

# Laboratorio de Biomecánica

## Práctica 3

### Diseño de la estructura de un panorámico

#### 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

20 de septiembre 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]

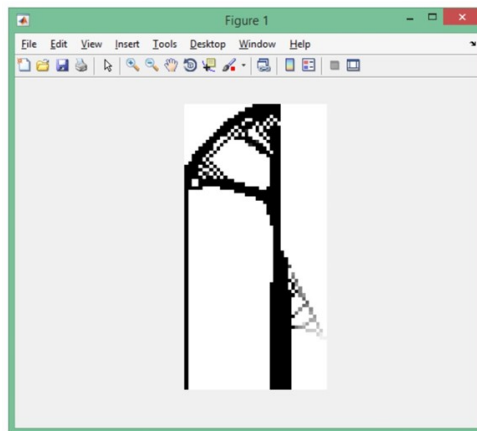


Figura 1: Ejemplo de la forma geométrica

## 2. Estado del arte

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 análisis de elementos finitos 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.

## 3. Propuesta de diseño de la geometría, alcances y limitaciones

Actualmente, su empresa necesita contar con los mejores anuncios para mostrar los productos y servicios que tiene disponibles. De esta manera, podrá atraer a usuarios que se convertirán en futuros clientes y colaboraran a aumentar su visibilidad. Es por ello, que cuando usted contrata los servicios de diseño y construcción de anuncios panorámicos, está dando un paso muy importante para potenciar su marca.

Las estructuras panorámicas son soportes en donde pueden ir ubicados diferentes tipos de anuncios publicitarios. La ventaja de estas estructuras es que pueden ir ubicadas en diferentes paisajes urbanos con el fin de promocionar un producto o servicio.

Ahora, para atraer la atención de los usuarios y futuros clientes es necesario crear una publicidad muy llamativa, y de ese objetivo nos encargamos nosotros. Por consiguiente, nos guiaremos por las normativas correspondientes para colocar la construcción del panorámico en un lugar apropiado. Los materiales adecuados para la construcción de panorámicos que pueden ser elaborados en plástico, lona, telas, PVC, acrílico o metal. Aunque también, puede optar por panorámicos electrónicos y agregarle elementos como luces, e incluso música.

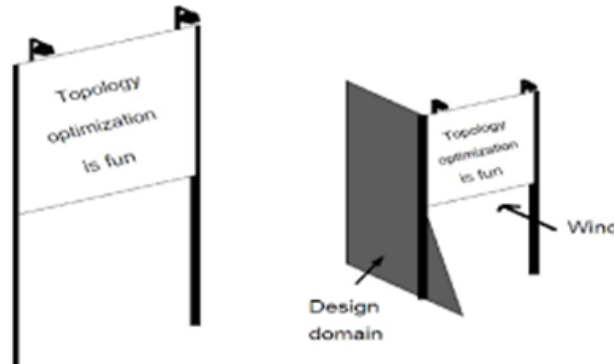


Figura 2:

## 4. Procedimiento de la programación

Codigo que se utiliza:

```
%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function topp3(nelx,nely,volfrac,penal,rmin,passive);
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
for ely = 1:nely
    for elx = 1:nelx
        if elx - 20 < (ely/2);
            passive(ely,elx) = 0;
        else
            passive(ely,elx)=1;
        end
    end
end
x(find (passive))=0.001;
change = 1.;
% START ITERATION
while change > 0.01
    loop = loop + 1;
    xold = x;
    % FE-ANALYSIS
    [U]=FE(nelx,nely,x,penal);
%13 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; %19
            dc(ely,elx) = 0.;
            for i = 1:5
                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) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
            end
        end
    end
%25 FILTERING OF SENSITIVITIES
    [dc] = check(nelx,nely,rmin,x,dc);
%27 DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
    [x] = OC(nelx,nely,x,volfrac,dc,passive);
%29 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
%40 %%%%%%%%%% 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
%65 %%%%%%%%%%%%%% 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),5); U =zeros(2*(nely+1)*(nelx+1),5);
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*nelx*(nely+1)+2,1) = 1;
F(2*nelx*(nely+1)+(nely/4),2) = 1;
F(2*nelx*(nely+1)+(nely/2),3) = 1;
F(2*nelx*(nely+1)+(nely),4) = 1;
F(2*nelx*(nely+1)+(nely*1.2),5) = 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 127

```

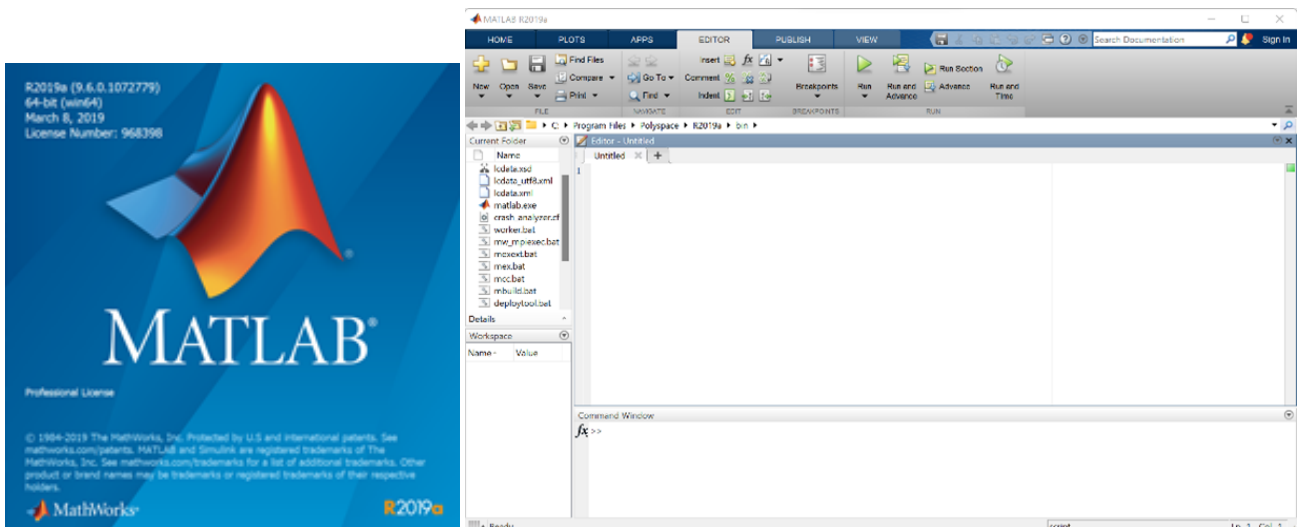
```

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)];

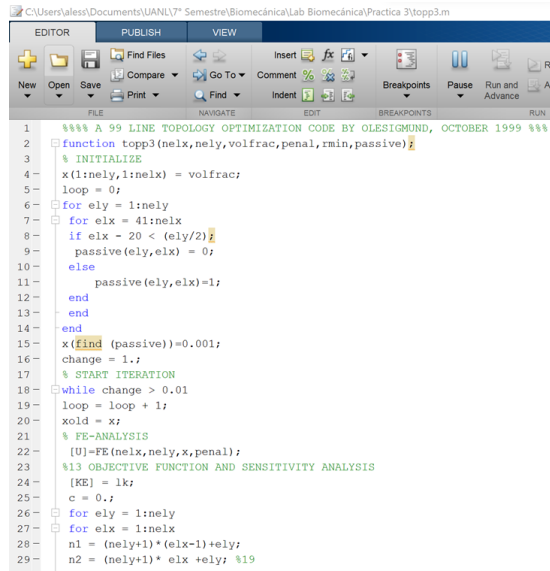
```

## 5. Pasos que seguir para la programación en MATLAB

1. Abrir MATLAB y esperar a que éste se inicialice, y muestre su pantalla principal.



- 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.

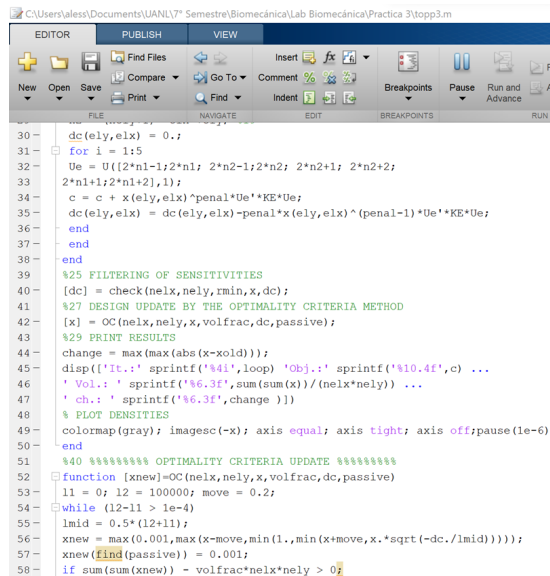


```

1  %%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESGIMUND, OCTOBER 1999 %%%
2  function topp3(nelx,nely,volfrac,penal,rmin,passive)
3  % INITIALIZE
4  x(1:nely,1:nelx) = volfrac;
5  loop = 0;
6  for ely = 1:nely
7  for elx = 1:nelx
8  if elx - 20 < (ely/2)
9  passive(ely,elx) = 0;
10 else
11 passive(ely,elx)=1;
12 end
13 end
14 end
15 x(find (passive))=0.001;
16 change = 1.;
17 % START ITERATION
18 while change > 0.01
19 loop = loop + 1;
20 xold = x;
21 % FE-ANALYSIS
22 [U]=FE(nelx,nely,x,penal);
23 %13 OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
24 [KE] = lk;
25 c = 0.;
26 for ely = 1:nely
27 for elx = 1:nelx
28 n1 = (nely+1)*(elx-1)+ely;
29 n2 = (nely+1)* elx +ely; %19

```

Figura 3: Código de Matlab P1



```

30 dc(ely,elx) = 0.;
31 for i = 1:5
32 Ue = U((2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1; 2*n2+2;
33 2*n1+1;2*n1+2),1);
34 c = c + x(ely,elx)*penal*Ue'*KE*Ue;
35 dc(ely,elx) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
36 end
37 end
38 end
39 %25 FILTERING OF SENSITIVITIES
40 [dc] = check(nelx,nely,rmin,x,dc);
41 %27 DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
42 [x] = OC(nelx,nely,x,volfrac,dc,passive);
43 %29 PRINT RESULTS
44 change = max(max(abs(x-xold)));
45 disp(['It.: ' sprintf('%4i',loop) 'Obj.: ' sprintf('%10.4f',c) ...
46 ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
47 ' ch.: ' sprintf('%6.3f',change) ])
48 % PLOT DENSITIES
49 colormap(gray); imagesc(-x); axis equal; axis tight; axis off; pause(1e-6);
50 end
51 %40 %%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%%
52 function [xnew]=OC(nelx,nely,x,volfrac,dc,passive)
53 l1 = 0; l2 = 100000; move = 0.2;
54 while (l2-l1 > 1e-4)
55 lmid = 0.5*(l2+l1);
56 xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
57 xnew(find (passive)) = 0.001;
58 if sum(sum(xnew)) - volfrac*nelx*nely > 0;

```

Figura 4: Código de Matlab P2

```

59- l1 = lmid;
60- else
61- l2 = lmid;
62- end
63- end
64- %%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%
65- function [dcn]=check(nelx,nely,rmin,x,dc)
66- dcn=zeros(nely,nelx);
67- for i = 1:nelx
68- for j = 1:nely
69- sum=0.0;
70- for k = max(i-round(rmin),1):min(i+round(rmin),nelx)
71- for l = max(j-round(rmin),1):min(j+round(rmin),nely)
72- fac = rmin-sqrt((i-k)^2+(j-l)^2);
73- sum = sum+max(0,fac);
74- dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
75- end
76- end
77- dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
78- end
79- end
80- %65 %%%%%%%%% FE-ANALYSIS %%%%%%%%%
81- function [U]=FE(nelx,nely,x,penal)
82- [KE] = lk;
83- K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
84- F = sparse(2*(nely+1)*(nelx+1),5); U =zeros(2*(nely+1)*(nelx+1),5);
85- for ely = 1:nely
86- for elx = 1:nelx
87- nl = (nely+1)*(elx-1)+ely;

```

Figura 5:Codigo de Matlab P3

```

C:\Users\alest\Documents\UAN\I7* Semestre\Biomecánica\Lab Biomecánica\Practica 3\topp3.m
EDITOR PUBLISH VIEW
+ Find Files Insert
Compare Go To Comment
New Open Save Print Find Indent Breakpoints Pause
FILE NAVIGATE EDIT BREAKPOINTS
90- K(edof,edof) = K(edof,edof) + x(ely,elx)*penal*KE;
91- end
92- end
93- % DEFINE LOADSAND SUPPORTS(HALF MBB-BEAM)
94- F(2*nelx*(nely+1)+2,1) = 1;
95- F(2*nelx*(nely+1)+(nely/4),2) = 1;
96- F(2*nelx*(nely+1)+(nely/2),3) = 1;
97- F(2*nelx*(nely+1)+(nely),4) = 1;
98- F(2*nelx*(nely+1)+(nely*1.2),5) = 1;
99- fixeddofs =2*(nely+1):2*(nely+1):2*(nelx+1)*(nely+1);
100- alldofs = [1:2*(nely+1)*(nelx+1)];
101- freedofs = setdiff(alldofs,fixeddofs);
102- % SOLVING 127
103- U(freedofs,:) = K(freedofs,freedofs) \F(freedofs,:);
104- U(fixeddofs,:) = 0;
105- %%%%%%%%% ELEMENT STIFFNESS MATRIX %%%%%%%%%
106- function [KE]=lk
107- E = 1.;
108- nu = 0.3;
109- k=[ 1/2-nu/6 1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
110- -1/4+nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8];
111- KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
112- k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
113- k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
114- k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
115- k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
116- k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
117- k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
118- k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];

```

Figura 6:Codigo de Matlab P4

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.

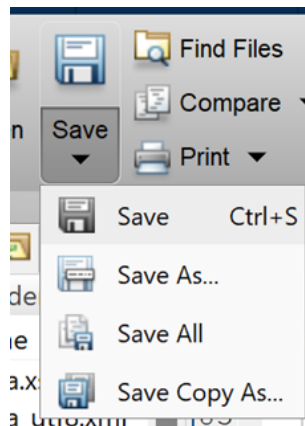


Figura 7: Como salvar un archivo

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 “`topp1(40,80,0.2,3.0,0.5,0.001)`”.
  - 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

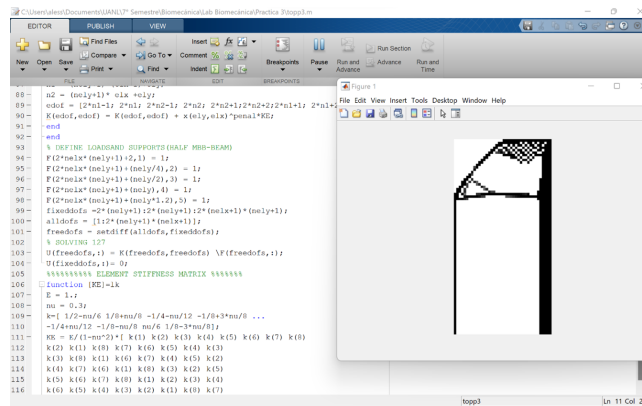


Figura 8: Pantalla de MatLab con código y figura

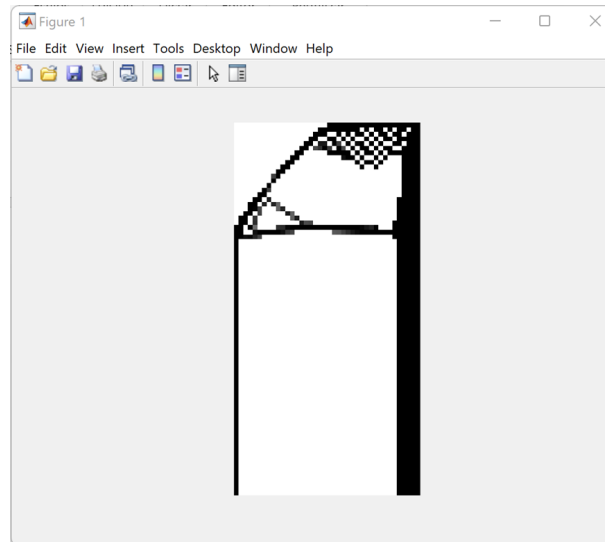


Figura 9: Figura que se crea en MatLab

## 7. Conclusiones

Christopher Alejandro Monsivais Ramos:

Esta práctica nos ayudo principalmente a seguir practicando o reforzando los conocimientos sobre el código de optimización topológica que se implementó desde la práctica 1, 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 ultima imagen.

Angela Itzel Rodriguez Torres:

Los resultados presentados en la sección anterior incluyen los tres aspectos que proporciona el análisis estructural del panorámico. En este caso, los valores que se obtienen nos permiten darnos cuenta de cómo responde al estar en la intemperie con las condiciones del problema, el cual presenta valores que no son críticos para llevarlo a una situación de falla en la estructura, por lo que al soportar las cargas que se ejercen sobre el espectacular no se corre el riesgo de una caída del mismo debido a las condiciones mencionadas.

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:

En conclusión, esta actividad nos permite apreciar una vez más la flexibilidad de usar este código. Mirando la viga, luego el cuadro de la bicicleta y ahora la estructura general, puede usar la función de simulación para: Evalúa físicamente los materiales y geometrías a utilizar, pero tiene la ventaja de no utilizar demasiados recursos.

Martín Alejandro Villanueva Cortes:

Para esta practica hemos seguido practicando con el mismo tipo de programación para realización de las piezas que se nos dan, como lo es en esta practica y la anterior solo se modifican un poco estos ya que son la base para seguir desarrollando otras piezas que se nos puedan dar en otras prácticas.

## Referencias

[1] Optimización topológica, 2017.