con adaptatividad se refiere al nivel de malla inicial.
`.KindOfTO`	Variable tipo string que indica el tipo de proceso que se está tratando. Para todos los explicados en la tesis de máster (Optimización tipo A y B con/sin adaptatividad) toma los caracteres '`Topology_Optimization`'. En el caso de realizar una optimización como la explicada en la sección 5.1 de la tesis de máster toma los caracteres '`APBPI`' (Adaptative Process By Projecting Information).
`.check4On`	Variable lógica (0/1) que indica si está activada la opción para realizar el alisado de pesos mediante el tipo de filtro `check4`, adecuado para evitar problemas de capacidad debido a un elevado número de elementos, ya que trabajo con una matriz virtual.

Estructura	Descripción
`.VrbParam`	Variable raíz que almacena parámetros utilizados en el criterio de convergencia del proceso de optimización topológica y el criterio de convergencia del bucle de adaptatividad de la malla.
`.VrbParam.SESlope` `.VrbParam.SESlope2`	(SESlope: Strain Energy Slope) Variables que almacenan valores utilizados en el criterio de convergencia del proceso de optimización. Son valores límite que se comparan con el decremento porcentual de la energía de deformación.
`.VrbParam.Change2` `.VrbParam.Change3`	Variables que almacenan valores utilizados en el criterio de convergencia del proceso de optimización. Son valores límite que se comparan con el máximo cambio en los pesos de todos los elementos en una actualización los de pesos.
`.VrbParam.TRC`	(TRC: Total Relative Change) Variable que almacena un valor utilizado en el criterio de convergencia del proceso de optimización. Es un valor límite que se compara con el cambio medio de los pesos en todos los elementos en una actualización de los pesos.
`.VrbParam.SESlopeAP`	(SESlopeAP: Strain Energy Slope at Adaptative Process) Variable que almacena un valor utilizado en el criterio de convergencia del proceso de adaptatividad de la malla. Es un valor límite que se compara con el decremento porcentual de la energía de deformación que hay entre las soluciones finales de dos mallas consecutivas.
`.VrbParam.ElsSubAP`	(ElsSub: Elements Subdivided) Variable que almacena un valor utilizado en el criterio de convergencia del proceso de adaptatividad de la malla. Es un valor límite que se compara con el número de elementos refinados al obtener una nueva malla.

Estructura	**Descripción**
`.Vrb(iMesh)`	Variable raíz que almacena datos del proceso utilizados en el criterio de convergencia del proceso de optimización topológica y el criterio de convergencia del bucle de adaptatividad de la malla.
`.Vrb(iMesh).c`	Vector que almacena la energía de deformación de cada una de las iteraciones del proceso de optimización topológica que se realiza en la malla `iMesh.`
`.Vrb(iMesh).change`	Vector que almacena el cambio máximo en los pesos de cada una de las iteraciones del proceso de optimización topológica que se realiza en la malla `iMesh.`
`.Vrb(iMesh)` `.TotalRelChange`	(TotalRelChange: Total Relative Change). Vector que almacena el cambio medio en los pesos de cada una de las iteraciones del proceso de optimización topológica que se realiza en la malla `iMesh.`
`.Vrb(iMesh)` `.SESlopex100`	(SESlopex100:Strain Energy Slope in percentage) Vector que almacena el decremento porcentual de la energía de deformación de cada una de las iteraciones del proceso de optimización topológica que se realiza en la malla `iMesh.`
`.Vrb(iMesh).APElsSub`	Variable que almacena el número de elementos que se subdividen al pasar de la malla `iMesh` a la `iMesh` +1.
`.Vrb(iMesh).Weights`	Vector de tamaño el número de elementos totales (de todos los niveles) que almacena, en la posición de cada elemento, el peso asignado en la solución final del proceso de optimización topológica que se realiza en la malla `iMesh`.
`.FixedElm`	Vector que almacena aquellos elementos (sus índices) que están dentro de una geometría fija (sólo los elementos del nivel de malla inicial)

Estructura	**Descripción**
`.NonFixedElm`	Vector que almacena aquellos elementos (sus índices) que no están dentro de ninguna geometría fija. Esta variable es útil porque son estos elementos los que su utilizan en el proceso de optimización, al contrario de los almacenados en `OptTop.FixedElm`.
`.NoStressElsandAEls`	Vector que almacena todos los elementos (sus índices) activos que no están dentro de una geometría para desestimar tensiones. Es útil para buscar la tensión máxima del componente, ya que solo se busca en estos elementos.
`.Level(iLevel)`	Subestructura que almacena información referente a cada uno de los niveles de la malla `iLevel`.
`.Level(iLevel).dx` `.Level(iLevel).dy`	Variables que almacenan la cantidad de elementos que hay en dirección *x* e *y* respectivamente para cada nivel de malla.
`.Level(iLevel).FEl`	(FEl: First Element) Variable que almacena el índice del primer elemento de cada nivel de malla.
`.TotalArea`	Variable que almacena el área de la geometría de trabajo.
`.lmid`	Vector que almacena el valor al que converge el multiplicador de Lagrange en el algoritmo de bisección (utilizado en la rutina `OC_3`) para cada una de las iteraciones. Se utiliza este valor para estudiar la evolución de dicho parámetro.
`.SerieVolFrac`	Vector que almacena la fracción de volumen con la que se trabaja en cada una de las iteraciones en el caso de que se utilice optimización de tipo B.
`.SerieStress`	Vector que almacena la tensión máxima alcanzada en el componente en cada una de las iteraciones en el caso de que se utilice optimización de tipo B.

Por otra parte hay dos variables que se utilizan en el proceso de optimización pero se han definido en la variable global `Elm` porque hacen referencia a características asociadas a cada elemento. Se definen a continuación.

Estructura	**Descripción**
`Elm.Weight`	Vector del tamaño del número total de elementos que almacena en el índice de cada elemento el peso (densidad) asociado a cada elemento. Esta variable se va actualizando a medida que los pesos van cambiando.
`Elm.dc`	Vector del tamaño del número total de elementos que almacena en el índice de cada elemento la derivada de la energía de deformación asociada a cada elemento. Esta variable se va actualizando a medida que dichos valores van cambiando.

3 Principales funciones

El programa principal de GFEM se arranca desde la ventana *Command Window* de MATLAB mediante la función **RunGUI**.. Esta función abre la interfaz gráfica. Una vez definido completamente el problema, el proceso se inicia desde la ventana *Mesh & Solve* con el botón *Solve*, que llama a la función general del programa: **MainGE**.

La función **MainGE** es común para todo tipo de proceso en el programa de GFEM. En esta se obtienen datos del problema que se han introducido a través de la interfaz gráfica. En el caso concreto del proceso de optimización topológica esto ocurre en la función **Situacion**, donde básicamente se pasan los valores de la variable globlal `FEMGui.OptTop` a `OptTop`, que es la que se utiliza en todo el programa.

Si el tipo de problema que se está resolviendo es el de optimización topológica al final de **MainGE** se invoca **Topology_Optimization**. A continuación se describen las principales funciones que tienen lugar en el proceso.

Topology_Optimization

Es la función principal. En esta se realizan primeramente dos pasos:

- Cálculo de elementos dentro de geometrías definidas como "geometrías fijas" y cálculo de elementos dentro de geometrías definidas como "geometrías para desestimar tensiones". Esto se realiza en un script llamado **FSElem**.
- Inicialización de variables. Esto se hacen en el script **IniVlesTO**, y sirve para inicializar ciertas variables utilizadas posteriormente en el programa. Incluye además cuatro sub-funciones que calculan ciertas variables necesarias para el proceso:
    - **LevelsInf**: Calcula para cada nivel de malla, el número de filas, columnas y el índice del primer elemento.
    - **ElementArea**: Calcula áreas de los elementos de diferentes formas (absolutas y relativas).
    - **Elementcenters**: Calcula la posición del centro de cada elemento.
    - **MapWeightsDim**: Calcula ciertos parámetros para las rutinas de dibujo del mapa de pesos.

Después se accede al tipo de problema específico en función de la elección del usuario en la interfaz gráfica:

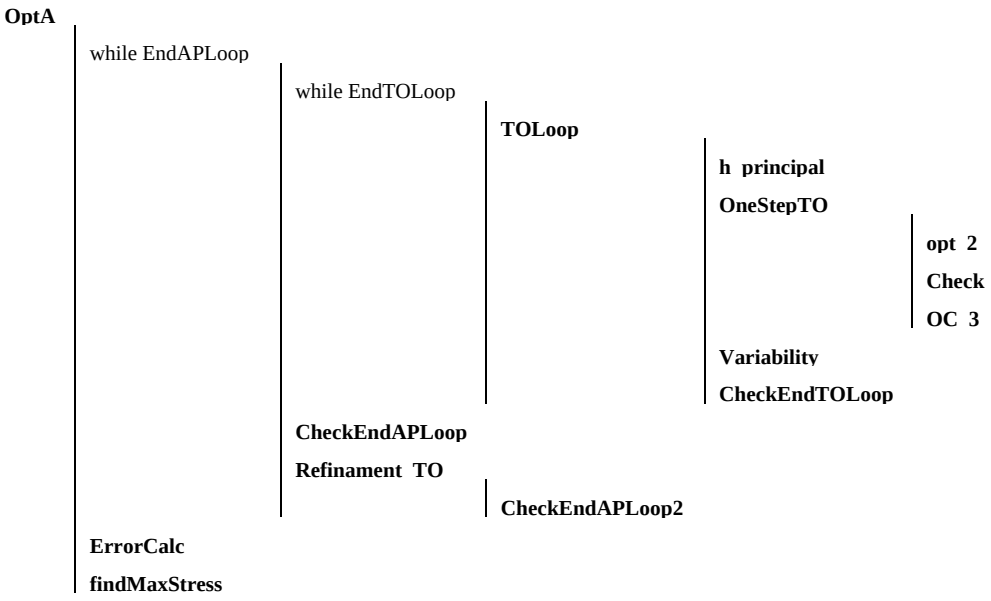
- Optimización A: se invoca entonces a la función **OptA**.
- Optimización B: se invoca entonces a la función **OptB**.

Existe la posibilidad de iniciar también el tipo de problema "Adaptative Process By Projecting Information", explicado en la tesis de máster en la sección 5.1. Esta opción no está disponible el la interfaz gráfica porque es un tipo de proceso que tuvo únicamente una fase experimental. Si se desea iniciarlo se tiene que entrar en la sub-función **Situacion** y darle a la variable `OptTop.KindOfTO` el valor `'APBPI'`.

OptA

Este script realiza las operaciones pertinentes al tipo de optimización A. La estructura consiste dos bucles anidados. El primero de ellos se refiere al proceso de adaptatividad de la malla, en cada paso se refina la malla según un determinado criterio de refinamiento. Para cada una de las mallas existe otro bucle que se refiere al proceso de optimización topológica propiamente dicho. Este algoritmo está explicado de forma detallada en el apartado 4.1.5 de la tesis de máster.

A continuación se presentan, de forma estructurada, las principales funciones, sub-funciones o scripts que se encuentran dentro del script **OptA**, y seguidamente se explica cada una de ellas.



Todos los nombres en el esquema anterior hacen referencia a una función o script excepto el comando “while EndALoop” y “while EndTOLoop” que se refieren al inicio de los bucles de adaptatividad de la malla y de optimización topológica respectivamente, ya que no hay una función explícita para definirlos, si no que se encuentran dentro de **OptA**.

Dentro del bucle de optimización topológica se invoca a **TOLoop**, que representa una iteración en el proceso de optimización, y se explica a continuación. Después de cada paso iterativo (una vez ha convergido el proceso de optimización) se tiene la distribución de pesos definitiva que constituye el componente para una malla determinada.

Se pasa entonces al script **CheckEndAPLoop**. Aquí, lo primero que se comprueba es si la opción de adaptatividad de la malla no está activada, en cuyo caso el proceso finaliza y se pasa directamente al cálculo de tensiones mediante las funciones **ErrorCalc** y **findMaxStress**. Si por el contrario la opción de adaptatividad de la malla sí está activada, se comprueba si el proceso de adaptatividad ha convergido en **CheckEndAPLoop**, atendiendo a criterios basados en la evolución de la energía de deformación.

Si no es así, se refina la malla mediante la sub-función **Refinament_TO**, y posteriormente se analiza de nuevo si el proceso ya había convergido mediante otro criterio, que se define en **CheckEndAPLoop2N**, y que se atiende al número de elementos refinados.

Si el proceso de adaptatividad no converge, se repite el bucle. Cuando este termina, se calculan las tensiones en la función **ErrorCalc**. La función **findMaxStress** sirve para encontrar la máxima tensión en el componente, desestimando, si es el caso, aquellos elementos que estén dentro de las geometrías creadas para tal efecto.

TOLoop

Este script representa una iteración del bucle del proceso de optimización topológica. Como punto de partida se tiene una distribución de pesos determinada. El primer paso dentro de **TOLoop** es calcular los desplazamientos para dicha distribución de pesos, y esto se hace en **h_Principal**, donde se realiza un análisis de elementos finitos.

A continuación se calculan los nuevos pesos a partir de los pesos actuales y el campo de desplazamientos. Esto se realiza en **OneStepTO**, que se explica más detalladamente a continuación. Acto seguido se calcula la variabilidad del proceso comparando el nuevo campo de pesos con el anterior. Esto se realiza en el script **Variability**, y los datos obtenidos sirven para analizar después la convergencia del proceso, hecho que ocurre en el script **CheckEndTOLoop**. Tanto los datos que se obtienen en **Variability** como los cálculos que se realizan con estos en

CheckEndTOLoop se explican de forma detallada en la sección 3.1.2 de la tesis de máster.

El script **TOLoop** se repite sucesivamente dentro del bule de optimización hasta que se obtiene una distribución de pesos que define finalmente el componente.

OneStepTO

Esta sub-función representa un paso iterativo en el proceso de optimización topológica en el sentido de que le entra una distribución de pesos (asociada a un campo de desplazamientos) y sale una nueva distribución actualizada. Para conseguir esto se realizan tres pasos:

- Cálculo de las derivadas de la energía de deformación en cada elemento. Tiene lugar en la función **opt_2**.
- Alisado (o filtro) de las derivadas de la energía de deformación. Tiene lugar en el script **check**.
- Cálculo de la nuevo distribución de pesos. Tiene lugar en **OC_3** y requiere de las derivadas de la energía de deformación en cada elemento (una vez alisadas).

opt_2

Esta función devuelve la derivada de la energía de deformación en cada elemento a partir de los desplazamientos y matriz de rigidez de cada elemento. Utiliza un bucle for que recorre todos los elementos activos. Una explicación más detallada se puede encontrar en el punto 2.3.3 de la tesis de máster.

check

Aquí se realiza el filtro de las derivadas de la energía de deformación en cada elemento. Esta es una función que se ha sido ampliamente modificada desde su versión original. Se dispone de una explicación detallada de los cambios realizados en la sección 3.1.1.1 de la tesis de máster. Se exponen aquí las principales características.

Dependiendo del tipo de proceso se utiliza un filtro u otro. Si se trata de un proceso de optimización topológica sin adaptatividad se utiliza generalmente la

sub-función **check3**. Esta es una versión mejorada del filtro original que utiliza un único bucle (a diferencia de los cuatro bucles anidados del programa original) que recorre los elementos activos. La idea básica es generar una matriz (llamada en la función `matrix`) que representa los índices de los elementos de la malla regular. Esta sirve para calcular los elementos vecinos del elemento `iElem` que intervendrán en la operación de alisado.

Trabajando sin adaptatividad de la malla existe la posibilidad de utilizar la sub-función **check4**, en vez de **check3**, que trabaja de la misma manera pero la matriz que representa los índices de los elementos de la malla es una matriz virtual, es decir, no se crea en realidad. Esta sub-función se ideó para problemas en los que el número de elementos sea tan alto que no sea eficiente (o incluso resulte imposible) almacenar una matriz de dimensiones tan grandes. Tiene la ventaja por tanto de evitar problemas de almacenamiento de información. Por el contrario es una sub-función computacionalmente más costosa que **check3**. Por tano, se utiliza a priori la sub-función check3. En caso de que el usuario desee utilizar **check4** debe determinar esta opción en la función **Situación**, donde se le asignará a la variable `OptTop.check4On` un valor de 1.

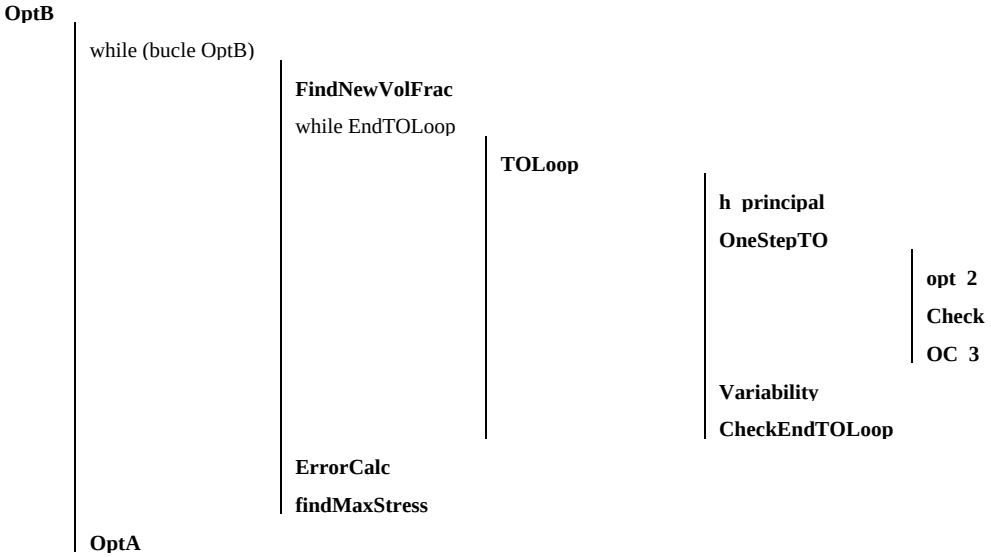
Si se está trabajando con un problema que utiliza adaptatividad de la malla, entonces el filtro se realiza automáticamente mediante la sub-función **check5**, ya que **check3** y **check4** quedan obsoletas al ser sólo válidas para mallas regulares. **check5** es capaz de realizar el alisado con elementos de diferente tamaño. Aunque **check5** funciona de forma correcta con mallas regulares, es conveniente utilizar **check3** o **check4** en estos casos ya que son computacionalmente menos costosos.

OC_3

Esta función tiene como entradas los pesos actuales y las derivadas de energía de deformación en los elementos (no son entradas como argumentos, ya que estas variables son parte de la variable global `OptTop`). Con estos datos y ciertos parámetros calcula los nuevos pesos en los elementos. Realmente sólo se actualizan los pesos de los elementos que no son fijos, es decir, aquellos elementos que no están dentro de una geometría que define una zona de material fijo, y se llaman `OptTop.NonFixedElm`. Los `OptTop.FixedElm` tienen el peso asignado directamente al valor de 1. Una explicación más detallada del funcionamiento de esta función se puede encontrar en la sección 2.3.3.1 de la tesis de máster.

OptB

Este script realiza las operaciones pertinentes al tipo de optimización B. El parámetro de entrada principal es la tensión máxima admisible del material utilizado, que son parámetros introducidos por el usuario vía la interfaz gráfica. A partir de aquí se trata de encontrar la fracción de volumen para la cual se satisfaga dicha tensión en todo el componente. Para ello, se recurre a un proceso iterativo en el cual se van calculando componentes para diferentes fracciones de volumen. El cálculo de cada componente se realiza mediante un proceso de optimización topológica independiente. La fracción de volumen que se va iterando se calcula mediante un algoritmo matemático que ejecuta la función **FindNewVolFrac**. Este proceso iterativo se construye mediante un bucle que termina cuando la diferencia entre la máxima tensión encontrada en el componente y la máxima tensión admisible está por debajo de cierta tolerancia definida por el usuario. Se presenta a continuación un pequeño esquema estructurado de las principales funciones que tienen lugar en **OptB**. El comando `while` que aparece se refiere a la condición del bucle mencionada.



En cada iteración del bucle se busca primeramente la fracción de volumen que se va a utilizar en **TOLoop**, y esto se hace mediante la función **FindNewVolFrac**. Cuando el proceso de optimización topológica converge se calculan las tensiones mediante **ErroCalc**, y la máxima tensión en el componente mediante

findMaxStess. El criterio de continuar o acabar el bucle se encuentra en el mismo enunciado del `while`.

Una vez finalizado este bucle, se tiene una fracción de volumen determinada y una distribución de pesos en el componente. Si está activa la opción de malla adaptativa se entra entonces en el script **OptA**, donde se realizará un proceso de optimización topológica con malla adaptativa. Si no está activa la opción de malla adaptativa el proceso termina sin entrar en **OptA**.

FindNewVolFrac

Esta función tiene como objetivo calcular la nueva fracción de volumen que se utilizará para el siguiente proceso de optimización topológica. Las dos primeras iteraciones del bucle de **OptB** trabajan con fracciones de volumen de 1 y 0.5 consecutivamente. A partir de ahí, se calculan las nuevas fracciones a partir de una aproximación lineal primeramente y aproximaciones cuadráticas y tercer grado a medida que se tienen más datos de iteraciones anteriores.

A parte de las funciones explicadas existen dos rutinas de dibujo para representar el componente con sus pesos: **DrawMap2** y **DrawMap4**.

DrawMap2

Esta función crea un mapa de bits utilizando la función de la biblioteca de MATLAB llamada `patch`. Ésta asocia a cada punto del espacio un valor determinado. Es una función rápida pero con el inconveniente de que dibuja enteros los elementos que están intersectados por la geometría de trabajo. Es por esto que se utiliza esta función para los pasos intermedios del proceso de optimización. Cuando se llega a la solución se utiliza **DrawMap4** para dibujar los elementos intersectados correctamente.

DrawMap4

Esta función hace uso de la sub-función **DrawLocalEffectivity** para dibujar mediante polinomios el mapa de pesos en los elementos. Es una función más lenta que **DrawMap2**, pero consigue dibujar solo la parte activa de los elementos intersectados por la geometría de trabajo. Esta función se utiliza al final del proceso de optimización topológica, una vez obtenida la solución.