Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

LABORATORIO DE BIOMECANICA

PRÁCTICA 1

Optimización Topológica

Docente: Dra. Yadira Moreno Vera

Día: Martes

Hora: V2

Brigada: 214 ***Equipo***: 2

Nombre	**Matricula**	**Carrera**
Viviana Nathalie Tienda Téllez	1919910	IMTC
Mauricio Guerrero Hernández	1905306	IMTC
Karla Gabriela Torres García	1910427	IMTC
Brayan Uriel Grimaldo Salazar	1908530	IMTC
Alina Martínez Escobedo	1912818	IMTC

Fecha de entrega: 05-septiembre-2022

OBJETIVO:

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

INTRODUCCION

En este presente trabajo se presenta una propuesta de análisis de formas de programación, de sus características de trabajo específicas que se presentan siguiendo una metodología con el objetivo de conocer cada una de las secciones que integran el código de optimización topológica, como crear el archivo (.m) en el programa de MATLAB y como ejecutar el análisis correspondiente.

Se propone un código de optimización topológica definiendo las condiciones iniciales que declaramos al inicio de la práctica para después poder realizar la simulación del programa y notar el cómo funciona la optimización topológica. La estructura por realizarse se le aplicara una optimización a una barra a la cual se le someten distintas cargas por lo que el objetivo mismo es que el programa vaya optimizando dicha estructura para obtener las mismas o mejores propiedades con menor material y con un acabado más estético.

NOMBRE Y DEFINICIÓN DE LA PROGRAMACIÓN

La pieza en un inicio es un rectángulo dividido en una cantidad finita de cuadros, con esto el número de elementos y nodos son sencillos de saber, de igual manera, el conocer el radio de nuestra pieza lo podremos conocer gracias al radio de los elementos que estén de manera horizontal y vertical. Los problemas que conllevan este tipo de estructuras pueden ser resueltos mediante distintos métodos, por ejemplo, Criterios de Optimalidad, Programas de Secuencia Lineal, Método de Asíntotas en Movimiento; para nuestra pieza utilizaremos los Criterios de Optimalidad ya que este es el más sencillo. En cuanto restricciones, se utilizará una clase de filtro que ayudará a hacer mallas independientes en la pieza y mejorará su diseño.

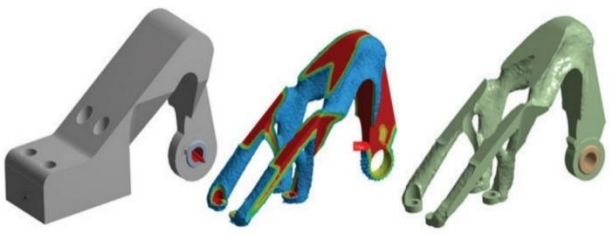
ESTADO DEL ARTE

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, como se puede ver en el ejemplo de la Figura1. 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 [1].

Algunas aplicaciones en las que se han utilizado métodos de optimización son las siguientes:

-Figura1: Ejemplo de optimización topológica

- Problemas en dinámica. Problema relevante para el diseño de máquinas o estructuras en las que se involucren cargas dinámicas, en los que se busca maximizar o minimizar las eigenfrecuencias dependiendo de la aplicación.
- Problemas de pandeo. Para la maximización de la primera carga crítica de pandeo en una estructura.
- Limitaciones de esfuerzo. Se consideran distintos criterios para poder resolver y proponer problemas de estructuras dado un valor de esfuerzo definido.
- Cargas de presión. Son problemas comunes en los que se busca minimizar los parámetros de diseño como las densidades del material volumétrico a través de todo el diseño. El problema radica en cómo se considera el tipo de carga cuando los dominios de la superficie están sujetos fuerzas que cambian en tamaño, dirección y tamaño debida a la distribución del material. Ejemplos de estas son las presiones de algún fluido.
- Problemas geométricos no lineales. Son aquellos en los que se utiliza el análisis de elemento finito con geometrías no lineales



para elementos como estructuras blandas, estructuras esbeltas o mecanismos.

- Mecanismos compatibles. Aquellas estructuras o mecanismos que están diseñados para ser flexibles y otorgar cierto grado permisible de movilidad sin utilizar piezas móviles. Se requiere un diseño que minimice el riesgo de fallo por fatiga.
- Diseño de soportes. Son los problemas en los que no se requieren diseños por distribución de material; sino que se diseña y se analiza, la cantidad de soportes y su posición de ellos en una estructura.
- Diseño de materiales. Son problemas en los que se aplican ideas de la optimización topológica para diseñar la estructura de un material.

Los softwares dedicados a la optimización topológica:

No todos los software CAD ofrecen esta función de optimización topológica. Aunque no se utiliza necesariamente en el proceso de modelado y fabricación, puede ser mejor optar por un programa que la incluya, para asegurarse de diseñar una pieza óptima. Es por ello que algunas empresas han decidido desarrollar un software dedicado a este proceso. Uno de los pioneros es sin duda Altair, con su solución OptiStruct, que luego llevó a otra solución, Altair Inspire. También existen otras opciones como la de Ansys, Dassault Systèmes, Autodesk o incluso nTopology. Además, hay muchos programas de CAD que integran funciones para optimizar las piezas, como Solidworks, Creo o Fusion 360 [2].

¿Dónde se utiliza actualmente la optimización topológica?

La industria automotriz abordó rápidamente este problema debido a la reducción de costes mediante el ahorro en materias primas asociadas con los tamaños de serie. De hecho, la reducción de unos pocos gramos por cada vehículo, en una producción de varios millones de unidades, representa toneladas de material ahorrado. Como ejemplo de esto tenemos el chasis impreso en 3D del Light Rider, una pieza que pesa solo 6 kilos gracias a una distribución óptima del material. Más recientemente, está la parte de suspensión del Fiat Chrysler Automobiles, que reúne más de 12 componentes diferentes en uno mismo. Al centrarse en la optimización topológica, los diseñadores redujeron su peso final en un 36%.

La aeronáutica es sin duda otro sector interesado en la optimización topológica, con el objetivo de reducir costes indirectos. Un avión más ligero consume menos combustible, lo que, a la larga, genera importantes ahorros para una aerolínea. Esto es lo que el diseñador

Andreas Bastian demostró con sus asientos de avión. Los diseñó un 54% más ligeros, que, en su conjunto, supondría una reducción muy significativa del peso de un avión. Más allá del peso, la optimización topológica permite, especialmente al sector aeronáutico, imaginar formas mucho más complejas ya que la industria se libera de las limitaciones impuestas por los moldes [3].

Los pasos que se requieren para realizar la optimización topológica:

1. Dibujar o Importar geometría
2. Simplificar la pieza y definir el espacio de diseño
3. Establecer uniones, juntas y contactos
4. Asignar materiales
5. Definir los casos de carga
6. Generar la optimización
7. Refinar la geometría
8. Exportar a CAD o generar STL
9. Verificar el rendimiento
10. Fabricar

MATLAB es un lenguaje de programación, así como un entorno computacional interactivo, que permite realizar diferentes tareas, tales como: manipulaciones matriciales, trazado de funciones y datos, implementación de algoritmos, creación de interfaces de usuario, programas en otros lenguajes (C, C ++, Java y FORTRAN), analizar datos, desarrollar algoritmos y crear modelos o aplicaciones [5].

MATLAB también permite escribir una serie de comandos en un archivo y ejecutar el archivo como una unidad completa, como escribir una función y llamarla [4].

Los .m files son archivos de texto donde guardamos una secuencia de comandos. Dichos archivos son guardados con extensión .m y pueden ser de dos tipos:

- Scripts: En estos archivos, escribe una serie de comandos que desea ejecutar juntos. Los scripts no aceptan entradas y no devuelven ninguna salida. Operan con datos en el espacio de trabajo.
- Functions: Las funciones pueden aceptar entradas y devolver salidas. Las variables internas son locales a la función.

La manera de crear archivos de secuencias de comandos (.m) es utilizando un editor de texto, este se puede abrir el editor MATLAB de dos formas:

- Usando el símbolo del sistema
- Usando el IDE

Si se está utilizando el símbolo del sistema, se debe escribir **edit** en este para abrir el editor. Igualmente, se puede escribir directamente **edit** y luego el nombre del archivo con extensión.m [4].

PROCEDIMIENTO DE PROGRAMACION

Las 99 líneas del programa están divididas en 36 líneas para el programa principal, 12 líneas para la optimización, 16 para el filtro de malla y 35 líneas para el código de elemento finito.

Programa principal (líneas 1-36)

El programa principal (líneas 1-36) empieza distribuyendo el material en partes iguales en el dominio del diseño (línea 4). después de otras inicializaciones, el bucle principal empieza llamando a la subrutina del elemento finito (línea 12) que regresa el desplazamiento del vector U.

Ya que la matriz de rigidez de elemento para el material solido es la misma para todos los elementos, la subrutina es llamada solo una vez (línea 14). Después de esto, un bucle sobre todos los elementos determinara la función objetiva y las sensibilidades (4). Las variables n1 y n2 indican los números de nodos de elementos superiores izquierdo y derecho y son usados para extraer el vector de desplazamiento Ue del desplazamiento global del vector U. El análisis de sensibilidad es seguido al llamar al filtro de malla independiente (línea 26) y al optimizador (línea 28).

El cumplimiento, así como otros parámetros son impresos por las líneas 30-33 y la distribución de densidad resultante se traza (línea 35). El bucle principal se termina si el cambio en las variables de diseño es menos al 1%. De lo contrario se repetirán los pasos anteriores.

Optimización basada en criterios de optimalidad (líneas 37-48)

Las variables de diseño actualizadas se encuentran en el optimizador (líneas 37-48). Sabiendo que el volumen del material (sum(sim(xnew)) es una función decreciente monótona de los multiplicadores de Lagrange (lag), el valor de los multiplicadores de Lagrange que satisface la restricción de volumen se puede encontrar mediante un algoritmo de bifurcación (líneas 40-48). Este algoritmo se inicializa adivinando un valor l1 inferior y un l2 superior para el lagrangiano (línea 39). El intervalo que limita el multiplicador de Lagrange se reduce a la mitad repetidamente hasta que su tamaño es menor a los criterios de convergencia (línea 40).

Filtrado de malla independiente (líneas 49-64)

Las líneas 49-64 representan la implementación de Matlab de (5). Es importante notar que no todos los elementos en el dominio del diseño son buscados con el fin de encontrar los elementos que se encuentran dentro del radio rmin pero solo los que se encuentren dentro de un cuadrado con lados de longitud dos veces redondas (rmin) alrededor del elemento considerado. Al seleccionar menos rmin que uno en la rutina, las sensibilidades filtradas serán iguales a las sensibilidades originales que hacen el filtro inactivo.

Código de elemento finito (líneas 65-99)

El código de elemento finito esta escrito en las líneas 65-99. Tenga en cuenta que el solucionador hace uso de las escasas opciones en Matlab. La matriz global rígida se forma por un bucle sobre todos los elementos (líneas 70-77). Como fue en el caso del programa principal, las variables n1 y n2 denotan los números de nodos de elementos superior izquierda y derecha y son usados para insertar la matriz de rigidez de los elementos en los lugares correctos en la matriz global. Como se menciono antes, ambos nodos y elementos son numerados en columnas de izquierda a derecha. Además cada nodo tiene dos grados de libertad (horizontal y vertical), por lo que se aplica el comando F(2,1)=-1 (línea 79) una fuerza unitaria vertical en la esquina superior izquierda.

Los soportes se implementan al eliminar los grados de libertad arreglados de las ecuaciones lineales. Matlab puede hacer esto con la línea:

84 U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);

Donde freedofs indican los grados de libertad que no están restringidos. Mayormente, es más fácil definir los grados de libertad que están arreglados (fixeddofs) por lo que los freedofs son encontrados automáticamente usando el operador de Matlab setdiff que encuentra los grados de libertad libres como la diferencia entre todos los grados de libertad y los grados de libertad arreglados. La matriz de rigidez es calculada en las lieas 86-99. La matriz de 8x8 por el cuadrado bi-linear elemento de 4 nodos fue determinado analíticamente usando un software de manipulación simbólica. El modulo de Young y el redio de Poison nu puede ser alterado en las líneas 88 y 89.

IMPLEMENTACION DE LA PROGRAMACION

En base a la explicación del procedimiento realizado para la programación de la optimización topología en Matlab, ahora el siguiente paso a realizar es el corroborar que el código propuesto funcione correctamente. Para la implementación de este código lo único necesario es el copiar el siguiente programa en el editor de Matlab.

```
%%%% Codigo de 99 Lineas OPTIMIZACION TOPOLOGICA %%%%
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-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
%%%%%%%%%% 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)];
```

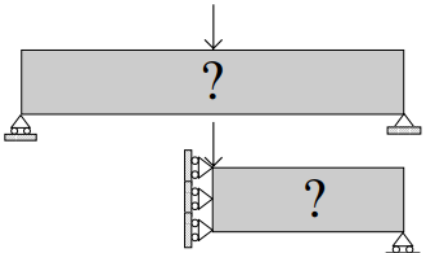
-Código de optimización topológica

Para correr la simulación de la optimización se deben de llamar con la siguiente línea:

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

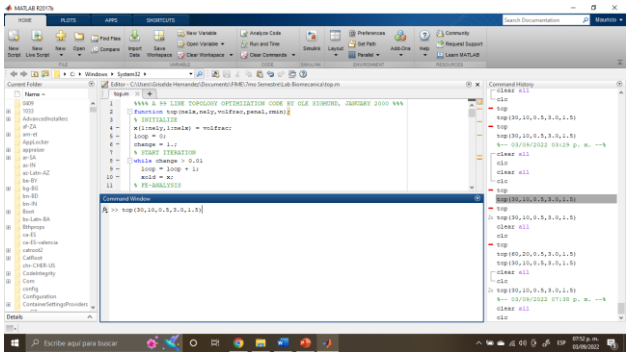
Donde ***nelx*** y ***nely*** son el número de elementos en el direcciones horizontal y vertical respectivamente, ***volfrac*** es la fracción de volumen, ***penal*** es el poder de penalización y ***rmin*** es el tamaño del filtro (dividido por el tamaño del elemento). Otras variables, así como las condiciones de contorno se definen dentro del propio código de Matlab. Dichas condiciones se pueden modificar en caso de querer ciertas propiedades específicas y de esta manera tener distintas observaciones del comportamiento del código ante distintas entradas. Por lo que por razones prácticas en este caso las condiciones a utilizar son las siguientes:
>> top (30,10,0.5,3.0,1.5)

Una vez definidas las condiciones iniciales de la práctica se proseguirá a realizar la simulación del programa para poder notar el cómo funciona la optimización topológica. Cabe mencionar que la estructura a realizarle la optimización es una barra a la cual se le someten distintas cargas por lo que a continuación se mostraran imágenes de la estructura original (Figura2) y de cómo el mismo programa va optimizando dicha estructura para obtener las mismas o mejores propiedades con menor material y con un acabado más estético.

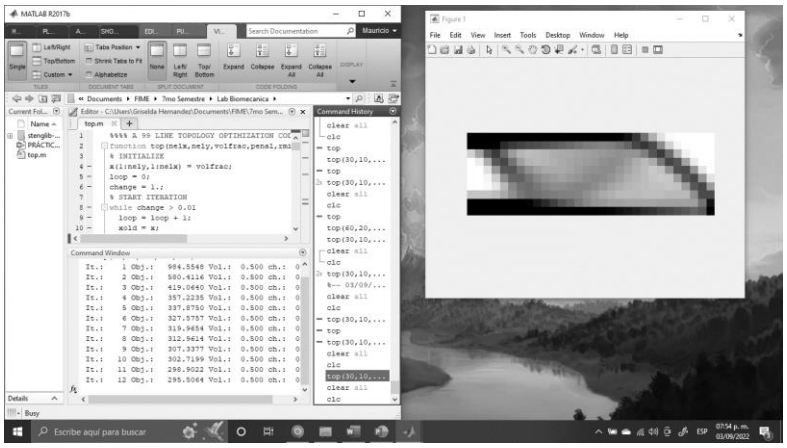


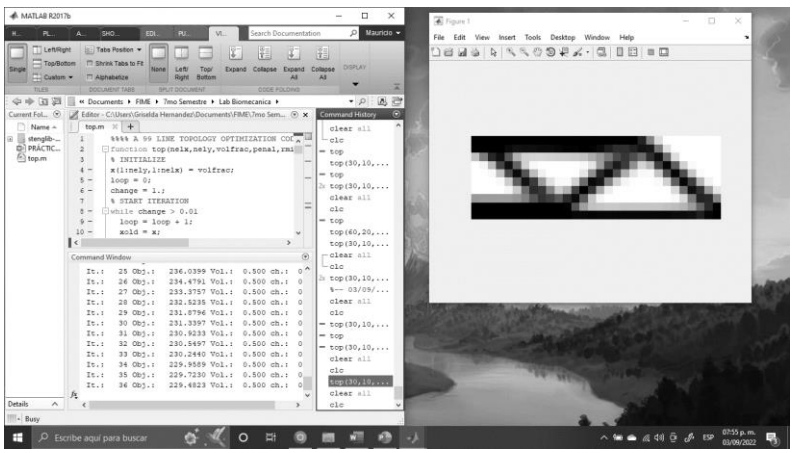
-Figura2: Estructura

Optimización:

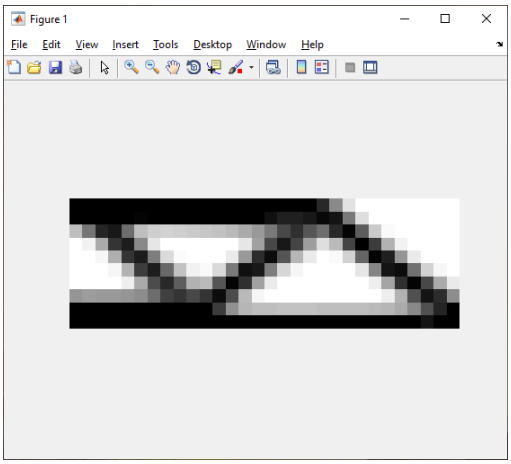


-Definición de condiciones iniciales en Matlab





-Simulación de la optimización topológica



En base al código propuesto, se puede observar que, con respecto a la figura propuesta de la barra uniforme sometida a cargas, se puede observar que efectivamente existe una gran diferencia entre cada una de las formas, obteniendo en la optimización una estructura compuesta de barras y con mucho menor material necesario para ser elaborada, más sin embargo cumpliendo aun así con las propiedades establecidas en la figura uniforme.

-Optimización topológica completa

CONCLUSIONES

Mauricio Guerrero Hernández:

En base a lo visto en esta práctica, lo que más se logra destacar es la comprensión de del concepto de optimización topológica para que, de esta manera, reconozcamos la importancia de la utilización de este concepto hoy en día en la industria, como podría ser en el diseño automotriz para poder mejorar la aerodinámica de una carrocería, en la industria para mejorar las propiedades de alguna estructura de una pieza de maquinaria, en el diseño y elaboración de prótesis, o inclusive para ahorrar el costo de material necesario para su elaboración, entre muchas otras aplicaciones las cuales como se mencionó anteriormente son muy solicitadas en distintos sectores laborales. De igual manera cabe destacar que la programación de un código en Matlab resulto un gran reto debido a que no es muy habitual el realizar esta clase de optimizaciones en algún lenguaje de programación y por lo general es más habitual el verlo dentro de algún software de diseño 3D, por lo que me resulto realmente interesante las condiciones necesarias para poder hacer esta clase de optimizaciones en un lenguaje de programación destinado principalmente en las matemáticas, ayudándonos de igual manera a comprender la importancia de conocer acerca de las optimizaciones topológicas y no cerrarnos de mente y explorar otros softwares en los cuales somos capaces de realizar diversas tareas relacionadas a nuestra carrera.

Brayan Uriel Grimaldo Salazar:

Al término de este reporte, se cumplió el objetivo planteado al inicio, el cual era conocer cada una de las secciones que integran el código de optimización topología, por lo que al momento de estar realizando la práctica logramos la comprensión del concepto de optimización topológica y reconocer su aplicación en los diversos campos, logramos crear el archivo correspondiente de MATLAB (m.) y la ejecución del mismo, además de la implementación o desarrollo de la programación en sus diferentes vistas, por lo que gracias a estos se pueden lograr diversas cosas como la optimización topológica y de las diferentes aplicaciones que estas conlleva.

Karla Gabriela Torres García:

Realizando la debida investigación para la práctica, pude observar lo que la optimización topológica era, lo que sus beneficios y análisis representa a las estructuras o componentes modernos.

En esta práctica se utilizó el lenguaje de programación que Matlab, donde se estuvo trabajando con la optimización de una estructura. Al igual, fue interesante observar las diferentes aplicaciones que el programa tiene para ofrecer, pues anteriormente solo veíamos un área más matemática con un poco de programación, pero para obtener graficas.

Viviana Nathalie Tienda Tellez:

En esta primera practica investigamos primeramente un código de optimización topológica en Matlab, el programa consiste en 99 líneas, las primeras 36 líneas son correspondientes al programa principal que se encarga de mandar a llamar las secciones para el correcto funcionamiento del análisis, las líneas 37-48 corresponden a la optimización, las líneas 49-64 es la sección del filtro de malla independiente y las líneas 65-99 corresponden al código de elemento finito.

La optimización topológica es de gran ayuda en la industria y fundamental en el análisis estructural, ya que significa un gran ahorro y como su nombre lo dice optimización para la fabricación de estructuras, piezas, etc. al realizar el análisis, colocar las fuerzas y

dependiendo como se carga la pieza, a partir de ahí empezar a revisar en donde no necesitamos material para eliminarlo y así obtener como resultado una estructura más liviana, funcional y con reducción de costos.

Alina Martínez Escobedo

En nuestra primera práctica del Laboratorio de Biomecánica teníamos como objetivo conocer las distintas secciones que componen a un código de optimización topológica. Para poder lograr esto utilizamos el software llamado Matlab que fue de gran ayuda para poder realizar el código de la estructura que elegimos. Gracias a esta práctica pudimos ver cómo es importante conocer este tipo de códigos ya que nos ayudan a poder analizar una estructura con más rapidez y detectar que es necesario modificar o eliminar.

REFERENCIAS

[1]A. (s. f.). *La optimización estructural y sus aplicaciones*. Instituto Politécnico Nacional. Recuperado 4 de septiembre de 2022, de https://www.boletin.upiita.ipn.mx/index.php/ciencia/916-cyt-numero-82/1889-la-optimizacion-estructural-y-sus-aplicaciones

[2]C., L. (2020, 16 diciembre). *La optimización topológica en la impresión 3D*. 3Dnatives. Recuperado 4 de septiembre de 2022, de https://www.3dnatives.com/es/optimizacion-topologica-10012017/#!

[3]*Optimización Topológica | Catec*. (s. f.). CATEC. Recuperado 4 de septiembre de 2022, de http://www.catec.aero/es/materiales-y-procesos/l%C3%ADnea-de-investigaci%C3%B3n/optimizaci%C3%B3n-topol%C3%B3gica#:%7E:text=La%20optimizaci%C3%B3n%20topol%C3%B3gica%20es%20una,funcionalidades%20mec%C3%A1nicas%20del%20componente%20objetivo.

[4] MATLAB - Archivos M. (2020, December 10). Retrieved September 5, 2022, from Stack website: https://isolution.pro/es/t/matlab/matlab-m-files/matlab-archivos-m

[5] MATLAB: descripción general. (2020, December 10). Retrieved September 5, 2022, from Stack website: https://isolution.pro/es/t/matlab/matlab-overview/matlab-descripcion-general