



UANL

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN



FIME

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Laboratorio De Biomecanica

Practica 1

Equipo 11 - N5

Andrik David Salas Carranza
Juan Carlos Saldaña González
Jeiddy Michel Martinez Navéjar
Ana Sofía Limón González
Joel Zuñiga Olvera
Yuliana Lizbeth Bravo Salaza
Fred Raúl Peña Mata
Raúl Alexandro Vega López
Jesús Alberto Medina González

13 de septiembre de 2022

Índice

1. Objetivo	3
2. Resumen	3
3. Introducción	3
4. Marco teórico	3
5. Desarrollo	5
5.1. Programación	5
5.2. Estado del arte	6
5.3. Procedimiento de la programación	7
5.4. Implementación o desarrollo de la programación en sus diferentes vistas.	12
6. Código implementado	14
7. Resultados	17
8. Conclusiones	19

1. Objetivo

El estudiante conocerá una de las secciones que integran el código de optimización topológica, como se debe crear el archivo (.m) en MATLAB y como se ejecuta el análisis.

2. Resumen

Para este reporte se va a implementar el código de 99 líneas para la optimización topológica de una pieza; en este caso será el de una viga. Para su implementación se estará trabajando con el software de Matlab ya que nos permite manipular el código de la manera en la que nosotros deseamos para cumplir con el objetivo planteado en la práctica.

3. Introducción

El código de optimización topológica de 99 líneas en Matlab que se estará utilizando para la correspondiente práctica uno del laboratorio de biomecánica; se divide en 36 líneas para la programación principal, 12 para los criterios de optimización, 16 para el filtro de mallado y 35 para el código de elemento finito. Este código fue desarrollado por O.Sigmund, Departamento f Solid Mechanics, Building 404, Technical University of Denmark, DK-2800 Lyngboy, Denmark. El código puede ser descargado desde la página del autor.

4. Marco teórico

La optimización topológica es una técnica englobada dentro del campo de análisis estructural. Se basa en el análisis mecánico de un componente o estructura. Su principal objetivo es el aligeramiento estructural manteniendo las funcionalidades mecánicas del componente objetivo. A diferencia de otros tipos de optimización, la optimización topológica ofrece un nuevo concepto de diseño estructural enfocado a aquellas aplicaciones donde el peso del componente es crucial (por ejemplo, la industria aeroespacial).

Gracias a los nuevos métodos computacionales, es posible llevar la optimización a un nivel más complejo de análisis a nivel estático, dinámico, plástico, modal o de impacto, entre otros, los cuales pueden considerarse durante el proceso de optimización.

El desarrollo de esta metodología tiene un amplio campo de aplicación para las tecnologías de fabricación aditiva, como por ejemplo la fabricación SLM (Selective Laser Melting), debido a las grandes posibilidades en términos de diseño (geometrías muy complejas).

La optimización topológica es una técnica que pertenece al análisis estructural, y consiste, básicamente, en analizar un componente o estructura y, en función de cómo se cargue, eliminar material ahí donde no es necesario.

En el proceso de optimización topológica, se deben de tener en cuenta varios aspectos; el espacio de diseño, el o los casos de carga que va a sufrir la pieza en cuestión, el material y la tecnología con que se va a realizar su fabricación, la reducción de costes mediante la minimización de soportes y aprovechamiento de la cuba de impresión, en caso de utilizar tecnologías aditivas, y muchos más. Los pasos por seguir para una Optimización Topológica son el dibujar o importar geometría,



Figura 1:

simplificar la pieza y definir el espacio de diseño, establecer uniones, juntas y contactos, asignar materiales, definir los casos de carga, generar la optimización, refinar la geometría, exportar a CAD o generar STL, verificar el rendimiento y fabricar.

MATLAB es un sistema de cómputo numérico que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, macOS y GNU/Linux.

Entre sus prestaciones básicas se hallan la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

5. Desarrollo

5.1. Programación

La programación es el proceso de crear un conjunto de instrucciones que le dicen a una computadora como realizar algún tipo de tarea. Pero no solo la acción de escribir un código para que la computadora o el software lo ejecute. Incluye, además, todas las tareas necesarias para que el código funcione correctamente y cumpla el objetivo para el cual se escribió.

En la actualidad, la noción de programación se encuentra muy asociada a la creación de aplicaciones de informática y videojuegos. En este sentido, es el proceso por el cual una persona desarrolla un programa, valiéndose de una herramienta que le permita escribir el código (el cual puede estar en uno o varios lenguajes, como C++, Java y Python, entre muchos otros) y de otra que sea capaz de “traducirlo” a lo que se conoce como lenguaje de máquina, que puede comprender.^{el} microprocesador.

Un lenguaje de programación es un lenguaje formal (o artificial, es decir, un lenguaje con reglas gramaticales bien definidas) que le proporciona a una persona, en este caso el programador, la capacidad de escribir (o programar) una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico o lógico de un sistema informático, de manera que se puedan obtener diversas clases de datos o ejecutar determinadas tareas. A todo este conjunto de órdenes escritas mediante un lenguaje de programación se le denomina programa informático

La optimización topológica es uno de los métodos potencialmente más prometedores de cara a mejorar las características de una estructura. Generalmente este método ha sido aplicado a geometrías continuas, laminares o volumétricas, en las cuales la optimización parte del mallado MEF de la pieza. A partir del mismo, los algoritmos encuentran la solución topológicamente óptima en función del peso, la resistencia, la rigidez u otros parámetros de diseño. Sin embargo, en el caso de querer optimizar estructuras reticulares, se hace necesario crear un reticulado inicial, a partir del cual iniciar el proceso de optimización. Esta malla de partida implica la optimización únicamente se realiza sobre los elementos presentes y que no sea factible evaluar la totalidad de los caminos de carga posibles.

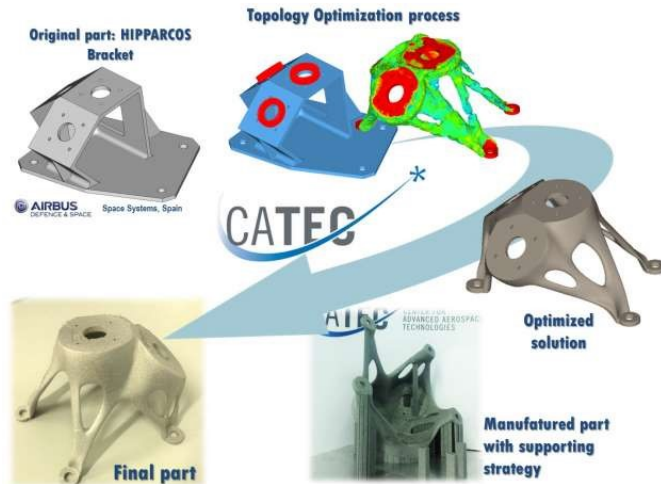


Figura 2:

5.2. Estado del arte

La optimización de la topología (TO) es un método matemático que optimiza el diseño del material dentro de un espacio de diseño dado, para un conjunto dado de cargas, condiciones de contorno y restricciones con el objetivo de maximizar el rendimiento del sistema. TO es diferente de la optimización de la forma y la optimización del tamaño en el sentido de que el diseño puede alcanzar cualquier forma dentro del espacio de diseño, en lugar de tratar con configuraciones predefinidas. La formulación convencional de TO utiliza un método de elementos finitos [FEM] para evaluar el desempeño del diseño. El diseño se optimiza utilizando técnicas de programación matemática basadas en gradientes, como el algoritmo de criterios de optimalidad y el método de mover asíntotas o algoritmos no basados en gradientes, como los algoritmos genéticos.

La optimización de topología tiene una amplia gama de aplicaciones en ingeniería aeroespacial, mecánica, bioquímica y civil. Actualmente, los ingenieros usan principalmente TO en el nivel de concepto de un proceso de diseño. Debido a las formas libres que ocurren naturalmente, el resultado es a menudo difícil de fabricar. Por esa razón, el resultado que surge de TO a menudo se ajusta para la fabricación. Agregar restricciones a la formulación para aumentar la capacidad de fabricación es un campo activo de investigación. En algunos casos, los resultados de TO se pueden fabricar directamente utilizando fabricación aditiva; Por lo tanto, TO es una parte clave del diseño para la fabricación aditiva.

Adentrándonos un poco en la historia podemos resaltar que este método derivado de las matemáticas fue claramente definido, explicado y hecho utilizable para la mecánica en la década de 2000, en particular con el artículo fundador de Ole Sigmund

5.3. Procedimiento de la programación

El código de Matlab se compone como un código de optimización topológica de tipo estándar, el cual esta preparado para que lo interprete el software de MATLAB, para hacerlo se deben seguir los pasos básicos que son los siguientes:

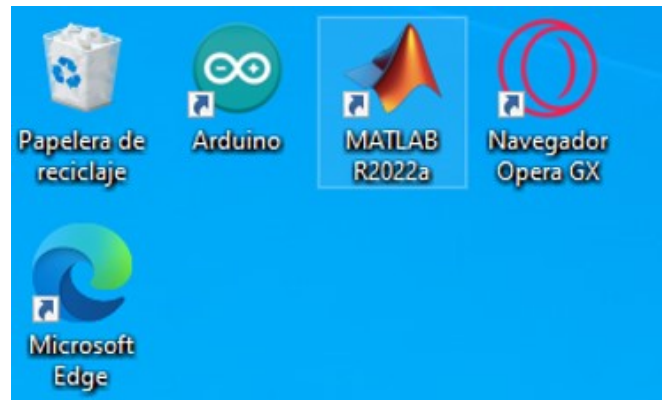


Figura 3: 1. Abrir MATLAB seleccionando el icono en nuestro ordenador.

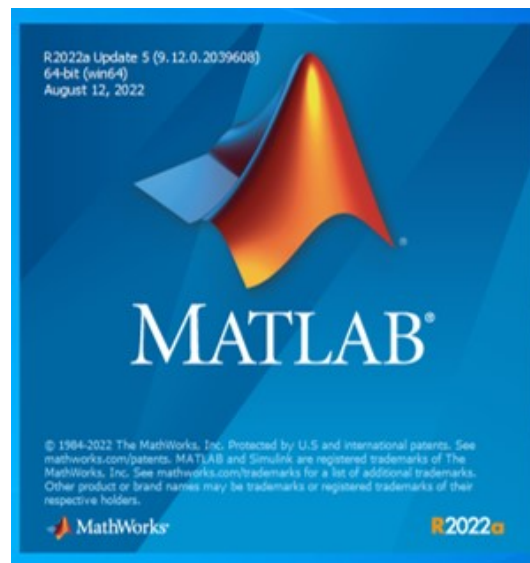


Figura 4: 2. Esperar a que se inicialice.

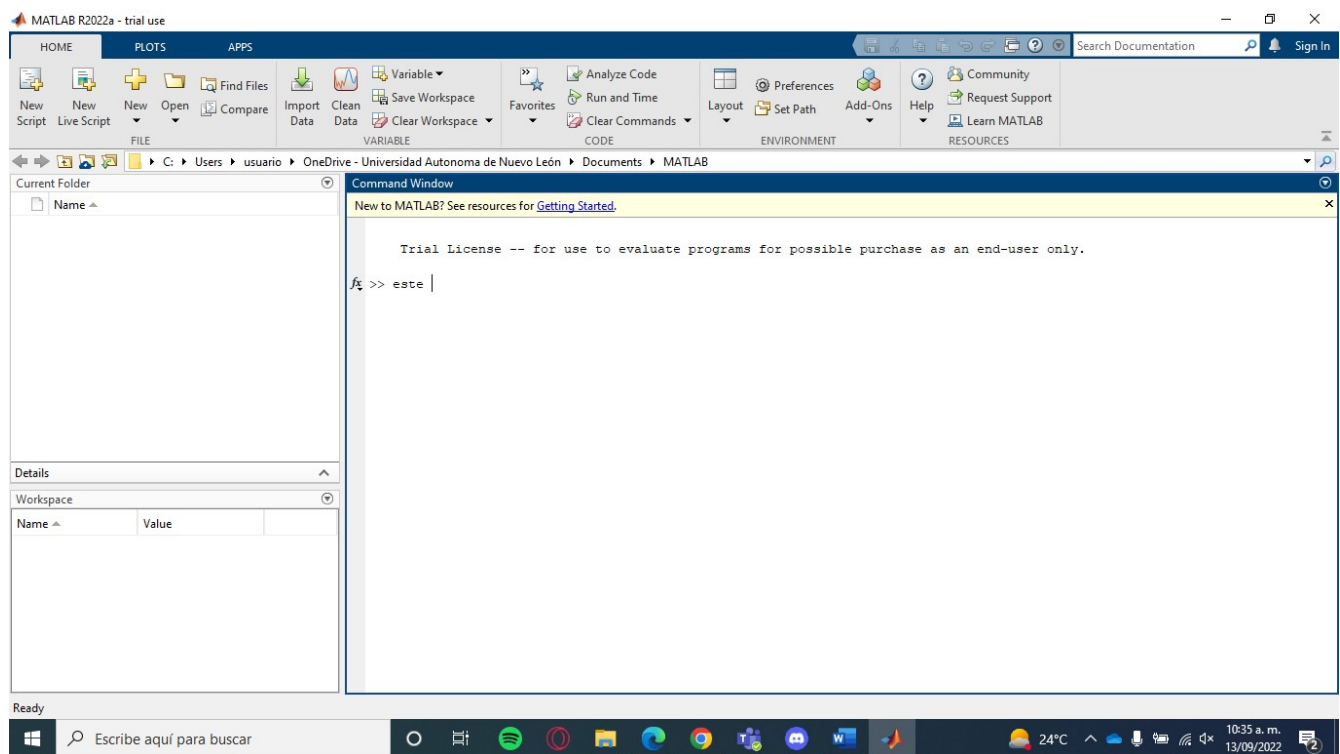


Figura 5: 3. Pantalla de inicio.

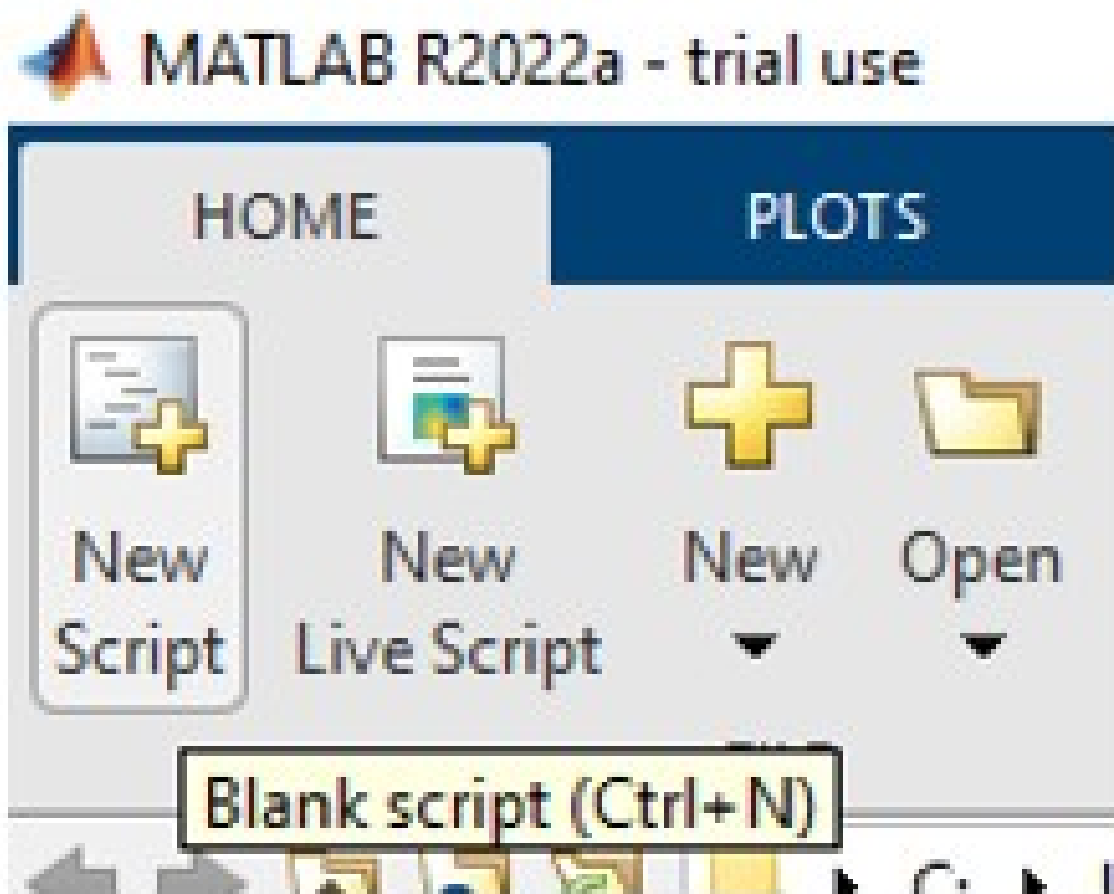


Figura 6: 4. Seleccionar en la barra de herramientas, seguir los siguientes pasos: HOME; New Script.

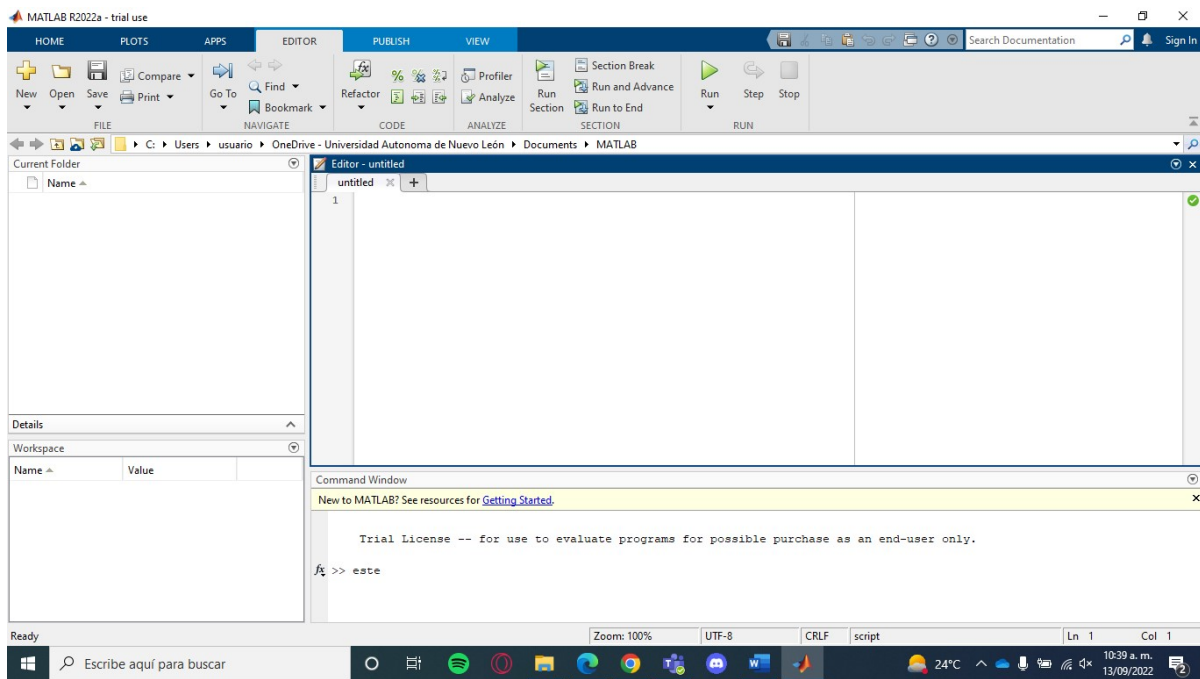


Figura 7: 5. Una vez desplegada el área de trabajo, se le asigna un nombre al archivo para poder guardar.

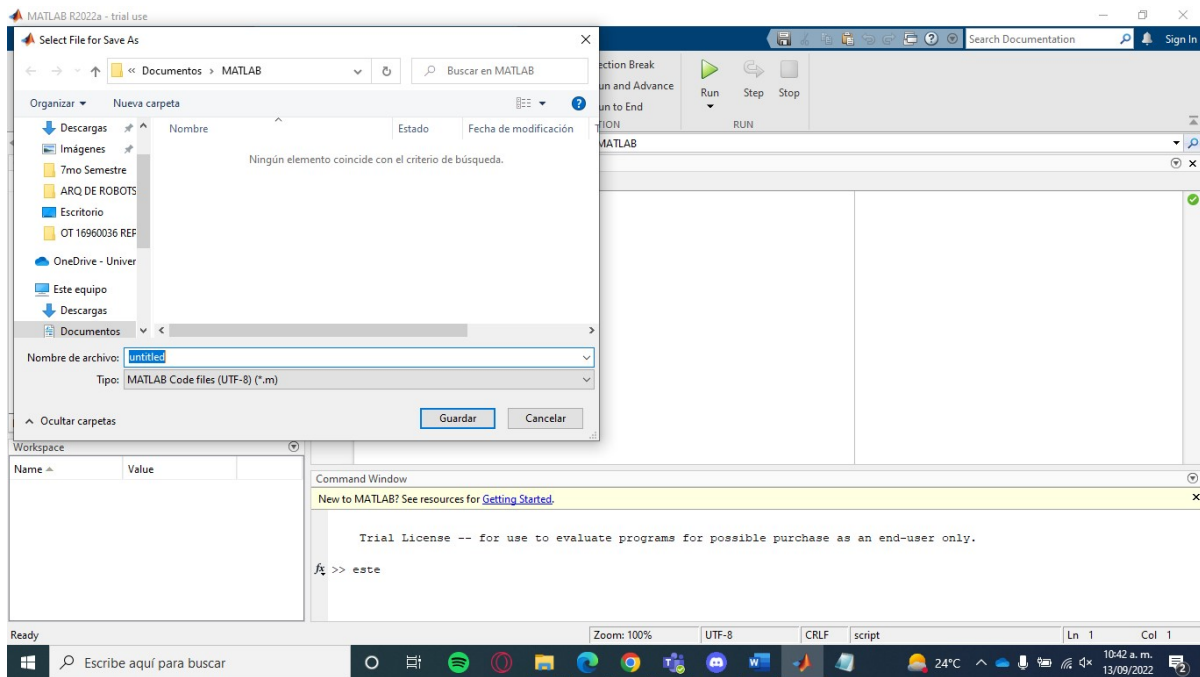
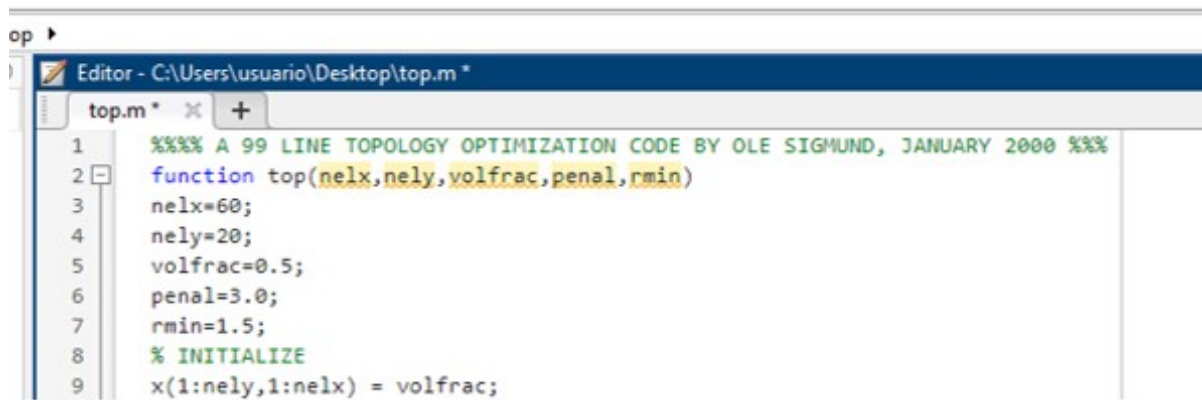


Figura 8: 6. Se procede a guardar. EDITOR, Save, Save As. Se le colocan los datos y/o nombre designado.



The image shows a MATLAB editor window titled "Editor - C:\Users\usuario\Desktop\top.m *". The code is as follows:

```
1 %%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
2 function top(nelx,nely,volfrac,penal,rmin)
3 nelx=60;
4 nely=20;
5 volfrac=0.5;
6 penal=3.0;
7 rmin=1.5;
8 % INITIALIZE
9 x(1:nely,1:nelx) = volfrac;
```

Figura 9: 7. Se coloca el código correctamente. Y se procede a guardar. 8. Definir valores.

5.4. Implementación o desarrollo de la programación en sus diferentes vistas.

99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000
CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND

```
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)Apenal*Ue'*KE*Ue;
dc(ely,elx) = -penal*x(ely,elx)A(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('
' ch.: ' sprintf('
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*nex*nely < 0;
l1 = lmid;
else
l2 = lmid;
end
end
MESH-INDEPENDENCY FILTER
function [dcn]=check(nex,nely,rmin,x,dc)
dcn=zeros(nely,nex);
for i = 1:nex
for j = 1:nely
sum=0.0;
for k = max(i-floor(rmin),1):min(i+floor(rmin),nex)
for l = max(j-floor(rmin),1):min(j+floor(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(nex,nely,x,penal)
KE
= lk;
K = sparse(2*(nex+1)*(nely+1), 2*(nex+1)*(nely+1));
F = sparse(2*(nely+1)*(nex+1),1); U = zeros(2*(nely+1)*(nex+1),1);
for elx = 1:nex
for ely = 1:nely
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)Apenal*KE;
end
end
DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
F(2,1) = -1;
fixeddofs = union([1:2:2*(nely+1)],[2*(nex+1)*(nely+1)]);
alldofs = [1:2*(nely+1)*(nex+1)];
freedofs = setdiff(alldofs,fixeddofs);

```

```

SOLVING
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-nuA2)*[ 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)];

```

This Matlab code was written by Ole Sigmund, Department of Solid Mechanics, Technical University of Denmark, DK-2800 Lyngby, Denmark. Please sent your comments to the author: sigmund@fam.dtu.dk

The code is intended for educational purposes and theoretical details are discussed in the paper

.^A 99 line topology optimization code written in Matlab”

by Ole Sigmund (2001), Structural and Multidisciplinary Optimization, Vol 21, pp. 120–127.

The code as well as a postscript version of the paper can be downloaded from the web-site: <http://www.topopt.dtu.dk>

Disclaimer:

The author reserves all rights but does not guaranty that the code is free from errors. Furthermore, he shall not be liable in any event caused by the use of the program.

6. Código implementado

```

Editor - C:\Users\usuario\Practical.m *
Practical.m *  +
1  %%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
2  %%%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%
3  %Equipo Lineares LABORATORIO BIOMECANICA BRIGADA 509%
4  function Practical(nelx,nely,volfrac,penal,rmin);
5  % INITIALIZE
6  x(1:nely,1:nelx) = volfrac;
7  loop = 0;
8  change = 1.;
9  % START ITERATION
10 while change > 0.01
11     loop = loop + 1;
12     xold = x;
13     % FE-ANALYSIS
14     [U]=FE(nelx,nely,x,penal);
15     % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
16     [KE] = lk;
17     c = 0.;
18     for ely = 1:nely
19         for elx = 1:nelx
20             n1 = (nely+1)*(elx-1)+ely;
21             n2 = (nely+1)* elx +ely;
22             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);
23             c = c + x(ely,elx)^penal*Ue'*KE*Ue;
24             dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
25         end
26     end
27 % FILTERING OF SENSITIVITIES

```

Figura 10: 7. Código implementado.

```

Editor - C:\Users\usuario\Practical.m
Practical.m  +
24     x(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
25     end
26 end
27 % FILTERING OF SENSITIVITIES
28 [dc] = check(nelx,nely,rmin,x,dc);
29 % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
30 [x] = OC(nelx,nely,x,volfrac,dc);
31 % PRINT RESULTS
32 change = max(max(abs(x-xold)));
33 disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',c) ...
34       ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
35       ' ch.: ' sprintf('%6.3f',change )])
36 % PLOT DENSITIES
37 colormap(gray); imagesc(-x); axis equal; axis tight; axis off; pause(1e-6);
38 end
39 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
40 function [xnew]=OC(nelx,nely,x,volfrac,dc)
41 l1 = 0; l2 = 100000; move = 0.2;
42 while (l2-l1 > 1e-4)
43     lmid = 0.5*(l2+l1);
44     xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
45     if sum(sum(xnew)) - volfrac*nelx*nely > 0;
46         l1 = lmid;
47     else
48         l2 = lmid;
49     end
50 end
51 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

Figura 11: 7. Código implementado.

```

Editor - C:\Users\usuario\Practical.m
Practical.m
50 end
51 %%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%
52 function [dcn]=check(nelx,nely,rmin,x,dc)
53 dcn=zeros(nely,nelx);
54 for i = 1:nelx
55     for j = 1:nely
56         sum=0.0;
57         for k = max(i-floor(rmin),1):min(i+floor(rmin),nelx)
58             for l = max(j-floor(rmin),1):min(j+floor(rmin),nely)
59                 fac = rmin-sqrt((i-k)^2+(j-l)^2);
60                 sum = sum+max(0,fac);
61                 dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
62             end
63         end
64         dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
65     end
66 end
67 %%%%%%%%% FE-ANALYSIS %%%%%%%%%
68 function [U]=FE(nelx,nely,x,penal)
69 [KE] = lk;
70 K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
71 F = sparse(2*(nely+1)*(nelx+1),1); U = zeros(2*(nely+1)*(nelx+1),1);
72 for elx = 1:nelx
73     for ely = 1:nely
74         n1 = (nely+1)*(elx-1)+ely;
75         n2 = (nely+1)* elx +ely;
76         edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];

```

Figura 12: 7. Código implementado.

```

Editor - C:\Users\usuario\Practical.m
Practical.m
76 edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
77 K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
78 end
79 end
80 % DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
81 F(2,1) = -1;
82 fixeddofs = union([1:2*(nely+1)], [2*(nelx+1)*(nely+1)]);
83 alldofs = [1:2*(nely+1)*(nelx+1)];
84 freedofs = setdiff(alldofs, fixeddofs);
85 % SOLVING
86 U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
87 U(fixeddofs,:)= 0;
88 %%%%%%%%% ELEMENT STIFFNESS MATRIX %%%%%%%%%
89 function [KE]=lk
90 E = 1.;
91 nu = 0.3;
92 k=[ 1/2-nu/6    1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
93    -1/4+nu/12 -1/8-nu/8    nu/6    1/8-3*nu/8];
94 KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
95                  k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
96                  k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
97                  k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
98                  k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
99                  k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
100                 k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
101                 k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
102 %
103 %%%%%%%%%

```

Figura 13: 7. Código implementado.

7. Resultados

Parámetros: (30,10,0.5,3.0,1.5)

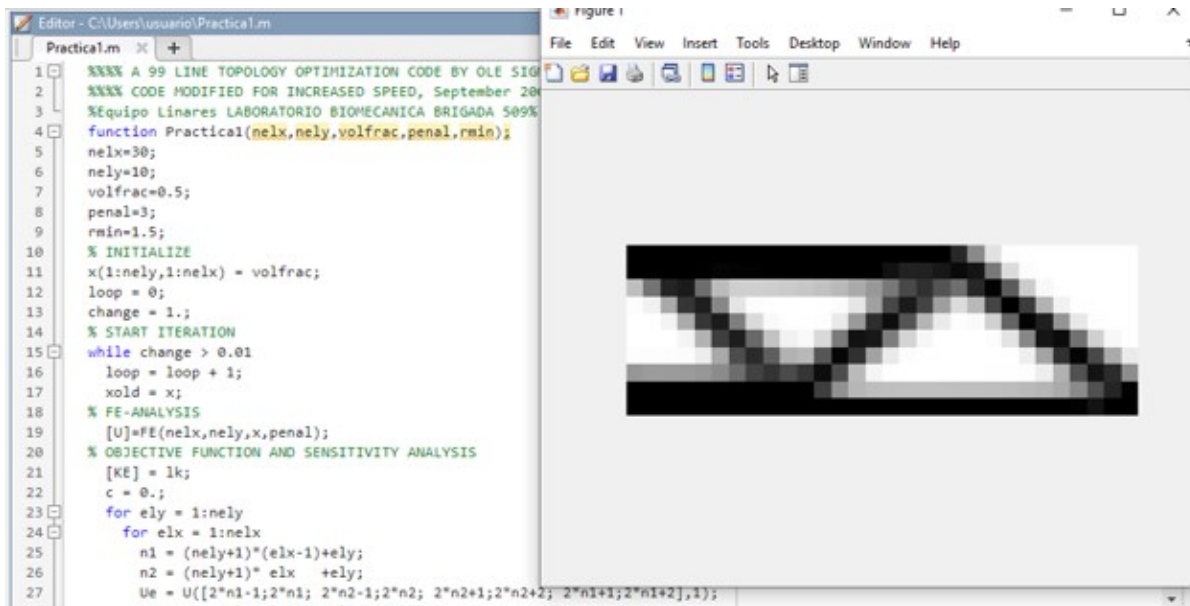


Figura 14: Parámetros: (40,10,0.5,3.0,1.5)

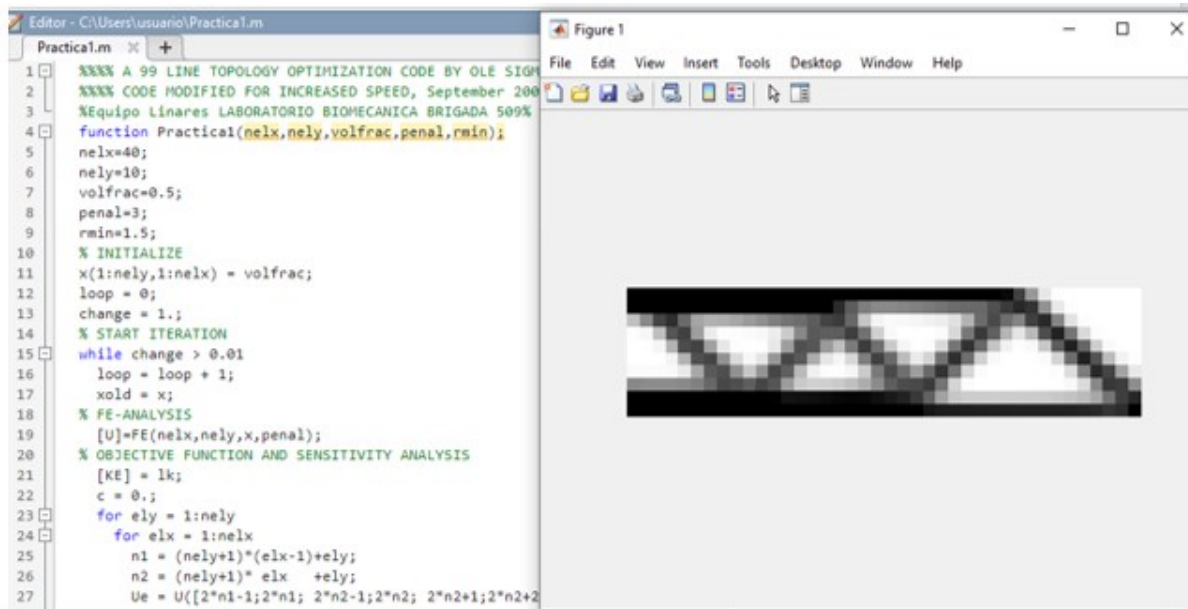


Figura 15: Parámetros: (60,10,0.5,3.0,1.5)

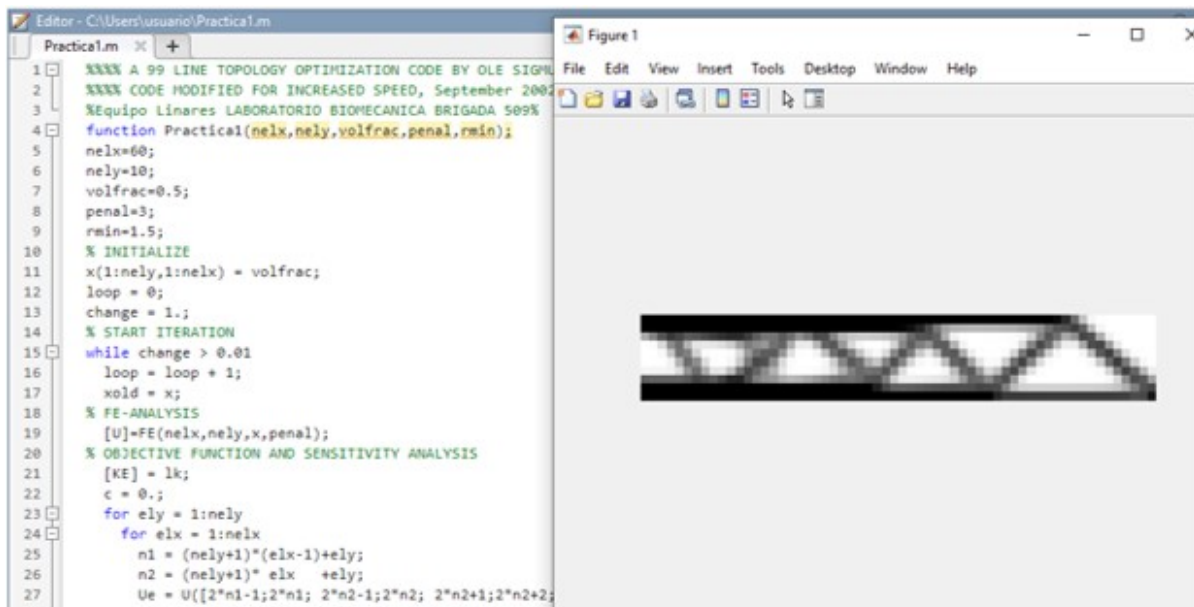


Figura 16: 7. Código implementado.

8. Conclusiones

Raúl Alexandro Vega López

Gracias a esta tecnología es posible observar donde se concentra la carga en una estructura “robusta” que usualmente ocupa mucho material desperdiciado y convertirla en una estructura más limpia que no ocupe tanto material desperdiciado. Con esto se optimizan y aprovechan de una mejor manera los espacios de una estructura y los gastos de material son menores.

Juan Carlos Saldaña González

En conclusión, esta práctica fue algo tediosa ya que tuvimos que recurrir a varios recursos como lo es matlab. Fue algo complicado al principio volver a utilizar matlab ya que este software ya tenía un tiempo considerable que no lo había vuelto a manejar así que tuve que recurrir a apuntes de libretas, tutoriales entre otras cosas para poder volver a manipular el software en conclusión el objetivo se logró cumplir y pudimos así cumplir cada punto que se nos pidió.

Yuliana Lizbeth Bravo Salazar

Para finalizar esta práctica; fue de gran utilidad ya que trabajamos con un programa que yo ya conocía de semestres anteriores (Matlab). Así que la realización de la actividad fue fácil de comprender.

Ana Sofía Limón González

Para realizar esta práctica fue necesario conocer la definición del término optimización topológica, la cuál es una técnica donde se analiza una estructura y posteriormente se aligera conservando sus propiedades mecánicas. Luego de entender lo que significaba la optimización topológica fue posible aplicarlo dentro de un ejemplo. Para lograr realizar la práctica se hizo uso del software MATLAB con el cual fue posible realizar el código para analizar la viga que se propuso.

Jesús Alberto Medina González

Para concluir con esta práctica se tuvo que recurrir a la instalación del software ya que yo no lo había utilizado desde hace 2 años y volver a analizar toda la información y comandos para poder utilizarlo de Manera correcta para la práctica, los datos de ingresaron y después se obtiene el resultado necesario para la práctica.

Fred Raúl Peña Mata 1866587 Concluimos este trabajo recordando conocimientos previos de la carrera sobre el uso del software de Matlab y obteniendo nuevos conocimientos como la optimización topología que se utiliza para analizar estructuras y aligerarlas, pero manteniendo sus propiedades mecánicas.

Jeiddy Michel Martínez Navéjar

Puedo concluir que el objetivo planeado para esta práctica de laboratorio, el cual es que el estudiante conocerá una de las secciones que integran el código de optimización topológica, como se debe crear el archivo en MATLAB y como se ejecuta el análisis; fue cumplido exitosamente ya que se completaron los objetivos, así como los puntos de los que debía constar nuestro reporte. Esta práctica la considero sencilla ya que tenemos conocimientos previos con el software, y su utilización; siguiendo los pasos todo fue simple y poco tedioso. Todo esto apoyándonos con herramientas en línea para cumplir con los parámetros establecidos.

Andrik David Salas Carranza

Mediante la elaboración de esta práctica entendí como implementaremos el código de optimización topológica de 99 líneas en Matlab y como este se puede dividir ya sea 36 para la programación, 12 para los criterios de optimización, 16 para el filtro de mallado y 35 para el código de elemento finito. También vi cómo la optimización topológica es una técnica englobada está dentro del campo del análisis estructural y esta se basa en el análisis mecánico de un componente.

Referencias

[de Ingeniería y Tecnología Avanzada S.L.(2013)] EITA (Estudio de Ingeniería y Tecnología Avanzada S.L. Optimización topológica, septiembre 2013.

[de Mola(2012)] Marco Loret de Mola. Mathworks, matlab, agosto 2012.

[director general Joaquin Rodriguez Grau(2019)] director general Joaquin Rodriguez Grau. Optimización topológica, septiembre 2019.

[grupo TopOpt es líder mundial en desarrollo y aplicaciones de métodos de optimización de topología basados en densidad. DTU. El grupo TopOpt es líder mundial en desarrollo y aplicaciones de métodos de optimización de topología basados en densidad. Un código de optimización de topología de 99 líneas escrito en matlab, Marzo 2018.

[HiSoUR(2012)] HiSoUR. Optimización de topología, septiembre 2012.

[Mathieu(2014)] Mihaela Juganaru Mathieu. Introducción a la programación, septiembre 2014.

[O.Sigmund(2018)] Departamento of Solid Mechanics O.Sigmund. Código de optimización de topología de 99 líneas, Marzo 2018.

[R. Lagunilla Sánchez(2016)] B. Vallés Fernández E. Alcalá Fazio R. Lagunilla Sánchez, D. Arribas Mantelli. Optimización topológica de estructuras reticulares 3d con malla variable, septiembre 2016.

[grupo TopOpt es líder mundial en desarrollo y aplicaciones de métodos de optimización de topología basados en densidad. DTU. El grupo TopOpt es líder mundial en desarrollo y aplicaciones de métodos de optimización de topología basados en densidad. Un código de optimización de topología de 99 líneas escrito en matlab, Marzo 2018.

[director general Joaquin Rodriguez Grau(2019)] [de Ingeniería y Tecnología Avanzada S.L.(2013)]

[O.Sigmund(2018)] [de Mola(2012)] [Mathieu(2014)] [R. Lagunilla Sánchez(2016)] [HiSoUR(2012)]