Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

- *Tarea: Código de 99 líneas para la optimización topológica.*

Maestro: Isaac Estrada.

Nombre:	**Matricula:**
Felipe Daniel Zamarripa Valdez	1755483
Miguel Angel Morales Arredondo	1987049
Severo Wenceslao Arreola Platas	1915391
Alvaro Alexis García Garza	1605716
Juan Angel Reyna Medrano	1669008

Materia: Laboratorio de Biomecánica Hora: Viernes N5

Cd. Universitaria, San Nicolás de los Garza, N.L.

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.

Marco Teórico:

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).

Un problema clásico de la ingeniería consiste en determinar la configuración geométrica óptima de un cuerpo que minimice o maximice una cierta función objetivo, al mismo tiempo que satisface las restricciones o condiciones de contorno del problema. La solución de este problema puede ser planteada utilizando dos estrategias: como un problema de optimización de forma o de optimización de la topología.

La optimización de forma consiste en modificar la geometría del dominio preservando su topología, es decir sin crear huecos o cavidades en su interior. Este tipo de análisis es usualmente conocido como análisis de sensibilidad al cambio de forma y sus bases matemáticas se encuentran bien establecidas. El principal inconveniente del análisis de sensibilidad al cambio de forma es que sólo permite cambios en la frontera del dominio, lo que limita su campo de aplicación.

Una manera más general de controlar un dominio es mediante modificaciones de su topología, lo que permite obtener la configuración deseada partiendo de una morfología inicial distante de la óptima. Los métodos de homogenización son posiblemente los más utilizados para la optimización topológica. Estos consisten en caracterizar la topología a través de su densidad, es decir, los huecos se identifican con regiones de densidad nula. De esta forma la solución del programa resulta en una distribución ficticia de material.

Matlab es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.

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.**

Desarrollo:

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 éste 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 figura 1, con lo que se abre un editor de texto, dentro del cual será necesario escribir el código proporcionado.

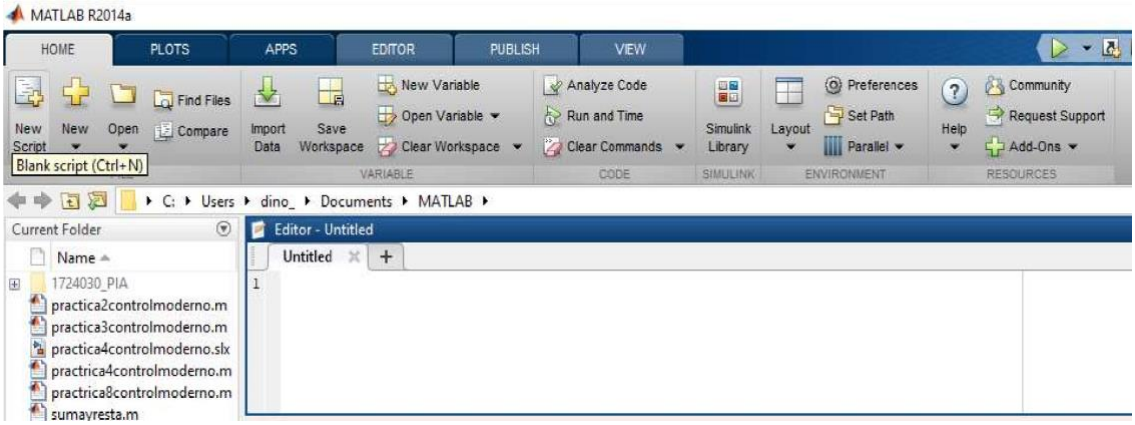

Figura 1.- Opción a seleccionar para crear un nuevo script en MATLAB.

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 figura 1.2. En caso de no ser así, debemos de navegar a "Mis Documentos/MATLAB" y guardar aquí el script recién creado. La figura 1.2 ejemplifica una script que está siendo guardado en el directorio de MATLAB con el nombre "topp1".

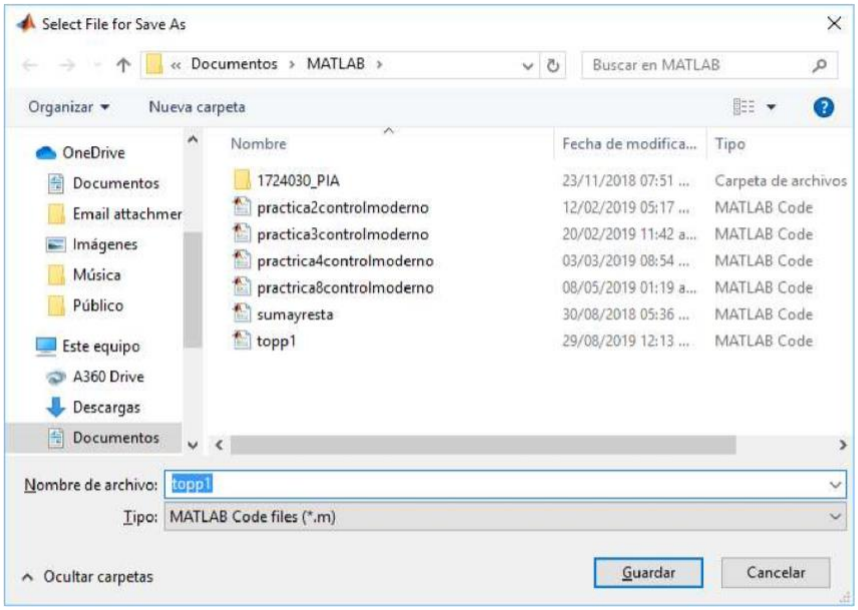

Figura 1.2 Ventana para guardar script, dentro de la carpeta raíz de MATLAB.

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 R2010a del software, el directorio actual del intérprete se encuentra en la barra de herramientas superior, como muestra la figura 1.3.

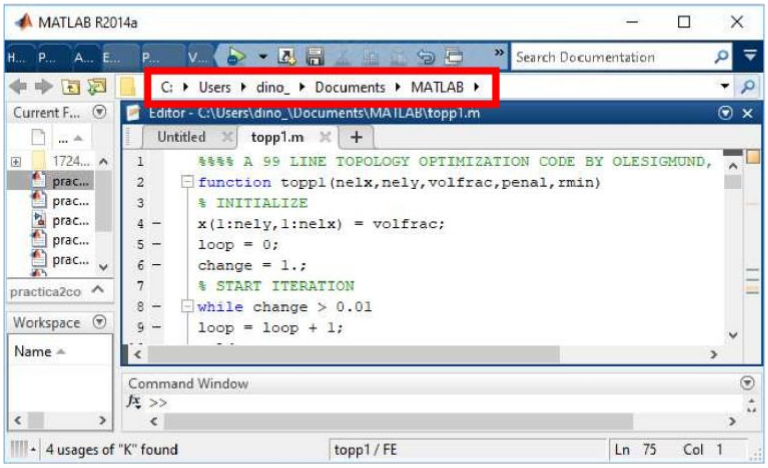

Figura 1.3 Directorio actual del intérprete de MATLAB

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 con cargas y restricciones, este caso en particular, es evaluado y simulado cuando escribimos desde la línea de comando de MATLAB “topp1(60,20,0.5,3.0,1.5)”.

Resultados:

a) El documento presenta una implementación compacta de Matlab de un código de optimización de topología para minimizar el cumplimiento de estructuras cargadas estáticamente. El número total de líneas de entrada de Matlab es 99, incluido el optimizador y la subrutina de elementos finitos. Las 99 líneas se dividen en 36 líneas para el programa principal, 12 líneas para el optimizador basado en Criterios de Optimidad, 16 líneas para un filtro de independencia de malla y 35 líneas para el código de elementos finitos. De hecho, excluyendo las líneas de comentarios y las líneas asociadas con la salida y el análisis de elementos finitos, se muestra que solo se requieren 49 líneas de entrada de Matlab para resolver un problema de optimización de topología bien planteado. Al agregar tres líneas adicionales, el programa puede resolver problemas con múltiples casos de carga. El código está destinado a fines educativos.

b) Código de 99 líneas

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function topp1(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)
```

Resultados:

a) El documento presenta una implementación compacta de Matlab de un código de optimización de topología para minimizar el cumplimiento de estructuras cargadas estáticamente. El número total de líneas de entrada de Matlab es 99, incluido el optimizador y la subrutina de elementos finitos. Las 99 líneas se dividen en 36 líneas para el programa principal, 12 líneas para el optimizador basado en Criterios de Optimidad, 16 líneas para un filtro de independencia de malla y 35 líneas para el código de elementos finitos. De hecho, excluyendo las líneas de comentarios y las líneas asociadas con la salida y el análisis de elementos finitos, se muestra que solo se requieren 49 líneas de entrada de Matlab para resolver un problema de optimización de topología bien planteado. Al agregar tres líneas adicionales, el programa puede resolver problemas con múltiples casos de carga. El código está destinado a fines educativos.

b) Código de 99 líneas

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function topp1(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)
```

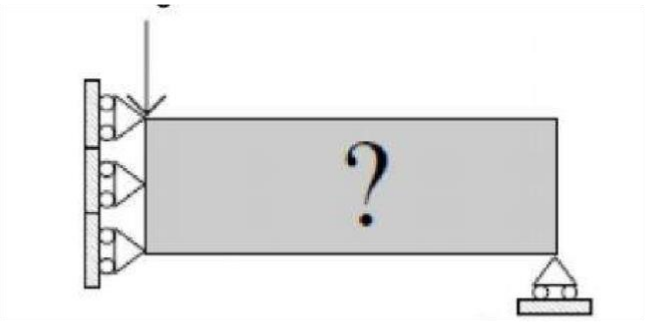

Figura 1.4 Dominio de diseño, carga y restricciones propuestas.

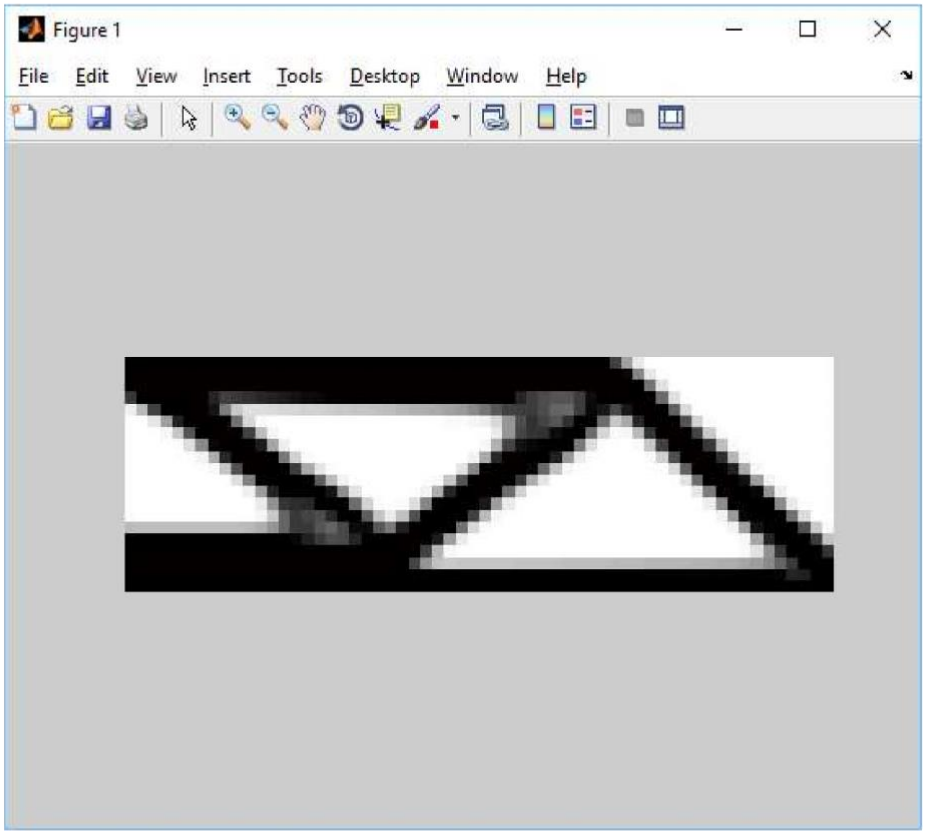

Figura 1.5 Imagen del diseño optimizado.

c) Definir cada una de las variables de entrada. Conociendo que pusimos dentro del programa 'topp1(60,20,0.5,3.0,1.5)¿

- 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).

Conclusiones:

En esta practica aprendimos a usar MATLAB para realizar análisis de elemento finito para objetos de ámbito simple, dicho análisis nos ayuda mucho a optimizar piezas para hacerlas más seguras, resistentes y al menor costo siendo prueba de que el hacer uso de la tecnología puede facilitarnos el trabajo y realizarlo con una precisión mayor.