UNIVERSIDAD AUTÓNOMA

DE NUEVO LEÓN

FACULTAD DE INGENIRÍA

MECÁNICA Y ELÉCTRICA

Laboratorio de **B**iomecánica

Práctica 1

Catedrático: Yadira Moreno Vera

Martes V2

Brigada 214

Integrantes del Equipo #1	**Matricula**
Susana Rubio Medina	1798151
Melissa Lizeth Galindo Reyes	1856086
Oziel Alberto Torres Villarreal	1900260
Gabriel Eduardo Vázquez Ortega	1903060
José Manuel Reséndiz García	1907334
Miguel Ángel Martínez Villanueva	1934489

Semestre Agosto – Diciembre 2022

San Nicolás de los Garza, N.L.

06 de Septiembre de 2022

Índice

Optimización Topológica

El objetivo fundamental de la optimización topológica es encontrar la distribución de masa en una estructura que está sometida a condiciones de carga y restricción de frontera específicas, de tal forma que cumpla con una función objetivo que se debe optimizar.

La optimización topológica, es un método numérico usado para determinar la mejor distribución de material dentro de un volumen denominado dominio de diseño. La estructura dentro de este espacio soporta un determinado conjunto de cargas y condiciones de frontera, de tal manera que la distribución de material resultante cumple con las restricciones de diseño impuestas. La optimización topológica puede verse como un método para encontrar la forma óptima de distribuir cierta cantidad de material en una región para formar una estructura.

De entre todas las aplicaciones, una de las más potentes y depuradas es la solución "Inspire" de Altair, está permite a los ingenieros de diseño, crear e investigar conceptos estructuralmente eficientes, rápida y fácilmente. Las simulaciones estructurales tradicionales permiten a los ingenieros comprobar si un diseño soportará las cargas requeridas; lo que hace Inspire es mejorar este proceso generando un nuevo diseño mediante optimización topológica, siguiendo 3 pasos:

- Introducción de las cargas
- Optimización
- Validación del diseño

Los fabricantes de automatización necesitan innovar constantemente y la incorporación de nuevas geometrías más eficientes, nuevos procesos de fabricación y nuevos materiales los empuja a adoptar la optimización topológica como una de las mayores apuestas para la industria. Si hablamos del mundo de la competición de motor la importancia aún es mayor. Poniendo como ejemplo la Fórmula 1, la posibilidad de reducir de manera significativa el peso de un monoplaza y reducir el tiempo por vuelta, hace que todas las escuderías inviertan tiempo y dinero en las tecnologías que se lo permiten, fue así con los materiales compuestos y es así con la optimización topológica.

Está es una tecnología de reciente aplicación que se impondrá en los próximos años como método de diseño en determinados sectores y que será una competencia clave para los ingenieros de diseño.

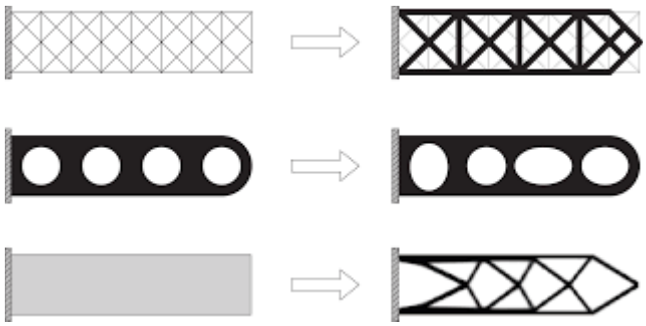


Figura 3 Ejemplo de Optimización Topológica
en el diseño de elementos estructurales.

Estado del Arte

La optimización topológica se concibió como una metodología avanzada de diseño estructural capaz de generar configuraciones innovadoras, ligeras y de alto rendimiento difíciles de obtener mediante procedimientos convencionales. Se trata de un método independiente de la configuración inicial, a diferencia de otros tipos de optimización como la optimización de forma o de tamaño. A nivel industrial, en el sector aeronáutico, uno de los ejemplos más famosos es la aplicación de optimización topológica en el diseño de las costillas del borde de atache del ala del Airbus (Figura 1) que permitió reducir hasta 500 kg en cada avión.

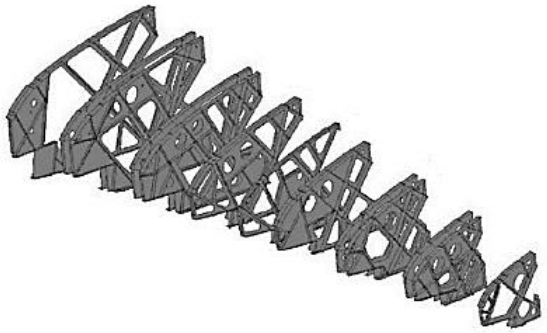


Figura 1. Costillas del borde de ataque de Airbus A360 obtenidas mediante la aplicación de optimización topológica

La cantidad de material y su distribución está limitada en el dominio del diseño: Ω, el dominio mencionado puede contener regiones de material o con vacíos que no pertenecen al análisis, y por tanto, se denominan regiones pasivas. A diferencia del dominio pasivo, el dominio activo debe ser optimizado bajo una función (objetivo).

Para la solución del problema de optimización se considera un dominio bidimensional y teoría de elasticidad lineal.

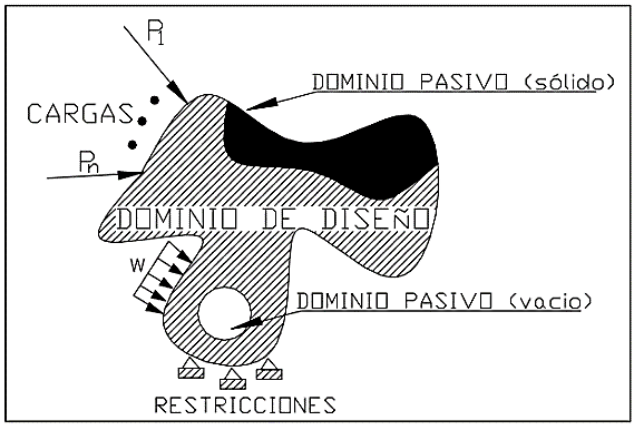


Figura 2. Dominio para el problema de Optimización Topológica.

Optimización de forma

Matemáticamente, la optimización de forma puede plantearse como el problema de encontrar un conjunto acotado Ω, minimizando una cierta función

$$\mathcal{F}(\Omega)$$

donde Ω es el dominio de diseño el cual puede estar sujeto a restricciones de forma.

$$\mathcal{G}(\Omega) = 0$$

Usualmente se está interesado en determinar Ω, sobre el cual se especifican las restricciones deseadas especificadas no afecten la evolución de la pieza, ya que en tal caso se corre el riesgo que la pieza no sea correctamente optimizada.

Al discretizarla frontea en puntos, la optimización de forma resulta un problema más complejo que la optimización topológica, ya que las coordenadas de los puntos sobre las fronteras de la estructura son las variables de diseño. De tal forma que, si se busca mayor exactitud, mayor será la cantidad de variables de diseño requeridas, y que se tengas que variar durante el proceso de optimización.

En un problema de optimización de forma existen diferentes tipos de restricciones, entre las cuales las más comunes son:

- Restricción de desplazamiento: Esto permite mantener a validez geométrica de los contornos generados. También evita que, debido a las perturbaciones excesivas en los nodos, dos contornos lleguen a cruzarse e invaliden la configuración geométrica del modelo.
- Restricción de masa: Con el fin de conservar la coherencia del problema, se incluyen restricciones de masa dependiendo de lo que se busque con esta, minimizar o maximizar.
- Restricción de esfuerza: La restricción de esfuerzo para el problema de optimización de forma busca garantizar la resistencia del diseño optimizad para la condición de carga aplicada.

Aplicaciones

La optimización topológica se utiliza en la fase conceptual de proceso de diseño para llegar a una propuesta de diseño conceptual que sea funcional y fabricable. Esto nos ayuda a reducir el tiempo del ciclo de diseño y el costo total, mientras se obtiene un diseño confiable.

Las aplicaciones de la optimización topológica son muy amplias, entre las principales se pueden mencionar el diseño de elementos de máquinas, donde generalmente el objetivo es minimizar la cantidad del material usado (masa) y las deformaciones, restringiendo los esfuerzos por medio de alguna teoría de falla. Igualmente, la optimización topológica tiene aplicaciones en el área de dinámica de fluidos, en el diseño de elementos con aplicación térmica y aplicaciones en el campo de la biomecánica.

Implementación del código en MATLAB

El código de optimización topológica tiene muchas implementaciones, vamos a analizar algunos ejemplos donde se estudia la optimización topológica en un material frente a una carga térmica uniforme. Cambiando algunas partes del código (se puede consultar en el *TFG* por *Román Abad Castro* [4]) para modificar la estructura y utilizando la ventana de comandos de Matlab debemos llamar la función top e introducir el siguiente código de entrada:

top(40,40,0.4,3.0,1.2)

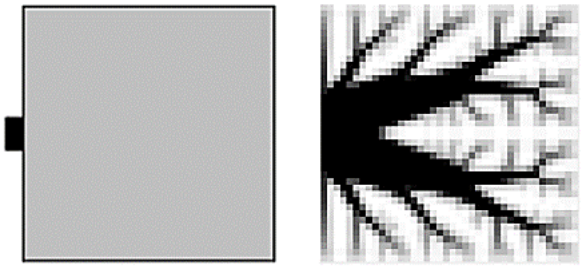
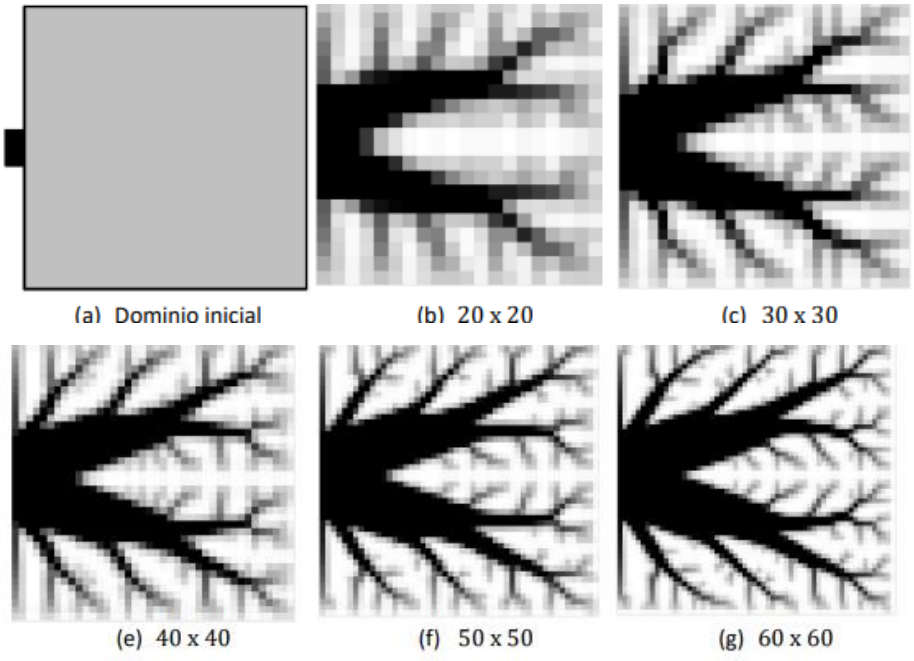


Figura 3. Ejemplo de la optimización topológica frente a una carga térmica uniforme.

En la parte izquierda de la figura 3 se muestra el dominio inicial (área gris) y la zona por la que saldrá el calor (recuadro negro en un lateral de la figura). Y a la derecha se puede observar la distribución optima de material ocupando el 40% del volumen inicial.
Es posible obtener una gran cantidad de resultados haciendo modificaciones en el código de Matlab o en los parámetros de diseño y comparar los resultados para concluir cual solución sería la más apropiada para la optimización topológica. Las variaciones más notables (para más información consultar el *TFG* por *Román Abad Castro* [4]):

- **Variación en el número de elementos:** sometiendo el mismo cuadro a una carga térmica uniforme, dividiendo el dominio inicial en un número de elementos, desde 20 x 20 hasta 100 x 100. Ejemplos:



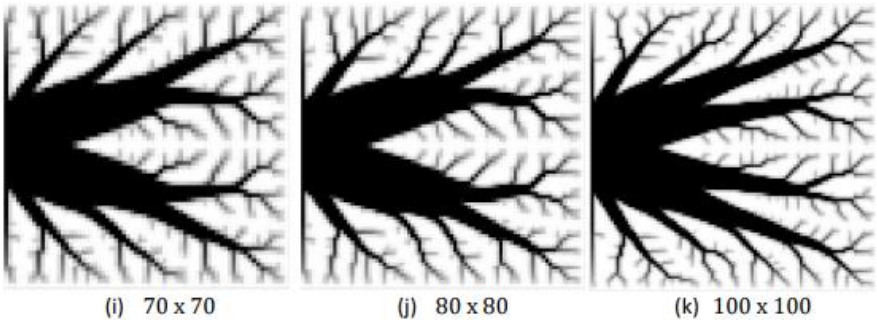


Figura 4. Ejemplos de soluciones con variación en el número de elementos.

Podemos observar como la nitidez de la imagen va aumentando a medida que se incrementan los elementos, aunque con 20 x 20 elementos se puede observar se forma aproximada como serán las ramificaciones. El único inconveniente con el que cuenta incrementar el número de elementos es la capacidad de la computadora que este realizando las pruebas ya que al incrementar los elementos la cantidad de interacciones se incrementa demasiado, esto puede generar sobre calentamientos en algunos equipos y tiempos de espera prolongados para dar con una solución.

- **Variación del factor de penalidad *p*:** El factor de penalidad reduce la aparición de zonas "grises" que puedan complicar la interpretación de los resultados.

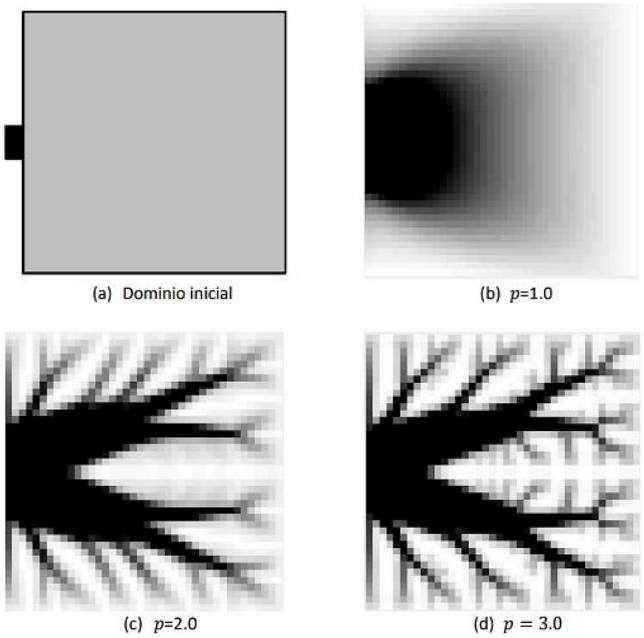


Figura 5. Resultados variando el factor de penalidad "*p*".

Un factor de penalidad de 1, significa que el resultado no se modificara, este tipo de imagen no aporta ningún tipo de información para la solución. Con *p*=2 se pueden distinguir algunas ramificaciones sin embargo esta solución no cumple con la precisión necesaria para ser tomada como una buena solución. Finalmente, con *p*=3 se pueden diferenciar claramente las zonas blancas y negras, habiendo ramificaciones suficientes. Si se introduce un factor de penalidad superior a 3, en la programación de *Román Abad Castro* [4], el programa entra en bucle sin converger en una solución. Por lo que en este caso el factor de penalidad más apropiado es *p*=3.

- **Variación de la restricción de volumen:** Se modifica el porcentaje del dominio inicial que será ocupado por material sólido.

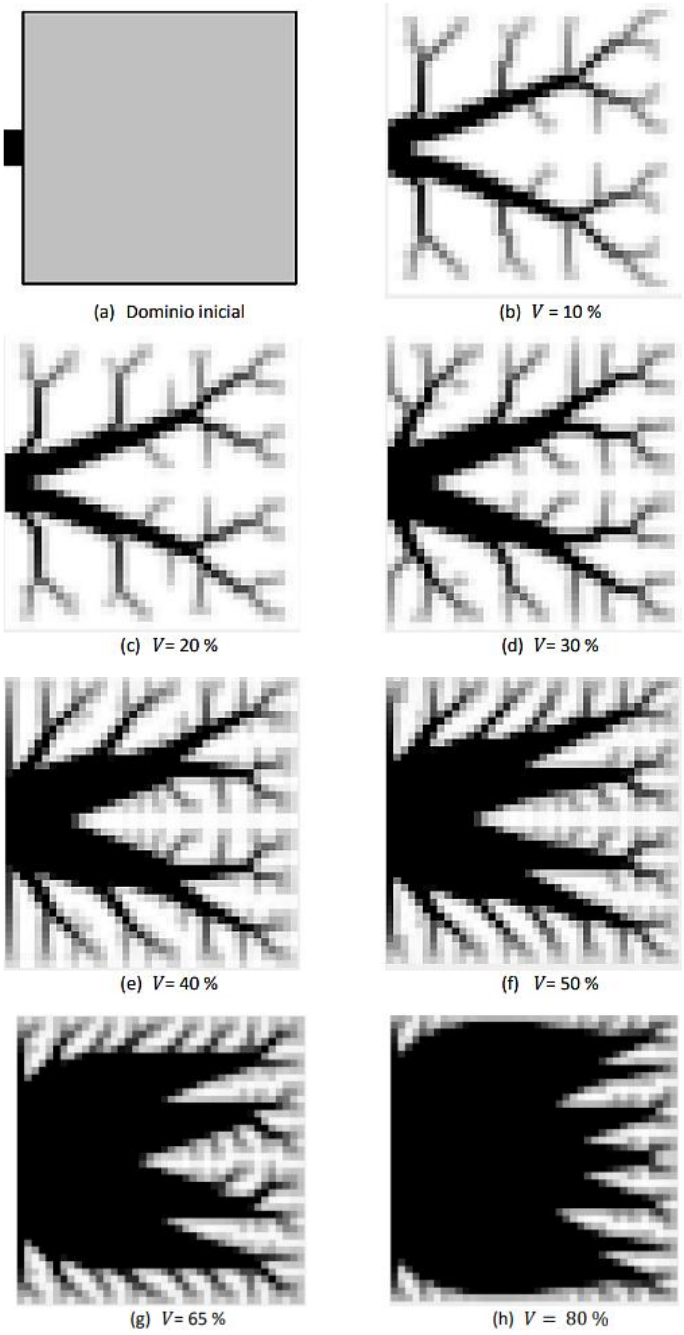


Figura 6. Solución para distinto porcentaje de volumen sólido.

A mayor volumen crece la parte central, compuesta por material sólido, las ramificaciones toman formas similares, y entre mayor volumen menores son los huevos entre ellas. Aquí se intenta encontrar el equilibrio entre el costo de utilizar más material para la pieza y la eficacia para evacuar el calor. Entre menos volumen de material se use menor será el costo, pero acumulara más tensiones y fallara antes.

- **Variación del radio de filtro:** parámetro utilizado para que no haya diferencias grandes de densidad entre nodos. Al aumentar el radio de filtro, se incrementan los elementos que entran en el área de filtro, los elementos separados por una distancia menor al radio no podrán tener densidades muy diferente.

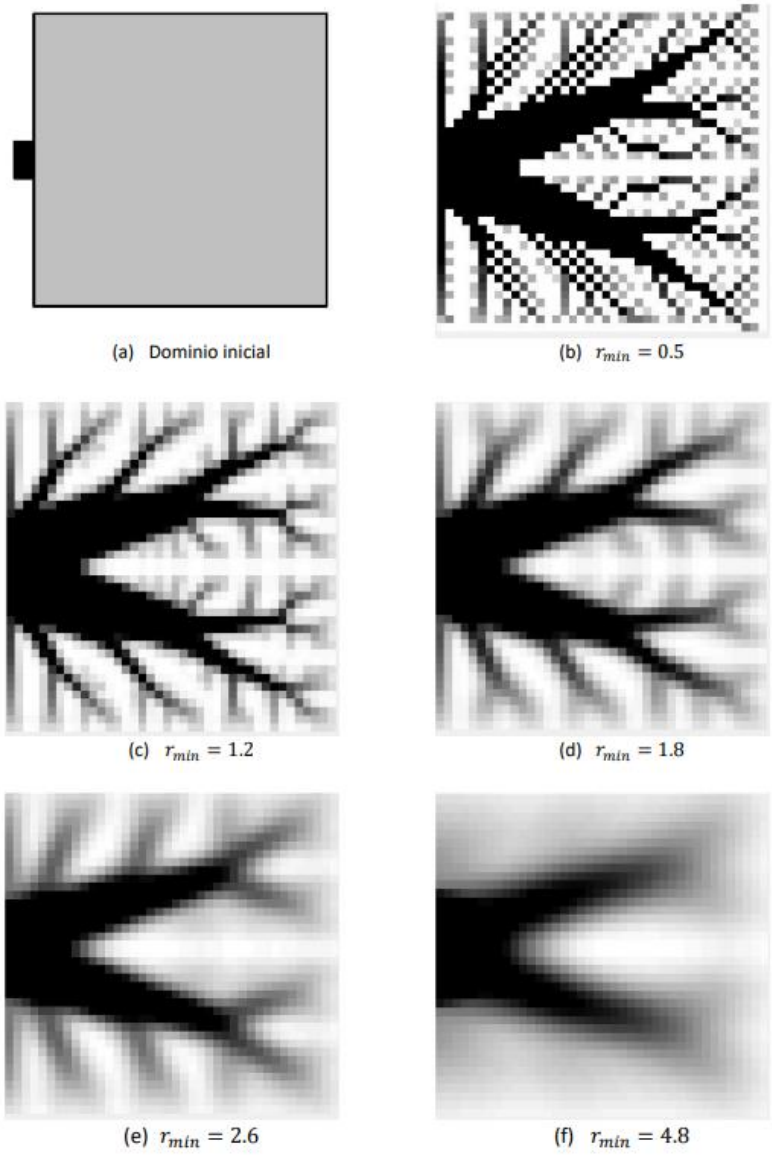


Figura 7. Solución para distinto radio de filtro en la solución.

En la imagen (b) de la figura 7, se observa un problema conocido como "checkerboard" esta es la primera razón por la que se implementa un filtro y con el valor de *r = 0.5* no interactúa con los valores que lo rodean por lo que el filtro no está funcionando. Sin embargo, incrementar demasiado el filtro se observa un efecto parecido a cuando no se aplicaba el factor de penalidad, sin aportar información útil. Para este caso el filtro más optimo es *r = 1.2* suficiente para eliminar el efecto "checkerboard" pero no tan grande para perder el efecto del factor de penalidad.

Otras modificaciones que se pueden hacer, como diferentes formas y restricciones

- Cambiar el dominio inicial a un cuadrado: Se obtiene modificando la llamada al programa inicial:

top(80,40,0.4,3.0,1.2)

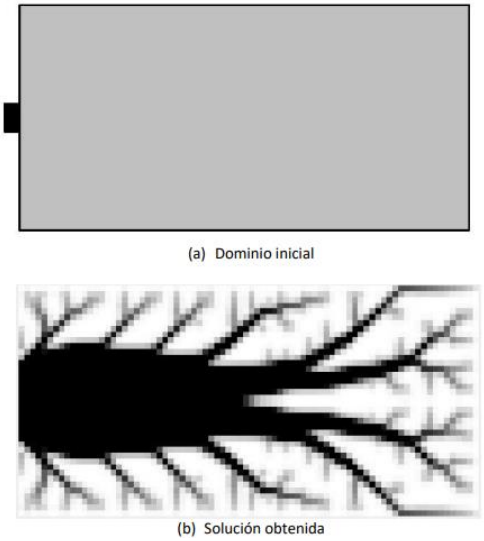


Figura 8. Optimización topológica de un dominio rectangular.

- **Doble zona de evacuación de calor:** para obtener esta variación se modifica el código en Matlab para incrementar los nuevos elementos de salida variable "ficeddoffs".

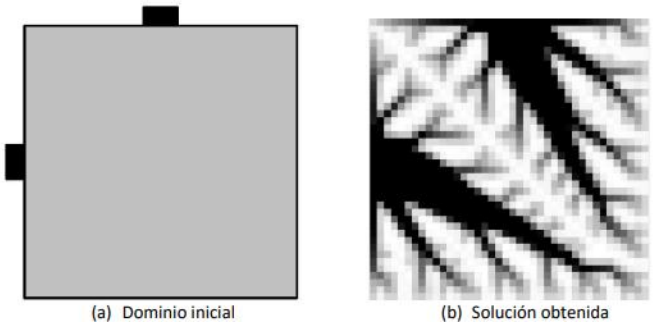


Figura 9. Evaluación por partes izquierda y superior.

Se observa una simetría al trazar un eje diagonal, desde la esquina superior izquierda hasta la inferior derecha, incluso se pueden visualizar 2 triángulos simétricos respecto al mismo eje.

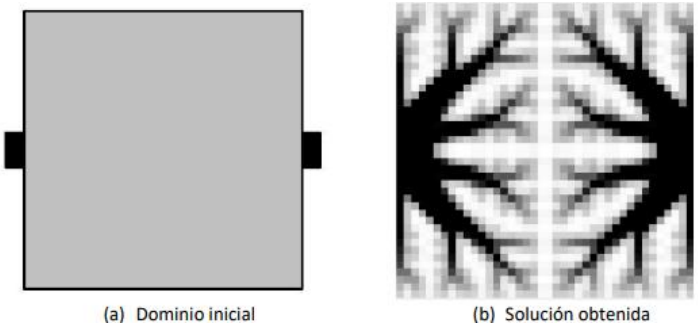


Figura 10. Evaluación por partes izquierda y derecha.

Se observa la simetría entre ambas partes respecto a un eje central, en estos casos se recomienda aprovechar la simetría para reducir el coste operacional ya que sencillamente se pudo analizar un dominio inicial de 20 x 40 y obtener ambas soluciones con una imagen espejo.

Carga térmica solamente a la mitad del dominio inicial: Para obtener este resultado se modifica el código de Matlab de *Román Abad Castro* [4] línea 75 por la siguiente:

```
F((nelx+1)*nely/2+1:1:(nelx+1)*(nely+1),1) = 1;
```

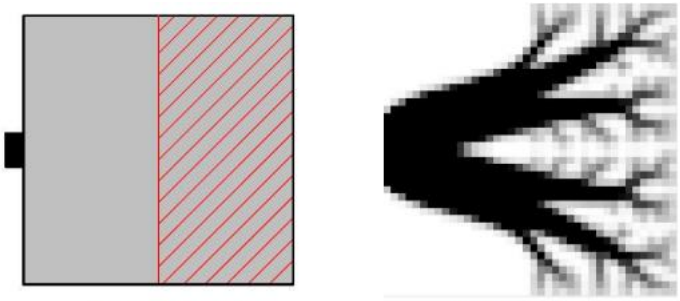


Figura 11. Optimización topología con carga térmica en solo la mitad del dominio

Se observa como las ramificaciones centrales siguen existiendo, pero las ramificaciones más pequeñas solo se extienden en la zona sometida a la carga térmica, donde se busca optimizar el material.

- **Dominio cuadrado con zona no optimizable y evacuación de calor por zona no optimízable:** Se estudia el dominio inicial, pero con una zona no optimizable, es decir, con una zona vacia. Para ello en la subsección OC donde se aplica el criterio de optimalidad, se añase tras la línea 40 del código de *Román Abad Castro* [4], la siguiente línea:

```
xnew(find(passive))=0.001;
```

La sentencia asignada es la densidad mínima 0.001 equivalente a un valor nulo a todos los elementos que se encuentren en la variable “passive”. Esta variable se define al principio del código y la vamos a modificar para generar una zona no optimizable circular con un diámetro de 15 elementos, en el centro del cuadrado, con el siguiente código después de la línea 4:

```
for ely = 1:nely
    for elx = 1:nelx
        if sqrt((ely-nely+1/2.)^2 + (elx-nelx/2.)^2) < 7.5
            passive(ely,elx)=1;
            x(ely,elx)=0.001;
        else
            passive(ely,elx)=0;
        end
    end
end
```

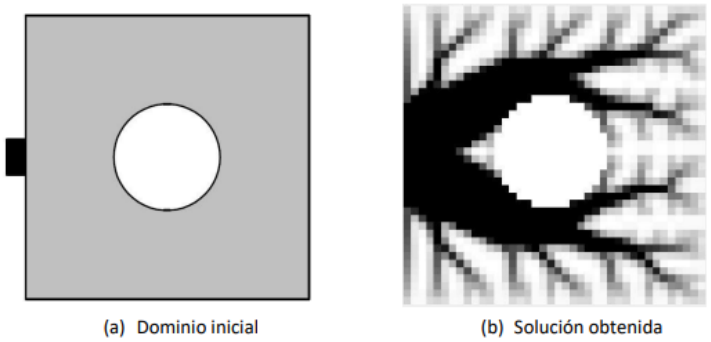


Figura 12. Dominio con sección no optimizable

El diámetro del círculo puede cambiar, pero lo interesante es observar como las raíces se adaptan a la zona no optimizable, rodeándola completamente. Es posible añadir más zonas no optimizables incluso con diferentes formas, relacionando la ecuación de la superficie no optimizable con las coordenadas de cada elemento.

Otra posible aplicación de la zona no optimizable es que la zona de evacuación sea el área no optimizable:

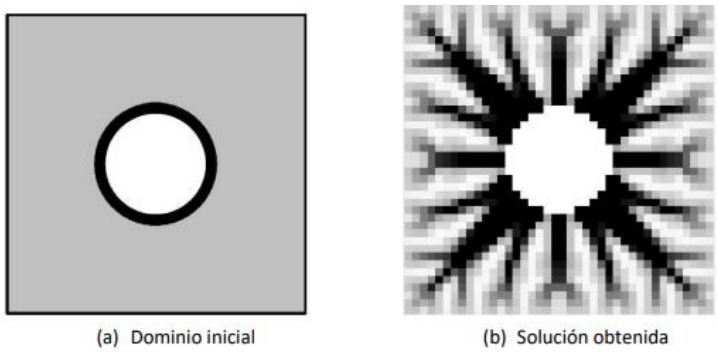


Figura 13. Evacuación de calor por la zona no optimizable.

Al igual que en los otros ejemplos las ramificaciones crecen desde el área por donde evacua el calor y se extienden las ramificaciones.

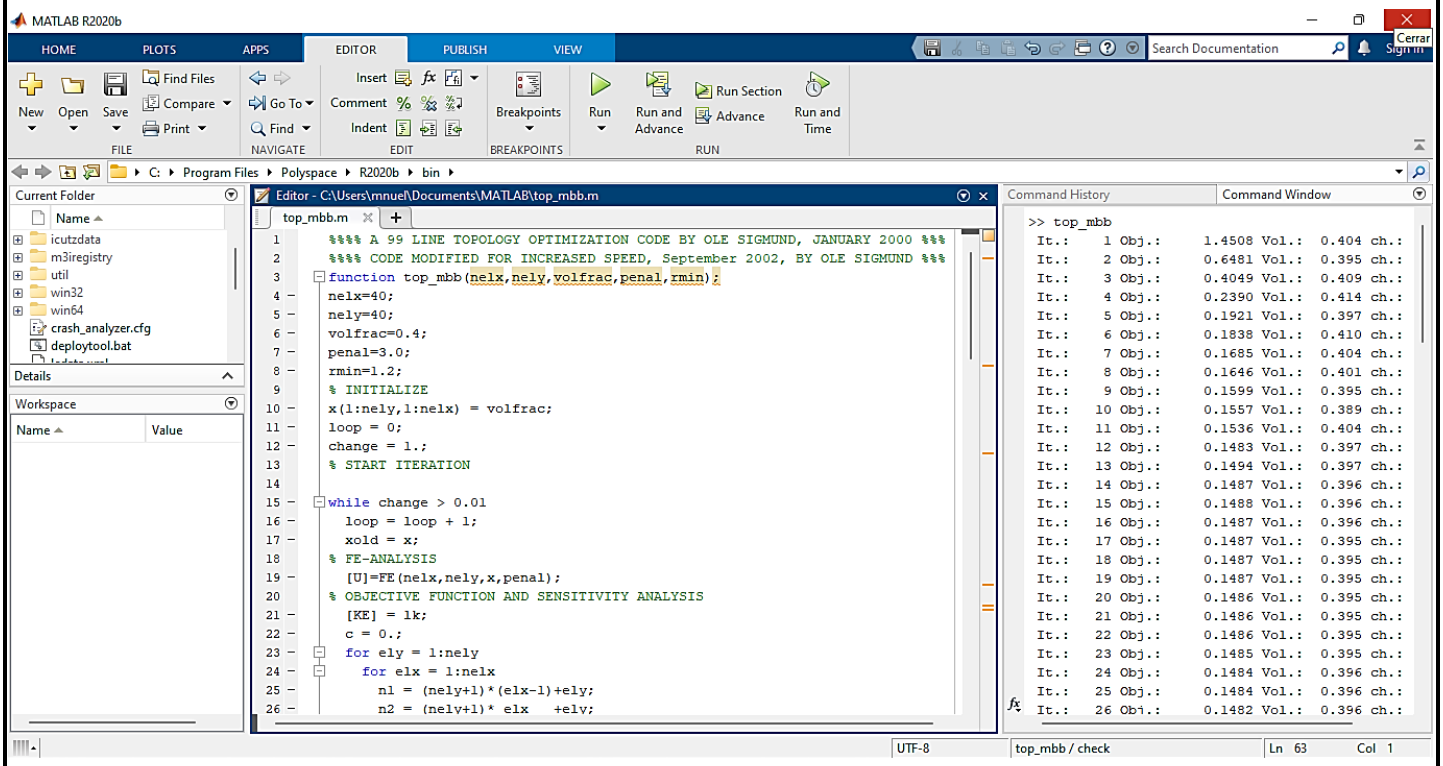
Código en Matlab.

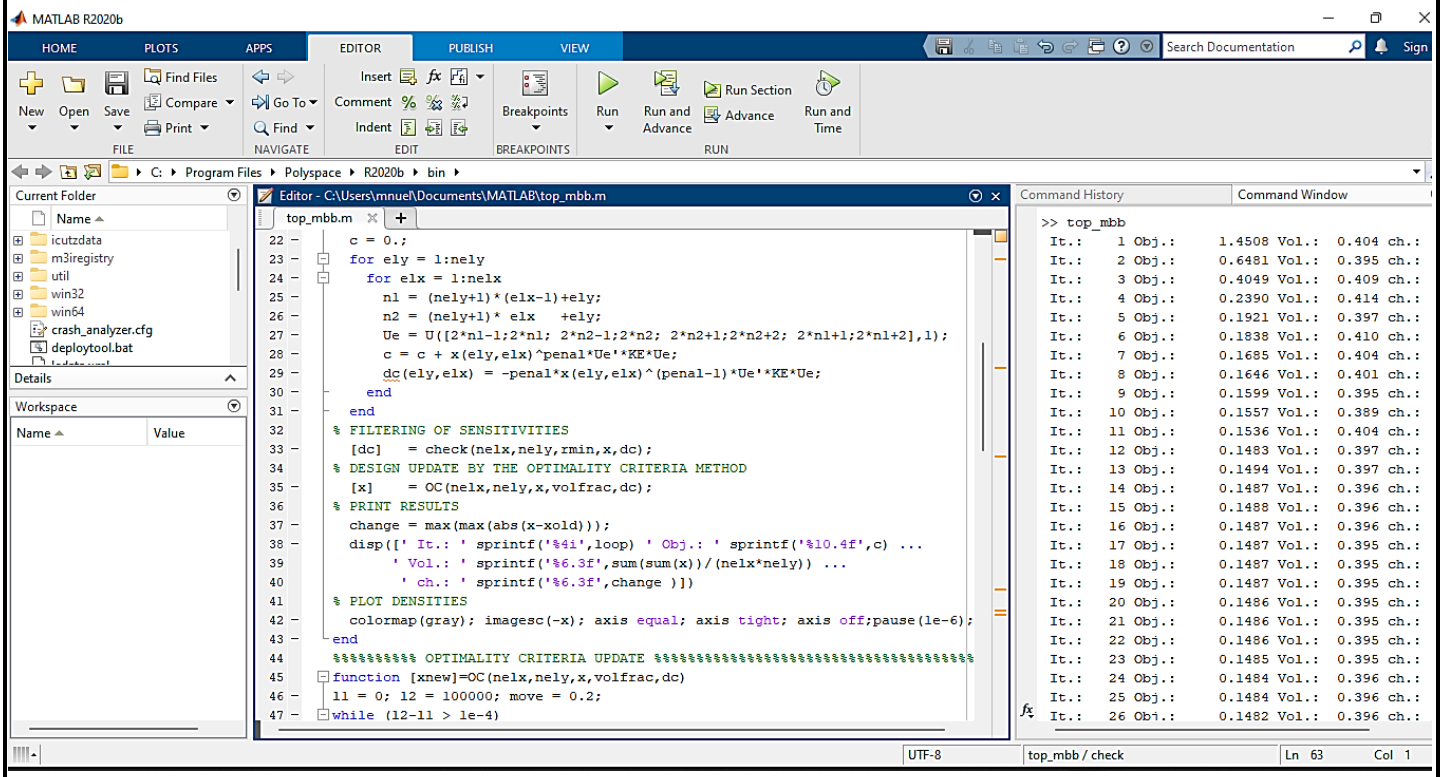
Se analiza una estructura de viga tipo cantiléver, la cual esta empotrada del lado izquierdo y cuenta con una carga puntual en la esquina inferior derecha. Este tipo de problema fue utilizado por O. Sigmund como ejemplo en su publicación "A 99 line topology optimization code written in Matlab" [1], y nos permite evaluar el desempeño de la herramienta desarrollada teniendo en cuenta que los resultados óptimos para este problema de optimización son bien conocidos.

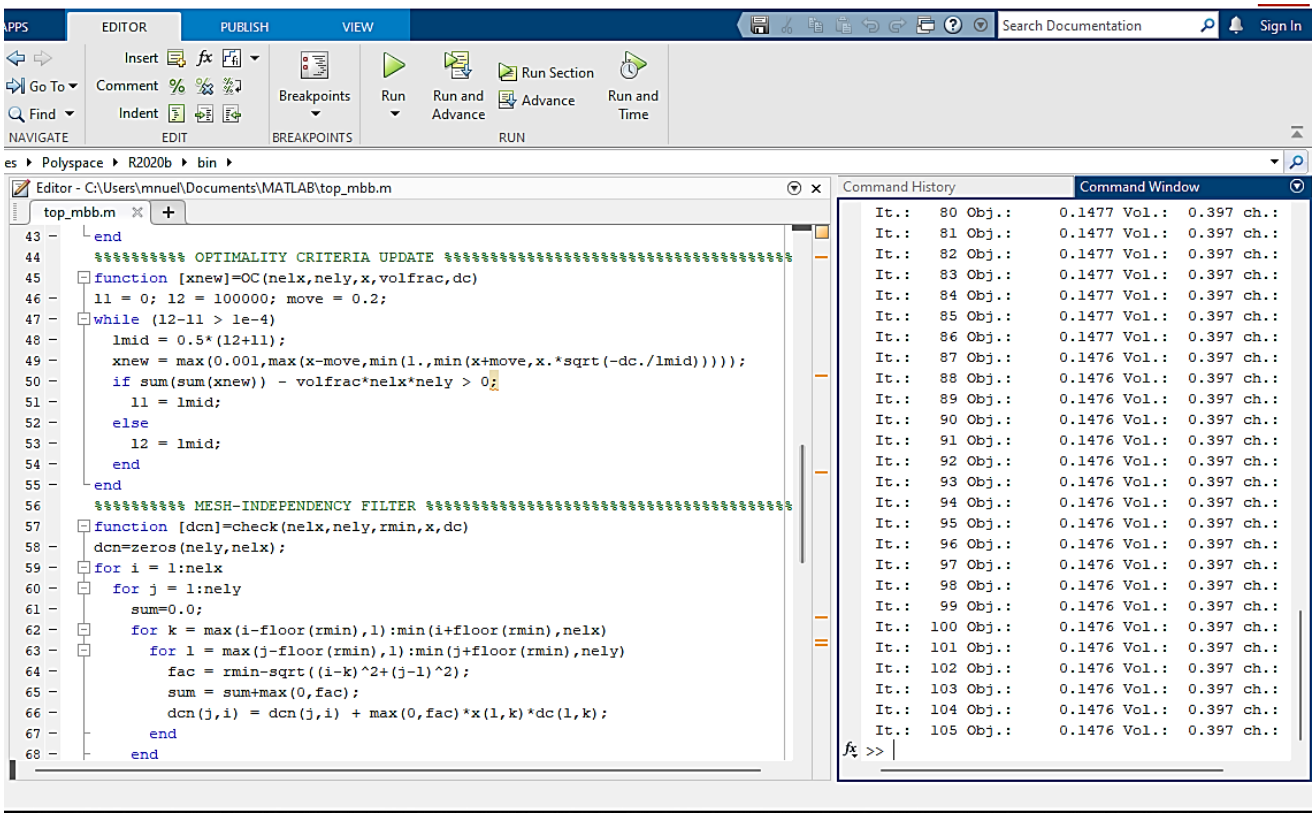
```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
%%%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%
function top(nelx,nely,volfrac,penal,rmin);
nelx=40;
nely=40;
volfrac=0.4;
penal=3.0;
rmin=1.2;
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
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
title('Análisis de estructura de viga tipo cantiléver con MATLAB');
%%%%%%%%%% 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-floor(rmin),1):min(i+floor(rmin),nelx)
      for l = max(j-floor(rmin),1):min(j+floor(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),1); U = zeros(2*(nely+1)*(nelx+1),1);
for elx = 1:nelx
  for ely = 1:nely
    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 LOADS AND SUPPORTS (HALF MBB-BEAM)
F(2,1) = -1;
fixeddofs   = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
alldofs     = [1:2*(nely+1)*(nelx+1)];
freedofs    = setdiff(alldofs,fixeddofs);
% SOLVING
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)];
```

Implementación/desarrollo de la programación en sus diferentes vistas

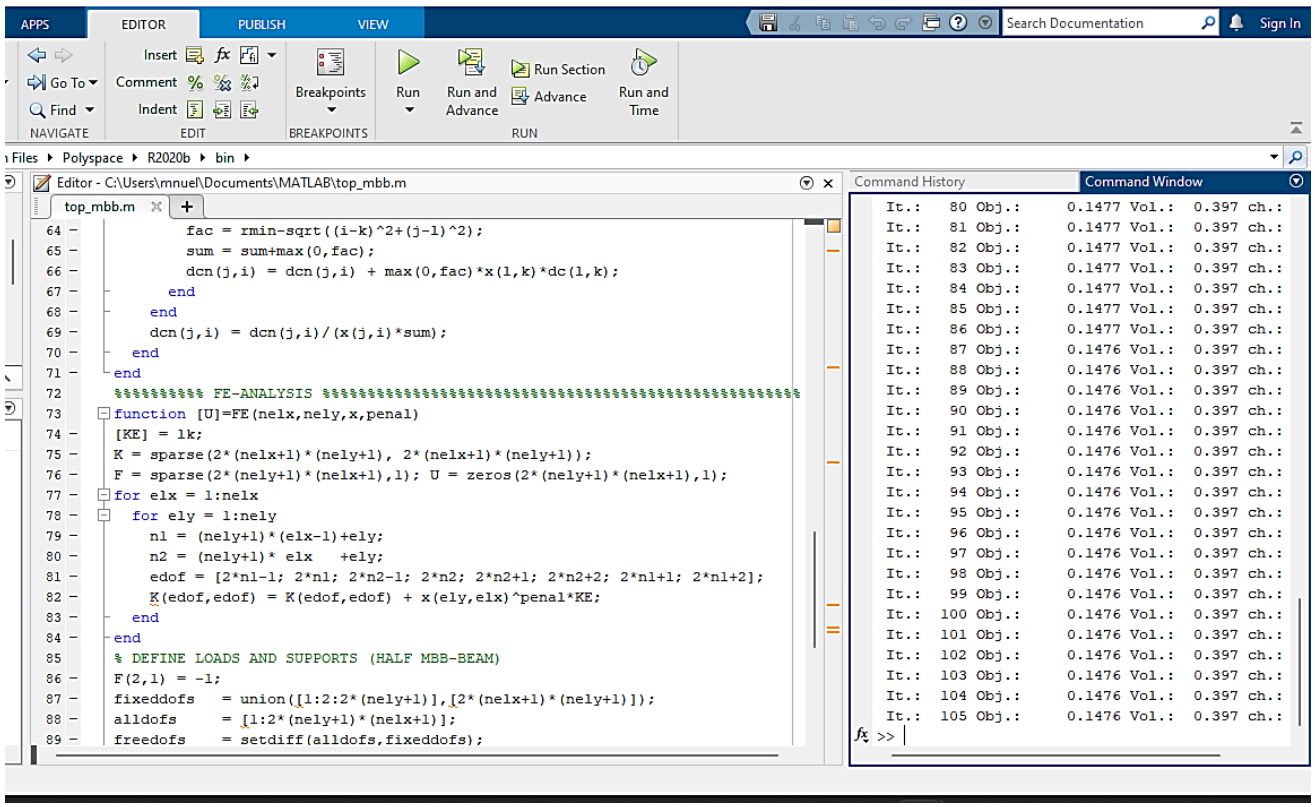







```
end
%%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [xnew]=OC(nelx,nely,x,volfrac,dc)
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
  lmid = 0.5*(l2+l1);
  xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
  if sum(sum(xnew)) - volfrac*nelx*nely > 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-floor(rmin),1):min(i+floor(rmin),nelx)
      for l = max(j-floor(rmin),1):min(j+floor(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
```

```
It.:   80 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   81 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   82 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   83 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   84 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   85 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   86 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   87 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   88 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   89 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   90 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   91 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   92 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   93 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   94 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   95 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   96 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   97 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   98 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   99 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  100 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  101 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  102 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  103 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  104 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  105 Obj.:    0.1476 Vol.:  0.397 ch.:
fx >>
```




```
        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),1); U = zeros(2*(nely+1)*(nelx+1),1);
for elx = 1:nelx
  for ely = 1:nely
    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 LOADS AND SUPPORTS (HALF MBB-BEAM)
F(2,1) = -1;
fixeddofs   = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
alldofs     = [1:2*(nely+1)*(nelx+1)];
freedofs    = setdiff(alldofs,fixeddofs);
```

```
It.:   80 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   81 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   82 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   83 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   84 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   85 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   86 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   87 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   88 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   89 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   90 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   91 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   92 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   93 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   94 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   95 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   96 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   97 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   98 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   99 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  100 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  101 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  102 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  103 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  104 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  105 Obj.:    0.1476 Vol.:  0.397 ch.:
fx >>
```

PLOTS APPS EDITOR PUBLISH VIEW Search Documentation Sign In

Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

NAVIGATE EDIT BREAKPOINTS RUN

C: ▸ Program Files ▸ Polyspace ▸ R2020b ▸ bin ▸

Editor - C:\Users\mnuel\Documents\MATLAB\top_mbb.m

top_mbb.m

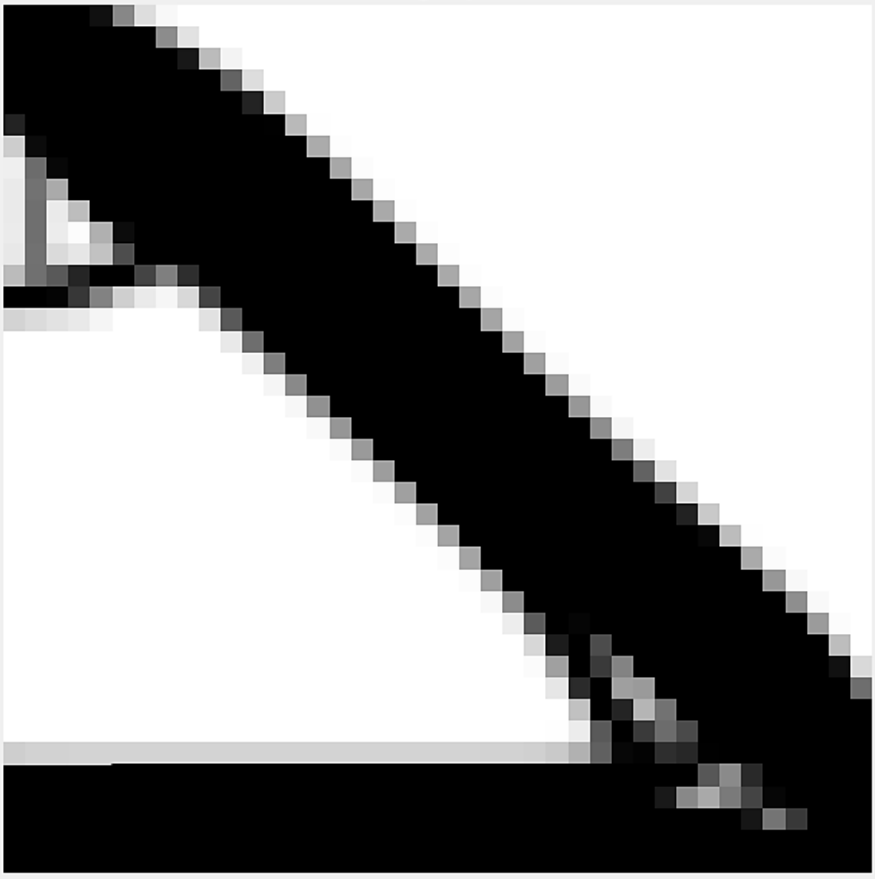
```
81 -        edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
82 -        K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
83 -      end
84 -    end
85      % DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
86 -    F(2,1) = -1;
87 -    fixeddofs   = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
88 -    alldofs     = [1:2*(nely+1)*(nelx+1)];
89 -    freedofs    = setdiff(alldofs,fixeddofs);
90      % SOLVING
91 -    U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
92 -    U(fixeddofs,:)= 0;
93      %%%%%%%%%% ELEMENT STIFFNESS MATRIX %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
94      function [KE]=lk
95 -    E = 210.;
96 -    nu = 0.3;
97 -    k=[ 1/2-nu/6   1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
98 -       -1/4+nu/12 -1/8-nu/8  nu/6       1/8-3*nu/8];
99 -    KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
100                      k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
101                      k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
102                      k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
103                      k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
104                      k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
105                      k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
106                      k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
```

Command History

Command Window

```
It.:   80 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   81 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   82 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   83 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   84 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   85 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   86 Obj.:    0.1477 Vol.:  0.397 ch.:
It.:   87 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   88 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   89 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   90 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   91 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   92 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   93 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   94 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   95 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   96 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   97 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   98 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:   99 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  100 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  101 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  102 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  103 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  104 Obj.:    0.1476 Vol.:  0.397 ch.:
It.:  105 Obj.:    0.1476 Vol.:  0.397 ch.:
fx >>
```

Análisis de estructura de viga tipo cantiléver con MATLAB



Conclusiones del Equipo 1

Susana Rubio Medina. - De acuerdo con la investigación realizada podemos determinar que la optimización topológica es un tema de mucha importancia en el ámbito industrial ya que gracias a esto se pueden generar mejoras en las distintas estructuras de maquinaria lo cual no sólo beneficia a un solo sector sino a toda la comunidad ingenieril puesto que uno de nuestros objetivos como ingenieros es mejorar la calidad de vida e innovar en la industria. Para nuestra primera practica se genero un código en Matlab para comprender y realizar el análisis de una viga tipo cantiléver que, aunque no sea una pieza biomecánica su estructura fue la más adecuada para la aplicación de nuestro código.

Melissa Lizeth Galindo Reyes. - Para la investigación del marco teórico de esta práctica se realizó la investigación d la optimización topológica en diversas fuentes como tesis, artículos científicos, publicaciones de libros y artículos de congresos los cuales nos ayudaron a comprender el funcionamiento y la importancia de la optimización topológica. Por otra parte, la asistencia de Matlab para nuestro análisis represento una importante parte de nuestro trabajo ya que se realizaron diversas funciones que no conocíamos.

Oziel Alberto Torres Villarreal. – En esta práctica se trabajó con la generación del código de Matlab para el análisis de una pieza, para ello se estudió el cómo generar el código paso a paso ya que se trabajó con la publicación de O. Sigmund como ejemplo en "A 99 line topology optimization code written in Matlab" [1], de forma que se estudió a fondo su funcionamiento. Conocimos cada una de las secciones que integran el código de la optimización topológica y el cómo se realiza la ejecución de su análisis.

Gabriel Eduardo Vázquez Ortega. – Gracias a esta práctica pudimos ver que el código de optimización topológica tiene muchas implementaciones, esto ayuda a reducir el tiempo de diseño y con ello el costo total de la fabricación de piezas, lo que se busca en el laboratorio de biomecánica es analizar y proponer una mejor a una pieza alguna prótesis y como primer paso se realizó el análisis de una pieza mediante el código de optimización topológica de O. Sigmund.

José Manuel Reséndiz García. – Ya que la optimización topológica se utiliza en la fase conceptual de proceso de diseño para llegar a una propuesta de diseño, en nuestra practica 1 se utilizó para realizar el análisis de una viga tipo cantiléver, análisis que se trabajó en el software de Matlab modificando un código base para poder aplicarlo al análisis de nuestra pieza obteniendo como resultado la graficación de la estructura analizada.

Miguel Ángel Martínez Villanueva. – Como parte de esta práctica se analizaron las acciones que inter digo de la optimización topológica para ello se realizó el código necesario para generar el análisis de nuestra estructura de forma que también se pueda visualizar gráficamente el cómo es que funciona nuestro código además gracias a la investigación realizada previamente como parte del estado del arte se pudo comprender los resultados obtenidos.

Bibliografía

- ✓ De Facultad de Ingeniería, Universidad Autónoma de San Luis Potosí. (Ed.). (2013). Herramienta de diseño bidimensional que integra optimización, topología y forma. MEMORIAS DEL XIX CONGRESO INTERNACIONAL ANUAL DE LA SOMIM. http://somim.org.mx/memorias/memorias2013/pdfs/A1/A1_35.pdf [1]

- ✓ Duque D., C. A., Garzón A., D. A., & Roa G., M. A. (2005). Análisis del regeneramiento óseo bajo el esquema de optimización topológica. Ingeniería e Investigación, 25(1), 22–29. https://www.redalyc.org/articulo.oa?id=64325104 [2]

- ✓ Plaza Rodríguez, E. (2021). Automatización de un programa de optimización topológica de un elemento tridimensional a escala mesoscópica. 174806. https://riunet.upv.es/handle/10251/174806 [3]

- ✓ Castro, R. A. (2018). IMPLEMENTACIÓN EN MATLAB DE UN PROCEDIMIENTO DE OPTIMIZACIÓN TOPOLÓGICA PARA ESTRUCTURAS CON CARGAS TÉRMICAS. Bilboko Ingeniaritza Eskola. https://addi.ehu.es/bitstream/handle/10810/29399/TFG%20Rom%c3%a1n%20Abad%20Castro.pdf?sequence=1&isAllowed=y [4]

- ✓ González, J. L. D. (2019, mayo 22). La Optimización Topológica. El futuro inmediato del diseño en automoción. Linkedin.com; LinkedIn. https://www.linkedin.com/pulse/la-optimizaci%C3%B3n-topol%C3%B3gica-el-futuro-inmediato-del-en-dur%C3%A1n-gonz%C3%A1lez [5]