

Práctica 1

Optimización topológica con MATLAB

Equipo 4

Jorge Fuentes, Tania Hernandez, Anahi Herrera, Gustavo Díaz, Miriam Mata, Alejandro Ramos

9 de septiembre de 2022

Resumen

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

La optimización de la topología comenzó en 1904 con las estructuras de Michell. Aquí, Michell consideró las estructuras de vigas y también desarrolló una teoría de diseño según la cual todas las estructuras de vigas se cruzan entre sí en un ángulo de 90° . En consecuencia, se creó una disposición óptima de los esfuerzos de tracción y compresión. Una vez mencionado esto hablemos sobre lo que estaremos trabajando, el código a utilizar se le llama código de optimización de topología de 99 líneas, en la que se presenta una cantidad de 99 líneas de código en Matlab en la que las 99 líneas de entrada corresponden al optimizador y a la subrutina de elemento finito, El código de 99 líneas se divide en varias estructuras que componen cada una a cierta parte del programa que realiza cierta acción. De las 99 líneas de código: 36 líneas pertenecen al programa principal 12 líneas pertenecen a criterio optimizador 16 líneas a la independencia de mallado y 35 líneas al código de elemento finito Viendo el código sin líneas de comentarios obtenemos que con tan solo 19 líneas de código son realmente Necesarias para hacer dicha optimización de geometrías. Agregando 3 líneas más de códigos seremos capaces de agregar múltiples cargas. Donde lo implementaremos en MATLAB que es un programa muy potente con el cual podremos realizar cálculos numéricos con vectores y matrices, trabajar con números escalares, tanto reales como complejos y utilizar una amplia variedad de gráficos en dos y tres dimensiones. MATLAB tiene un lenguaje propio de programación, El lenguaje de cálculo técnico desarrollado por MathWorks, es un entorno de programación para el desarrollo de algoritmos, análisis de datos, visualización y cálculo numérico.

Índice

1. Introducción	2
2. Desarrollo	3
2.1. Nombre y definición de la programación	3
2.1.1. Sobre MATLAB	3
2.1.2. La Optimización topológica	3
2.1.3. Hablemos del Código de 99 líneas	3
2.2. Estado del arte	3
2.2.1. Ventajas que ofrece la optimización de la topología	3
2.2.2. Tipos de optimización de la topología existentes	4
2.2.3. Optimización discreta	4
2.2.4. Optimización continua	4
2.3. Procedimiento de la programación	5
2.4. Implementación de la programación	7
3. Conclusiones	10
3.1. Jorge Fuentes	10
3.2. Tania Hernandez	10
3.3. Anahi Herrera	10
3.4. Gustavo Díaz	10
3.5. Miriam Mata	10
3.6. Alejandro Ramos	10

1. Introducción

En esta práctica se estará realizando el código de optimización topológica de 99 líneas 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. Para poder llevar a cabo la práctica se tuvo que realizar una investigación grupal para poder entender la metodología y para poder llegar a una implementación exitosa en el programa de MATLAB. Las evidencias de lo mismo se anexarán más adelante.

2. Desarrollo

2.1. Nombre y definición de la programación

2.1.1. Sobre MATLAB

Es un programa muy potente con el cual podremos realizar cálculos numéricos con vectores y matrices, trabajar con números escalares, tanto reales como complejos y utilizar una amplia variedad de gráficos en dos y tres dimensiones. MATLAB tiene un lenguaje propio de programación, El lenguaje de cálculo técnico desarrollado por MathWorks, es un entorno de programación para el desarrollo de algoritmos, análisis de datos, visualización y cálculo numérico[1].

2.1.2. La Optimización topológica

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[3]. 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).

2.1.3. Hablemos del Código de 99 líneas

Se le llama código de optimización de topología de 99 líneas, en la que se presenta una cantidad de 99 líneas de código en Matlab en la que las 99 líneas de entrada corresponden al optimizador y a la subrutina de elemento finito, El código de 99 líneas se divide en varias estructuras que componen cada una a cierta parte del programa que realiza cierta acción[2]. De las 99 líneas de código: 36 líneas pertenecen al programa principal 12 líneas pertenecen a criterio optimizador 16 líneas a la independencia de mallado y 35 líneas al código de elemento finito Viendo el código sin líneas de comentarios obtenemos que con tan solo 19 líneas de código son realmente Necesarias para hacer dicha optimización de geometrías. Agregando 3 líneas más de códigos seremos capaces de agregar múltiples cargas.

2.2. Estado del arte

La optimización de la topología comenzó en 1904 con las estructuras de Michell. Aquí, Michell consideró las estructuras de vigas y también desarrolló una teoría de diseño según la cual todas las estructuras de vigas se cruzan entre sí en un ángulo de 90° . En consecuencia, se creó una disposición óptima de los esfuerzos de tracción y compresión. Además, hoy en día también es un método de desarrollo de productos asistido por ordenador con el que se puede reconocer el potencial de optimización en una fase temprana del proceso de desarrollo. Por ello, varios proveedores de software ofrecen hoy en día soluciones para ello.

2.2.1. Ventajas que ofrece la optimización de la topología

Dependiendo de la aplicación, hay casi una multitud de objetivos. Sin embargo, el objetivo fundamental es aprovechar al máximo el espacio de la instalación. También otros ejemplos son estos:

- Reducción de peso con una capacidad de carga invariable
- Aumento de la carga con un peso constante
- Optimización de la distribución de la fuerza en varios puntos de apoyo
- Optimización de la frecuencia natural mediante la relación correcta de masa y rigidez

2.2.2. Tipos de optimización de la topología existentes

En resumen, la optimización topológica se ocupa de encontrar la forma básica (topología) más favorable para los componentes con carga mecánica. En este contexto, se distingue entre optimización continua y discreta, además de los ámbitos de aplicación y los objetivos de optimización descritos anteriormente.

2.2.3. Optimización discreta

El representante más conocido es Anthony George Maldon Michell y su personal trabaja. Estas optimizaciones por trabajos de barra se siguen utilizando hoy en día, ya que las ventajas residen principalmente en el bajo tiempo de cálculo.

2.2.4. Optimización continua

Para ello, el proceso se realiza generalmente en 3 pasos, a saber.

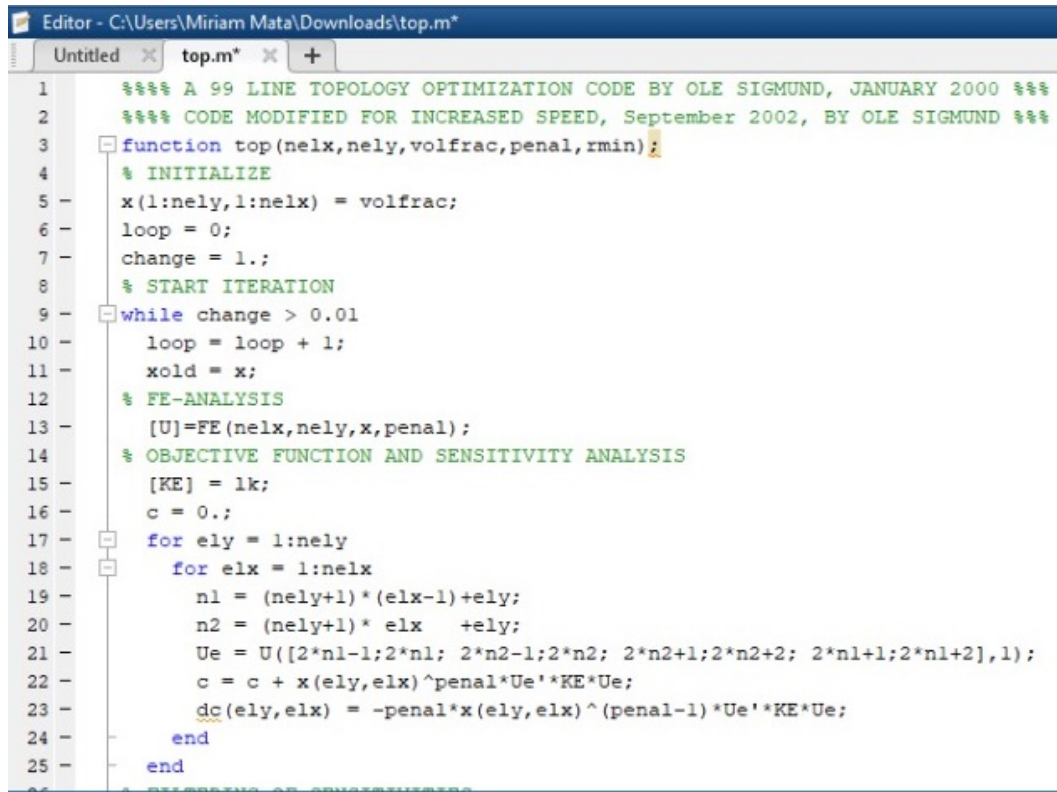
Hay que definir el espacio máximo de instalación disponible. De esta manera dEsto puede hacerse mediante CAD o un sistema de elementos finitos. Es necesario establecer otros objetivos de optimización, como: de baja masa o alta rigidez.

El software utiliza los datos disponibles para determinar uno o más Propuestas de diseño que sirven de plantilla para el concepto de diseño del diseñador. está disponible para su uso.

Estaremos encantados de asesorarle sobre cómo puede mejorar y, sobre todo, acelerar su proceso de desarrollo de productos con la ayuda de la optimización topológica.

2.3. Procedimiento de la programación

A continuación, veremos la codificación de la programación en MATLAB.



```
Editor - C:\Users\Miriam Mata\Downloads\top.m*
Untitled x top.m* x +
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 % ENDING OF SUBROUTINE
```

Figura 1: Configuraciones iniciales y análisis de sensibilidad.

```

Editor - C:\Users\Miriam Mata\Downloads\top.m*
Untitled x top.m* x +
25 - end
26 % FILTERING OF SENSITIVITIES
27 [dc] = check(nelx,nely,xmin,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

```

Figura 2: Optimización e impresión de resultados.

```

Editor - C:\Users\Miriam Mata\Downloads\top.m*
Untitled x top.m* x +
49 - end
50 %***** MESH-INDEPENDENCY FILTER *****
51 function [dcn]=check(nelx,nely,xmin,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(xmin),1):min(i+floor(xmin),nelx)
57             for l = max(j-floor(xmin),1):min(j+floor(xmin),nely)
58                 fac = xmin-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
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         nl = (nely+1)*(elx-1)+ely;

```

Figura 3: Filtro de independencia - Mesh.

```

Editor - C:\Users\Miriam Mata\Downloads\top.m*
Untitled x top.m* x +
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*(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)

```

Command Window

Figura 4: Matriz de rigidez del elemento.

2.4. Implementación de la programación

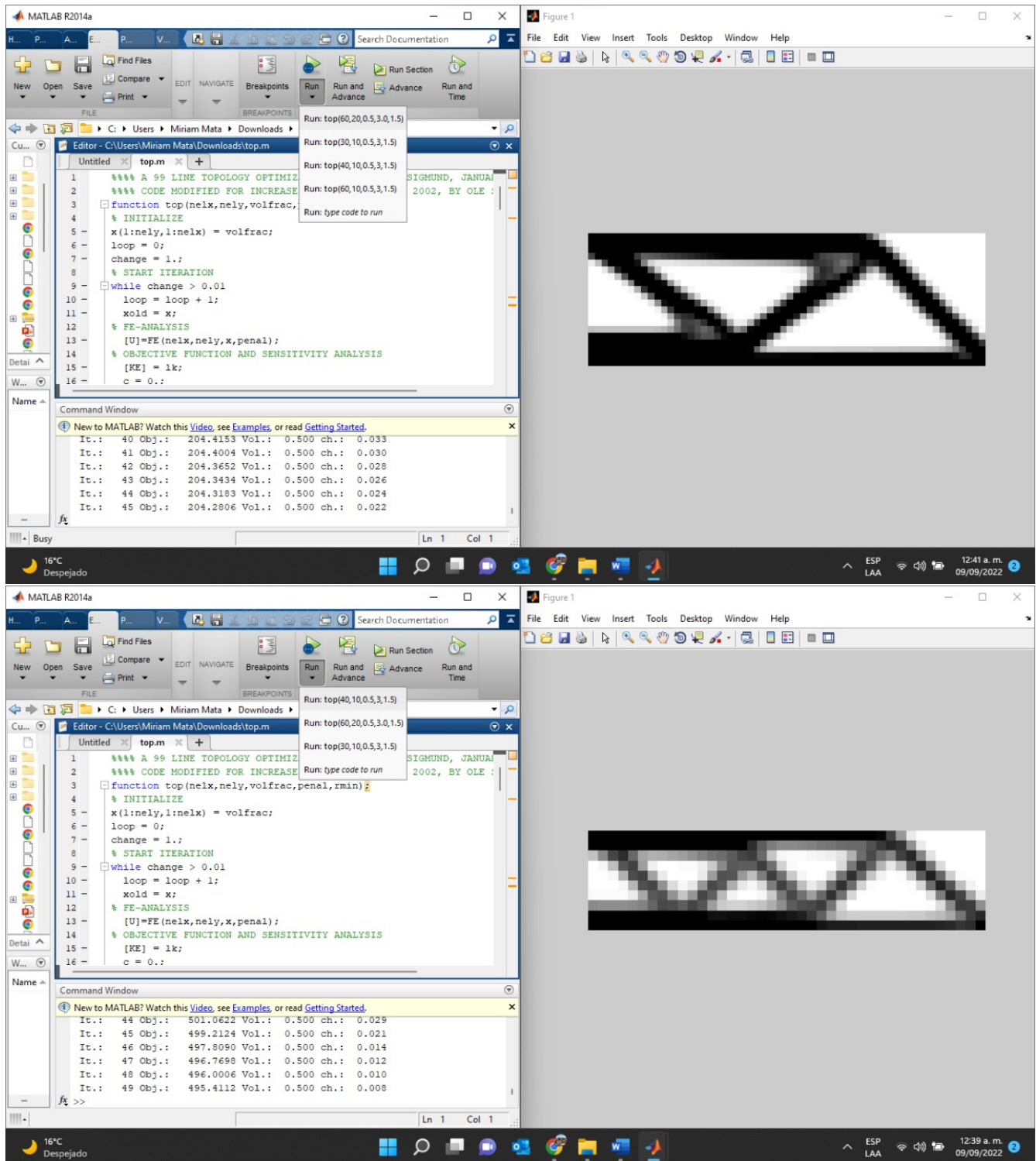


Figura 5: Resultados de la Optimización topológica 1.

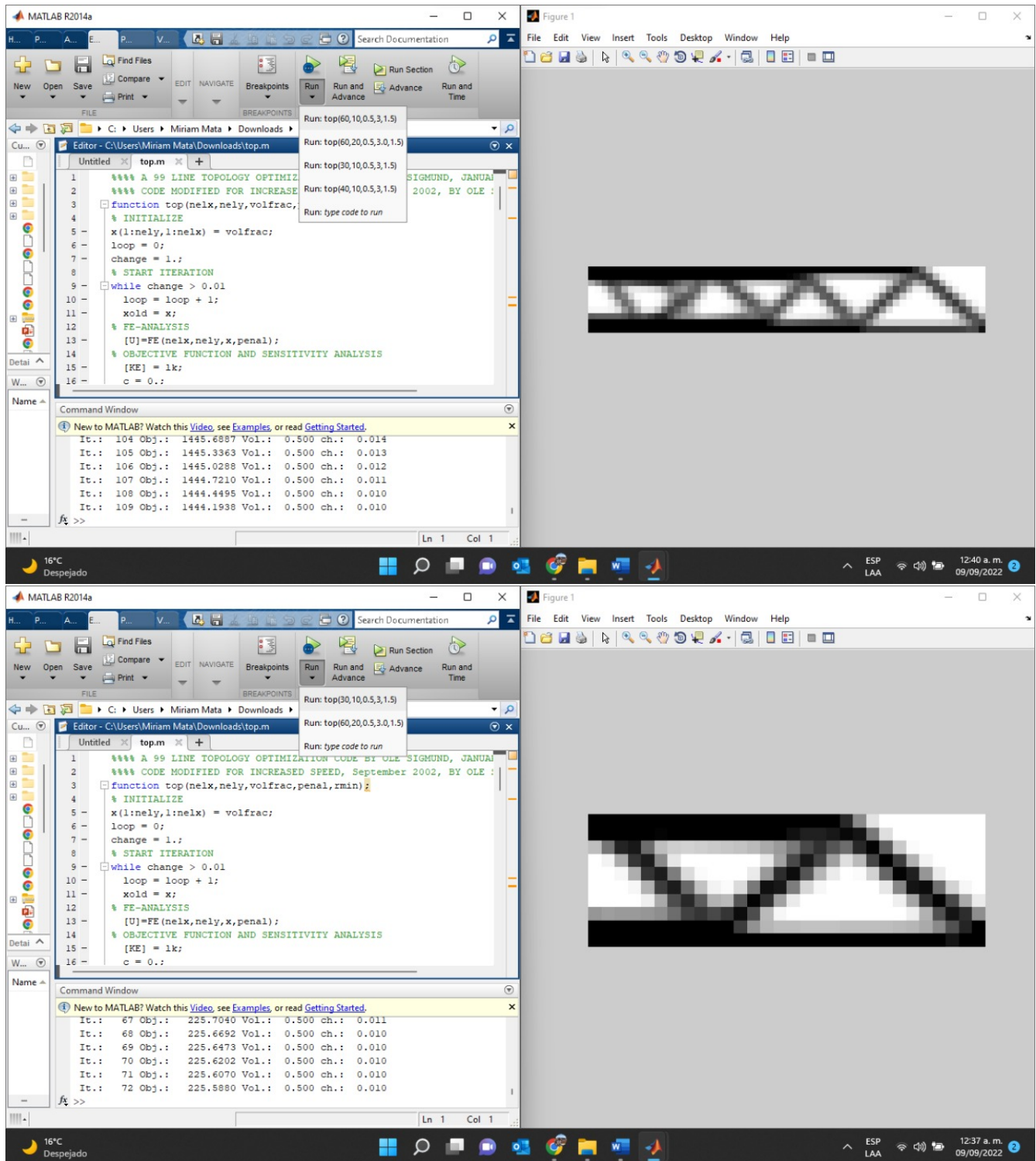


Figura 6: Resultados de la Optimización topológica 2.

3. Conclusiones

3.1. Jorge Fuentes

Trabajar nuevamente con MATLAB fue una sorpresa para mí, no recordaba mucho de como utilizar dicha herramienta de simulación de matrices por lo que me costó un poco entender sobre que estaba haciendo exactamente el código utilizado en nuestra práctica. Una vez entendido fue fácil modificar las variables para ver los resultados que mostraba la optimización de la topología de los elementos que estábamos usando. La ejecución del análisis así como se hace el código en MATLAB fueron de gran ayuda para repasar la programación, ya que, como sabemos la optimización topológica va de la mano con la biomecánica al momento de optimizar los elementos a un grado óptimo y funcional.

3.2. Tania Hernandez

La practica se enfoco en la programacion de matlab, como forma del proceso que con lleva el programar en el mencionado programa, especificamente de un codigo extenso, asi como tambien en el estado del arte de la pieza elaborada, ya que todo lo investigado, nos fue util para la realizacion de la pieza.

3.3. Anahi Herrera

Gracias a la elaboración de esta práctica se entendió de una mejor manera el funcionamiento de MATLAB aplicado hacia la materia de biomecánica, se realizaron investigaciones grupales llevando esta teoría recopilada a la práctica de manera exitosa.

3.4. Gustavo Díaz

Durante el desarrollo de esta practica vimos el funcionamiento de cierto programa dentro del software de Matlab en el cual se nos pidió hacer un modelado de una viga dentro de la optimización topológica y así poder hacer el diseño de la pieza de manera rápida y eficiente esta simulación se hace con la intención de optimizar la menor cantidad de material o tiempo requerido para realizarla pero sin deformarse, hacerlo en Matlab ayuda dado que puedes hacer el diseño de la pieza y ver cada uno de los análisis de las piezas.

3.5. Miriam Mata

Esta práctica estuvo muy interesante porque a pesar de ya haber de usar Matlab en semestres anteriores en lo personal nunca había usado un código como este, fue un poco complicado, pero se logró, siento que a nosotros como futuros ingenieros en mecatrónica se nos debió de haber inculcado más la programación desde los primeros semestres, ya que si bien la mayoría se va a el área de diseño en cierta parte seguimos viendo programación, pero esta actividad me ayudo para poder refrescar un poco mis conocimientos en Matlab porque en sí es un compilador muy sencillo de usar también depende mucho de la librería que te sepas por qué va cambiando cada año.

3.6. Alejandro Ramos

En esta práctica lo que vimos más que nada fue la teoría, y la programación esto nos viene muy bien para ir agarrando las bases o los fundamentos para que en las siguientes prácticas ya las realicemos de la mejor manera y así obtener el mejor resultado posible.

Referencias

- [1] V. J. Challis. A discrete level-set topology optimization code written in matlab., December 2010.
- [2] Yamada T. Izui K. Nishiwaki S. Otomori, M. Matlab code for a level set-based topology optimization method using a reaction diffusion equation., March 2015.
- [3] K. Suresh. A 199-line matlab code for pareto-optimal tracing in topology optimization., June 2009.