Laboratorio de Biomecánica
Práctica 3
Refuerza el cable de un teleférico

EQUIPO 6
Christopher Alejandro Monsivais Ramos 1899218
Alessandra González Torres 1895846
Angela Itzel Rodríguez Torres 1925284
Abdiel Isaac Rodríguez Pinales 1842044
Martin Alejandro Villanueva Cortes 1870820
Alejandro Sebastian Marin Cota 1814201

18 de octubre de 2022

1. Marco Teórico

La optimización topológica es una técnica englobada dentro del campo de análisis estructural. Se basa en el análisis mecánico de un componente o estructura. Su principal objetivo es el aligeramiento estructural manteniendo las funcionalidades mecánicas del componente objetivo. A diferencia de otros tipos de optimización, la optimización topológica ofrece un nuevo concepto de diseño estructural enfocado a aquellas aplicaciones donde el peso del componente es crucial (por ejemplo, la industria aeroespacial).
Gracias a los nuevos métodos computacionales, es posible llevar la optimización a un nivel más complejo de análisis a nivel estático, dinámico, plástico, modal o de impacto, entre otros, los cuales pueden considerarse durante el proceso de optimización.
El desarrollo de esta metodología tiene un amplio campo de aplicación para las tecnologías de fabricación aditiva, como por ejemplo la fabricación SLM (Selective Laser Melting), debido a las grandes posibilidades en términos de diseño (geometrías muy complejas).[1]

2. Definicion de la forma geométrica.

Cable de teleférico
Está formado por varios torones (formados por un determinado número de alambres que son enrollados helicoidalmente alrededor de un centro en una o varias capas) que son enrollados helicoidalmente alrededor de un alma, que es el eje central del cable y puede ser de acero, fibras naturales o de polipropileno.
Como ya se ha dicho anteriormente, el cable es el producto final y se identifica por el número de torones y el número de alambres de cada torón, su tipo de alma y si son negros o galvanizados.

Características

- Cable de alambre para cables de transporte (tracción) y para cables carril/conductor
- Resistencia a la tensión entre 1370N/mm^2 y 2160N/mm^2
- Revestimiento: alambre fosfatado pulido, alambre galvanizado común, alambre revestido Bezinal.

- Variación del diámetro del alambre: entre 0,30 mm y 4,00 mm. Alambres más gruesos disponibles a pedido.

Ventajas

- Alta resistencia de tensión, lo que resulta en el desempeño de cables de rendimiento superior (= alta carga de rotura para el diámetro de cable establecido).
- Excelente ductilidad del alambre, lo cual resulta en propiedades de torsión de la cuerda óptimas a la fatiga.
- Alambre adecuado para usos compactos y no compactos.
- Gran utilidad y confiabilidad de rendimiento.

3. Estado del arte

Objetivo: Dar a conocer cada una de las secciones que integran el código de optimización topológica de 99 líneas en Matlab, saber ejecutar el análisis del código y observar los resultados obtenidos.

Contenido: El artículo presenta una implementación compacta en Matlab de un código de optimización topológica para la minimización de la conformidad de estructuras cargadas estáticamente. El número total de líneas de entrada de Matlab es de 99 incluyendo el optimizador y la subrutina de elementos finitos. Las 99 líneas están divididas en 36 líneas para el programa principal, 12 líneas para el optimizador basado en criterios de optimización, 16 líneas para un filtro de dependencia de malla y 35 líneas para el código de elementos finitos. De hecho, excluyendo las líneas de comentario y las asociadas a la salida y a analisis de elemenios tintos se anserva que solo se necesitan 4u ineas de entrada, de Matlab para resolver un problema de optimización topológica bien planteado. Añadiendo tres líneas adicionales, el programa puede resolver problemas con múltiples casos de carga. El código está pensado para fines educativos.

4. Procedimiento de la programación

Codigo que se utiliza:

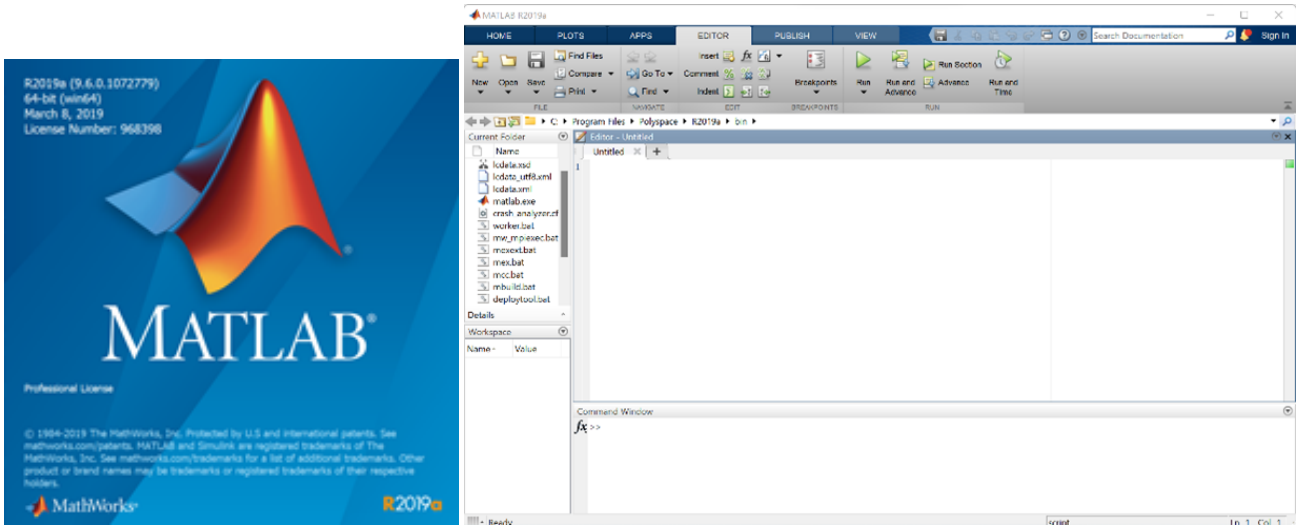
```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, OCTOBER 1999 %%%
%%% PRACTICA4 (60,80,0.33,3.0,1.5)
function Practica4(nelx,nely,volfrac,penal,rmin);
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
for ely = 1:nely
 for elx = 1:nelx
 if ely>21
 if elx<21
 passive(ely,elx) = 1;
 elseif elx>41
 passive(ely,elx) = 1;
 else
 passive(ely,elx) = 0;
 end
 end
 end
end
x(find(passive))=0.001;
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;
dc(ely,elx)=0.;
for i=1:2
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);
c = c + x(ely,elx)^penal*Ue'*KE*Ue;
dc(ely,elx) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)* Ue'*KE*Ue;
end
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,passive);
% 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
% 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,passive)
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)))));
xnew(find(passive))=0.001;
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),2); U = sparse(2*(nely+1)*(nelx+1),2);
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(40,1) = -1; F(9760,2)=1.;
fixeddofs = 2*(nely+1):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)];
```

5. Pasos que seguir para la programación en MATLAB

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 abrir un archivo nuevo en el editor de texto, dentro del cual será necesario escribir el código proporcionado.

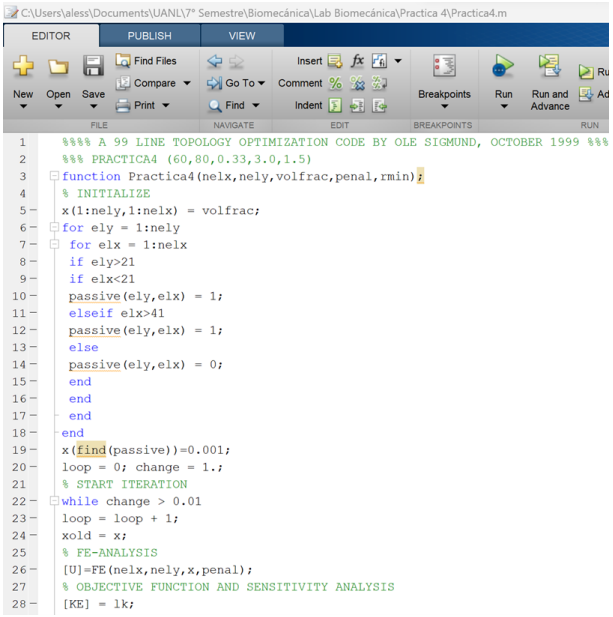
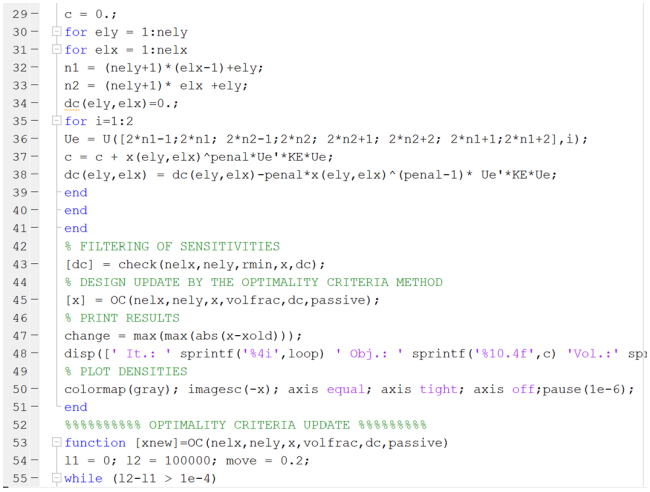
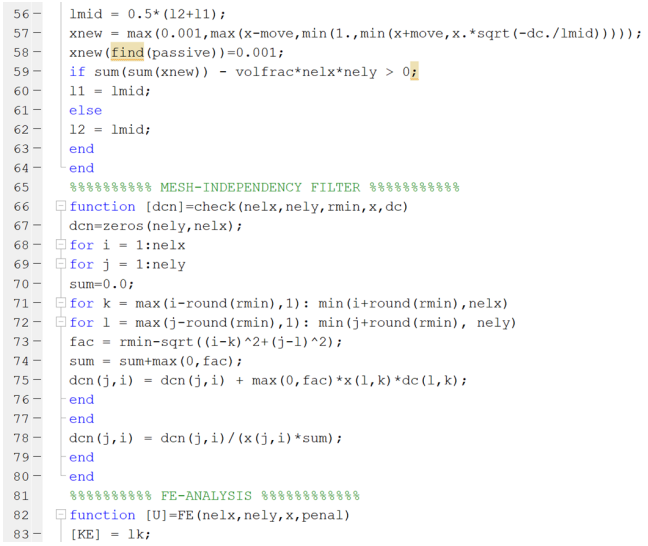


Figura 1: Codigo de Matlab P1



```
c = 0.;
for ely = 1:nely
for elx = 1:nelx
n1 = (nely+1)*(elx-1)+ely;
n2 = (nely+1)* elx +ely;
dc(ely,elx)=0.;
for i=1:2
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);
c = c + x(ely,elx)^penal*Ue'*KE*Ue;
dc(ely,elx) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)* Ue'*KE*Ue;
end
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,passive);
% PRINT RESULTS
change = max(max(abs(x-xold)));
disp([' It.: ' sprintf('%4i',loop) ' Obj.: ' sprintf('%10.4f',c) 'Vol.:' sp
% 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,passive)
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
```

Figura 2: Codigo de Matlab P2



```
lmid = 0.5*(l2+l1);
xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
xnew(find(passive))=0.001;
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;
```

Figura 3: Codigo de Matlab P3

```
K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
F = sparse(2*(nely+1)*(nelx+1),2); U = sparse(2*(nely+1)*(nelx+1),2);
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(40,1) = -1; F(9760,2)=1.;
fixeddofs = 2*(nely+1):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;
```

Figura 4: Codigo de Matlab P4

```
%%%%%%%%%% 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 5: Codigo de Matlab P5

3. El siguiente paso consiste en salvar el archivo. Se recomienda que el archivo se guarde en el directorio raíz de MATLAB que por default muestra. En caso de no ser así, debemos de navegar a “Mis Documentos MATLAB” y guardar ahí el script recién creado.

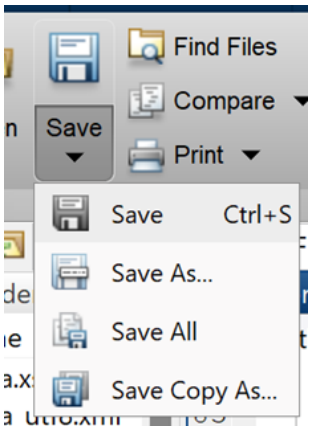


Figura 6: Como salvar un archivo

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 R2019a del software, el directorio actual del intérprete se encuentra en la barra de herramientas superior.

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, este caso en particular es evaluado y simulado cuando escribimos desde la línea de comando de MATLAB “Practica4(60,80,0.33,3.0,1.5)”.

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

6. Resultados de la optimización

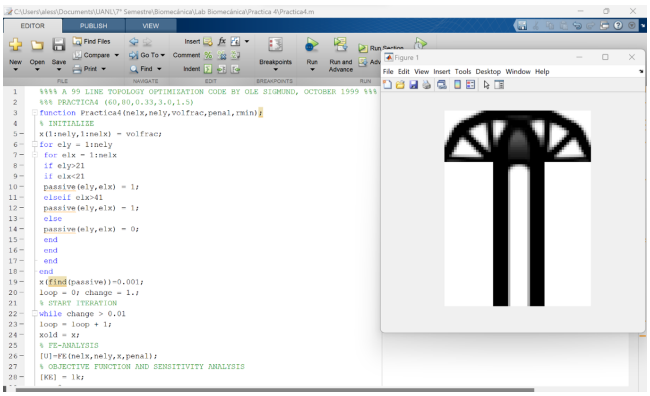


Figura 7: Pantalla de MatLab con código y figura

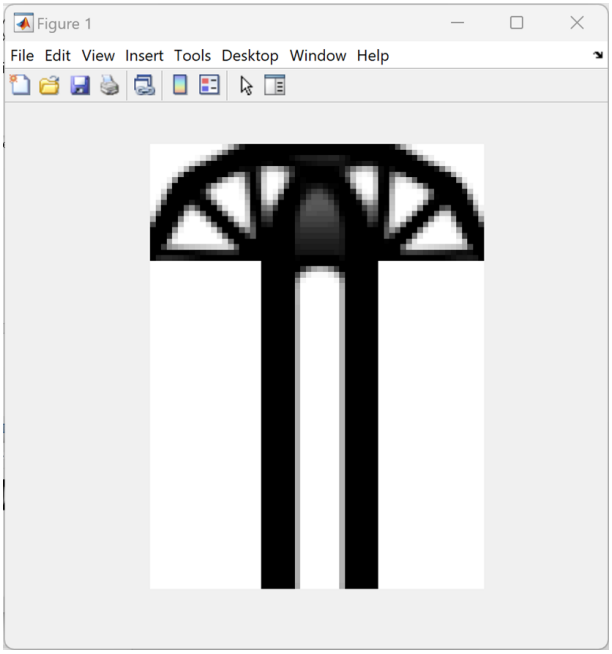


Figura 8: Figura que se crea en MatLab

7. Conclusiones

Christopher Alejandro Monsivais Ramos:
Esta práctica nos ayudo principalmente a reforzar los conocimientos que se han ido adquiriendo acerca de los códigos de optimización topologica, solo que esta vez se le realizaron distintos cambios para poder obtener como resultado la pieza final.

Alessandra Gonzalez Torres:
En esta práctica continuamos utilizando el código original que se vio en la primera práctica, solo que se tuvieron que hacer ciertas modificaciones para que se acoplara a lo pedido en la práctica actual. También se tuvieron que cambiar los valores que antes se estaban utilizando para poder crear la pieza final, como se muestra en la última imagen.

Angela Itzel Rodriguez Torres:
Como podemos observar en los resultados de los casos propuestos se tiene una geometrica muy similar entre ellos. En el caso de dos cargas, este como las fuerzas son aplicadas en opuestos simetricos la forma de pieza es simetrica en el eje Y. Empezando con lo que hicimos podemos concluir que aunque se crea que algo no se toma en cuenta dentro de un sistema de esfuerzos por ser un espacio en blanco, esto no debe ser así, debemos darle la importancia para el diseño óptimo del diagrama.

Abdiel Isaac Rodriguez Piñales:
Se concluye que los avances de tecnología han sido tales que se ha conseguido la optimización de los nuevos diseños. Como en el método de optimización topológica la cual sirve para reducir los materiales a utilizar los nuevos diseños. Esto ha sido útil en áreas como la ingeniería aeroespacial donde es crucial el volumen y materia de los componentes a utilizar. Todo es gracias al avance en el desarrollo de software.

Alejandro Sebastian Marin Cota:
Se concluyó que los avances tecnológicos permitieron la optimización de nuevos diseños. Al igual que las técnicas de optimización de topología, minimiza el uso de materiales rediseñados. Esto es útil en áreas como la ingeniería aeroespacial, donde el tamaño y el material de los componentes que se utilizarán es crítico. Todo gracias a los avances en el desarrollo de software.

Martín Alejandro Villanueva Cortes:
Gracias a esta práctica he podido reforzar mis conocimientos sobre la optimización de la topología de las piezas que se nos han pedido tanto en esta como en las practicas anteriores por lo que sigue siendo de gran ayuda el poder realizar cada una las practicas que llevamos hechas

Referencias

[1] Optimización topológica, 2017.