

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Laboratorio Biomecánica

Practica #1

Nombre	Matricula
Luis García González	1604958
Adan Asis Briones Torres	1732258
Sergio Esteban Cantú Carrasco	1863714
Oscar Marcelo Fragoso Martínez	1894650
Alfredo Cárdenas Mena	1902495

Instructor: Dra. Yadira Moreno Vera

Brigada: 109

No. De equipo: #3

Semestre: Agosto-Diciembre 2022

Hora: Lunes N5

Fecha: 19/09/2022

**Lugar: Ciudad Universitaria, San Nicolas de los Garza, N.L,
México**

INTRODUCCION

El Método de optimización de topología (MOT) es una técnica computacional que permite el diseño de una estructura óptima, utilizando solo la fracción de volumen del dominio de diseño total, sujeta a ciertas condiciones de contorno y cargas, dentro de las cuales la cantidad de material se distribuye de manera óptima. Este método maximiza o minimiza iterativamente la función objetivo al combinar técnicas de optimización lineal con métodos de elementos finitos (FEM) [2]. La optimización topológica es una técnica incluida en el campo del análisis estructural. Se basa en el análisis mecánico de componentes o estructuras. Su objetivo principal es reducir el peso estructural manteniendo la función mecánica de la pieza de destino. A diferencia de otros tipos de optimización, la optimización de topología ofrece un nuevo concepto de diseño estructural que se enfoca en aquellas aplicaciones donde el peso de los componentes es crítico (por ejemplo, la industria aeroespacial). Con la ayuda de nuevos métodos de cálculo, la optimización puede avanzar a un nivel de análisis más complejo a nivel estático, dinámico, plástico, modal o de impacto, todos los cuales pueden considerarse en el proceso de optimización. Debido a las enormes posibilidades en cuanto a diseños (geometrías muy complejas), el desarrollo de este método tiene amplias áreas de aplicación para técnicas de fabricación aditiva como la fabricación SLM (selective laser melting).

APLICACIONES

La industria automotriz resolvió rápidamente el problema gracias a la reducción de costos a través del ahorro de materia prima asociado con los tamaños de serie. De hecho, en el caso de producir millones de coches, unos gramos menos por coche suponen un ahorro de toneladas de material. Como ejemplo tenemos el chasis impreso en 3D del Light Rider, que pesa tan solo 6kg debido a la óptima distribución de los materiales. Más recientemente, los componentes de suspensión de Fiat Chrysler Automóviles combinaron más de 12 componentes diferentes en uno. Al centrarse en la optimización de la topología, los diseñadores redujeron el peso final en un 36 por ciento. Al centrarse en la optimización de la topología, los diseñadores redujeron el peso final en un 36 por ciento. La aviación es sin duda otra área de interés en la optimización topológica con el objetivo de reducir los costes generales. Los aviones más ligeros consumen menos combustible, lo que ahorrará mucho combustible a las aerolíneas a largo plazo. Eso es lo que muestra el diseñador Andreas Bastián con sus asientos de avión. Combinado, el diseño 54% más liviano representa una reducción significativa en el peso de la aeronave. Además del peso, la optimización topológica permite, especialmente en

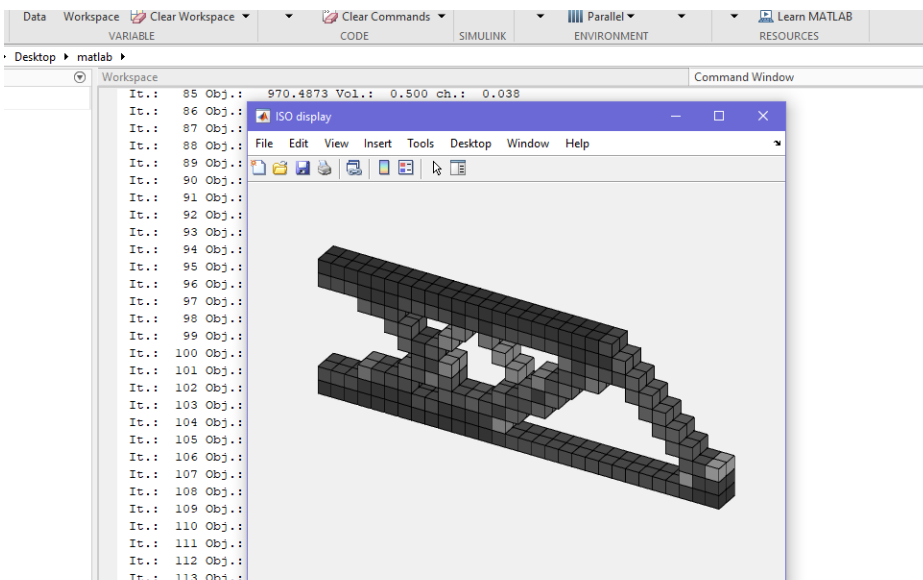
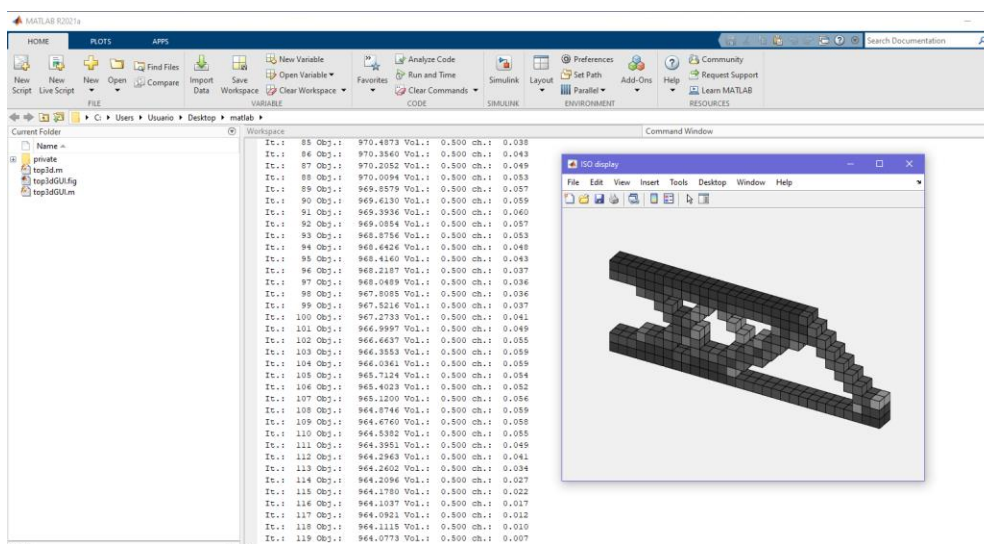
aeronáutica, imaginar formas más complejas a medida que la industria escapa a las limitaciones impuestas por los moldes. Finalmente, la comunidad médica también está investigando este método de diseño, especialmente para la fabricación de implantes personalizados. La optimización de la topología puede simular la densidad y la rigidez de los huesos al tiempo que reduce su peso total. De hecho, muchos implantes contienen estructuras reticulares y siguen siendo tan fuertes como los implantes de diseño convencional, o incluso más fuertes para algunos.

ESTADO DEL ARTE

Sigmund, O. Un código de optimización de topología de 99 líneas escrito en Matlab. En este documento educativo, el autor demuestra una implementación de Matlab de un código de optimización de topología para minimizar el ajuste de estructuras cargadas estáticamente. Este código consta de 99 líneas que se dividen en 36 líneas para el programa principal, 12 líneas para el optimizador basado en criterios de optimización, 16 líneas para el filtro dependiente de la cuadrícula y 35 líneas para el término de código de elemento orgánico. En la literatura, se pueden encontrar muchos enfoques para resolver problemas de optimización de topología. En el artículo original de Bendsøe y Kikuchi (1988), se utilizó un enfoque microestructural o basado en la homogeneización basado en estudios de persistencia de la solución. El enfoque basado en el consenso ha sido Se ha aplicado en muchos trabajos, pero tiene el inconveniente de que la determinación y evaluación de las microestructuras óptimas y sus orientaciones son complejas, incluso no resueltas (para problemas no solucionables), y además, las estructuras resultantes no se pueden construir, ya que no hay una escala de longitud definida asociada con las microestructuras. Sin embargo, el enfoque uniforme para la optimización de la topología sigue siendo importante porque puede proporcionar límites en el rendimiento teórico de las estructuras. Abad R. Matlab implementación de un procedimiento de optimización topológica para estructuras cargadas térmicamente. En este trabajo, el autor ha realizado un estudio sobre las técnicas de optimización topológica utilizadas en el diseño de estructuras. Este estudio se centró en estructuras sujetas a cargas térmicas uniformes, por lo que buscábamos una topología que proporcionara una conducción térmica óptima. El autor implementa el problema en un fragmento de código del programa Matlab, para resolver las estructuras, y realiza un análisis en profundidad de los resultados obtenidos por este programa para diferentes condiciones. Mencionó que

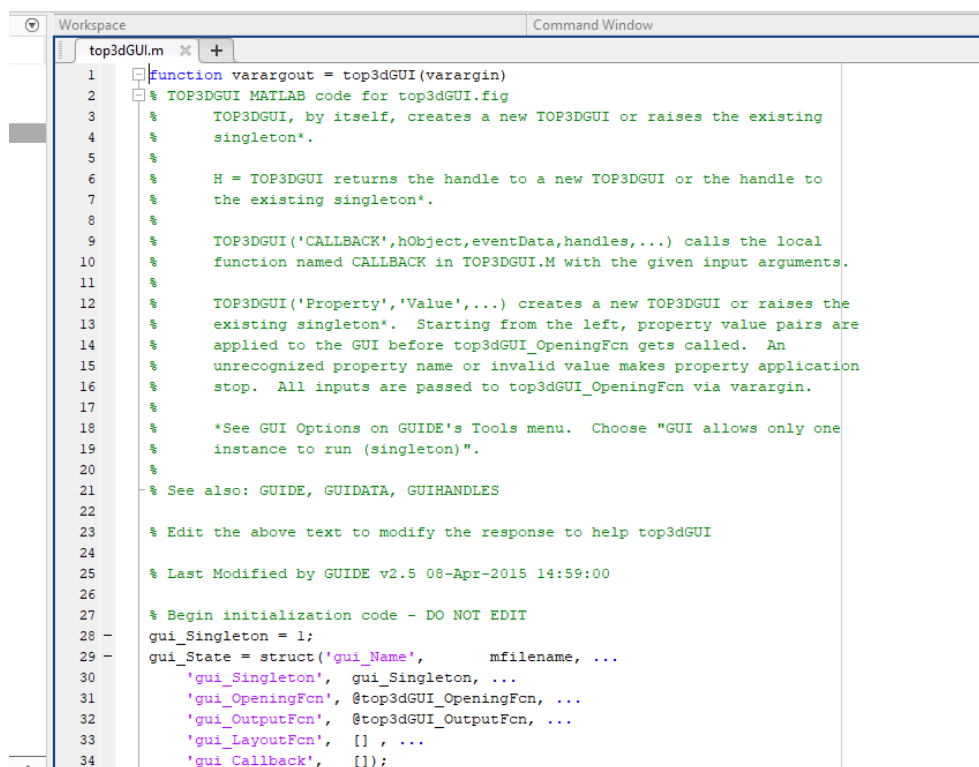
el método utilizado para el cálculo se basa en el método de elementos finitos y utiliza el criterio de optimización para optimizar la conducción térmica.

Procedimiento de la programación



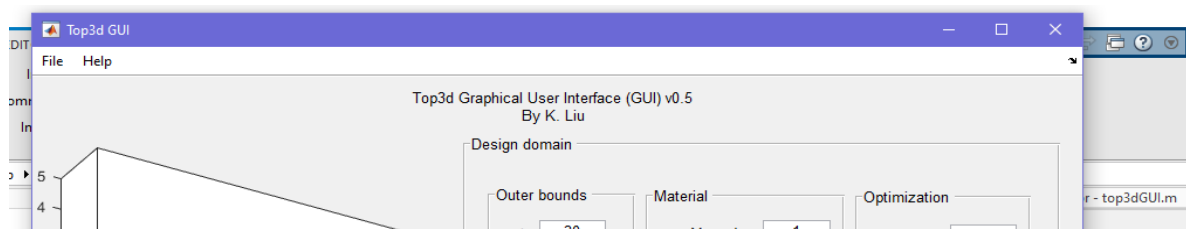
top3d(30, 10, 2, 0.5, 3.0, 1.2)

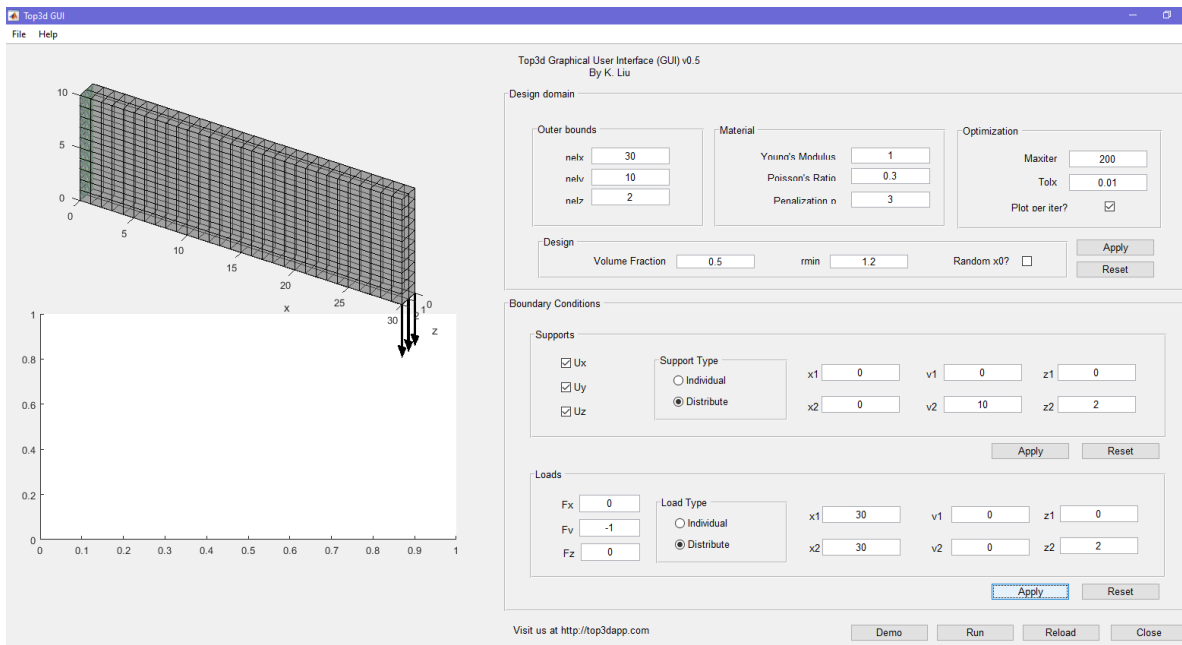
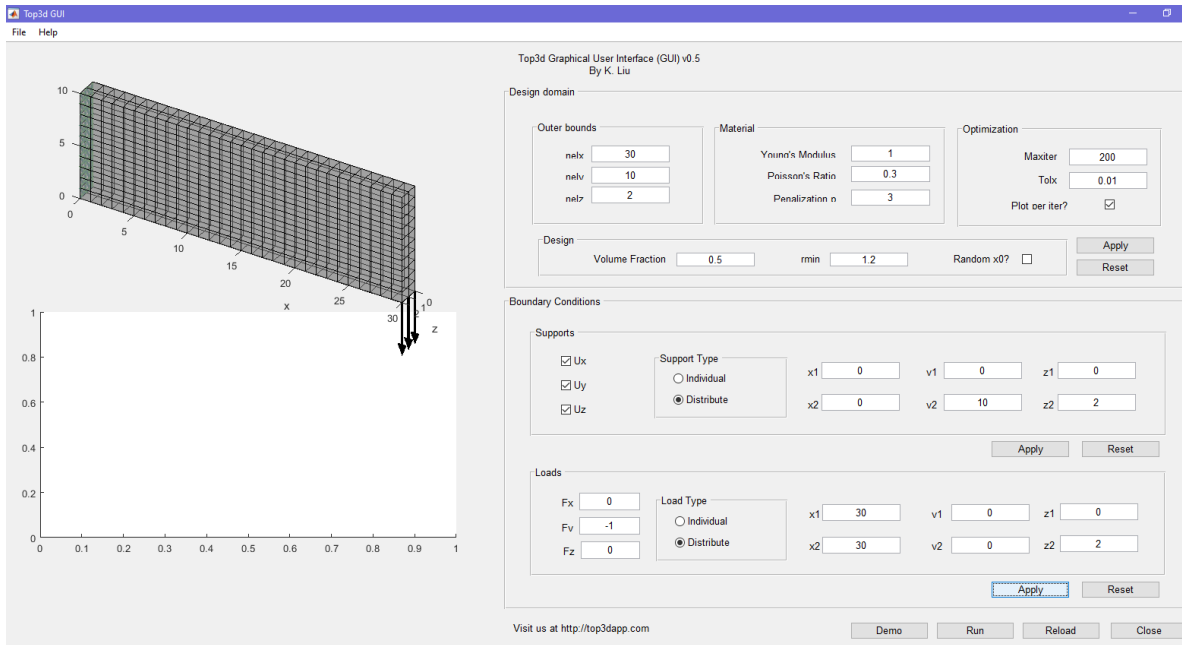
Se está resolviendo el problema de la viga en voladizo predeterminada en el programa como se muestra abajo a la izquierda y el resultado de la optimización topológica debería ser similar al de abajo a la derecha, todo gracias a la línea de promoción.

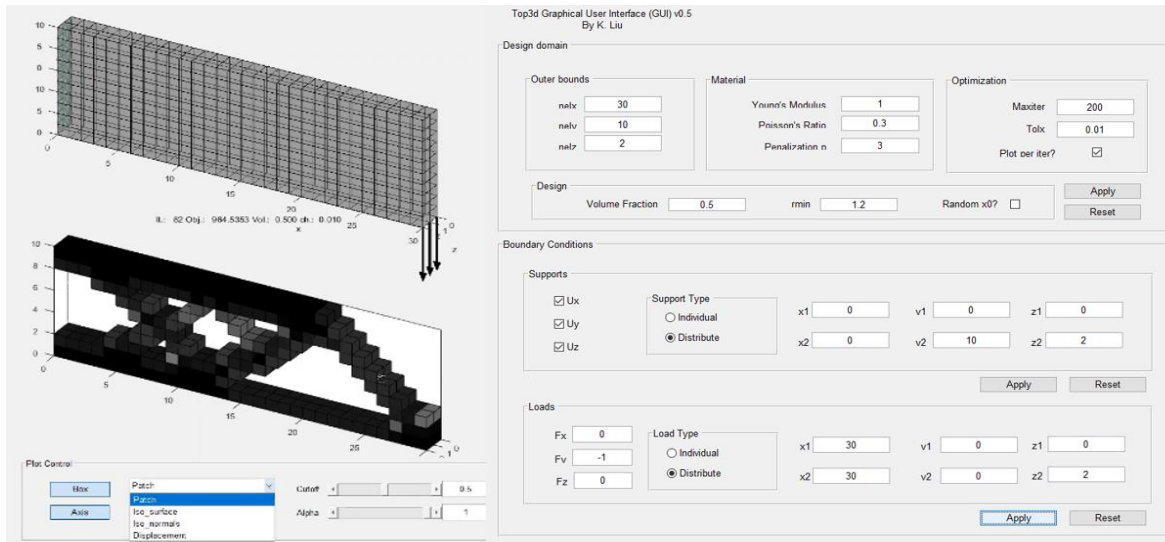
A screenshot of a MATLAB script editor showing the code for 'top3dGUI.m'. The script is a function that takes variable arguments and returns a handle to a GUI. It includes comments explaining the function's purpose, usage, and initialization code. The code is as follows:

```
1 function varargout = top3dGUI(varargin)
2 % TOP3DGUI MATLAB code for top3dGUI.fig
3 % TOP3DGUI, by itself, creates a new TOP3DGUI or raises the existing
4 % singleton*.
5 %
6 % H = TOP3DGUI returns the handle to a new TOP3DGUI or the handle to
7 % the existing singleton*.
8 %
9 % TOP3DGUI('CALLBACK',hObject,eventData,handles,...) calls the local
10 % function named CALLBACK in TOP3DGUI.M with the given input arguments.
11 %
12 % TOP3DGUI('Property','Value',...) creates a new TOP3DGUI or raises the
13 % existing singleton*. Starting from the left, property value pairs are
14 % applied to the GUI before top3dGUI_OpeningFcn gets called. An
15 % unrecognized property name or invalid value makes property application
16 % stop. All inputs are passed to top3dGUI_OpeningFcn via varargin.
17 %
18 % *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
19 % instance to run (singleton)".
20 %
21 % See also: GUIDE, GUIDATA, GUIHANDLES
22
23 % Edit the above text to modify the response to help top3dGUI
24
25 % Last Modified by GUIDE v2.5 08-Apr-2015 14:59:00
26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name',       mfilename, ...
30 'gui_Singleton',   gui_Singleton, ...
31 'gui_OpeningFcn', @top3dGUI_OpeningFcn, ...
32 'gui_OutputFcn',  @top3dGUI_OutputFcn, ...
33 'gui_LayoutFcn',  [], ...
34 'gui_Callback',   []);
```

Corremos el comando Gui para que nos muestre la interfaz geométrica del programa





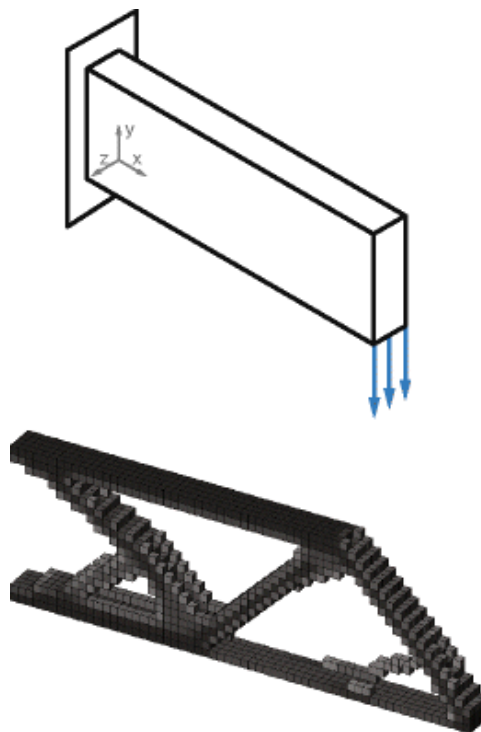


```

1 % AN 169 LINE 3D TOPOLOGY OPTIMIZATION CODE BY LIU AND TOVAR (JUL 2013)
2 function top3d(nelx,nely,nelz,volfrac,penal,rmin)
3 % USER-DEFINED LOOP PARAMETERS
4 maxloop = 200; % Maximum number of iterations
5 tolx = 0.01; % Termination criterion
6 displayflag = 0; % Display structure flag
7 % USER-DEFINED MATERIAL PROPERTIES
8 E0 = 1; % Young's modulus of solid material
9 Emin = 1e-9; % Young's modulus of void-like material
10 nu = 0.3; % Poisson's ratio
11 % USER-DEFINED LOAD DOFS
12 [il,jl,kl] = meshgrid(nelx, 0, 0:nelz); % Coordinates
13 loadnid = kl*(nelx+1)*(nely+1)+il*(nely+1)+(nely+1-jl); % Node IDs
14 loaddof = 3*loadnid(:) - 1; % DOFs
15 % USER-DEFINED SUPPORT FIXED DOFS
16 [iif,jf,kf] = meshgrid(0,0:nely,0:nelz); % Coordinates
17 fixednid = kf*(nelx+1)*(nely+1)+iif*(nely+1)+(nely+1-jf); % Node IDs
18 fixeddof = [3*fixednid(:); 3*fixednid(:)-1; 3*fixednid(:)-2]; % DOFs
19 % PREPARE FINITE ELEMENT ANALYSIS
20 nele = nelx*nely*nelz;
21 ndof = 3*(nelx+1)*(nely+1)*(nelz+1);
22 F = sparse(loaddof,1,-1,ndof,1);
23 U = zeros(ndof,1);
24 freedofs = setdiff(1:ndof,fixeddof);
25 KE = 1k_H8(nu);
26 nodegrd = reshape(1:(nely+1)*(nelx+1),nely+1,nelx+1);
27 nodeids = reshape(nodegrd(1:end-1,1:end-1),nely*nelx,1);
28 nodeidz = 0:(nely+1)*(nelx+1):(nelz-1)*(nely+1)*(nelx+1);
29 nodeids = repmat(nodeids,size(nodeidz))+repmat(nodeidz,size(nodeids));
30 edofVec = 3*nodeids(:)+1;
31 edofMat = repmat(edofVec,1,24)+ ...
32 repmat([0 1 2 3*nely + [3 4 5 0 1 2] -3 -2 -1 ...
33 3*(nely+1)*(nelx+1)+[0 1 2 3*nely + [3 4 5 0 1 2] -3 -2 -1]],nele,1);
34 iK = reshape(kron(edofMat,ones(24,1))',24*24*nele,1);
35 kK = reshape(kron(edofMat,ones(1,24))+24*24*nele,1);

```

```
Workspace Command Window Editor - top3dGUI.m
top3dGUI.m
57
58 % Welcom message
59 fprintf(1, ' Top3d Graphical User Interface (GUI) v0.5\n');
60 fprintf(1, ' An efficient 3D Topology Optimization program\n');
61 fprintf(1, ' written by K. Liu and A. Tovar\n');
62 fprintf(1, ' Indiana Univ-Purdue Univ Indianapolis, U.S.A\n');
63 fprintf(1, ' visit us at <a href="http://www.top3dapp.com">Top3dAPP.com</a>\n');
64 % Set unit
65 % set(hObject,'Unit','normalized')
66 % Warning
67 warning('off', 'MATLAB:hg:patch:CannotUseFaceVertexCDataOfSize0');
68 % Apply default values
69 set_DesignPanel(handles);
70 get_DesignPanel(hObject, eventdata, handles, 'initial');
71 handles = guidata(hObject);
72 % - BC
73 get_BCPanel(hObject, eventdata, handles, 'initial');
74 handles = guidata(hObject);
75 % - LC
76 get_LCPanel(hObject, eventdata, handles, 'initial');
77 handles = guidata(hObject);
78 % Initial domains
79 x = 0.5*ones(5,10,2);
80 axes(handles.axes_design);
81 plot_3d(x, 1, 0.5);
82 % Plot arrow
83 sp = [0 0 5];
84 ep1 = sp + [2 0 0];
85 ep2 = sp - [0 0 2];
86 ep3 = sp + [0 2 0];
87 ar = plot_arrow(sp, ep1);
88 plot_arrow(ar,'Width',2);
89 ar = plot_arrow(sp, ep2);
90 plot_arrow(ar,'Width',2);
91 ar = plot_arrow(sp, ep3);
92 plot_arrow(ar,'Width',2);
```



La configuración de los datos fue dada gracias a al top3d utilizado
Y configuramos los datos según el top3d anteriormente usado

Implementación o desarrollo de la programación en sus diferentes vistas

EXPLICACION GENERAL SOBRE EL CODIGO

Este código, denominado programa de 177 líneas, es el sucesor del programa de 99 líneas de Sigmund (2001) que hereda y amplifica los mismos inconvenientes. Nuestro artículo presenta un programa de 169 líneas denominado top3d que incorpora estrategias eficientes para la optimización topológica tridimensional. Este programa se puede usar de manera efectiva en computadoras personales para generar estructuras de tamaño considerable. Este documento explica el uso de top3d en problemas de optimización de topología de conducción de calor, mecanismo de cumplimiento y cumplimiento mínimo.

Por defecto, el código resuelve un problema de cumplimiento mínimo para la viga en voladizo. El dominio de diseño prismático está completamente restringido en un extremo y se aplica una carga vertical unitaria distribuida hacia abajo en el borde libre inferior. La Figura muestra los resultados de optimización de topología para resolver el problema de cumplimiento mínimo con las siguientes líneas de entrada de MATLAB:

```
top3d(60,20,4,0.3,3,1.5)
```

En primer lugar, cambiar las condiciones de carga

```
12 il = nelx/2; jl = 0 ; kl = nelz/2;  
13 loadnid = kl*(nelx+1)*(nely+1)+il*(nely  
+1)+(nely+1-jl);  
14 loaddof = 3*loadnid(:) - 1;
```

En segundo lugar, definir las condiciones de contorno correspondientes

```
16 iif = [0 0 nelx nelx]; jf = [0 0 0 0];  
kf = [0 nelz 0 nelz];  
17 fixednid = kf*(nelx+1)*(nely+1)+iif*(  
nely+1)+(nely+1-jf);  
18 fixeddof = [3*fixednid(:); 3*fixednid(:)  
-1; 3*fixednid(:)-2];
```

entonces el problema puede ser promovido por línea:

```
top3d(40,20,40,0.2,3.0,1.5)
```

Apéndice C: Programa MATLAB top3d

```

1  % A 169 LINE 3D TOPOLOGY OPTIMIZATION CODE BY LIU AND TOVAR (JUL 2013)
2  function top3d(nelx,nely,nelz,volfrac,penal,rmin)
3  % USER-DEFINED LOOP PARAMETERS
4  maxloop = 200; % Maximum number of iterations
5  tolx = 0.01; % Termination criterion
6  displayflag = 1; % Display structure flag
7  % USER-DEFINED MATERIAL PROPERTIES
8  E0 = 1; % Young's modulus of solid material
9  Emin = 1e-9; % Young's modulus of void-like material
10 nu = 0.3; % Poisson's ratio
11 % USER-DEFINED LOAD DOFs
12 il = nelx; jl = 0; kl = 0:nelz; % Coordinates
13 loadnid = kl*(nelx+1)*(nely+1)+il*(nely+1)+(nely+1-jl); % Node IDs
14 loadidof = 3*loadnid(:) - 1; % DOFs
15 % USER-DEFINED SUPPORT FIXED DOFs
16 [jf,kf] = meshgrid(1:nely+1,1:nelz+1); % Coordinates
17 fixednid = (kf-1)*(nely+1)*(nelx+1)+jf; % Node IDs
18 fixeddof = [3*fixednid(:); 3*fixednid(:)-1; 3*fixednid(:)-2]; % DOFs
19 % PREPARE FINITE ELEMENT ANALYSIS
20 nele = nelx*nely*nelz;
21 ndof = 3*(nelx+1)*(nely+1)*(nelz+1);
22 F = sparse(loadidof,1,-1,ndof,1);
23 U = zeros(ndof,1);
24 freedofs = setdiff(1:ndof,fixeddof);
25 KE = lk.H8(nu);
26 nodegrd = reshape(1:(nely+1)*(nelx+1),nely+1,nelx+1);
27 nodeids = reshape(nodegrd(1:end-1,1:end-1),nely*nelx,1);
28 nodeidz = 0:(nely+1)*(nelx+1):(nelz-1)*(nely+1)*(nelx+1);
29 nodeids = repmat(nodeids,size(nodeidz))+repmat(nodeidz,size(nodeids));
30 edofVec = 3*nodeids(:)+1;
31 edofMat = repmat(edofVec,1,24)+...
32 repmat([0 1 2 3*nely + [3 4 5 0 1 2] -3 -2 -1 ...
33 3*(nely+1)*(nelx+1)+[0 1 2 3*nely + [3 4 5 0 1 2] -3 -2 -1]],nele,1);
34 iK = kron(edofMat,ones(24,1))';
35 jK = kron(edofMat,ones(1,24))';
36 % PREPARE FILTER
37 iH = ones(nele*(2*(ceil(rmin)-1)+1)^2,1);
38 jH = ones(size(iH));
39 sH = zeros(size(iH));
40 k = 0;
41 for kl = 1:nelz

```

```

42     for il = 1:nelx
43         for jl = 1:nely
44             e1 = (kl-1)*nelx*nely + (il-1)*nely+jl;
45             for k2 = max(kl-(ceil(rmin)-1),1):min(kl+(ceil(rmin)-1),nelz)
46                 for i2 = max(il-(ceil(rmin)-1),1):min(il+(ceil(rmin)-1),nelx)
47                     for j2 = max(jl-(ceil(rmin)-1),1):min(jl+(ceil(rmin)-1),nely)
48                         e2 = (k2-1)*nelx*nely + (i2-1)*nely+j2;
49                         k = k+1;
50                         iH(k) = e1;
51                         jH(k) = e2;
52                         sH(k) = max(0,rmin-sqrt((i1-i2)^2+(j1-j2)^2+(k1-k2)^2));
53                     end
54                 end
55             end
56         end
57     end
58 end
59 H = sparse(iH,jH,sH);
60 Hs = sum(H,2);
61 % INITIALIZE ITERATION
62 x = repmat(volfrac,[nely,nelx,nelz]);
63 xPhys = x;
64 loop = 0;
65 change = 1;
66 % START ITERATION
67 while change > tolx && loop < maxloop
68     loop = loop+1;
69     % FE-ANALYSIS
70     sK = KE(:)*(Emin+xPhys(:)'.^penal*(E0-Emin));
71     K = sparse(iK(:),jK(:),sK(:)); K = (K+K')/2;
72     U(freedofs,:) = K(freedofs,freedofs)\F(freedofs,:);
73     % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
74     ce = reshape(sum((U(edofMat)*KE).*U(edofMat)),2,[nely,nelx,nelz]);
75     c = sum(sum(sum((Emin+xPhys.^penal*(E0-Emin)).*ce)));
76     dc = -penal*(E0-Emin)*xPhys.^(penal-1).*ce;
77     dv = ones(nely,nelx,nelz);
78     % FILTERING AND MODIFICATION OF SENSITIVITIES
79     dc(:) = H*(dc(:)./Hs);
80     dv(:) = H*(dv(:)./Hs);
81     % OPTIMALITY CRITERIA UPDATE
82     l1 = 0; l2 = 1e9; move = 0.2;
83     while (l2-l1)/(l1+l2) > 1e-3

```

```

80 dv(:) = 1/(dv(:)./hs);
81 % OPTIMALITY CRITERIA UPDATE
82 l1 = 0; l2 = 1e9; move = 0.2;
83 while (l2-l1)/(l1+l2) > 1e-3
84     lmid = 0.5*(l2+l1);
85     xnew = max(0,max(x-move,min(1,min(x+move,x.*sqrt(-dc./dv/lmid)))));
86     xPhys(:) = (H*xnew(:))./Hs;
87     if sum(xPhys(:)) > volfrac*nele, l1 = lmid; else l2 = lmid; end
88 end
89 change = max(abs(xnew(:)-x(:)));
90 x = xnew;
91 % PRINT RESULTS
92 fprintf(' It.:%5i Obj.:%11.4f Vol.:%7.3f ch.:%7.3f\n',loop,c,mean(xPhys(:)),change);
93 % PLOT DENSITIES
94 if displayflag, clf; display_3D(xPhys); end
95 end
96 clf; display_3D(xPhys);
97 end
98 % ===== AUXILIARY FUNCTIONS =====
99 % GENERATE ELEMENT STIFFNESS MATRIX
100 function [KE] = lk.H8(nu)
101 A = [32 6 -8 6 -6 4 3 -6 -10 3 -3 -3 -4 -8;
102     -48 0 0 -24 24 0 0 0 12 -12 0 12 12 12];
103 k = 1/72*A'*[1; nu];
104 % GENERATE SIX SUB-MATRICES AND THEN GET KE MATRIX
105 K1 = [k(1) k(2) k(3) k(5) k(5);
106     k(2) k(1) k(2) k(4) k(6) k(7);
107     k(2) k(2) k(1) k(4) k(7) k(6);
108     k(3) k(4) k(4) k(1) k(8) k(8);
109     k(5) k(6) k(7) k(8) k(1) k(2);
110     k(5) k(7) k(6) k(8) k(2) k(1)];
111 K2 = [k(9) k(8) k(12) k(6) k(4) k(7);
112     k(8) k(9) k(12) k(5) k(3) k(5);
113     k(10) k(10) k(13) k(7) k(4) k(6);
114     k(6) k(5) k(11) k(9) k(2) k(10);
115     k(4) k(3) k(5) k(2) k(9) k(12);
116     k(11) k(4) k(6) k(12) k(10) k(13)];
117 K3 = [k(6) k(7) k(4) k(9) k(12) k(8);
118     k(7) k(6) k(4) k(10) k(13) k(10);
119     k(5) k(5) k(3) k(8) k(12) k(9);
120     k(9) k(10) k(2) k(6) k(11) k(5);
121     k(12) k(13) k(10) k(11) k(6) k(4);
122     k(2) k(12) k(9) k(4) k(5) k(3)];
123 K4 = [k(14) k(11) k(11) k(13) k(10) k(10);
124     k(11) k(14) k(11) k(12) k(9) k(8);
125     k(11) k(14) k(11) k(12) k(9) k(8);
126     k(13) k(12) k(12) k(14) k(7) k(7);
127     k(10) k(9) k(8) k(7) k(14) k(11);
128     k(10) k(8) k(9) k(7) k(11) k(14)];
129 K5 = [k(1) k(2) k(8) k(3) k(5) k(4);
130     k(2) k(1) k(8) k(4) k(6) k(11);
131     k(8) k(8) k(1) k(5) k(11) k(6);
132     k(3) k(4) k(5) k(1) k(8) k(2);
133     k(5) k(6) k(11) k(8) k(1) k(8);
134     k(4) k(11) k(6) k(2) k(8) k(1)];
135 K6 = [k(14) k(11) k(7) k(13) k(10) k(12);
136     k(11) k(14) k(7) k(12) k(9) k(2);
137     k(7) k(7) k(14) k(10) k(2) k(9);
138     k(13) k(12) k(10) k(14) k(7) k(11);
139     k(10) k(9) k(2) k(7) k(14) k(7);
140     k(12) k(2) k(9) k(11) k(7) k(14)];
141 KE = 1/((nu+1)*(1-2*nu))*...
142 [ K1 K2 K3 K4;
143   K2' K5 K6 K3';
144   K3' K6 K5' K2';
145   K4 K3 K2 K1'];
146 end
147 % DISPLAY 3D TOPOLOGY (ISO-VIEW)
148 function display_3D(rho)
149 [nely,nelx,nelz] = size(rho);
150 hx = 1; hy = 1; hz = 1; % User-defined unit element size
151 face = [1 2 3 4; 2 6 7 3; 4 3 7 8; 1 5 8 4; 1 2 6 5; 5 6 7 8];
152 set(gcf,'Name','ISO display','NumberTitle','off');
153 for k = 1:nelz
154     z = (k-1)*hz;
155     for i = 1:nelx
156         x = (i-1)*hx;
157         for j = 1:nely
158             y = nely*hy - (j-1)*hy;
159             if (rho(j,i,k) > 0.5) % User-defined display density threshold
160                 vert = [x y z; x y-hx z; x+hx y-hx z; x+hx y z; x y z+hx; x y-hx z+hx; x+hx y-hx z+hx; x+hx y z+hx];
161                 vert(:,2:3) = vert(:,3:2); vert(:,2,:) = -vert(:,2,:);
162                 patch('Faces',face,'Vertices',vert,'FaceColor',[0.2+0.8*(1-rho(j,i,k))
163                     ,0.2+0.8*(1-rho(j,i,k)),0.2+0.8*(1-rho(j,i,k))]);
163                 hold on;
164             end
165         end
166     end
167 end
168 axis equal; axis tight; axis off; box on; view([30 30]); pause(1e-6);

```

```

165         end
166     end
167 end
168 axis equal; axis tight; axis off; box on; view([30,30]); pause(1e-6);
169 end
170 %=====
171 % This code was written by K Liu and A Tovar, Dept. of Mechanical
172 % Engineering, Indiana University-Purdue University Indianapolis,
173 % Indiana, United States of America
174 %=====
175 % Please send your suggestions and comments to: kailiu@iupui.edu
176 %=====
177 % The code as well as step-by-step tutorials can be found from the
178 % website: http://top3dapp.com
179 %=====
180 %
181 % Disclaimer:
182 % The authors reserves all rights for the program.
183 % The code may be distributed and used for educational purposes.
184 % The authors do not guarantee that the code is free from errors, and
185 % they shall not be liable in any event caused by the use of the code.
186 %=====
187 %
188 %
189 %

```

Conclusiones

- **Luis García González 1604958**

Al momento de finalizar la elaboración de esta práctica, aprendemos sobre el uso de Matlab, que aun a pesar de que es un programa que se ha utilizado durante varias ocasiones a lo largo de a carrera universitaria hay momentos en los que aun se dificultan el uso de este programa, al hacer la practica comprendí como se puede usar para analizar la geometría de una pieza, dando como conclusión personal el que es necesario estudiar mas el programa y aprender las diversas formas de utilizarlo y sus aplicaciones a la industria

- **Alfredo Cárdenas Mena 1902495**

El uso de Matlab a mi parecer me sigue siendo complejo y difícil, no había tenido la oportunidad de utilizarlo en otras materias dentro de la carrera de FIME, pero hay mucho material para aprender y bases de donde agarrar, el proyecto del top3d es un programa ya elaborado y probado del cual se nos da la oportunidad tomar como referente hacia nuestras necesidades y procesos, el explicar su funcionamiento recreándolo en nuestros sistemas ayudan mucho a la comprensión de este mismo.

- **Adan Asis Briones Torres 1732258**

Este laboratorio funcional presenta una implementación muy simple de un algoritmo de optimización de topología, en este caso el código se implementa utilizando solo 99 líneas de entrada de Matlab. 3 El método de elementos finitos fue diseñado para su uso en computadoras y

permitió resolver ecuaciones diferenciales que involucraban un problema de física en geometrías complejas, en cuyo caso se llevó a cabo para optimizar la topología, con el fin de aligerar la estructura propuesta conservando las funciones mecánicas de . Finalmente, como podemos ver, todo se hace en Matlab, por lo que es importante darse cuenta de la importancia y utilidad de los diferentes softwares, ya que facilitan mucho nuestro trabajo y nos permiten hacer más precisos cálculos.

- **Sergio Esteban Cantú Carrasco 1863714**

Como conclusión opino que MATLAB se me hace un software fácil de utilizar a pesar de saber lo básico de él y de ser la segunda vez que trabajo con el, conformado por 99 líneas no fue complicado trabajar con eso; es importante aprender algo de este tipo de software que hacen el trabajo más fácil, rápido y preciso

- **Oscar Marcelo Fragoso Martínez 1894650**

Esta práctica me dejó aprendizaje acerca del software de MATLAB con el cual no había trabajado anteriormente, no me fue tan complicado entenderlo más sin embargo si tiene sus diferencias, aunque lo encuentro bastante útil porque sé que es un programa que tiene muchas herramientas

Referencias

- Meza, C. A., Tamayo, F., & Franco, E. E. (2015). Optimización topológica aplicada al diseño de componentes estructurales mecánicos de peso reducido. *El hombre y la máquina*, (46), 72-79.
- Ochoa Sánchez, C. A. (2018). Optimización topológica en estructuras de tres dimensiones usando elementos finitos en el campo elástico lineal. Departamento de Ingeniería Civil y Agrícola.
- 99 Line Topology Optimization Code – O. Sigmund, Department of Solid Mechanics, Building 404, Technical University of Denmark, DK-2800 Lyngby, Denmark.