UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FIME
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Práctica 1

Manuel Exiquio Barrera Suárez 1992101
Ernesto Axel Moreno García 1992038
Juan Ángel Hérnandez Antonio 1992154
Alberto Alan Ramirez Velazquez 1802607

September 5, 2022

1 Objetivo

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

2 Marco Teórico

La optimización topológica comienza con la creación de un modelo 3D en la fase de borrador, en el que se aplicaran las diferentes cargas o fuerzas para la pieza (una presión sobre las lengüetas de sujeción, por ejemplo). Un problema clásico de la ingeniería consiste en determinar la configuración geométrica óptima de un cuerpo que minimice o maximice una cierta función objetivo, al mismo tiempo que satisface las restricciones o condiciones de contorno del problema. La solución de este problema puede ser planteada utilizando dos estrategias: como un problema de optimización de forma o de optimización de la topología.

La optimización de forma consiste en modificar la geometría del dominio preservando su topología, es decir sin crear huecos o cavidades en su interior. Este tipo de análisis es usualmente conocido como análisis de sensibilidad al cambio de forma y sus bases matemáticas se encuentran bien establecidas. El principal inconveniente del análisis de sensibilidad al cambio de forma es que sólo permite cambios en la frontera del dominio, lo que limita su campo de aplicación.

Una manera más general de controlar un dominio es mediante modificaciones de su topología, lo que permite obtener la configuración deseada partiendo de una morfología inicial distante de la óptima. Los métodos de homogenización son posiblemente los más utilizados para la optimización topológica. Estos consisten en caracterizar la topología a través de su densidad, es decir, los huecos se identifican con regiones de densidad nula. De esta forma la solución del programa resulta en una distribución ficticia de material.

Matlab es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). 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.

El código de optimización topológica de 99 líneas en Matlab que se utilizara en este laboratorio se divide en 36 líneas para la programación principal, 12 líneas para los criterios de optimización, 16 líneas para el filtro de mallado y 35 líneas para el código de elemento finito. De hecho, excluyendo las líneas de comentarios y líneas asociadas con la producción y el análisis de elementos finitos, el código resultante es de solo 49 líneas. Este código fue desarrollado por O. Sigmund, Department of Solid Mechanics, Building 404, Technical University of Denmark, DK-2800 Lyngby, Denmark. El código puede ser descargado desde la página del autor: http://www.topopt.dtu.dk[1].

3 Desarrollo

• Nombre y definición de la programación, mencionar un ejemplo de forma de la GEOMETRÍA

La optimización, también denominada programación matemática, sirve para encontrar la respuesta que proporciona el mejor resultado, la que logra mayores ganancias, mayor producción o felicidad o la que logra el menor costo, desperdicio o malestar. Con frecuencia, estos problemas implican utilizar de la manera más eficiente los recursos, tales como dinero, tiempo, maquinaria, personal, existencias, etc. Los problemas de optimización generalmente se clasifican en lineales y no lineales, según las relaciones del problema sean lineales con respecto a las variables. Existe una serie de paquetes de software para resolver problemas de optimización. Por ejemplo, LINDO o WinQSB resuelven modelos de programas lineales y LINGO y What'sBest! resuelven problemas lineales y no lineales.

La Programación Matemática, en general, aborda el problema de determinar asignaciones óptimas de recursos limitados para cumplir un objetivo dado. El objetivo debe representar la meta del decisorio. Los recursos pueden corresponder, por ejemplo, a personas, materiales, dinero o terrenos. Entre todas las asignaciones de recursos admisibles, queremos encontrar la/s que maximiza/n o minimiza/n alguna cantidad numérica tal como ganancias o costos.

• Estado del Arte

La fabricación aditiva, a diferencia de las técnicas de mecanizado tradicionales, permite producir piezas con geometrías complejas. El peso total de las mismas se puede optimizar mediante un método digital denominado «optimización topológica». Esto también maximiza la resistencia mecánica de la pieza creada. La optimización topológica es, de hecho, un subcampo del diseño digital que permite encontrar, gracias a fórmulas matemáticas, la distribución óptima de material en un volumen determinado sometido a tensiones mecánicas más o menos significativas. La optimización topológica, por tanto, consiste en utilizar un software concreto para «eliminar» el material que no posee los soportes. Entre los programas más conocidos se encuentran las soluciones Ansys Discovery, Tosca de Dassault Systèmes, Within Labs de Autodesk, Inspire de SolidThinking, Netfabb y Simufact Additive.

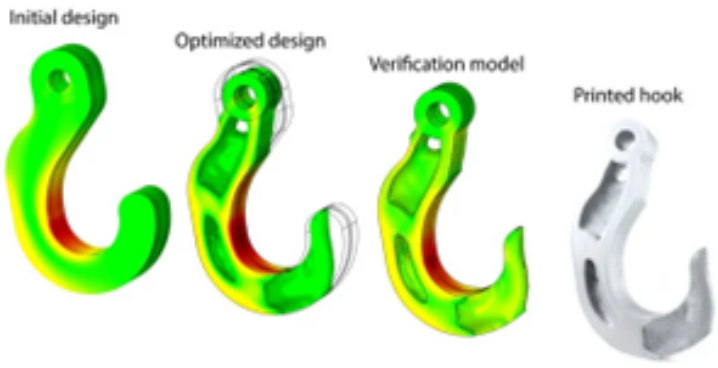

Figure 1: Ejemplo de Optimización Topológica

¿Cómo funciona la optimización topológica?

Los procesos tradicionales de diseño digital conllevan aplicar cargas a una pieza ya fabricada y evaluar dónde se está debilitando. Luego, los ingenieros deben repensar el diseño hasta que la pieza cumpla con las restricciones mecánicas dadas. Con la optimización topológica, el sentido es diferente: las cargas mecánicas son los datos de entrada que permitirán al software proponer una nueva geometría de la pieza. Así, en principio hay menos iteraciones, lo que reduce considerablemente los tiempos de diseño y fabricación.

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. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

- Lenguaje. Las aplicaciones de MATLAB se desarrollan en un lenguaje de programación propio. Este lenguaje es interpretado, y puede ejecutarse tanto en el entorno interactivo, como a través de un archivo de script (archivos *.m). Este lenguaje permite operaciones de vectores y matrices, funciones, cálculo lambda, y programación orientada a objetos.

- Limitaciones y alternativas. Durante mucho tiempo hubo críticas porque MATLAB es un producto propietario de The Mathworks, y los usuarios están sujetos y bloqueados al vendedor. Recientemente se ha proporcionado una herramienta adicional llamada MATLAB Builder bajo la sección de herramientas ≪Application Deployment≫ para utilizar funciones MATLAB como archivos de biblioteca que pueden ser usados con ambientes de construcción de aplicación .NET o Java. Pero la desventaja es que el computador donde la aplicación tiene que ser utilizada necesita MCR(MATLAB Component Runtime) para que los archivos MATLAB funcionen correctamente. MCR se puede distribuir libremente con los archivos de biblioteca generados por el compilador MATLAB.

- Interfaz con otros lenguajes de programación. MATLAB puede llamar funciones y subrutinas escritas en C o Fortran. Se crea una función envoltorio que permite que sean pasados y devueltos tipos de datos de MATLAB. Los archivos objeto dinámicamente cargables creados compilando esas funciones se denominan MEX-files, aunque la extensión de nombre de archivo depende del sistema operativo y del procesador.

• Procedimiento de la programación

El código de Matlab está compuesto como un código de optimización topológica estándar, el cual está listo para ser interpretado por MATLAB luego de llevar a cabo la siguiente serie de sencillos pasos:
1.- Abrir MATLAB y esperar a que éste se inicialice, y muestre su pantalla principal.
2.- Una vez en la pantalla de inicio de MATLAB es necesario seleccionar en la barra de herramientas Home → New → Script, tal como muestra la figura 1, con lo que se abre un editor de texto, dentro del cual será necesario escribir el código proporcionado.

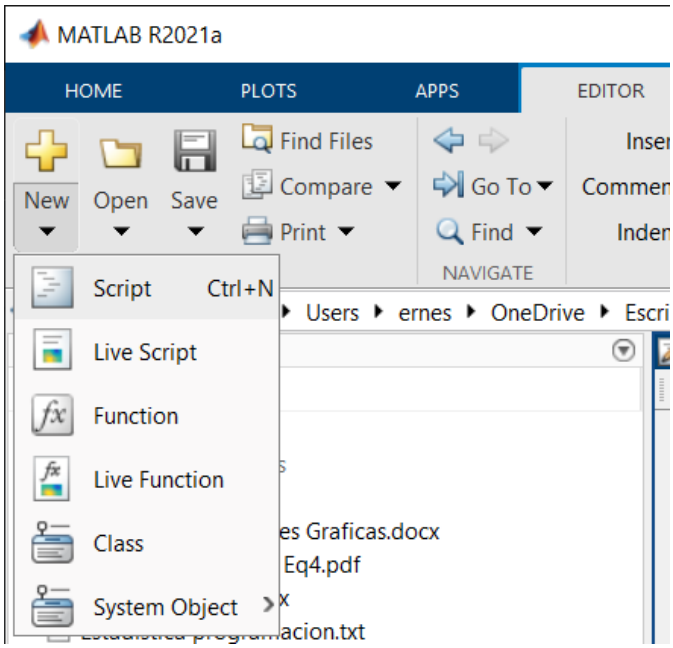

Figure 2: Opción a seleccionar para crear un nuevo script en MATLAB

3.- Una vez con el código completamente escrito en el editor de texto, es necesario salvar el archivo, teniendo especial atención en la ubicación donde se va a guardar el script, así como en el nombre que se le va a asignar al archivo. Se recomienda que el archivo se guarde en el directorio raíz de MATLAB que por default muestra es en el que el editor de texto nos ubica al seleccionar File → Save como muestra la figura 1.2. En caso de no ser así, debemos de navegar a "Mis Documentos/MATLAB" y guardar aquí el script recién creado. La figura 1.2 ejemplifica una script que está siendo guardado en el directorio de MATLAB con el nombre "topp1".

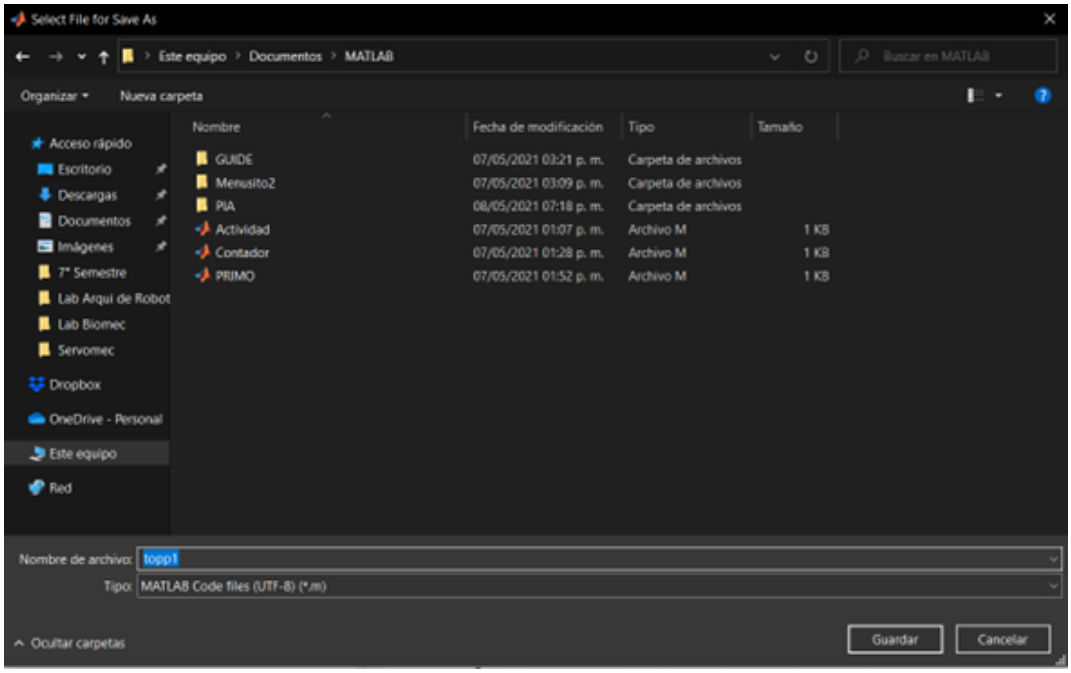

Figure 3: Ventana para guardar script, dentro de la carpeta raíz de MATLAB

4.- Una vez guardado el script en el directorio correcto, solo hace falta corroborar que el intérprete de MATLAB se encuentre en el mismo directorio. Esto se hace desde la pantalla principal de MATLAB. Para la versión R2010a del software, el directorio actual del intérprete se encuentra en la barra de herramientas superior, como muestra la figura 1.3.

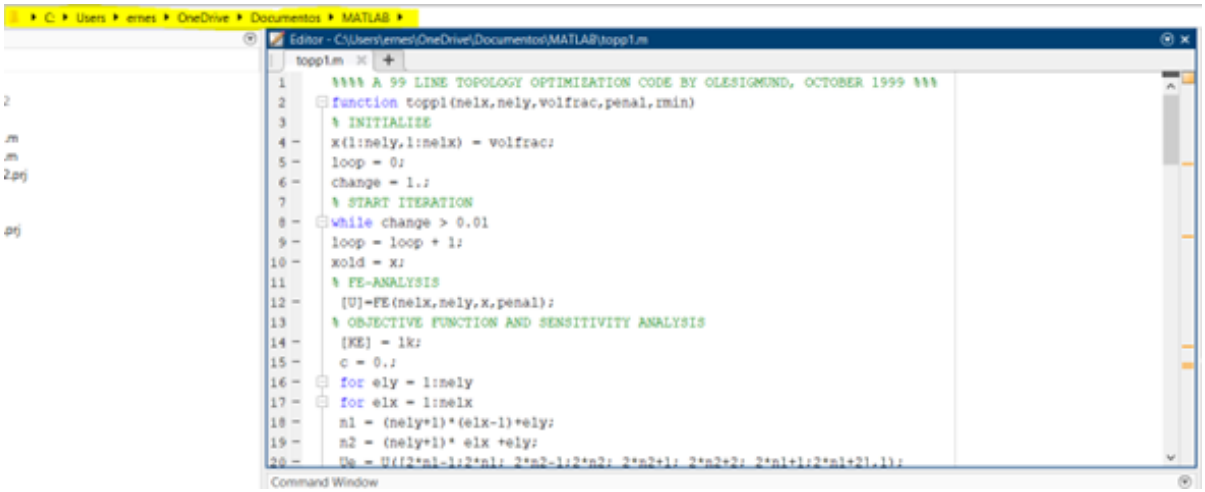

Figure 4: Directorio actual del intérprete de MATLAB

5.- Por último, hay que ejecutar el programa desde la ventana de comando de MATLAB. El código que se proporcionó viene preparado para optimizar un dominio de diseño con cargas y restricciones, este caso en particular es evaluado y simulado cuando escribimos desde la línea de comando de MATLAB "topp1(60,20,0.5,3.0,1.5)".

4 Resultados

• Implementación o desarrollo de la programación en sus diferentes vistas

a) El documento presenta una implementación compacta de Matlab de un código de optimización de topología para minimizar el cumplimiento de estructuras cargadas estáticamente. El número total de líneas de entrada de Matlab es 99, incluido el optimizador y la subrutina de elementos finitos. Las 99 líneas se dividen en 36 líneas para el programa principal, 12 líneas para el optimizador basado en Criterios de Optimidad, 16 líneas para un filtro de independencia de malla y 35 líneas para el código de elementos finitos. De hecho, excluyendo las líneas de comentarios y las líneas asociadas con la salida y el análisis de elementos finitos, se muestra que solo se requieren 49 líneas de entrada de Matlab para resolver un problema de optimización de topología bien planteado. Al agregar tres líneas adicionales, el programa puede resolver problemas con múltiples casos de carga. El código está destinado a fines educativos.

b) Código de 99 líneas

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function topp1(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)];
```

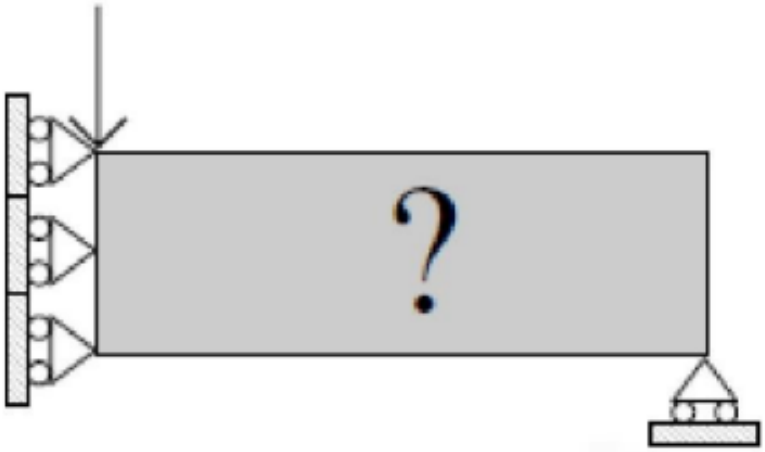

Figure 5: Dominio de diseño, carga y restricciones propuestas.

Describir cada una de las partes principales del código de 99 líneas en Matlab.

Main program.
El programa principal se inicia mediante la distribución del material uniformemente en el dominio de diseño. Después de algunas otras inicializaciones, el bucle principal comienza con una llamada a la subrutina de Elementos Finitos, que devuelve el vector de desplazamiento U.

Optimality criteria based optimizer.
Las variables de diseño actualizados se encuentran por el optimizador. Sabiendo que el volumen de material (sum (sum (xNew))) es una función monótonamente decreciente del multiplicador de Lagrange (lag), el valor de la Lagrangian multiplier que satisface la restricción de volumen puede ser encontrada por un algoritmo biseccionar.

Mesh-independency filtering.
Líneas 49 a 64 representan la aplicación de Matlab. Tenga en cuenta que no todos los elementos en el dominio de diseño son buscados con el fin de encontrar los elementos que se encuentran dentro la rmin radio, pero sólo aquellos dentro de un cuadrado de lado longitudes dos veces la vuelta (Rmin) alrededor del considerado elemento.

Finite element code.
El código de elementos finitos está escrito en las líneas 65-99. Nota que el solucionador hace uso de la opción escasa en Matlab. La matriz de rigidez global está formada por un bucle sobre todos los elementos. Como fue el caso en el principal programa, las variables n1 y n2 denotan superior izquierda y derecha números de nodo elemento y son utilizados para insertar el elemento de matriz de rigidez a la derecha de la matriz de rigidez global.

Después de haber realizado el código 99 líneas en el editor, en la ventana de comando de MATLAB escribimos "topp1(60,20,0.5,3.0,1.5)" y obtenemos lo siguiente en la simulación:

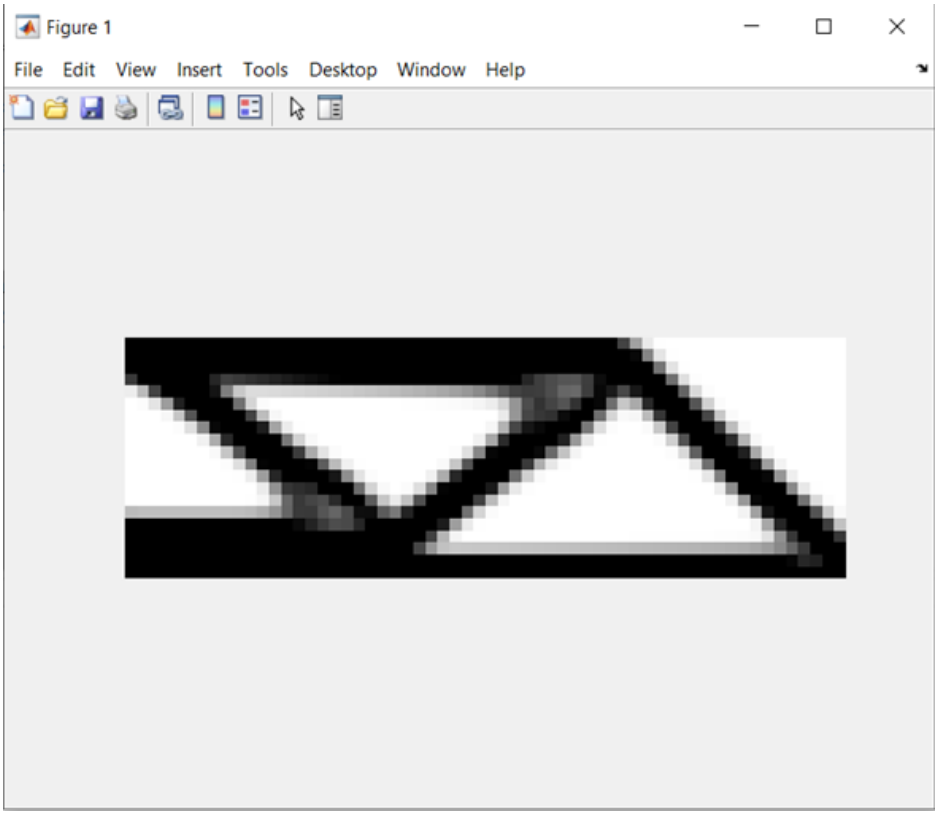

Figure 6: Imagen del diseño optimizado.

c) Definir cada una de las variables de entrada. Conociendo que pusimos dentro del programa 'topp1(60,20,0.5,3.0,1.5)'

nelx, nely (60, 20): Son el número de elementos en las direcciones horizontales (x) y verticales (y).
volfrac (.5): Es la fracción de volumen.
penal (3): Es el poder de penalización.
rmi (1.5): Es el tamaño del filtro (dividido por el tamaño del elemento).

5 Conclusiones

• Manuel Exiquio Barrera Suárez
Gracias a esta actividad comprendimos mejor el potencial que tiene el software de matlab y llevamos a cabo la creación de interfaces gráficas así como diseño de mecanismos. Tambien entendimos la optimización en la creación de ibjetos 3D.

• Ernesto Axel Moreno García
Para concluir con esta actividad nos dimos cuenta que se puede hacer uso del software Matlab para generar un análisis de elemento finito para objetos de ámbito simple y que se pueden usar en diferentes casos semejantes a este. Y por último puedo decir que obtuvimos algunos conocimientos importantes para la realización de simulaciones en MATLAB con los que antes no contábamos ya que conocimos las características y com es que funciona el código de 99 líneas el cual fue muy importante para la realización de esta primera práctica.

• Alberto Alan Ramirez Velazquez
En esta práctica pudimos conocer un poco más a fondo de lo que trata Matlab y todo su entorno de trabajo, siendo un software bastante versátil en la creación de interfaces gráficas, sistemas lineales complejos y poderosa en el

uso de programación para diseño de mecanismos, también fue interesante conocer sobre la optimización topológica para la creación e 3D ya que en esta podemos visualizar cómo se comportan distintas cargas en la pieza a diseñar por nosotros, estas herramientas de computo avanzadas nos sirven para obtener un producto final bastante bien establecido y que brinda de cierta confiabilidad, en general fue una práctica bastante entretenida debido a que logramos diseñar cuando menos un borrador de nuestro proyecto.

• Juan Ángel Hernández Antonio
En la actualidad existen múltiples plataformas y software dedicado al procesamiento matemático, MATLAB es un excelente ejemplo para esto, posee un entorno muy versátil para la creación de interfaces y también para sistemas lineales, con el podemos modelar matemáticamente mecanismos para poder ver el comportamiento que tiene cada uno y así poder comprender más a fondo el funcionamiento, límites y posibilidades de muchos mecanismos empleados día a día.

References

[1] O. Sigmund. A 99 line topology optimization code written in matlab. 2001.