UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FIME
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

LABORATORIO DE BIOMECÁNICA

PERIODO ESCOLAR AGOSTO – DICIEMBRE 2022

Nombre de la actividad
Practica 1

Nombre del alumno	Matricula
Leonardo Daniel De Leon Fuentes	1991978
Fatima Montserrat Castro Nuñez	1991834
Brandon Geovanny Espinosa Alcocer	1938292
Erick Eduardo Landa Gonzalez	1992037

Grupo	Salon	Dia clase	Hora
309	12BMC	Miércoles	N5

Nombre del profesor: Yadira Moreno Vera

Fecha de entrega:

21/09/2022

Objetivo:
El estudiante conocerá cada 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.

Marco teórico:
MATLAB es un acrónimo de "Laboratorio de Matrices". Esto se debe a que fue creado por primera vez como un lenguaje de programación matricial. Se diseñó para ingenieros y científicos. Pero cualquier persona que tenga interés en este programa o lo necesite puede utilizarlo. Matlab es un lenguaje de programación muy conocido y de cuarta generación, como Java, C+. Ahora se utiliza en aplicaciones avanzadas como el aprendizaje automático, el aprendizaje profundo y la ciencia de los datos.

MATLAB es un programa muy importante que muchos estudiantes de ingeniería e ingenieros deben aprender. Ayuda a realizar cálculos matemáticos, diseño, análisis y optimización (estructural y matemática), además de dar velocidad, exactitud y precisión a los resultados.

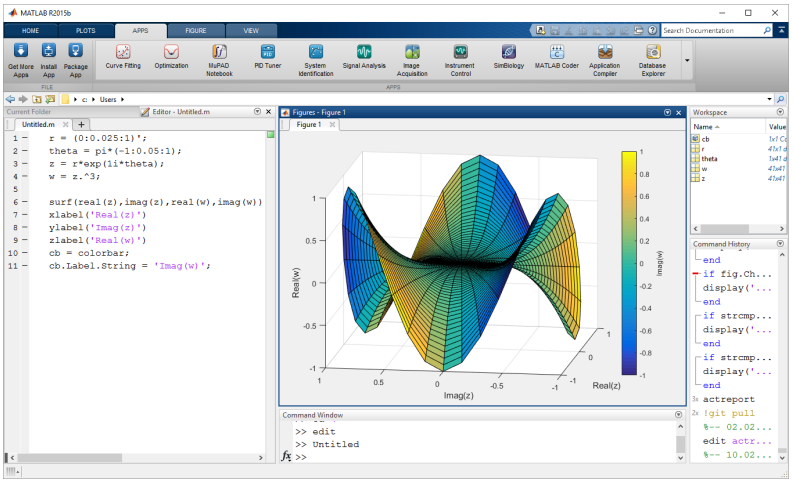


Figura 1. Interfaz de MATLAB

MATLAB se utiliza especialmente en aplicaciones de ingeniería. Se utiliza con frecuencia en el análisis de sistemas y cálculos matemáticos y su visualización. Los principales usos son:

- Cálculos de álgebra lineal numérica
- Aprendizaje automático
- Aprendizaje profundo
- Ciencia de los datos
- Simulación
- Creación de gráficos para Big Data
- Análisis y visualización de datos
- Desarrollo de algoritmos

- Creación de una interfaz gráfica de usuario y de una interfaz de programación de aplicaciones

Puede aplicar y diseñar diferentes algoritmos con MATLAB.
Puede cargar datos de diferentes fuentes, como archivos, bases de datos o Internet, en MATLAB. Al mismo tiempo, puede analizarlos. Además, puedes visualizarlos con varias opciones.

MATLAB dispone de una amplia biblioteca de funciones matemáticas. Así, puedes realizar el cálculo de álgebra lineal y matrices como producto matemático.

También puede simular modelos de datos, prototipos y productos que diseñe o calcule con Matlab. También puede optimizarlos para utilizarlos eficazmente con otros programas. Puede diseñar varias interfaces específicas para usted.
La optimización de la topografía ayuda a los fabricantes a diseñar y optimizar cualquier pieza de paredes finas. Al igual que la piel de un tambor, estas estructuras de láminas finas pueden excitarse fácilmente, provocando ruidos indeseables, vibraciones e incluso daños en determinadas condiciones.

La optimización de la topografía permite diseñar estructuras de paredes finas, como los paneles del suelo de los automóviles, para obtener la máxima rigidez, respuesta en frecuencia u otros objetivos de rendimiento.

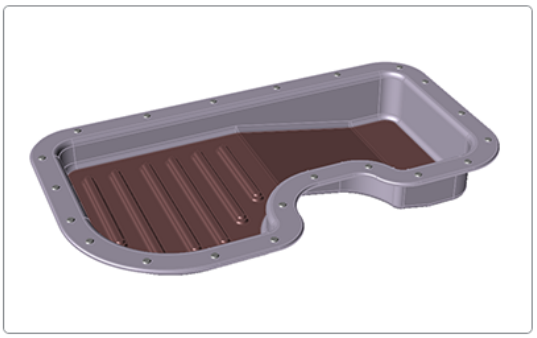


Figure 1. Design Space

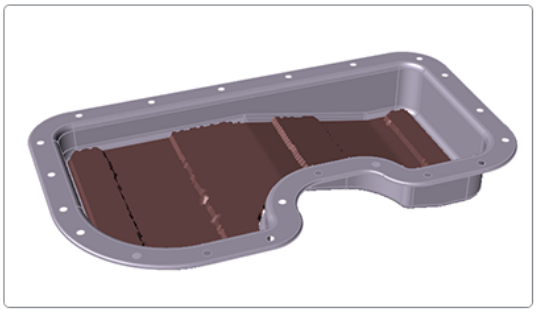


Figure 2. Optimized Shape

Figura 2. Optimización de una topografía.

Para mejorar las características de las vibraciones, se añaden modificaciones locales de la forma, como los cordones, para aumentar la rigidez. La mayoría de las veces, la ubicación, la forma y la orientación de estos cordones se basan en la geometría natural de la pieza y en la experiencia del diseñador. La optimización de la topografía permite a los diseñadores definir los espacios en los que pueden y no pueden añadirse cordones, la anchura de los cordones, así como la dirección de trazado, el ángulo y la altura. Esto significa que sólo se generan diseños prácticos, con patrones óptimos.

Un patrón de cuentas optimizado por el software suele superar con creces un diseño tradicional, maximizando la rigidez, la respuesta en frecuencia u otros objetivos de rendimiento, sin añadir masa ni complejidad de fabricación.

Desarrollo de la práctica:

1) **Nombre y definición de la programación mencionar un ejemplo de la forma de la geometría:**

El código Matlab que se presenta en este documento está destinado para la enseñanza de la ingeniería.

Se han introducido una serie de simplificaciones para simplificar el código Matlab.
En primer lugar, se supone que el dominio de diseño es rectangular y discretizado por elementos finitos cuadrados.

De este modo, la numeración de los elementos y los nodos es sencilla y la relación de aspecto de la estructura viene dada por la relación de elementos en la dirección horizontal (nelx) y vertical (nely).

Para garantizar la existencia de soluciones al problema de optimización de la topología, hay que introducir algún tipo de restricción en el diseño resultante debe introducirse

El código de Matlab, se construye como un código de optimización topológica estándar. El programa principal es se llama desde el prompt de Matlab con la línea

```
top(nelx,nely,volfrac,penal,rmin)
```

donde ***nelx*** y ***nely*** son el número de elementos en las direcciones horizontal y vertical, respectivamente, ***volfrac*** es la fracción de volumen, ***penal*** es la potencia de penalización y ***rmin*** es el tamaño del filtro (dividido por el tamaño del elemento). Otras variables de variables, así como las condiciones de contorno, se definen en el propio código de Matlab y pueden editarse si es necesario. Para cada iteración del bucle de optimización de la topología, el código genera una imagen de la distribución de la densidad de corriente.

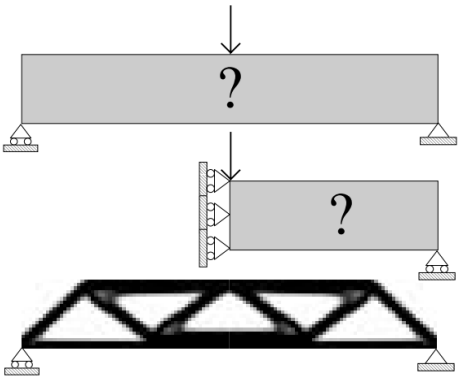


Figura 1.1. Optimización topológica de viga.
Arriba: dominio de diseño completo.
Medio: dominio de medio diseño con simetría condiciones de borde.
Abajo: viga optimizada de topología resultante (ambas mitades).

2) Estado del arte.

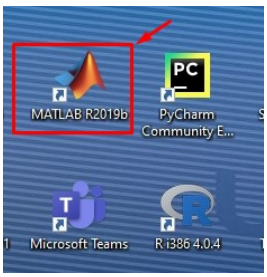
La unión ideal entre la fabricación aditiva (AM) y la optimización estructural (SO) es el elemento clave en el desarrollo de productos hoy en día. Por un lado, los modelos se producen mediante la adición de miles de capas con el uso de la fabricación aditiva (AM). Esto ofrece a los diseñadores una enorme flexibilidad geométrica, sin coste adicional, en comparación con la fabricación tradicional. La AM engloba muchas tecnologías, como la impresión 3D, la creación rápida de prototipos y la fabricación digital directa (DDM). Por otra parte, la optimización estructural reduce el uso de materiales, acorta el ciclo de diseño y mejora la calidad del producto. El SO puede aplicarse según el tamaño, la forma y la topología

La optimización de la topología suele denominarse como optimización general de la forma. La mayoría de las técnicas optimizan o bien la topología o tanto el tamaño como la forma. Sólo hay unos pocos ejemplos que han tratado de afrontar el problema de forma integral.

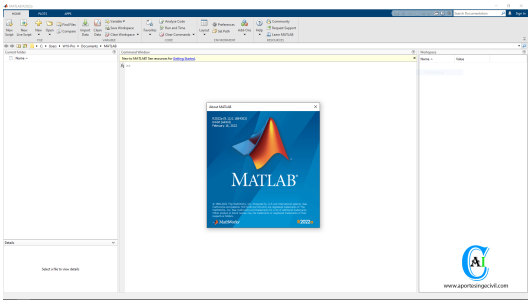
El estado actual del arte de la optimización de la topología (TO) está más orientado en la fase de fase de diseño conceptual. La idea general es encontrar la distribución óptima de materiales de una estructura con respecto a con respecto a su diseño y a las restricciones de los límites. Sin embargo, el principal reto de la TO es proporcionar una parametrización de diseño diseño que conduzca a un diseño físicamente óptimo también.

Los métodos más conocidos de optimización topológica son: el material sólido isotrópico con penalización (SIMP) y la optimización estructural evolutiva (ESO) o la optimización estructural estructural evolutiva (BESO)

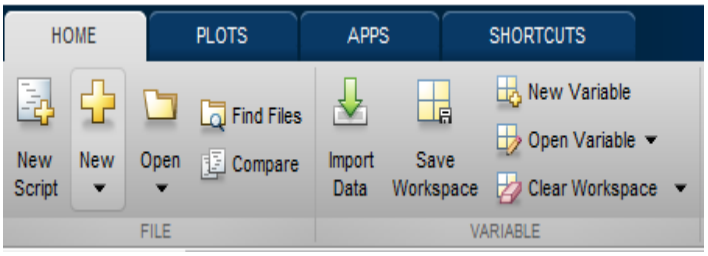
3) Procedimiento de la programación.



El código de matlab se compone como un código de optimización topológica estándar y este ala ves está preparado para interpretarlo.

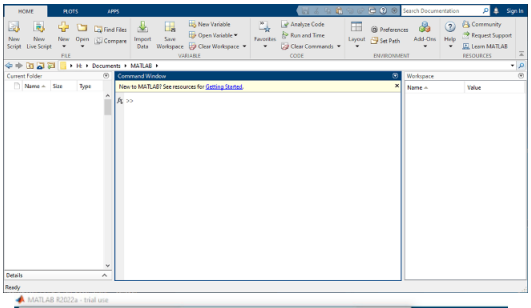


Pasamos a abrir el programa Matlab

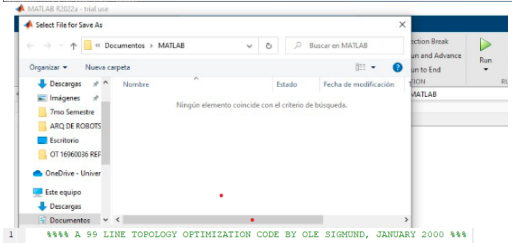


seleccionamos

HOME/New Script



En la pantalla de trabajo se le pone el nombre para guardarlo.



Se guarda en SAVE AS y pones la ubicación y nombre

```
%%%% A 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)^penal*Ue'*KE*Ue;
      dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
    end
  end
```

Se pone el codigo y se guarda con los datos correctos

4) Implementación y desarrollo de la programación en sus diferentes vistas.

El código de 99 líneas se divide en cuatro secciones principales, las cuales son: creación del programa principal, optimizador basado en el criterio de optimización, filtro de mallado independiente, y código de elemento finito.

Programa principal (Líneas 1-37).

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

Figura 4.1. Código principal.

Se comienza distribuyendo el material uniformemente en el dominio del diseño (*%INITIALIZE*), se hacen unas otras inicializaciones (*% START ITERATION*) y después se obtiene el vector U llamando a la subrutina de elementos finitos (*%FE-ANALYSIS*). Se inicia un ciclo el cual determina función objetivo y sensibilidades (*% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS*), en donde las variables n1 y n2 denotan arriba a la izquierda y a la derecha números de nodo de elementos en números de nodo globales. Después se llama al filtro de independencia de malla (*% FILTERING OF SENSITIVITIES*) y aplicar el optimizador en base al criterio de optimización. (*% DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD*). Se imprime el resultado y también la representación gráfica (*% PRINT RESULTS* y *% PLOT DENSITIES*) El bucle principal se termina si el cambio en las variables de diseño es menor que 1%.

```
26   % FILTERING OF SENSITIVITIES
27     [dc]   = check(nelx,nely,rmin,x,dc);
28   % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
29     [x]    = OC(nelx,nely,x,volfrac,dc);
30   % PRINT RESULTS
31     change = max(max(abs(x-xold)));
32     disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',c) ...
33           ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
34            ' ch.: ' sprintf('%6.3f',change )])
35   % PLOT DENSITIES
36     colormap(gray); imagesc(-x); axis equal; axis tight; axis off;pause(1e-6);
37   end
```

Optimizador basado en criterios de optimización (líneas 37–49).

```
%%%%%%%%%% 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
```

Figura 4.2. Actualización de los criterios de optimalidad.

En esta sección se encuentran las variables de diseño actualizadas. Como se sabe que el volumen del material es una función monótonamente decreciente del multiplicador de Lagrange (retraso), el valor del multiplicador Lagrangiano que satisface la restricción de volumen se puede encontrar mediante un algoritmo de bisección. El algoritmo de bisección se inicia proponiendo un l1 inferior y un límite l2 superior para el Lagrangiano multiplicador. El intervalo que limita el multiplicador lagrangiano se divide repetidamente por la mitad hasta que su tamaño es menor que los criterios de convergencia.

Filtrado de independencia de malla (líneas 50 a 65).

```
%%%%%%%%%% 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-floor(rmin),1):min(i+floor(rmin),nelx)
      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
```

Figura 4.3. Filtrado de independencia de malla.

Se selecciona *rmin* menor a uno en la llamada de la rutina y así las sensibilidades filtradas serán iguales a las sensibilidades originales haciendo el filtro inactivo. Tenga en cuenta que no se buscan todos los elementos en el dominio de diseño para encontrar los elementos que se encuentran dentro del radio *rmin*, sino solo aquellos dentro de un cuadrado con longitud de lados dos veces el valor de alrededor del elemento considerado.

Código de elementos finitos (líneas 65 a 99).

```
%%%%%%%%%% 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 = zeros(2*(nely+1)*(nelx+1),1);
for elx = 1:nelx
  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)^penal*KE;
  end
end
% DEFINE LOADS AND 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
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)];
```

Figura 4.4. Código de elementos finitos.

La matriz de rigidez global está formada por un bucle sobre todos los elementos. Las variables n1 y n2 denotan arriba a la izquierda y a la derecha números de nodo de elementos en números de nodo globales y se utilizan para insertar la matriz de rigidez del elemento en los lugares correctos en la matriz de rigidez global.

Además, cada nodo tiene dos grados de libertad (horizontal y vertical), por lo que el comando indica que se está aplicando una unidad de fuerza vertical en la esquina superior izquierda. Los apoyos son implementados mediante la eliminación de los grados de libertad de las ecuaciones lineales (*% DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)*).

Se calcula la matriz de rigidez del elemento (*% ELEMENT STIFFNESS MATRIX*). La matriz de 8 por 8 para un elemento cuadrado bilineal de 4 nodos se determinó analíticamente utilizando un software de manipulación simbólica. El módulo de Young E y la relación de Poisson *nu* pueden ser editados.

Resultados:

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

```
>> top(60,20,0.5,3.0,1.5)
 It.:    1 Obj.:  1007.0221 Vol.:  0.500 ch.:  0.200
 It.:    2 Obj.:   579.5598 Vol.:  0.500 ch.:  0.200
 It.:    3 Obj.:   412.5078 Vol.:  0.500 ch.:  0.200
 It.:    4 Obj.:   343.9883 Vol.:  0.500 ch.:  0.200
 It.:    5 Obj.:   322.0344 Vol.:  0.500 ch.:  0.193
 It.:    6 Obj.:   308.4706 Vol.:  0.500 ch.:  0.200
 It.:    7 Obj.:   298.5664 Vol.:  0.500 ch.:  0.171
 It.:    8 Obj.:   289.0954 Vol.:  0.500 ch.:  0.192
 It.:    9 Obj.:   280.6622 Vol.:  0.500 ch.:  0.131
 It.:   10 Obj.:   272.6553 Vol.:  0.500 ch.:  0.131
```

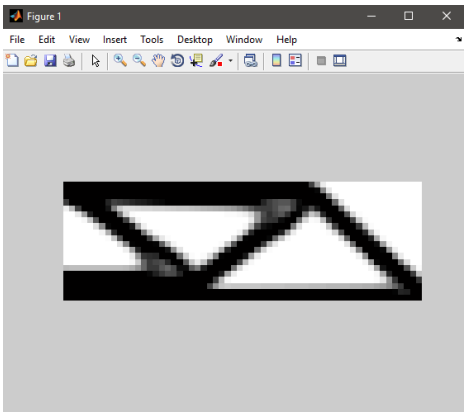


Figura 4.5. Resultados usando top(60,20,0.5,3.0,1.5).

```
top(30,10,0.5,3.0,1.5)
```

```
>> top(30,10,0.5,3.0,1.5)
 It.:    1 Obj.:   984.5548 Vol.:  0.500 ch.:  0.200
 It.:    2 Obj.:   580.4116 Vol.:  0.500 ch.:  0.200
 It.:    3 Obj.:   419.0640 Vol.:  0.500 ch.:  0.200
 It.:    4 Obj.:   357.2235 Vol.:  0.500 ch.:  0.194
 It.:    5 Obj.:   337.8750 Vol.:  0.500 ch.:  0.125
 It.:    6 Obj.:   327.5757 Vol.:  0.500 ch.:  0.141
 It.:    7 Obj.:   319.9654 Vol.:  0.500 ch.:  0.105
 It.:    8 Obj.:   312.9614 Vol.:  0.500 ch.:  0.113
 It.:    9 Obj.:   307.3377 Vol.:  0.500 ch.:  0.090
 It.:   10 Obj.:   302.7199 Vol.:  0.500 ch.:  0.095
```

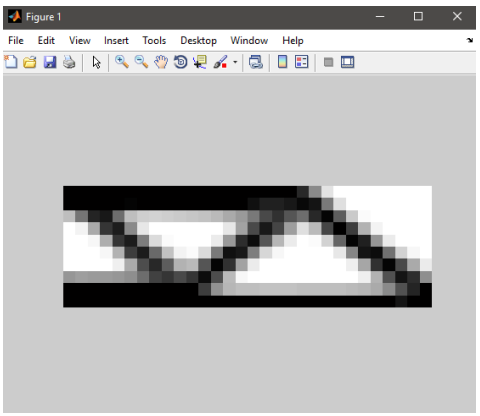


Figura 4.6. Resultados usando top(30,10,0.5,3.0,1.5).

5) Conclusiones:

Leonardo Daniel De Leon Fuentes 1991978

En esta práctica se conoció más acerca de la optimización de topología como esta nos puede ayudar para mejorar una pieza y las ventajas que se obtiene gracias a esta, por ejemplo se minimiza la cantidad de material, se obtiene un diseño más optimizado e incluso se puede reducir el precio de costo en algunos casos, también en esta práctica se hizo uso del software matlab que nos ayuda a resolver problemas con una gran cantidad de operaciones pero el principal conocimiento que me llevo de esta práctica es saber que es la optimización topológica, como se desarrolla y para qué nos sirve.

Fatima Montserrat Castro Nuñez 1991834

Aprendimos sobre los algoritmos matemáticos para la optimización topológica haciendo uso de elementos finitos dentro de un código en Matlab, esto es algo que me parece muy interesante, ya que es una nueva aplicación que no conocía de Matlab y me parece que es muy útil ya que optimiza estructuras con pocas lineas de codigo, como lo vimos en este ejemplo, en donde solo se necesitaron 99 líneas. Creo que la optimización topológica en el área de la biomecánica es muy necesaria ya que se requiere fabricar prótesis fáciles de llevar, que no generen tanto peso o que sean más robustas de lo necesario.

Brandon Geovanny Espinosa Alcocer

En esta práctica se pudo observar la optimización topológica y en cada tipo de optimización se pudo observar como cada una de las estructuras o cada objeto se puede llegar a mejorar según lo que se necesite o sus requerimientos que nos pida o que se puedan llegar a ver en ellos. En este podemos decir que es muy importante no saltarse ningún tipo de cálculo y establecer algunas normas predeterminadas para poder hacer una pieza o diseñar estructuras.

Referencias:

Huseyin, M. (2021, marzo 6). What is Matlab? Why we need it? Dev Genius. https://blog.devgenius.io/what-is-matlab-why-we-need-it-d61e405ef419

Ponginan, R. (2021, septiembre 2). What is topography optimization. Altair University. https://altairuniversity.com/52523-what-is-topography-optimization/

Sigmund, O. (2001). A 99 line topology optimization code written in Matlab. Structural and Multidisciplinary Optimization: Journal of the International Society for Structural and Multidisciplinary Optimization, 21(2), 120–127. https://doi.org/10.1007/s001580050176