Facultad de Ingeniería Mecánica y Eléctrica

UANL

Mecatrónica

Laboratorio de Biomecánica

Grupo:309

Optimización Topológica
Practica 1

Fecha de entrega: 21 Septiembre 2022

Profesor:

Nombre del Maestro

Nombre	Matricula	Carrera
Saul Moises Mendoza Cida	1942534	IMTC
Erick Daniel Esquivel Arguelles	1826021	IMTC
Omar Isaí Moreno Cruz	1849630	IMTC
Miguel Rodrigo Aguilar Moreno	1801380	IMTC
Betsaida Alejandra Ruedas Vázquez	1730437	IMTC

Índice

1. Objetivo

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

2. Estado del arte

La optimización topológica (OT) es una herramienta matemática que le permite al diseñador sintetizar topologías óptimas. En Ingeniería Mecánica se entiende como topología óptima a una pieza o parte mecánica diseñada especialmente para maximizar o minimizar alguna característica deseada. Por ejemplo, cuando se diseña el ala de un avión se desea obtener el menor peso posible, asegurando una rigidez y resistencia adecuadas. El problema de la máxima rigidez con restricción de volumen es de gran importancia en Ingeniería Mecánica e Ingeniería de Estructuras, pues permite reducir el peso final del elemento mecánico o estructural, conservando su rigidez y funcionalidad. Partes mecánicas de bajo peso implican menores costos por material y menor consumo de combustible en el caso de vehículos de transporte [1]. En general, la reducción de la inercia en partes en movimiento, sea maquinaria o vehículos, disminuye la cantidad de energía necesaria para su operación.

La OT es un campo de investigación de rápido crecimiento, donde intervienen distintas áreas como son las matemáticas, la mecánica y las ciencias computacionales, y que cuenta con importantes aplicaciones prácticas en la industria y en el sector de manufactura. En la actualidad, la OT es usada en las industrias aeroespacial, automotriz, de obras civiles, entre otras. Además tiene un papel muy importante en el campo de las micro y nanotecnologías, principalmente en el diseño de mecanismos flexibles [1].

Existen muchas otras aplicaciones de la OT en la Ingeniería Mecánica, como son: diseño de mecanismos flexibles y micromecanismos [4][7][2], diseño de MEM (Sistemas Micro-ElectroMecánicos) [5][3], diseño de materiales con coeficiente de Poisson o coeficiente de expansión térmica negativos (metamateriales)[6], resonadores, aletas para intercambio de calor, entre otras aplicaciones. Además, están comenzando a aparecer trabajos sobre aplicación de la técnica en la dinámica de fluidos [9], tales como diseño de mezcladores, muros de contención de represas [10], perfiles de ala de avión [8], entre otros. Un caso famoso donde se empleó exitosamente la OT es el diseño de las alas de los aviones Airbus A380, que por medio de OT quedaron considerablemente más livianas que las diseñadas por medios convencionales, ahorrando en combustible y al mismo tiempo aumentando la carga útil de la aeronave [1].

3. Marco teórico

Un código de optimización de topología de 99 líneas escrito en Matlab La propuesta de análisis de forma de programación es una implementación compacta en MATLAB de un código de op-

timización topológica para la minimización del cumplimiento de estructuras cargadas estáticamente obtenido de un artículo educacional de parte del autor Ole Sigmund (2001). En este caso se realizará para vigas (geometría).

El número total de líneas de entrada de MATLAB para este caso es de 99, incluidos un optimizador y una subrutina de elementos finitos. De tal modo, el código se dispondrá entonces de 36 líneas para el programa principal, 12 líneas para el optimizador basado en criterios, 16 líneas para un filtro de independencia de malla y 35 líneas para el código de elemento finito.

Un número de simplificaciones son introducidas en el código de MATLAB. Primero se asume que el dominio de diseño es rectangular y discretizada por elementos cuadrados finitos. De esta forma el número de elementos y nodos es simple y la relación de aspecto de la estructura se da por la relación de elementos en las direcciones horizontal (nelx) y vertical (nely).

De forma general, un problema de optimización topológica basado en el enforque de la ley de potencia, donde se busca minimizar el cumplimiento se puede escribir de la siguiente manera:

$$\left.\begin{aligned} \min_{\mathbf{x}}: \quad & c(\mathbf{x}) = \mathbf{U}^T\mathbf{K}\mathbf{U} = \sum_{e=1}^{N} (x_e)^p \, \mathbf{u}_e^T \, \mathbf{k}_0 \, \mathbf{u}_e \\ \text{subject to}: \quad & \frac{V(\mathbf{x})}{V_0} = f \\ : \quad & \mathbf{K}\mathbf{U} = \mathbf{F} \\ : \quad & \mathbf{0} < \mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{1} \end{aligned}\right\}$$

Donde U y F son los vectores globales de desplazamiento y fuerza, respectivamente, K es la matriz de rigidez global, Ue y Ke son el vector de desplazamiento del elemento y la matriz de rigidez , respectivamente, x es el vector de variables de diseño, xmin es un vector de densidades relativas mínimas, N (nelx X nely) es el número de elementos usados para discretizar el domino de diseño, p es el poder de penalización ,V(x) y V0 son el volumen de material y volumen de dominio de diseño, respectivamente y por último f es una fracción de volumen prescrita.

4. Desarrollo

Matlab implementation

El código de Matlab (véase el Apéndice), está construido como un código estándar de optimización de la topología. El programa principal se llamado desde el prompt de Matlab por 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, 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 la corriente. La figura 1 muestra la distribución de densidad resultante obtenida por el código dado en el Apéndice llamado con la entrada línea

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

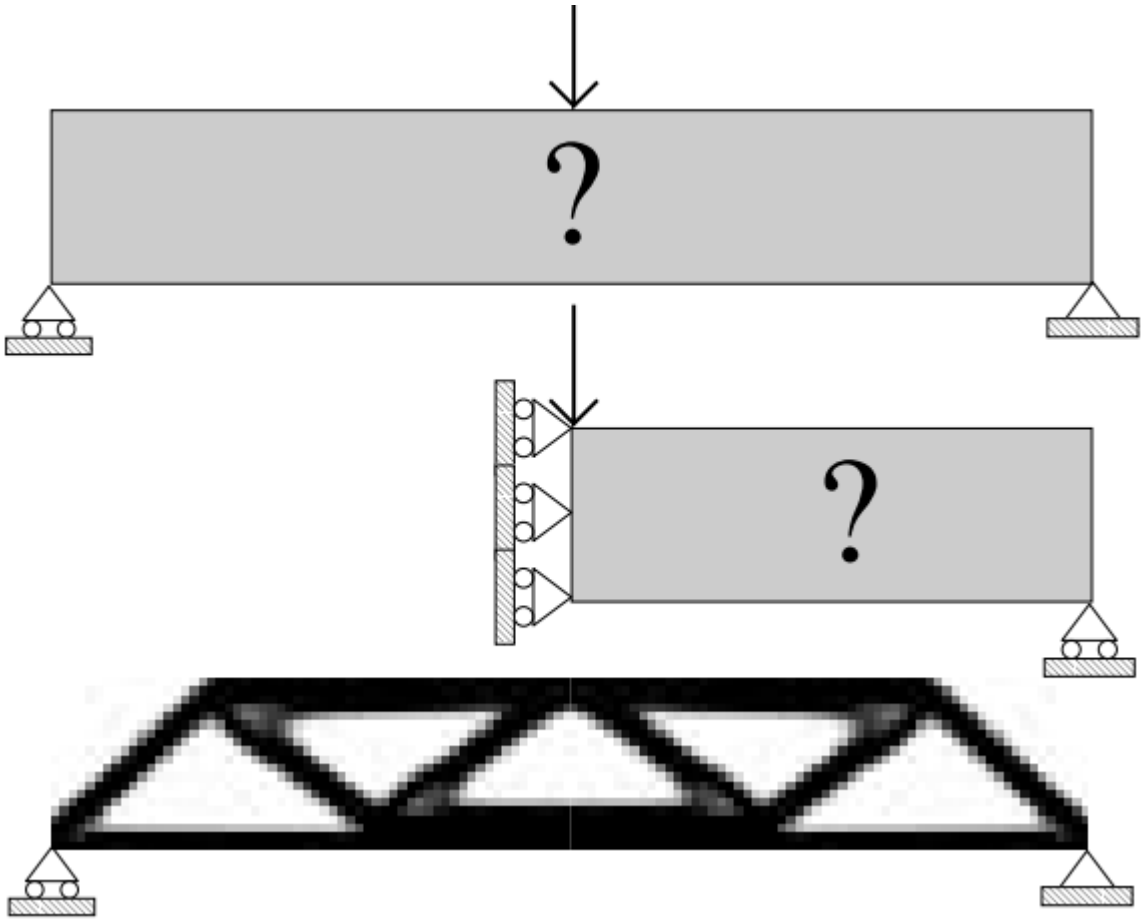

Figura 1: Optimización de la topología de la viga MBB. Arriba: dominio de diseño, en medio: medio dominio de diseño con condiciones de simetría y abajo: viga resultante con topología optimizada (ambas mitades)

Las condiciones de contorno por defecto corresponden a la mitad del "viga MBB"(Fig. 1). La carga se aplica verticalmente en la esquina superior izquierda y hay condiciones de contorno simétricas a lo largo del borde izquierdo y la estructura se apoya horizontalmente en la esquina inferior derecha. Los detalles importantes del código Matlab se discuten en las siguientes subsecciones.

Programa principal (líneas 1-36)

El programa principal (líneas 1-36) comienza distribuyendo el material uniformemente en el dominio de diseño (línea 4). Después de algunas inicializaciones, el bucle principal comienza con una llamada a la subrutina la subrutina de elementos finitos (línea 12) que devuelve el vector de desplazamiento U. vector de desplazamiento U. Dado que la matriz de rigidez del elemento para el material sólido es la misma para todos los elementos, la subrutina de la matriz de rigidez del elemento se llama sólo una vez (línea 14). A continuación, un bucle sobre todos los elementos (líneas 16- 24) determina la función objetivo y las sensibilidades (4). Las variables n1 y n2 denotan los números de nodo del elemento superior izquierdo y derecho en el nodo global. números de nodo del elemento superior izquierdo y derecho en los números de nodo global y se utilizan para extraer el vector de desplazamiento del elemento. para extraer el vector de desplazamiento del elemento Ue del vector de desplazamiento global U. del vector de desplazamiento global U. El análisis de sensibilidad es seguido por una llamada al filtro de independencia de malla (línea 26) y al optimizador de criterios de optimización (línea 28). El cumplimiento actual 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 (cambio determinado en línea 30) es inferior al 1 por ciento2. En caso contrario, se repiten los pasos anteriores. se repiten.

Optimizador basado en criterios de optimalidad (líneas 37-48)

El optimizador encuentra las variables de diseño actualizadas (líneas 37-48). Sabiendo que el volumen de material (sum(sum(xnew))) es una función monótonamente decreciente del multiplicador de Lagrange (lag), el valor del multiplicador de Lagrange que satisface la restricción de volumen puede encontrarse mediante un algoritmo de bioseccionamiento (líneas 40-48). El algoritmo de bipartición se inicializa adivinando un límite inferior l1 y superior l2 para el multiplicador lagrangiano multiplicador lagrangiano (línea 39). El intervalo que limita el multiplicador lagrangiano se reduce repetidamente a la mitad hasta que su tamaño es menos que el criterio de convergencia (línea 40).

Filtrado de independencia de la malla (líneas 49-64)

Las líneas 49-64 representan la implementación en Matlab de (5). Obsérvese que no se buscan todos los elementos del dominio de diseño para encontrar los elementos que se encuentran dentro de el radio rmin, sino sólo los que se encuentran dentro de un cuadrado de lado de lado dos veces redondo(rmin) alrededor del elemento elemento considerado. Seleccionando rmin menos que uno en la llamada de la rutina, las sensibilidades filtradas serán iguales a las sensibilidades originales haciendo que el filtro sea inactivo

Código de elementos finitos (líneas 65-99)

El código de elementos finitos está escrito en las líneas 65-99. Obsérvese que el solucionador hace uso de la opción sparse de Matlab. La matriz de rigidez global se forma mediante un bucle sobre todos los elementos (líneas 70-77). Como en el caso del programa principal programa principal, las variables n1 y n2 denotan los números de nodo del elemento superior izquierdo y derecho en el nodo global. números de nodo del elemento en los números de nodo global y se para insertar la matriz de rigidez del elemento en los lugares correctos en la matriz de rigidez global. Como se mencionó anteriormente, tanto los nodos como los elementos son numerados en columnas de izquierda a derecha. Además cada nodo tiene dos grados de libertad (horizontal y vertical), por lo que el comando F(2,1)=-1. (línea 79) aplica una fuerza unitaria vertical en la esquina superior izquierda. Los apoyos se implementan eliminando los grados de libertad fijos de las ecuaciones lineales. Matlab puede hacer esto muy elegantemente con la línea

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

donde freedofs indica los grados de libertad que no están restringidos. En la mayoría de los casos, es más fácil definir los grados de libertad que son fijos (fixeddofs) a partir de entonces los se encuentran automáticamente usando el operador de Matlab setdiff que encuentra los grados de libertad libres por diferencia entre todos los grados de libertad y los grados de libertad fijos (línea 82). (línea 82). La matriz de rigidez del elemento se calcula en las líneas 86- 99. La matriz de 8 por 8 para un elemento cuadrado bi-lineal de 4 nodos se determinó analíticamente utilizando un software de manipulación simbólica. El módulo de Young E y la relación de Poisson nu pueden modificarse en las líneas 88 y 89.

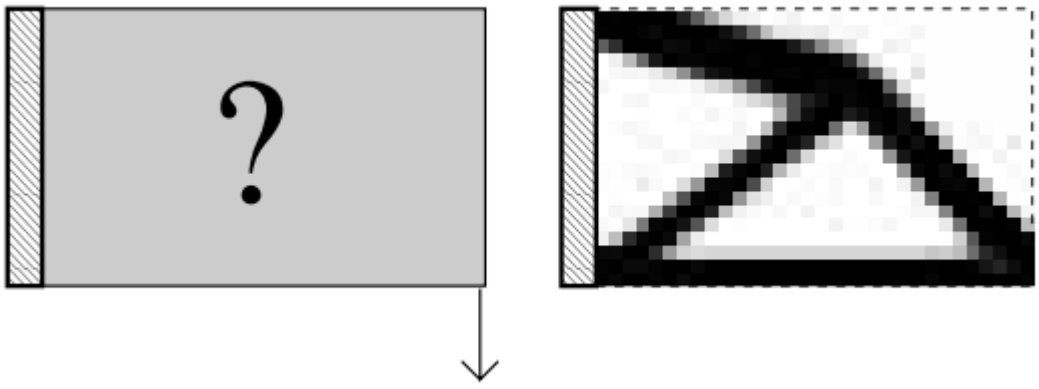

Figura 2: Optimización de la topología de una viga en voladizo. Izquierda: dominio de diseño y derecha: viga con topología optimizada.

Extensiones

El código de Matlab que se proporciona en el Apéndice resuelve el problema de optimizar la distribución del material en el MBBbeam (Fig. 1) de modo que se minimice su cumplimiento. Una serie de extensiones y cambios en el algoritmo se puede pensar, algunos de los cuales se mencionan.

Otras condiciones de contorno

Es muy sencillo cambiar las condiciones de contorno y de apoyo para resolver otros problemas de optimización. Para resolver el ejemplo del voladizo corto mostrado en la Fig. 2, sólo hay que cambiar las líneas 79 y 80 por

```
79 F(2*(nelx+1)*(nely+1),1) = -1;
80 fixeddofs = [1:2*(nely+1)];
```

Con estos cambios, la línea de entrada para el caso mostrado en la Fig. 2 es

```
top(32,20,0.4,3.0,1.2)
```

Casos de carga múltiples

También es muy sencillo ampliar el algoritmo para tener en cuenta casos de carga múltiple. De hecho, esto puede hacerse añadiendo tres líneas adicionales y haciendo pequeños cambios en otras 4 líneas. En el caso de dos casos de carga, los vectores de fuerza y desplazamiento deben definirse como vectores de dos columnas, lo que significa que la línea 69 se cambia por

```
69 F = sparse(2*(nely+1)*(nelx+1),2);
   U = sparse(2*(nely+1)*(nelx+1),2);
```

La función objetivo es ahora la suma de dos cumplimientos, es decir

$$c(\mathbf{x}) = \sum_{i=1}^{2} \mathbf{U}_i^T \mathbf{K} \mathbf{U}_i$$

Figura 3: Ecuación (7)

por lo que las líneas 20-22 se sustituyen por las líneas

```
19b dc(ely,elx) = 0.;
```

```
19c for i = 1:2
20 Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2;
        2*n2+1;2*n2+2;2*n1+1;2*n1+2],i);
21c = c + x(ely,elx)^penal*Ue'*KE*Ue;
22 dc(ely,elx) = dc(ely,elx) -
      penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
22b end
```

Para resolver el problema de dos cargas indicado en la Fig. 4, se añade una unidad hacia arriba en la esquina superior derecha se añade a la línea 79, que entonces se convierte en

```
79 F(2*(nelx+1)*(nely+1),1) = -1.;
           F(2*(nelx)*(nely+1)+2,2) = 1.;
```

La línea de entrada de la Fig. 4 es

```
top(30,30,0.4,3.0,1.2).
```

Elementos pasivos

En algunos casos, algunos de los elementos pueden tomar el valor mínimo de densidad (por ejemplo, un agujero para una tubería).Una matriz nely×nelx pasiva con ceros en los elementos libres de cambiar y unos en los elementos fijados a cero puede definirse en el programa principal y transferirse a la subrutina OC (añadiendo el pasivo a la llamada en las líneas 28 y 38). La línea añadida

```
42b xnew(find(passive)) = 0.001;
```

en la subrutina OC busca los elementos pasivos y establece su densidad igual a la densidad mínima (0,001).

La figura 5 muestra la estructura resultante obtenida con la entrada

```
top(45,30,0.5,3.0,1.5),
```

cuando se añadieron las siguientes 10 líneas al programa principal (después de la línea 4) para encontrar elementos pasivos dentro de un círculo con radio nely/3. y centro (nely/2., nelx/3.)

```
for ely = 1:nely
  for elx = 1:nelx
    if sqrt((ely-nely/2.)^2+(elx-nelx/3.)^2) < nely/3
     passive(ely,elx) = 1;  x(ely,elx) = 0.001;  else
     passive(ely,elx) = 0; end; end; end;
```

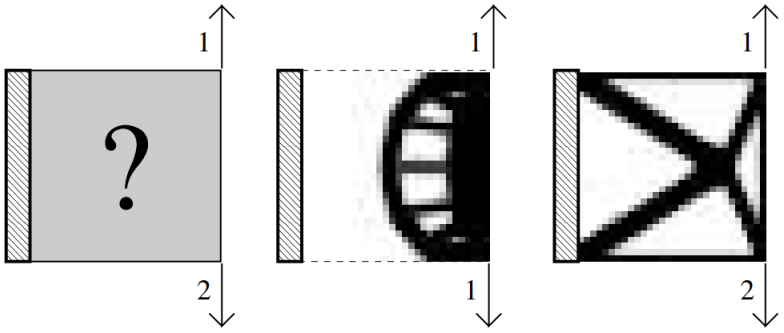

Figura 4: Optimización de la topología de una viga en voladizo con dos casos de carga. Izquierda: dominio de diseño, centro: topología optimizada topología utilizando un caso de carga y a la derecha: viga con topología optimizada utilizando dos casos de carga

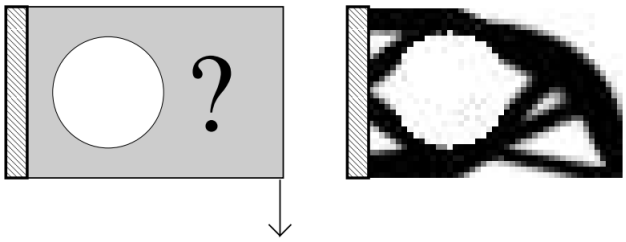

Figura 5: Optimización de la topología de una viga en voladizo con un orificio fijo. Izquierda: dominio de diseño y derecha: viga con topología optimizada.

Código de Matlab

```
%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, OCTOBER 1999 %%%
function top(nelx,nely,volfrac,penal,rmin);
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
change = 1.;
% START ITERATION
while change > 0.01
loop = loop + 1;
xold = x;
% FE-ANALYSIS
[U]=FE(nelx,nely,x,penal);
% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
[KE] = lk;
c = 0.;
```

```
16 for ely = 1:nely
17 for elx = 1:nelx
18 n1 = (nely+1)*(elx-1)+ely;
19 n2 = (nely+1)* elx +ely;
20 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);
21 c = c + x(ely,elx)^penal*Ue'*KE*Ue;
22 dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*
                Ue'*KE*Ue;
23 end
24 end
25 % FILTERING OF SENSITIVITIES
26 [dc] = check(nelx,nely,rmin,x,dc);
27 % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
28 [x] = OC(nelx,nely,x,volfrac,dc);
29 % PRINT RESULTS
30 change = max(max(abs(x-xold)));
31 disp([' It.: ' sprintf('%4i',loop) ' Obj.: '
    sprintf('%10.4f',c) ...
32 ' Vol.: ' sprintf('%6.3f',sum(sum(x))/
     (nelx*nely)) ...
33 ' ch.: ' sprintf('%6.3f',change )])
34 % PLOT DENSITIES
35 colormap(gray); imagesc(-x); axis equal; axis
   tight; axis off;pause(1e-6);
36 end
37 %%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%
38 function [xnew]=OC(nelx,nely,x,volfrac,dc)
39 l1 = 0; l2 = 100000; move = 0.2;
40 while (l2-l1 > 1e-4)
41 lmid = 0.5*(l2+l1);
42 xnew = max(0.001,max(x-move,min(1.,min(x+move,x.
         *sqrt(-dc./lmid)))));
43 if sum(sum(xnew)) - volfrac*nelx*nely > 0;
44 l1 = lmid;
45 else
46 l2 = lmid;
47 end
48 end
49 %%%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%%
50 function [dcn]=check(nelx,nely,rmin,x,dc)
```

```
51 dcn=zeros(nely,nelx);
52 for i = 1:nelx
53 for j = 1:nely
54 sum=0.0;
55 for k = max(i-round(rmin),1):
         min(i+round(rmin),nelx)
56 for l = max(j-round(rmin),1):
         min(j+round(rmin), nely)
57 fac = rmin-sqrt((i-k)^2+(j-l)^2);
58 sum = sum+max(0,fac);
59 dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)
   *dc(l,k);
60 end
61 end
62 dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
63 end
64 end
65 %%%%%%%%% FE-ANALYSIS %%%%%%%%%%%
66 function [U]=FE(nelx,nely,x,penal)
67 [KE] = lk;
68 K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*
       (nely+1));
69 F = sparse(2*(nely+1)*(nelx+1),1); U =
       sparse(2*(nely+1)*(nelx+1),1);
70 for ely = 1:nely
71 for elx = 1:nelx
72 n1 = (nely+1)*(elx-1)+ely;
73 n2 = (nely+1)* elx +ely;
74 edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1;
           2*n2+2;2*n1+1; 2*n1+2];
75 K(edof,edof) = K(edof,edof) +
          x(ely,elx)^penal*KE;
76 end
77 end
78 % DEFINE LOADSAND SUPPORTS(HALF MBB-BEAM)
79 F(2,1) = -1;
80 fixeddofs = union([1:2:2*(nely+1)],
              [2*(nelx+1)*(nely+1)]);
81 alldofs = [1:2*(nely+1)*(nelx+1)];
82 freedofs = setdiff(alldofs,fixeddofs);
83 % SOLVING
```

```
84 U(freedofs,:) = K(freedofs,freedofs) \
                   F(freedofs,:);
85 U(fixeddofs,:)= 0;
86 %%%%%%%%%% ELEMENT STIFFNESS MATRIX %%%%%%%
87 function [KE]=lk
88 E = 1.;
89 nu = 0.3;
90 k=[ 1/2-nu/6 1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
91 -1/4+nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8];
92 KE = E/(1-nu^2)*
[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
93 k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
94 k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
95 k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
96 k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
97 k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
98 k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
99 k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
```

5. Conclusiones

Saúl Moisés Mendoza Cida Esta práctica fue bastante interesante ya que el tema de optimización topológica fue uno el cual siempre quise saber como se como se obtenían las áreas optimizadas, con la investigación elaborada pude darme una idea de como es que se aplica la optimización topológica, gracias a esto podre utilizar este tipo de optimización en proyectos futuros especial mente en esta materia donde es necesario reducir el peso en la mayor medida posible manteniendo las propiedades de las piezas, esto con el fin de ser funcionales y que se ajusten de la mejor manera a las necesidades de los pacientes.

Erick Daniel Esquivel Arguelles Me parece interesante como un análisis mecánico estructural se lleva a cabo con la ayuda de un software de calculo especializado realizando un tipo de modelado matemático qué para esta práctica nos permite optimizar una estructura de tal manera que esta se ve alterada pero al mismo tiempo su función mínima de distribución de carga sea más eficiente, manteniendo la mayoría de funciones mecánicas de la estructura.

Miguel Rodrigo Aguilar Moreno Un ejemplo del concepto de optimización aplicado al programa MATLAB. implementando un modelo de analisis mecanico generando una mayor eficiencia, dando una muestra del expertiz de quien genero el codigo. manteniendo así, una estructura firme pese a su grado de detalle.

Omar Isaí Moreno Cruz En este práctica conocimos lo que es la optimización topológica. Que es un análisis estructural que nos permitirá hacer más ligero nuestro proyecto manteniendo la parte mecánica.

Betsaida Alejandra Ruedas Vázquez Al haber concluido está práctica pude comprender más acerca de las secciones que integran el código de optimización tropologica, esto con ayuda del programa MATLAB. Así mismo como el análisis de las formas de programación junto con características de programación específicas.

Referencias

[1] Elliott, C., Hintermüller, M., Leugering, G., and Sokolowski, J. (2011). Special issue on advances in shape and topology optimization: theory, numerics and new applications areas. *Optim. Methods Softw.*, 26(4-5):511–512.

[2] Fujii, D., Ejima, S., and Kikuchi, N. (2000). Topology optimization of compliant mechanisms using the homogenization design method. *J. Struct. Constr. Eng.*, 65(528):99–105.

[3] Katsuno, E. T., Dantas, J. L. D., and Silva, E. C. N. (2022). Topology optimization of low-friction painting distribution on a marine propeller. *Struct. Multidiscipl. Optim.*, 65(9).

[4] Kikuchi, N., N. S. F. L. S. O. . S. E. C. N. (1998). Design optimization method for compliant mechanisms microstructure. *Methods Appl. Mech. Engrg.*, 151(4-5):401 –417.

[5] Kota, S., J. J. L. Z. R. S. M. . S. J. (2001). Design of compliant mechanisms: Applications to mems. *Analog Integrated Circuits and Signal Processing*, 29:7 –15.

[6] Larsen, U. D., Signund, O., and Bouwsta, S. (1997). Design and fabrication of compliant micromechanisms and structures with negative poisson's ratio. *J. Microelectromech. Syst.*, 6(2):99–106.

[7] Lin, J., Luo, Z., and Tong, L. (2010). A new multi-objective programming scheme for topology optimization of compliant mechanisms. *Struct. Multidiscipl. Optim.*, 40(1-6):241–255.

[8] Maute, K. Allen, M. (2004). . conceptual design of aeroelastic structures by topology optimization. *Struct Multidisc Optim*, 27(27 - 42).

[9] Pingen, G., Evgrafov, A., and Maute, K. (2007). Topology optimization of flow domains using the lattice boltzmann method. *Struct. Multidiscipl. Optim.*, 34(6):507–524.

[10] Sigmund, O. and Clausen, P. M. (2007). Topology optimization using a mixed formulation: An alternative way to solve pressure load problems. *Comput. Methods Appl. Mech. Eng.*, 196(13-16):1874–1889.