

Laboratorio de Biomecánica

Práctica

Optimización de una prótesis de pie

EQUIPO 6

Christopher Alejandro Monsivais Ramos 1899218

Alessandra González Torres 1895846

Angela Itzel Rodríguez Torres 1925284

Abdiel Isaac Rodríguez Pinales 1842044

Martin Alejandro Villanueva Cortes 1870820

Alejandro Sebastian Marin Cota 1814201

1 de noviembre de 2022

1. 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).[1]

2. Definición de la forma geométrica.

Una prótesis se define como un dispositivo externo que sirve para reemplazar la falta parcial o total de un órgano o miembro con un objeto artificial, este busca cumplir las mismas funciones que la parte faltante. Desde los comienzos de la humanidad se han utilizado dispositivos artificiales para suplir estos miembros faltantes ya sean ocasionados por guerras, accidentes o incluso malformaciones de nacimiento. En el proceso de diseño de esta prótesis se decidió fijar la geometría y variar los materiales de los cuales se puede construir, la alternativa era fijar los materiales y el proceso de manufactura y variar la geometría.

3. Estado del arte

Objetivo: Dar a conocer cada una de las secciones que integran el código de optimización topológica de 99 líneas en Matlab, saber ejecutar el análisis del código y observar los resultados obtenidos.

Contenido: El artículo presenta una implementación compacta en Matlab de un código de optimización topológica para la minimización de la conformidad de estructuras cargadas estáticamente. El número total de líneas de entrada de Matlab es de 99 incluyendo el optimizador y la subrutina de elementos finitos. Las 99 líneas están divididas en 36 líneas para el programa principal, 12 líneas para el optimizador basado en criterios de optimización, 16 líneas para un filtro de dependencia de malla y 35 líneas para el código de elementos finitos. De hecho, excluyendo las líneas de comentario y las asociadas a la salida y a análisis de elementos finitos se ansevera que solo se necesitan 4u ineas de entrada, de Matlab para resolver un problema de optimización topológica bien planteado. Añadiendo tres líneas adicionales, el programa puede resolver problemas con múltiples casos de carga. El código está pensado para fines educativos.[2]

4. Procedimiento de la programación

Código 1

```

%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function topp(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;
dc(ely,elx)=0.;
for i=1:5
Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1; 2*n2+2;
2*n1+1;2*n1+2],i);
c = c + x(ely,elx)^penal*Ue'*KE*Ue;
dc(ely,elx) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)* Ue'*KE*Ue;
end
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))]);
% 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
%%%%%%%%%%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
%%%%%%%%%% 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),5); U =sparse(2*(nely+1)*(nelx+1),5);
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(3222,1) = -1;
F(3782,2) = -1;
F(2662,3) = -1;
F(2942,4) = -1;
F(3502,5) = -1;
fixeddofs = union([560:2*(nely+1):1260],[3920:2*(nely+1):4620]);
alldofs = [1:2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);
% SOLVING 127
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)];

```

Código 2

```

%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function topp(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;
dc(ely,elx)=0.;
for i=1:5
Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1; 2*n2+2;
2*n1+1;2*n1+2],i);
c = c + x(ely,elx)^penal*Ue'*KE*Ue;
dc(ely,elx) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
end
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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 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),5); U =sparse(2*(nely+1)*(nelx+1),5);
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(3222,1) = -1;
F(3782,2) = -1;
F(2662,3) = -1;
F(2942,4) = -1;
F(3502,5) = -1;
fixeddofs = [3920:2*(nely+1):4620];
alldofs = [1:2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);
% SOLVING 127
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)];

```

Código 3

```

%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function topp3(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;
dc(ely,elx)=0.;
for i=1:5
Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1; 2*n2+2;
2*n1+1;2*n1+2],i);
c = c + x(ely,elx)^penal*Ue'*KE*Ue;
dc(ely,elx) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
end
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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 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),5); U =sparse(2*(nely+1)*(nelx+1),5);
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(3222,1) = -1;
F(3782,2) = -1;
F(2662,3) = -1;
F(2942,4) = -1;
F(3502,5) = -1;
fixeddofs = [560:2*(nely+1):1260];
alldofs = [1:2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);

```

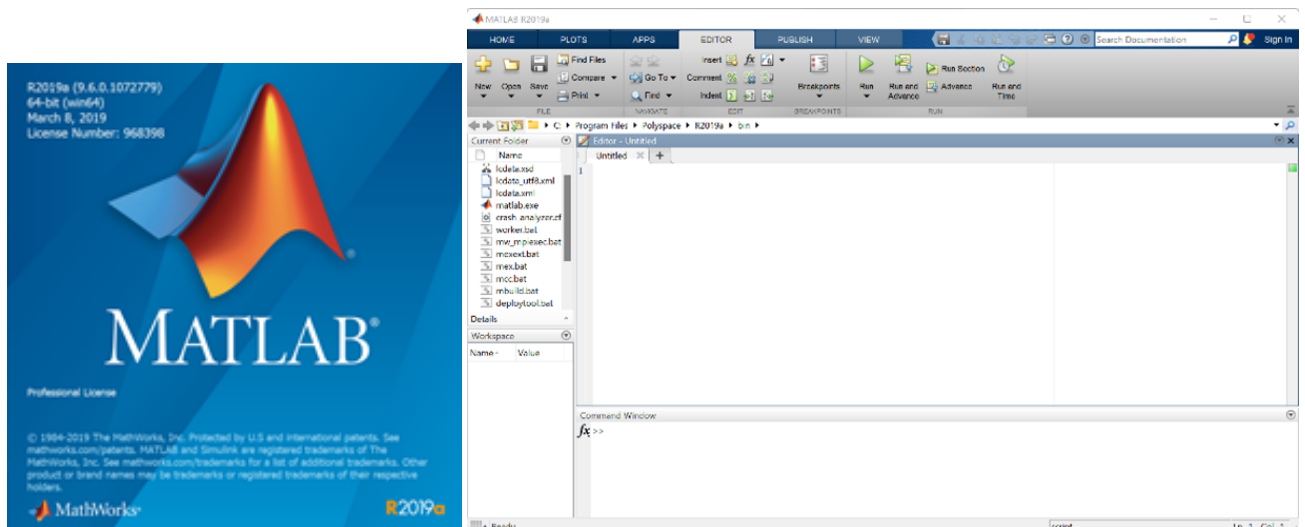
```

% SOLVING 127
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)];

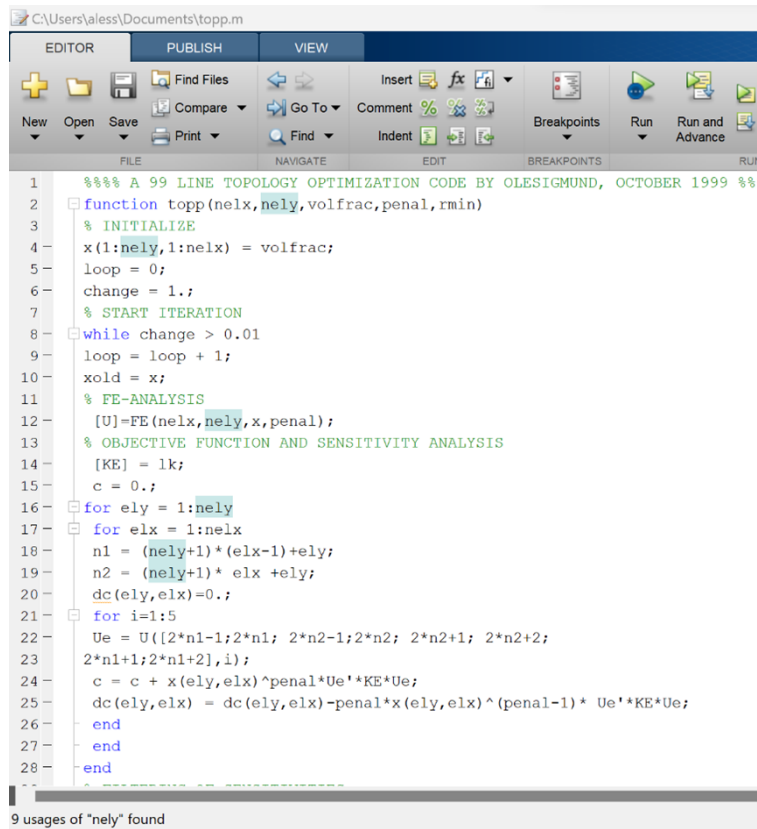
```

5. Pasos que seguir para la programación en MATLAB

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 abrir un archivo nuevo en el editor de texto, dentro del cual será necesario escribir el código proporcionado. (Los pasos se repetirán para cada código)



The screenshot shows the MATLAB Editor interface with the file 'topp.m' open. The code is a function for topology optimization. The interface includes a menu bar (FILE, NAVIGATE, EDIT, BREAKPOINTS, RUN) and a toolbar with icons for file operations, navigation, editing, and running. The code is as follows:

```
1  %%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
2  function topp(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 xold = x;
11 % FE-ANALYSIS
12 [U]=FE(nelx,nely,x,penal);
13 % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
14 [KE] = lk;
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 dc(ely,elx)=0.;
21 for i=1:5
22 Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1; 2*n2+2;
23 2*n1+1;2*n1+2],i);
24 c = c + x(ely,elx)^penal*Ue'*KE*Ue;
25 dc(ely,elx) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)* Ue'*KE*Ue;
26 end
27 end
28 end
```

At the bottom of the window, a status bar indicates "9 usages of 'nely' found".

Figura 1: Código 1

```

1  %%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
2  function topp2(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 xold = x;
11 % FE-ANALYSIS
12 [U]=FE(nelx,nely,x,penal);
13 % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
14 [KE] = lk;
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 dc(ely,elx)=0.;
21 for i=1:5
22 Ue = U([(2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1; 2*n2+2;
23 2*n1+1;2*n1+2;1),i]);
24 c = c + x(ely,elx)^penal*Ue'*KE*Ue;
25 dc(ely,elx) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
26 end
27 end
28 % FILTERING OF SENSITIVITIES
29

```

topp2 Ln 4 Col 28

Figura 2: Código 2

```

1  %%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
2  function topp3(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 xold = x;
11 % FE-ANALYSIS
12 [U]=FE(nelx,nely,x,penal);
13 % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
14 [KE] = lk;
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 dc(ely,elx)=0.;
21 for i=1:5
22 Ue = U([(2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1; 2*n2+2;
23 2*n1+1;2*n1+2;1),i]);
24 c = c + x(ely,elx)^penal*Ue'*KE*Ue;
25 dc(ely,elx) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
26 end
27 end
28 % FILTERING OF SENSITIVITIES
29

```

topp3 / FE Ln 90 Col 39

Figura 3: Código 3

3. El siguiente paso consiste en salvar el archivo. Se recomienda que el archivo se guarde en el directorio raíz de MATLAB que por default muestra. En caso de no ser así, debemos de navegar a “Mis Documentos MATLAB” y guardar ahí el script recién creado.

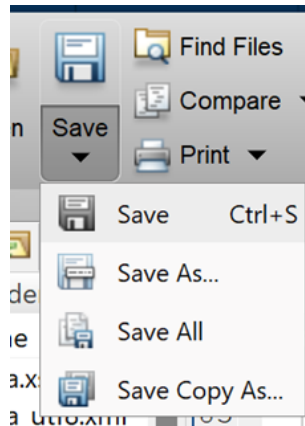


Figura 4: Como salvar un archivo

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 R2019a del software, el directorio actual del intérprete se encuentra en la barra de herramientas superior.
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, este caso en particular es evaluado y simulado cuando escribimos desde la línea de comando de MATLAB “`topp(72,34,0.33,3.0,1.5)`”.
 - 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 un radio de filtro para un filtro que hace que el diseño de malla-independiente.

6. Resultados de la optimización

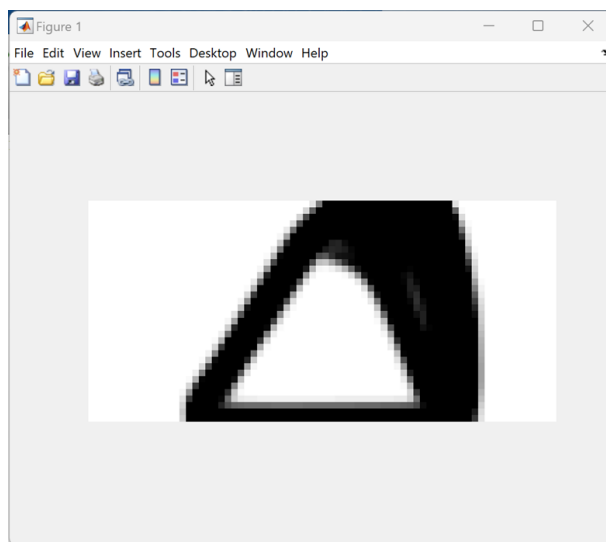


Figura 5: Resultados código 1

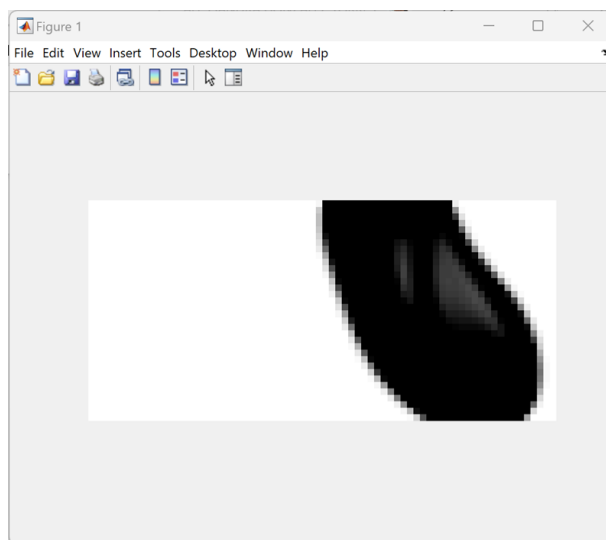


Figura 6: Resultados código 2

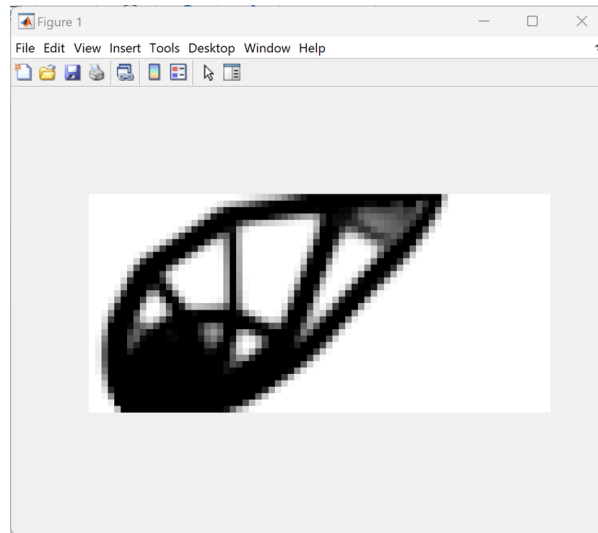


Figura 7: Resultados código 3

7. Conclusiones

Christopher Alejandro Monsivais Ramos:

En esta práctica reforzamos principalmente los conocimientos que se fueron adquiriendo a cerca de los códigos de optimización topologica, solo que esta vez se le realizaron distintos cambios para poder obtener como resultado la pieza final. Esto creo que es de mucha ayuda ya que nos mantiene en constante práctica del software de Matlab.

Alessandra Gonzalez Torres:

En esta práctica continuamos utilizando el código original que se vio en la primera práctica, solo que se tuvieron que hacer ciertas modificaciones para que se acoplara a lo pedido en la práctica actual. También se tuvieron que cambiar los valores que antes se estaban utilizando para poder crear las piezas finales, como se muestran en la última imagen.

Angela Itzel Rodriguez Torres:

En esta práctica utilizamos Matlab, nos damos cuenta de que los softwares de hoy en día nos apoyan mucho con cálculos e impresiones que nos facilitan el poder generar nuevas ideas e ir más rápido en nuestras investigaciones, así mismo generar el análisis de elemento finito para el diseño de una prótesis de pie y lo importante que puede ser para lugares de venta pueda ser una mejor producción debido a que es un mejor producto e innovador.

Abdiel Isaac Rodriguez Piñales:

Se concluye que los avances de tecnología han sido tales que se ha conseguido la optimización de los nuevos diseños. Como en el método de optimización topológica la cual sirve para reducir los materiales a utilizar los nuevos diseños. Esto ha sido útil en áreas como la ingeniería aeroespacial donde es crucial el volumen y materia de los componentes a utilizar. Todo es gracias al avance en el desarrollo de software.

Alejandro Sebastian Marin Cota:

Se concluyó que los avances tecnológicos permitieron la optimización de nuevos diseños. Al igual que las técnicas de optimización de topología, minimiza el uso de materiales rediseñados. Esto es útil en áreas como la ingeniería aeroespacial, donde el tamaño y el material de los componentes que se utilizarán es crítico. Todo gracias a los avances en el desarrollo de software.

Martín Alejandro Villanueva Cortes:

Tanto esta practica como las anteriores se ha seguido usando el mismo procedimiento para realizar las optimizaciones de cada una de las piezas que se nos han pedido lo cual nos permite obtener los resultados mostrados con solo cambiar unas partes del código en MatLab.

Referencias

- [1] Optimización topológica, Agosto 2017.
- [2] O. Sigmund. 99 line topology optimization code, 1999.