

Universidad Autónoma de Nuevo León
Facultad De Ingeniería Mecánica y Eléctrica

PRÁCTICA #1

Docente: Yadira Moreno Vera

Brigada: 109 Hora: N5 Día: lunes

Materia: Laboratorio de Biomecánica

Integrantes del equipo:

Aarón Lozano Aguilar 1844469

Montserrat Granados Salinas 1817165

Eunice Carolina Méndez Sosa 1851345

Aida Mata Moreno 1734743

Eimie Carolina Pereda Sanchez 1915035

Semestre: Enero-Junio 2022

Objetivo:

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

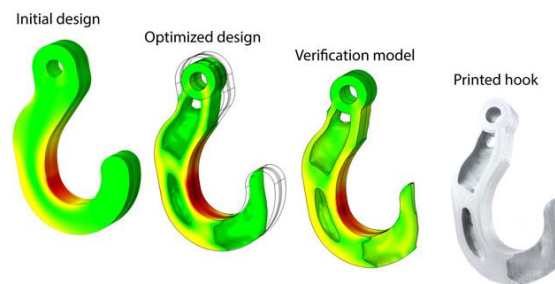
Marco Teórico:

La optimización topológica es una técnica que se engloba dentro del campo de análisis estructural. Que se basa en el análisis mecánico de un componente o estructura y su objetivo principal es el aligeramiento estructural, pero manteniendo las funcionalidades mecánicas del componente objetivo. Nos ofrece un concepto nuevo de diseño estructural enfocado hacia aquellas aplicaciones donde el peso del componente es parte crucial, u ejemplo muy claro es la industria aeroespacial

La optimización hoy en día tiene un gran avance gracias a los métodos computacionales que salen día con día, ya que con esto lo lleva a un nivel más complejo del análisis a un nivel estático, dinámico, plástico, modal o de impacto, los cuales se consideran durante todo el proceso de optimización.

Tiene un amplio campo de aplicación para las tecnologías de fabricación aditiva, u ejemplo es la fabricación SLM (Selective Laser Melting), ya que tiene grandes posibilidades en términos de diseño.

Podemos tomar de ejemplo una geometría que se usa día con día, ya sea en la construcción, así como en los grandes puertos de México ya sean marítimos a aéreos, un gancho.



Por medio de la optimización se va analizando y descartando las zonas que menor esfuerzo presenta a comparación de las zonas donde los esfuerzos son los mas considerables, repitiendo el análisis las veces necesarias hasta llegar al mejor resultado de optimización.

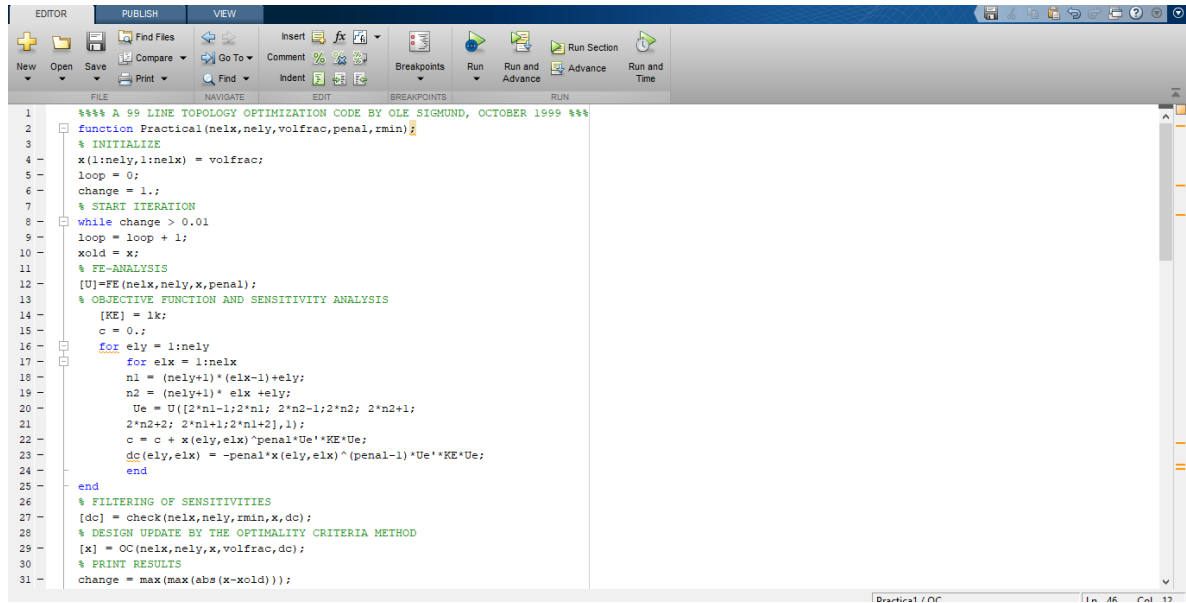
El código del cual haremos uso durante el curso de laboratorio fue hecho en Matlab y está destinado al campo de la optimización de topología y se puede usar para hacer extensiones como múltiples casos de carga, esquemas alternativos de independencia de malla, áreas pasivas, etc.

El código en uso consta de 99 líneas, las cuales 36 son para la programación principal, 12 para los criterios de optimización, 16 para el filtro de mallado y 35 líneas para el código de elemento finito.

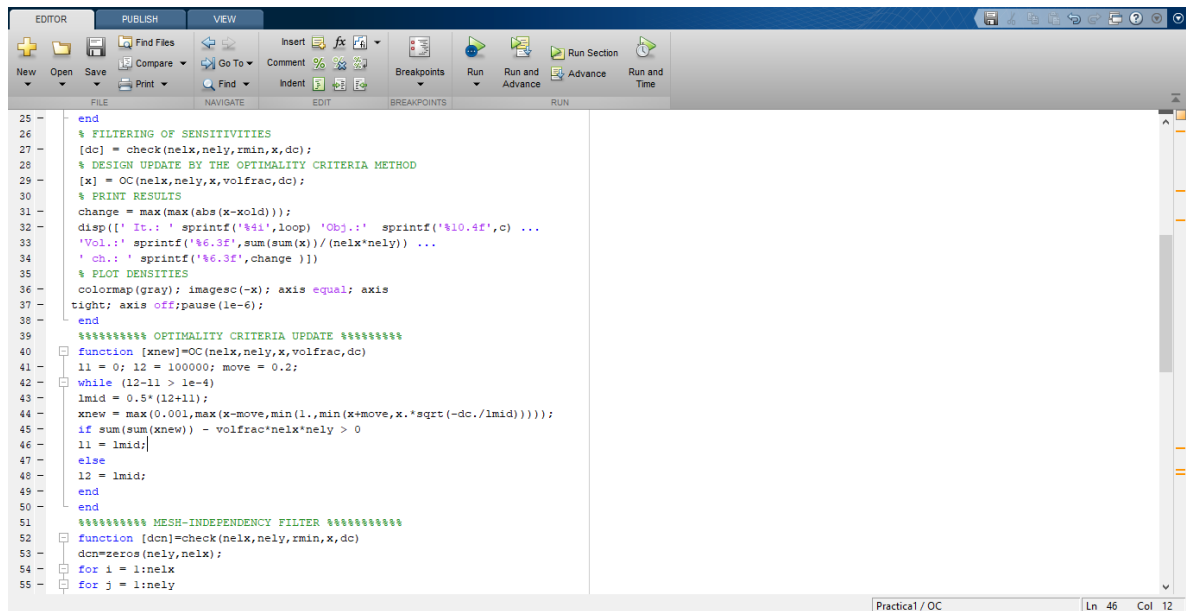
Este Código fue desarrollado por O. Sigmund, Department of Solid Mechanics, Building 404, Technical University of Denmark, DK-2800 Lyngby, Denmark. El código puede ser descargado desde la página del autor: <http://www.topopt.dtu.dk>.

Desarrollo:

1. Empezamos abriendo el software Matlab o en su caso algún similar como scilab y empezamos un nuevo script.
2. Escribimos el código de manera correcta corrigiendo sus errores



```
1  % A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, OCTOBER 1999 %
2  function Practical(nelx,nely,volfrac,penal,rmin)
3  % INITIALIZE
4  x(l:nely,1:nelx) = volfrac;
5  loop = 0;
6  change = 1.;
7  % START ITERATION
8  while change > 0.01
9  loop = loop + 1;
10 xold = x;
11 % FE-ANALYSIS
12 [U]=FE(nelx,nely,x,penal);
13 % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
14 [KE] = 1k;
15 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;
21             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
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)));
```



```
25 end
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
37 tight; axis off; pause(1e-6);
38 end
39 %***** OPTIMALITY CRITERIA UPDATE *****
40 function [xnew]=OC(nelx,nely,x,volfrac,dc)
41 l1 = 0; l2 = 100000; move = 0.2;
42 while (l2-l1 > 1e-4)
43     lmid = 0.5*(l2+l1);
44     xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
45     if sum(sum(xnew)) - volfrac*nelx*nely > 0
46         l1 = lmid;
47     else
48         l2 = lmid;
49     end
50 end
51 %***** MESH-INDEPENDENCY FILTER *****
52 function [dcn]=check(nelx,nely,rmin,x,dc)
53 dcn=zeros(nely,nelx);
54 for i = 1:nelx
55     for j = 1:nely
```

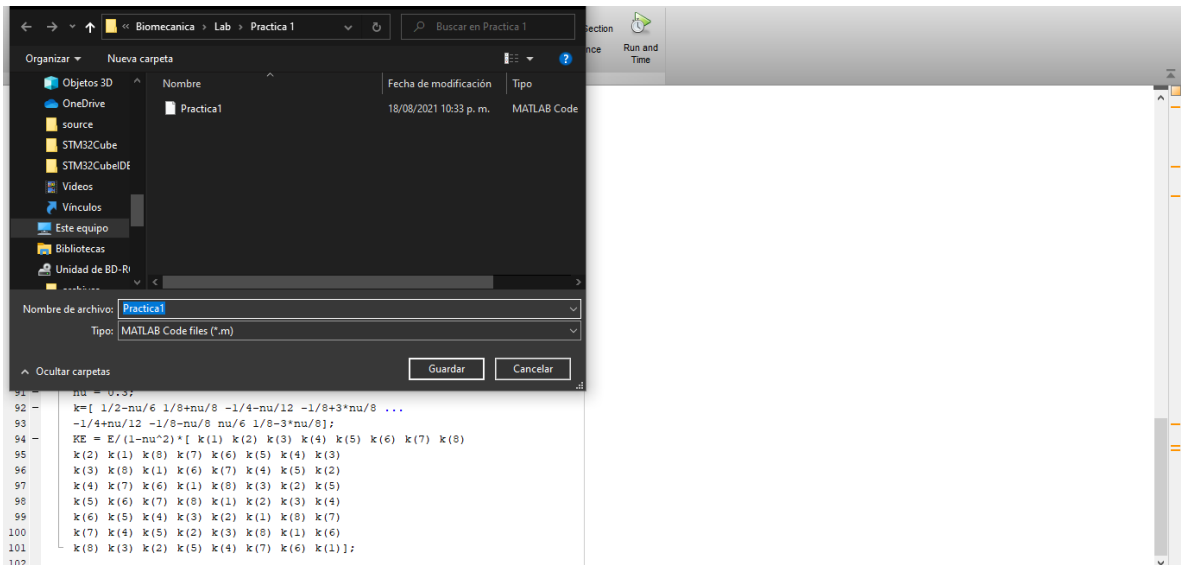
```
EDITOR PUBLISH VIEW
New Open Save Find Files Compare Go To Comment % Indent Breakpoints Run Run and Advance Run Section Run and Time
FILE NAVIGATE EDIT BREAKPOINTS RUN

49 end
50 end
51 %***** MESH-INDEPENDENCY FILTER *****
52 function [dcn]=check(nelx,nely,rmin,x,dc)
53 dcn=zeros(nely,nelx);
54 for i = 1:nelx
55 for j = 1:nely
56 sum=0.0;
57 for k = max(i-round(rmin),1):min(i+round(rmin),nelx)
58 for l = max(j-round(rmin),1):min(j+round(rmin), nely)
59 fac = rmin-sqrt((i-k)^2+(j-l)^2);
60 sum = sum+max(0,fac);
61 dcn(j,i) = dcn(j,i) + max(0,fac)*x(1,k)*dc(1,k);
62 end
63 end
64 dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
65 end
66 end
67 %***** FE-ANALYSIS *****
68 function [U]=FE(nelx,nely,x,penal)
69 [KE] = lk;
70 K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
71 F = sparse(2*(nely+1)*(nelx+1),1); U =sparse(2*(nely+1)*(nelx+1),1);
72 for ely = 1:nely
73 for elx = 1:nelx
74 n1 = (nely+1)*(elx-1)+ely;
75 n2 = (nely+1)* elx +ely;
76 edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1;2*n2+2;2*n1+1; 2*n1+2];
77 K(edof,edof) = K(edof,edof) + x(ely,elx)*penal*KE;
78 end
79 end
```

```
Practical / OC Ln 46 Col 12
EDITOR PUBLISH VIEW
New Open Save Find Files Compare Go To Comment % Indent Breakpoints Run Run and Advance Run Section Run and Time
FILE NAVIGATE EDIT BREAKPOINTS RUN

72 for ely = 1:nely
73 for elx = 1:nelx
74 n1 = (nely+1)*(elx-1)+ely;
75 n2 = (nely+1)* elx +ely;
76 edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1;2*n2+2;2*n1+1; 2*n1+2];
77 K(edof,edof) = K(edof,edof) + x(ely,elx)*penal*KE;
78 end
79 end
80 % DEFINE LOADSAND SUPPORTS (HALF MBB-BEAM)
81 F(2,1) = -1;
82 fixeddofs = union([1:2*2*(nely+1)], [2*(nelx+1)*(nely+1)]);
83 alldofs = [1:2*(nely+1)*(nelx+1)];
84 freedofs = setdiff(alldofs,fixeddofs);
85 % SOLVING
86 U(freedofs,:) = K(freedofs,freedofs) \F(freedofs,:);
87 U(fixeddofs,:) = 0;
88 %***** ELEMENT STIFFNESS MATRIX *****
89 function [KE]=lk
90 E = 1.;
91 nu = 0.3;
92 k=[ 1/2-nu/6 1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
93 -1/4+nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8];
94 KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
95 k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
96 k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
97 k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
98 k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
99 k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
100 k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
101 k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
102
```

- Guardamos el trabajo antes de correrlo y ver su resultado en la ventana de comando



- A continuación, corremos el código.

Código de 99 líneas

```

%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, OCTOBER 1999
function Practical(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
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-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),1); U =sparse(2*(nely+1)*(nelx+1),1);
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(2,1) = -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)];

```

Conclusiones:

Fue una práctica sencilla debido a que solamente explicamos el funcionamiento del código de manera que las practicas que siguen lo pondremos en práctica de manera eficaz en todo lo que se nos aborde aquí en el laboratorio al igual en clase. Personalmente se me hizo muy fácil ya que simplemente era explicar el funcionamiento del código, y aunque no soy muy bueno en Matlab pude reconocer las funciones que se hacían en el código y con ayuda de mis compañeros pude comprender mejor lo que se hacía.

Referencias:

- 99 Line Topology Optimization Code – O. Sigmund, Department of Solid Mechanics, Building 404, Technical University of Denmark, DK-2800 Lyngby, Denmark.