



Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Laboratorio de biomecánica

“PRÁCTICA 1”

Instructor(a): Ing. Isaac Estrada

Brigada: 109

Nombre	Matrícula	Carrera
Victor Emmanuel Cantú Corpus	1909659	IMC
Mauricio Julián Salazar Salzar	1906944	IMC
Brayan Orlando Belloc Castillo	1898242	IMC

Semestre Agosto – Diciembre 2022

Día 5 del mes Septiembre del año 2022

Ciudad Universitaria, San Nicolás de los Garza, Nuevo León

Objetivo

El estudiante conocer 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 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).

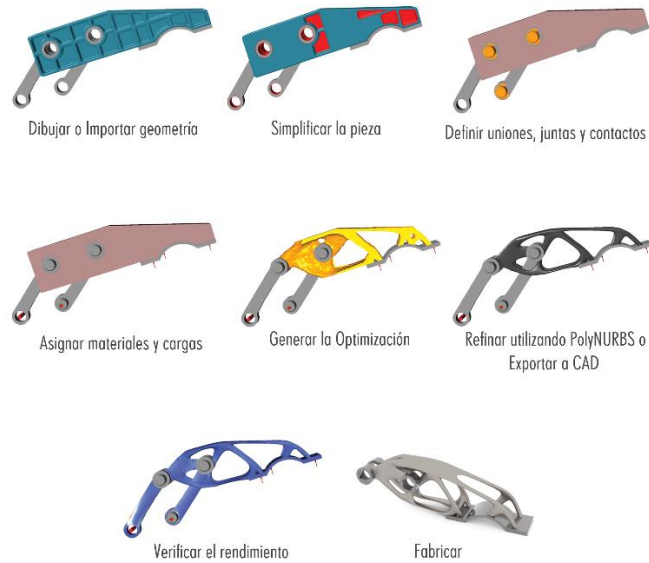
El desarrollo de esta metodología tiene un amplio campo de aplicación para las tecnologías de fabricación aditiva, como por ejemplo la fabricación SLM (Selective Laser Melting), debido a las grandes posibilidades en términos de diseño (geometrías muy complejas).

En el proceso de optimización topológica, se deben de tener en cuenta varios aspectos; el espacio de diseño, el o los casos de carga que va a sufrir la pieza en cuestión, el material y la tecnología con que se va a realizar su fabricación, la reducción de costes mediante la minimización de soportes y aprovechamiento de la cuba de impresión, en caso de utilizar tecnologías aditivas, y muchos más.

Pasos 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

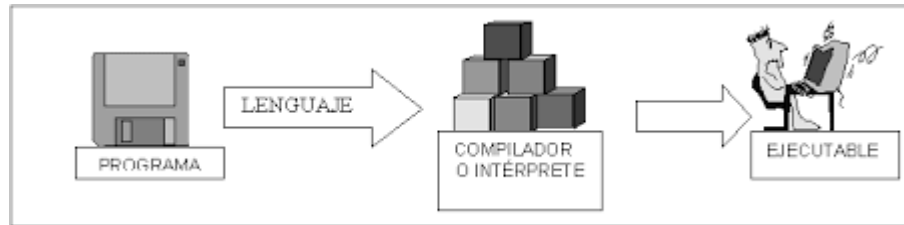


En el ámbito del desarrollo de software, el estado del arte (state of the art) típicamente ha estado vinculado con la evolución de los lenguajes de computación. Uno de los primeros grandes saltos en esa dirección fue el desarrollo del primer lenguaje de alto nivel, FORTRAN.

Una de las vertientes del “state of the art” tiene que ver con lo que denominamos “métodos formales” (MF).

En términos generales, los MF son técnicas mediante las cuales:

- Se escribe la especificación del sistema a desarrollar utilizando un lenguaje formal L1 (usualmente del paradigma declarativo), que luego es verificada con intervención humana y procesada mediante un compilador C1 para así generar el código en otro lenguaje formal L2 que representa un diseño de alto nivel.
- Este diseño escrito en el lenguaje L2 a su vez es verificado con intervención humana y procesado mediante un compilador C2 que genera el código en otro lenguaje formal L3 (usualmente del paradigma imperativo).
- Este código en el lenguaje L3 se procesa mediante un compilador C3 para obtener finalmente el sistema ejecutable.

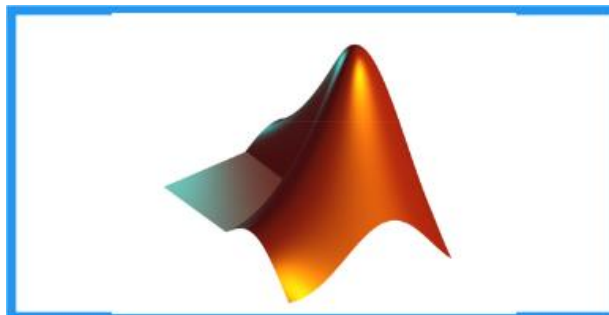


Matlab puede considerarse como un lenguaje de programación tal como C, Fortran, Java, etc. Algunas de las características de Matlab son:

- ❖ La programación es mucho más sencilla.
- ❖ Hay continuidad entre valores enteros, reales y complejos.
- ❖ La amplitud del intervalo y la exactitud de los números es mayor.
- ❖ Cuenta con una biblioteca matemática amplia.
- ❖ Abundantes herramientas gráficas, incluidas funciones de interfaz gráfica con el usuario.
- ❖ Capacidad de vincularse con los lenguajes de programación tradicionales.
- ❖ Transportabilidad de los programas.

Algunas de sus desventajas son:

- ❖ Necesita de muchos recursos de sistema como son Memoria, tarjeta de videos, etc. para funcionar correctamente.
- ❖ El tiempo de ejecución es lento.
- ❖ No genera código ejecutable.
- ❖ Es caro.



Procedimiento de la programación

```
%%%%%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE
%%%%%%%% CODE MODIFIED FOR INCREASED SPEED, SEPTEMBER 2002, 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.;
    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
```

```

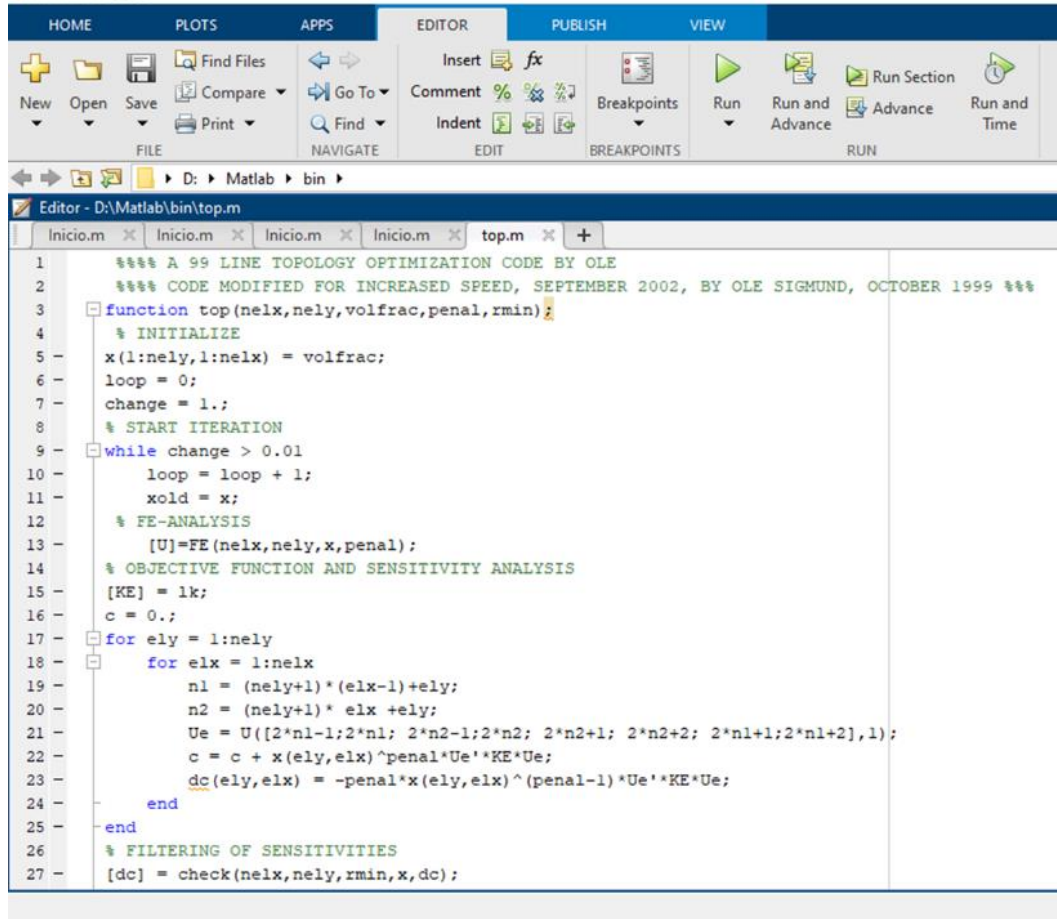
colormap(gray); imagesc(-x); axis equal; axis tight; axis off; pause(1e-6);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%%%%%%
function [xnew]=OC(nelx,nely,x,volfrac,dc)
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
    lmid = 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
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
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*(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)];

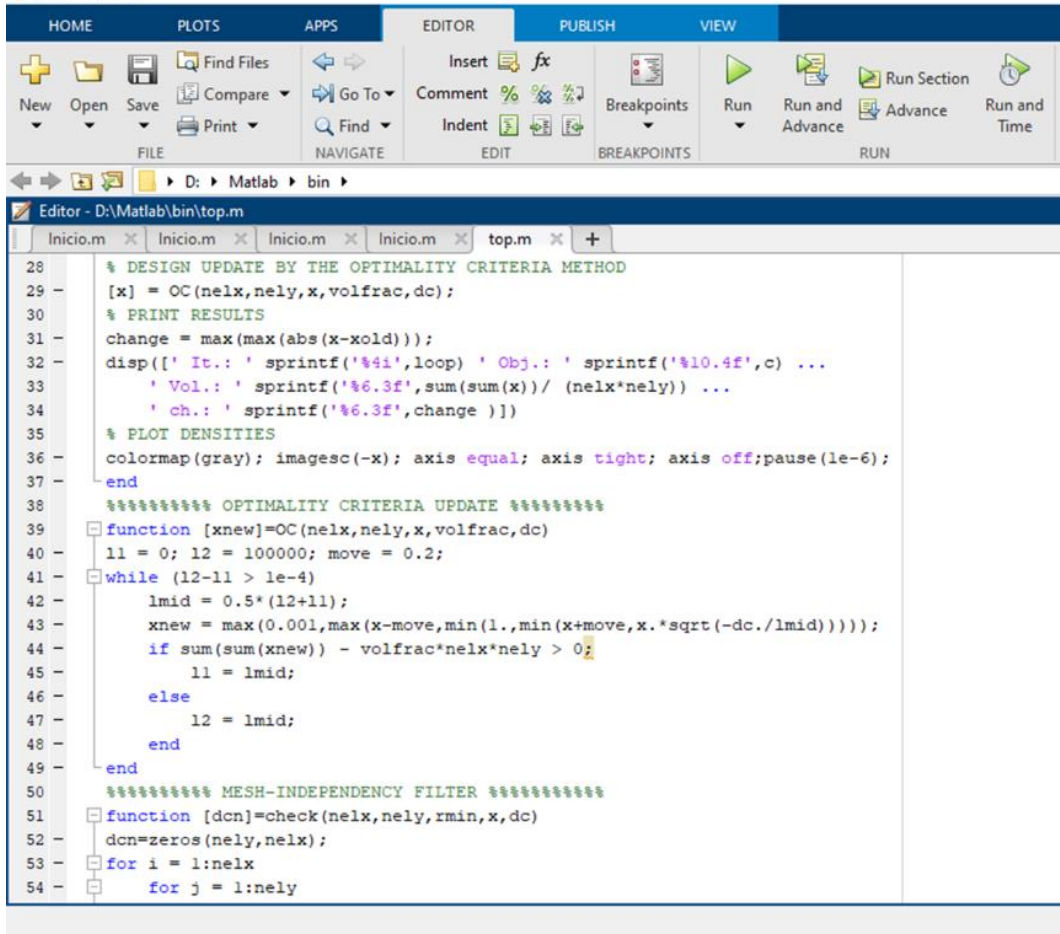
```



```

1  %%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE
2  %%% CODE MODIFIED FOR INCREASED SPEED, SEPTEMBER 2002, BY OLE SIGMUND, OCTOBER 1999 %%%
3  function top(nelx,nely,volfrac,penal,rmin);
4  % INITIALIZE
5  x(1:nely,1:nelx) = volfrac;
6  loop = 0;
7  change = 1.;
8  % START ITERATION
9  while change > 0.01
10     loop = loop + 1;
11     xold = x;
12     % FE-ANALYSIS
13     [U]=FE(nelx,nely,x,penal);
14     % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
15     [KE] = lk;
16     c = 0.;
17     for ely = 1:nely
18         for elx = 1:nelx
19             n1 = (nely+1)*(elx-1)+ely;
20             n2 = (nely+1)* elx +ely;
21             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);
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)));
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 tight; axis off; pause(1e-6);
37 end
38 %***** OPTIMALITY CRITERIA UPDATE *****
39 function [xnew]=OC(nelx,nely,x,volfrac,dc)
40 l1 = 0; l2 = 100000; move = 0.2;
41 while (l2-l1 > 1e-4)
42     lmid = 0.5*(l2+l1);
43     xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
44     if sum(sum(xnew)) - volfrac*nelx*nely > 0;
45         l1 = lmid;
46     else
47         l2 = lmid;
48     end
49 end
50 %***** MESH-INDEPENDENCY FILTER *****
51 function [dcn]=check(nelx,nely,rmin,x,dc)
52 dcn=zeros(nely,nelx);
53 for i = 1:nelx
54     for j = 1:nely

```

HOME	PLOTS	APPS	EDITOR	PUBLISH	VIEW
New Open Save Compare Print	Find Files Go To Find	Insert Comment Indent	Breakpoints Run Run and Advance Run Section Advance Run and Time	Breakpoints Run Run and Advance Run Section Advance Run and Time	Breakpoints Run Run and Advance Run Section Advance Run and Time

Editor - D:\Matlab\bin\top.m

```

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

```

MATLAB R2020a

HOME PLOTS APPS EDITOR PUBLISH VIEW

New Open Save Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run and Time

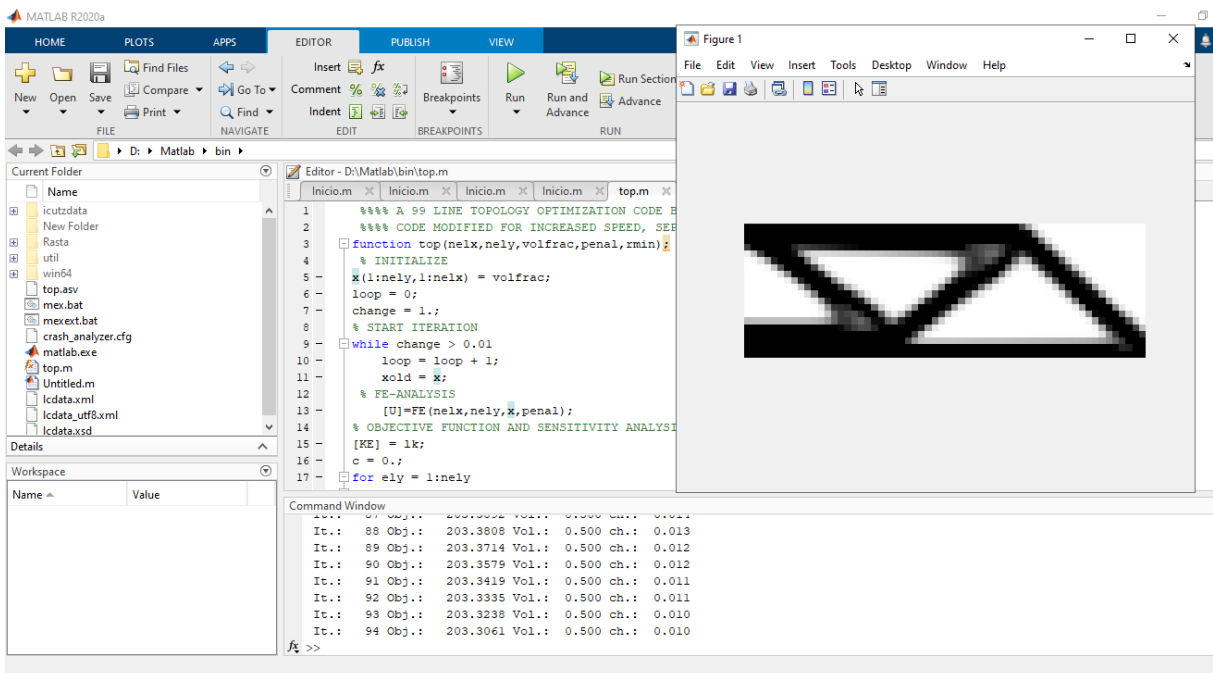
FILE NAVIGATE EDIT BREAKPOINTS RUN

Editor - D:\Matlab\bin\top.m

```

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

```



Conclusión

Victor Cantú: Gracias a esta práctica pudimos conocer diversos términos o conceptos que se ven relacionados con el trabajo de superficies o geometría mediante una codificación previa concluyendo con la construcción de algún sólido, utilizando el simulador o programa que se desee para la ocasión. Un ejemplo es el programa de Matlab, ya que permite realizar diferentes simulaciones dependiendo del campo para el cual se esté utilizando el simulador, es una gran herramienta en cuanto a fines educativos y es considerado uno de los softwares más completos. Nos quedamos con que la optimización topológica es una técnica muy utilizada en el ámbito de la estructura, principalmente se basa en analizar de manera mecánica una estructura o componente; gracias a todas las nuevas herramientas computacionales ya es posible llevar esto de la optimización a un nivel superior a tal nivel de aplicarlo en un enfoque donde ya tenga mayor importancia o impacto dicho componente o estructura.

Mauricio Salazar: Gracias a este trabajo comprendimos un poco más la manera en que se lleva a cabo una simulación en matlab, en esta ocasión se utilizó específicamente para calcular los diseños que una pieza y ver sus deformaciones que viene siendo lo que nos muestra el matlab, es algo muy práctico ya que el trabajo es realizar el código y el programa ya nos arroja todos los resultados, una posible falla o todo este tipo de factores externos que podrían afectar.

Brayan Belloc: La práctica en sí nos mostró una parte de la manera en la que se puede hacer una optimización topológica dentro de la aplicación MATLAB, esta misma optimización puede hacerse en otras aplicaciones de simulación. Por ejemplo, tenemos a SolidWorks, el cual es una aplicación de diseño de piezas, donde se puede simular la pieza cuando es sometida a ciertas cargas o fenómenos, pero el hecho de que se haya trabajado en MATLAB nos permite ver las ecuaciones usadas para este proceso. Claro que en las aplicaciones de diseño de piezas, estas ecuaciones actúan de manera discreta, y solo nos muestra la pieza y las partes donde ocurren fenómenos o deformaciones, de esta forma se puede decir que la práctica nos mostró las ecuaciones y procesos que se realiza dentro de las aplicaciones o programas de

diseño.

Bibliografía

- Optimización Topológica | Catec. (s. f.). Catec. Recuperado 29 de agosto de 2022, de <http://www.catec.aero/es/materiales-y-procesos/l%C3%ADnea-de-investigaci%C3%B3n/optimizaci%C3%B3n-topol%C3%B3gica#:~:text=La%20optimizaci%C3%B3n%20topol%C3%B3gica%20es%20una,funcionalidades%20mec%C3%A1nicas%20del%20compone%20objetivo.>
- Optimización Topológica. (2019, 9 enero). Estudio de Ingeniería y Tecnología Avanzada S.L. Recuperado 29 de agosto de 2022, de <https://eitaingenieros.com/optimizacion/>
- Introducción. (s. f.). Calderon. Recuperado 5 de septiembre de 2022, de <https://lc.fie.umich.mx/%7Ecalderon/Matlab/Introduccion.html>
- El Estado del Arte y la Prueba de Software. (s. f.). SG Buzz. Recuperado 5 de septiembre de 2022, de <https://sg.com.mx/revista/30/estado-arte-prueba-software#:~:text=En%20el%20%C3%A1mbito%20del%20desarrollo,lenguaje%20de%20alto%20nivel%2C%20FORTRAN.>