

# Practica No.1

## Equipo 6

Heber Adrián Casillas Gutiérrez 1894878

Luis Mateo Landa Rivera 1909998

Bruno Mendoza Palomo 1992283

Juan Erasmo Guerrero Treviño 1903220

Juan Javier missael Castillo Ruiz 1884560

Merary Castillo Sanchez 1895677

7 de septiembre de 2022

## Resumen

### 1. Objetivo

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

### 2. 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).

Como funciona la optimización topologica Los procesos tradicionales de diseño digital conllevan aplicar cargas a una pieza ya fabricada y evaluar dónde se está debilitando. Luego, los ingenieros deben repensar el diseño hasta que la pieza cumpla con las restricciones mecánicas dadas. Con la optimización topológica, el sentido es diferente: las cargas mecánicas son los datos de entrada que permitirán al software proponer una nueva geometría de la pieza. Así, en principio hay menos iteraciones, lo que reduce considerablemente los tiempos de diseño y fabricación.

La optimización topológica comienza con la creación de un modelo 3D en la fase de borrador, en el que se aplicaran las diferentes cargas o fuerzas para la pieza (una presión sobre las lengüetas de sujeción, por ejemplo). Después, el software se encarga de calcular todas las tensiones aplicadas. En este nivel, se puede realizar un corte de la pieza con el fin de retirar las partes no sometidas a las fuerzas. La geometría final, que cumple con los requisitos mecánicos y

de diseño, se puede obtener finalmente después de alisar la pieza. De esta forma, la optimización topológica responde a la necesidad de reducción de masa además del aumento de la resistencia mecánica de la pieza.

Los softwares dedicados a la optimización topológica

No todos los software CAD ofrecen esta función de optimización topológica. Aunque no se utiliza necesariamente en el proceso de modelado y fabricación, puede ser mejor optar por un programa que la incluya, para asegurarse de diseñar una pieza óptima. Es por ello que algunas empresas han decidido desarrollar un software dedicado a este proceso. Uno de los pioneros es sin duda Altair, con su solución OptiStruct, que luego llevó a otra solución, Altair Inspire. También existen otras opciones como la de Ansys, Dassault Systèmes, Autodesk o incluso nTopology. Además, hay muchos programas de CAD que integran funciones para optimizar las piezas, como Solidworks, Creo o Fusion 360.

¿Quién utiliza esta tecnología y con qué fin?

La industria automotriz abordó rápidamente este problema debido a la reducción de costes mediante el ahorro en materias primas asociadas con los tamaños de serie. De hecho, la reducción de unos pocos gramos por cada vehículo, en una producción de varios millones de unidades, representa toneladas de material ahorrado. La aeronáutica es sin duda otro sector interesado en la optimización topológica, con el objetivo de reducir costes indirectos. Un avión más ligero consume menos combustible, lo que, a la larga, genera importantes ahorros para una aerolínea.

### 3. Nombre y definicionde la programacion

El código de optimización topológica de 99 líneas en Matlab que se utilizara en este laboratorio se divide en 36 líneas para la programación principal, 12 líneas para los criterios de optimización, 16 líneas para el filtro de mallado y 35 líneas para el código de elemento finito. De hecho, excluyendo las líneas de comentarios y líneas asociadas con la producción y el análisis de elementos finitos, el código resultante es de solo 49 líneas. 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>.

### 4. Estado del Arte

En ingeniería siempre se desea determinar la mejor configuración de una pieza o un sistema para optimizar algunas características importantes en su desempeño final. En el caso de la optimización estructural, para obtener la solución óptima del problema se emplean diferentes técnicas, que van desde enfoques netamente empíricos, hasta métodos matemáticos analíticos y numéricos. En el caso de la optimización estructural, los métodos numéricos poseen la mayor relevancia y se pueden dividir en tres categorías: optimización paramétrica, optimización de forma y optimización topológica. La Figura 2 muestra gráficamente en qué consisten estas tres técnicas numéricas.

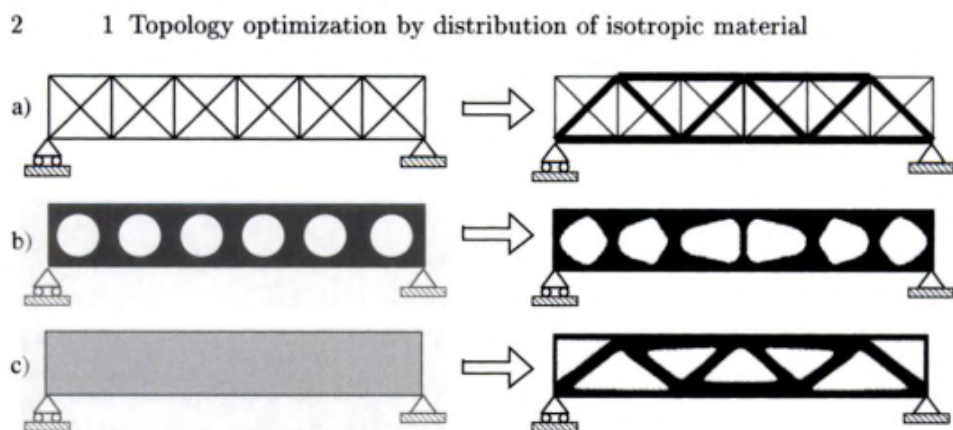


Figura 2: Categorías de optimización estructural: (a) optimización paramétrica, (b) optimización de forma y (c) optimización topológica

La primera categoría es la optimización paramétrica, que consiste en discretizar una estructura preestablecida con elementos cercha (barra articulada) para encontrar las dimensiones óptimas de la estructura. Las variables del diseño que pueden ser modificadas son el área transversal de cada elemento, longitudes, espesores y radios de entalle [1]. La segunda categoría es la optimización de forma, donde se busca encontrar la forma óptima de un dominio sin modificar su topología, es decir, sin agregar huecos o cavidades en su interior. En este caso se parametrizan los contornos internos y externos de la estructura mediante curvas splines o nurbs, para controlar la geometría del diseño. Los parámetros de estas curvas son las variables del diseño que dan como resultado la forma óptima de la estructura

La última categoría es la optimización topológica, donde el objetivo es distribuir el material en el dominio para encontrar la estructura. En este problema se introducen huecos o cavidades que en el inicio no estaban presentes. Inicialmente solo se dispone de la información de las condiciones de carga y las restricciones (apoyos), y el dominio inicial donde se desarrollará la estructura

### **OPTIMIZACIÓN TOPOLÓGICA DE ESTRUCTURAS CONTINUAS**

Cuando se considera la optimización topológica a partir de un medio continuo, lo que se busca es la óptima configuración de la estructura, donde el dominio se discretiza en elementos finitos que representan divisiones del material.

### **OPTIMIZACIÓN TOPOLÓGICA DE ESTRUCTURAS DISCRETAS**

La optimización topológica a partir de un dominio discreto se divide en dos categorías, optimización topológica de estructura de malla continua optimización topológica de estructuras discretas. Donde la optimización topológica de estructura de malla continua considera un dominio continuo de la estructura discretizando por un número infinito barras rígidas distanciadas por un espacio infinitesimal, cuya solución óptima es obtenida analíticamente por la teoría de la elasticidad. La optimización topológica de estructuras discretas considera un dominio donde hay varios puntos distribuidos y pueden ser posibles juntas, cuya solución óptima se obtiene numéricamente.

### **TÉCNICAS DE SOLUCIÓN DEL PROBLEMA DE OT**

A medida que ha ido evolucionando la optimización topológica, su implementación en problemas con un mayor grado de complejidad son más comunes. Esto ha requerido el desarrollo de nuevas técnicas de solución. Dentro de las técnicas de solución se pueden mencionar el criterio de optimalidad OC, el método de la asíntota móvil MMA y el método de programación lineal PLS, entre otros. El criterio de optimalidad OC [4] permite solucionar de manera eficiente los problemas simples de optimización topológica. Para problemas más complejos, como, por ejemplo, en el diseño de mecanismos electro-mecánicos, donde los problemas elástico y térmico deben ser solucionados simultáneamente, se destacan los métodos MMA [3] y PLS [2]. pues presentan buena convergencia.

## 5. Desarrollo

### Explicación de la programación

1. El programa principal comienza distribuyendo el material uniformemente en el dominio de diseño. Después de algunas otras inicializaciones, el ciclo principal comienza con una llamada a la subrutina de elementos finitos que devuelve el vector de desplazamiento  $U$ . Dado que la matriz de rigidez del elemento para material sólido es la misma para todos los elementos, la subrutina de matriz de rigidez del elemento se llama una sola vez. Después de esto, un ciclo sobre todos los elementos determina la función objetivo y las sensibilidades. Las variables  $n1$  y  $n2$  denotan números de nodo de elemento superior izquierdo y derecho en números de nodo global y se utilizan para extraer el vector de desplazamiento de elemento  $U_e$  del vector de desplazamiento global  $U$ . El análisis de sensibilidad es seguido por una llamada al filtro de independencia de malla y el optimizador Optimality Criteria. El cumplimiento actual, así como otros parámetros, se imprimen en las líneas 30 a 33 y se traza la distribución de densidad resultante. El bucle principal se termina si el cambio en las variables de diseño (cambio determinado en la línea 30) es inferior al 1 por ciento. De lo contrario, se repiten los pasos anteriores.

```
1  %%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
2  %%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%
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
```

2. El optimizador encuentra las variables de diseño actualizadas (líneas 37–48). Sabiendo que el volumen material ( $\text{sum}(\text{sum}(\text{xnew}))$ ) es una función monótonamente decreciente del multiplicador de Lagrange (retraso), el valor del Multiplicador lagrangiano que satisface la restricción de volumen se puede encontrar mediante un algoritmo de bisección. El algoritmo de bisección se inicializa adivinando un  $l1$  inferior y un límite  $l2$  superior para el Lagrangiano multiplicador. El intervalo que limita el multiplicador lagrangiano se divide repetidamente por la mitad hasta que su tamaño es menos que los criterios de convergencia.

```

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

```

3. Tenga en cuenta que no todos los elementos en el dominio de diseño son buscado para encontrar los elementos que se encuentran dentro el radio  $r_{min}$  pero solo aquellos dentro de un cuadrado con lado longitudes dos veces redondas ( $r_{min}$ ) alrededor de la considerada elemento. Seleccionando  $r_{min}$  menos de uno en la llamada del rutina, las sensibilidades filtradas serán iguales a las sensibilidades originales haciendo que el filtro esté inactivo.

```

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
55         sum=0.0;
56         for k = max(i-floor(rmin),1):min(i+floor(rmin),nelx)
57             for l = max(j-floor(rmin),1):min(j+floor(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(l,k)*dc(l,k);
61             end
62         end
63         dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
64     end
65 end

```

4. El código de elementos finitos está escrito en las líneas 65 a 99. Note que el solucionador hace uso de la opción `escasa` en Matlab. La matriz de rigidez global está formada por un bucle sobre todos los elementos. Como sucedía en los principales programas, las variables  $n1$  y  $n2$  indican la parte superior izquierda y derecha números de nodo de elementos en números de nodo globales y son se utiliza para insertar la matriz de rigidez del elemento a la derecha lugares en la matriz de rigidez global. Como se mencionó anteriormente, tanto los nodos como los elementos son columna numerada sabiamente de izquierda a derecha. Es más, cada nodo tiene dos grados de libertad (horizontal y vertical), por lo que el comando `F(2,1)=-1`. Se aplica una fuerza de fuerza unitaria vertical en la esquina superior izquierda. Los soportes se implementan eliminando los grados de libertad fijos de las ecuaciones lineales. Matlab puede hacer esta muy elegante con la línea donde `freedofs` indica los grados de libertad que no están restringidos. En general, es más fácil definir los grados de libertad que son fijos (`fixeddofs`) a partir de entonces el `freedofs` se encuentran automáticamente usando el operador de Matlab `setdiff` que encuentra los grados libres de libertad como la diferencia entre todos los grados de libertad y el fijo grados de libertad. La matriz de rigidez del elemento se calcula en las líneas 86– 99. La matriz de 8 por 8 para un elemento cuadrado bilineal 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.

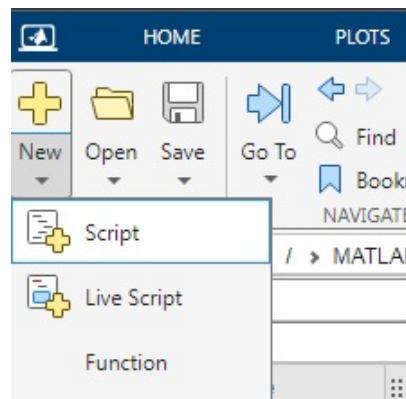
```

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 = zeros(2*(nely+1)*(nelx+1),1);
71 for elx = 1:nelx
72     for ely = 1:nely
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 LOADS AND SUPPORTS (HALF MBB-BEAM)
80 F(2,1) = -1;
81 fixeddofs = union([1:2*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)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
94                  k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
95                  k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
96                  k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
97                  k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
98                  k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
99                  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)];

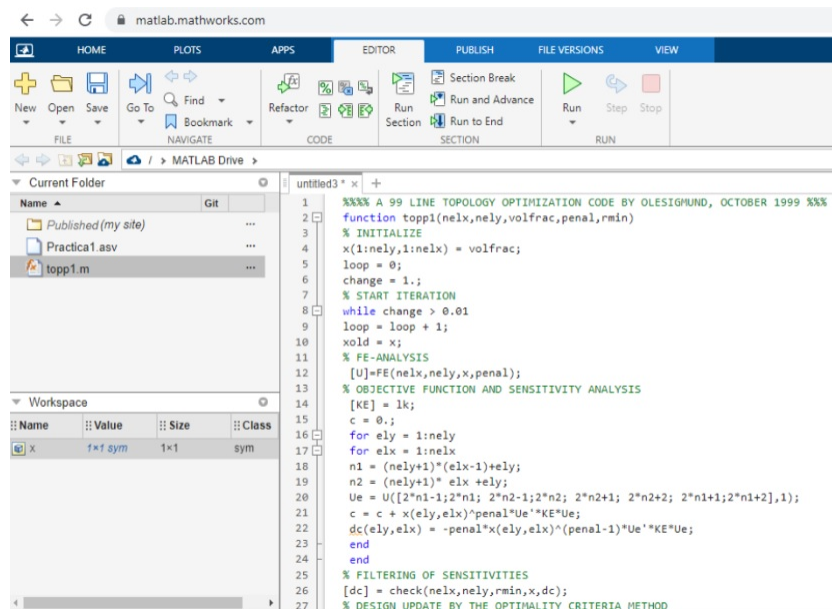
```

## Procedimiento de la programación

El código de Matlab está compuesto como un código de optimización topológica estándar, el cual está listo para ser interpretado por MATLAB luego de llevar a cabo la siguiente serie de sencillos pasos: 1.- Abrir MATLAB y esperar a que este se inicialice, y muestre su pantalla principal. 2.- Una vez en la pantalla de inicio de MATLAB es necesario seleccionar en la barra de herramientas Home → New → Script, tal como muestra la imagen, con lo que se abre un editor de texto, dentro del cual será necesario escribir el código proporcionado.



2. Escribimos nuestro código en el Script.

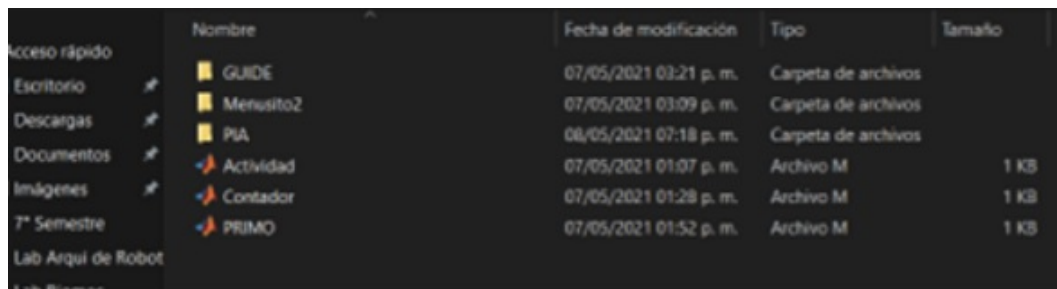


```

1 %%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIKUND, OCTOBER 1999 %%%
2 function topp1(nelx,nely,volfrac,penal,rmin)
3 % INITIALIZE
4 x(1:nely,1:nelx) = volfrac;
5 loop = 0;
6 change = 1.;
7 % START ITERATION
8 while change > 0.01
9     loop = loop + 1;
10    vold = 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; 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

```

3.- Una vez con el código completamente escrito en el editor de texto, es necesario salvar el archivo, teniendo especial atención en la ubicación donde se va a guardar el script, así como en el nombre que se le va a asignar al archivo. Se recomienda que el archivo se guarde en el directorio raíz de MATLAB que por default muestra es en el que el editor de texto nos ubica al seleccionar File → Save como muestra la siguiente imagen. En caso de no ser así, debemos de navegar a “Mis Documentos/MATLAB” y guardar el script que esta siendo guardado en el directorio de MATLAB con el nombre “topp1”.





4. Una vez guardado, agregamos la línea de entrada para nuestra topología, dicha línea de entrada será escrita en la ventana de comando de Matlab

```

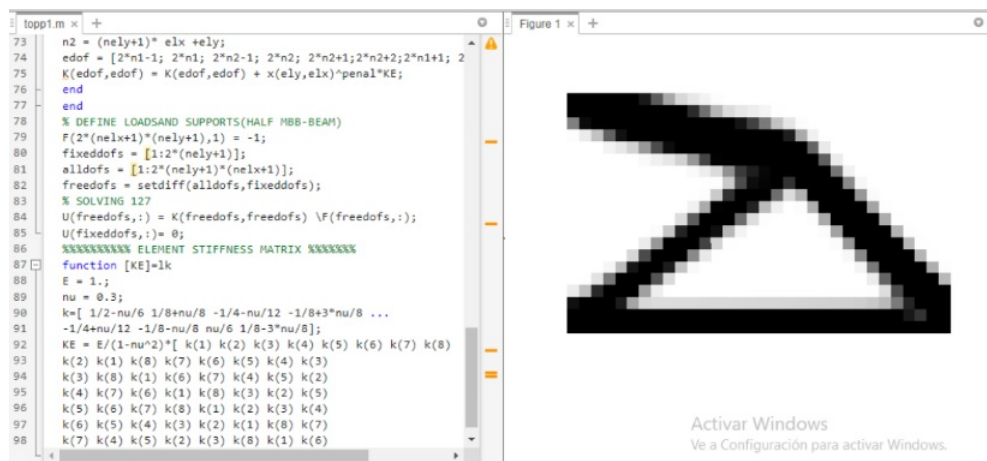
91 -1/4+nu/12 -1/8-nu/8 nu/8 1/8-3*nu/8];
92 KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
>> topp1(32,20,0.4,3.0,1.2)]

```

Activar Windows  
Ve a Configuración para activar Windows.

COMMAND WINDOW UTF-8 CRLF topp1 / k: Ln 89 Col 10

5. Una vez hecho esto, damos correr, y se observará nuestra topología, la cual se trata de una Topología de una Viga en Voladizo.



## 6. Conclusiones

Juan Javier Missael Castillo Ruiz 1884560: Tenia un poco de conocimiento sobre este tema que se puede llegar a hacer en matlab ya que anteriormente en clase de analisis de elemento finito tocamos el tema pero fue muy basico lo que llegamos a ver. Aprendi sobre como por mediante de algunos softwares podemos ayudarnos para ya que en caso de que no podamos ver algunas cosas como esta en la vida diaria podemos simularlas y darnos una idea de lo que podria llegar a pasar mediante las simulaciones y asi tener una visualizacion de lo que podria llegar a pasar.

Heber Adrián Casillas Gutiérrez 1894878: El manejo de matlab para simulaciones como por ejemplo la creada en este reporte es muy bueno por que llegar abarcar muchos temas en general tanto como las optimizaciones y ver que tecnicas de solucion son las mejores para la contruccion de un buen programa.

Luis Mateo Landa Rivera 1909998: Al ir creando el codigo en matlab me di cuenta que hay muchas cosas que no sabia en general que como unos temas diferentes a lo que es la creacion de un codigo se pueden acoplar tan ordenada y se ejecuta de una manera correcta para la compresion del tema en su totalidad.

Juan Erasmo Guerrero Treviño 1903220: Este reporte me ayudo mucho en generar nuevos conocimientos acerca de la programacion de un documento que no sabia que se podrian hacer y como se van relacionando muchos temas en matlab que anteriormente no conocia pero al ir haciendo el reporte me ayudo a comprender mucho mejor como se manejan diferentes estructuras en un codigo, mientras creas un tema relacionado con la materia.

Bruno Mendoza Palomo 1992283: Me gusto como tuvo relacion la optimizacion topologica en un analisis ejecutable en matlab por que no sabia como se manejaba en su totalidad la optimizacion pero en base que fui avanzando el reporte observar que muchas cosas pueden ser simularlas con ayuda de programas como matlab algo que no conocia anteriormente.

Merary Castillo Sanchez 1895677: El manejo de overlaf me gusto mucho para la creacion de documentos me ayuda a explorar mas cosas sobre el mundo de la programacion que no sea la orientacion en objetos y ver que se pueden



usar en diferentes temas cotidianos por ejemplo el tema de este reporte fue hacer el analisis en matlab y ver los resultados de una simulacion y fue muy bueno para la experiencia en documentos y reportes.

## 7. Bibliografía

- [1] Bendsee, M. P., and Sigmund, O. Topology Optimization: Theory, Methods and Applications. Springer Verlag, Berlin, 2003.
- [2] Carbonari, R. C. Project of piezéltricos flextensionais actuators using the method of topological optimization. Master's thesis, Polytechnic School of the University of São Paulo, 2003
- [3] Svanberg, K. The method of moving asymptotes a new method for structural optimization. Int. J. Numer. Meth. Engng. 24 (1987), 359-373
- [4] Yin. L. and Yang, W. Optimality criteria method for topology optimization under multiple constraints. Computers and Structures 79 (2001), 1839-1850