UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

LABORATORIO DE BIOMECANICA 509

PRACTICA 2.- DISEÑO DEL MARCO DE UNA BICICLETA

MATRICULA	NOMBRE ALUMNO		
1987049	MIGUEL ANGEL MORALES ARREDONDO		
1755483	FELIPE DANIEL ZAMARRIPA VALDEZ		
1915391	SEVERO WENCESLAO ARREOLA PLATAS		
1605716	ALVARO ALEXIS GARCIA GARZA		
1669008	JUAN ANGEL REYNA MEDRANO		
GRUPO / SALON	HORA	FRECUENCIA	SEMESTRE
509 / 12BMC	N5	VIERNES	AGOSTO – DICIEMBRE 2022
MAESTRO	ISAAC ESTRADA		

PEDRO DE ALBA S/N, CIUDAD UNIVERSITARIA, SAN NICOLAS DE LOS GARZA, NL.

Índice

Objetivo

Aprender a utilizar la lógica del código de Matlab para colocación de cargas, apoyos y fuerzas dentro de un espacio de diseño propuesto.

Marco Teórico

Hay diferencias entre las bicicletas de los hombres y de las mujeres y una de ellas es el diseño del cuadro. La bicicleta de las mujeres está diseñada para hacer más fácil de montar mientras que las bicicletas del hombre no. En esta práctica vamos a optimizar el diseño del marco para mejorarlo.

El problema de diseño se ilustra en la figura 2.1. Nuestra tarea es construir la parte frontal del marco para que la bicicleta sea lo más rígido posible. Esta parte se conecta con el manubrio y el asiento. El dominio de diseño se ilustra esquemáticamente en la figura 2.2.

Consideraciones

- El manubrio produce una fuerza en dirección vertical.
- Bastidor trasero actúa como soporte.
- Tenemos que declarar una parte vacía del domino de diseño para hacer espacio para la rueda delantera.

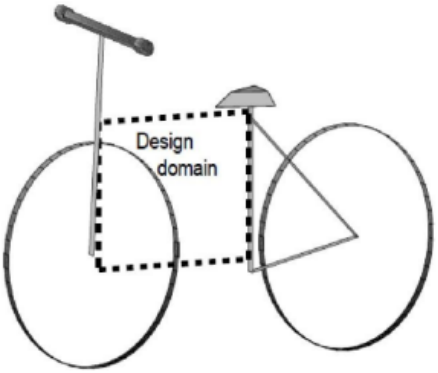


Figura 2.1 Problema de diseño

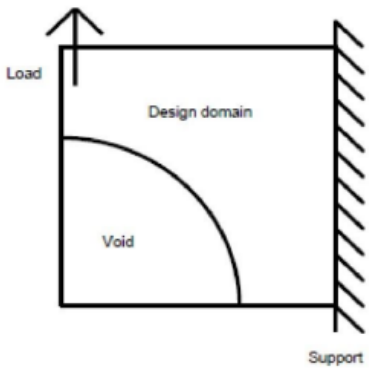


Figura 2.2. Vista esquemática del dominio de diseño

Desarrollo

Primero vamos a considerar solo la carga y el apoyo. Para esto editaremos líneas 80 y 81 del código de matlab:

80 F (2,1)=1;

81 fixeddofs = 2 * nelx * (nely + 1) + 1:2 * (nelx + 1) * (nely + 1);

Guarde el código en el mismo directorio. Luego ejecute Matlab con:

top(20,20,0.33,3.0,1.5)

El resultado debe ser similar a la figura 2.3. El dominio de diseño se discretiza 20 veces en 20 elementos finitos. Tal vez piense que la magnitud de la fuerza no es realista y que el módulo de Young debe ser corregido en la línea 89. Comúnmente $E = 2 * 10^{11} N/m^2$.

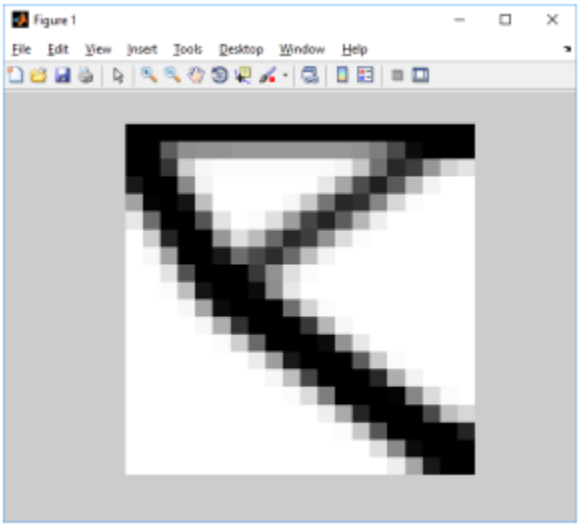


Figura 2.3 : Resultado de la primera optimización

También el tamaño de los elementos finitos se define como una1 vez por 1 unidad. Estos valores se tienen que cambiar para obtener valores correctos de optimización, pero no siempre es necesario cambiarlos ya que son solo correcciones de las escalas. Sin embargo, si se quieren corregir se puede modificar la línea 41 para mantener la precisión mientras se resuelven las ecuaciones:

41 while ((l2-l1)/l2 > 1e-4)

Penalización y filtro de radio

La sintaxis de la función es:

top(nelx,nely,volfrac,penal,rmin)

Donde las variables denotan lo siguiente:

nelx es el número de elementos finitos en la dirección horizontal.

nely es el número de elementos finitos en la dirección vertical.

volfrac es la fracción de volumen en el dominio de diseño.

penal es la penalización de las densidades intermedias. Una penalización alta hará la solución en blanco y negro, es decir los elementos finitos estarán llenos o vacíos. Una penalización = 1 significa que no hay penalización de las densidades intermedias.

▪ rmin es un radio de filtro para un filtro que hace que el diseño de malla-independiente.

Definición de regiones vacías.

El resultado en la figura 2.3 no deja ninguna zona hueca para la rueda delantera. Vamos a llamar a los elementos finitos en este pasivo vacío, y definir una matriz con ceros en elementos libres y seres en pasiva. Agregue las siguientes líneas al código de MATLAB entre la línea 5 y 6 para hacer esto:

```
for ely = 1:nely
    for elx = 1:nelx
        if ((elx)^2+(ely-nely)^2) < (0.65*nelx)^2
            passive(ely,elx) = 1;
        else
            passive(ely,elx) = 0;
        end
    end
end
x(find(passive))=0.001;
```

El último comando inicializa todos los elementos de la zona hueca en el bajo valor 0.001. También tenemos que actualizar la línea 29 y 40 e insertar una línea adicional entre 43 y 44:

```
29 [x] = OC(nelx,nely,x,volfrac,dc,passive);
40 function [xnew]=OC(nelx,nely,x,volfrac,dc,passive)
43b xnew(find(passive)) = 0.001;
```

Realiza estos cambios y ejecuta con:

```
top(20,20,0.33,3,1.5)
```

Esto va a generar algo más parecido a la bicicleta para un hombre, el marco frontal se muestra en la figura 2.4.

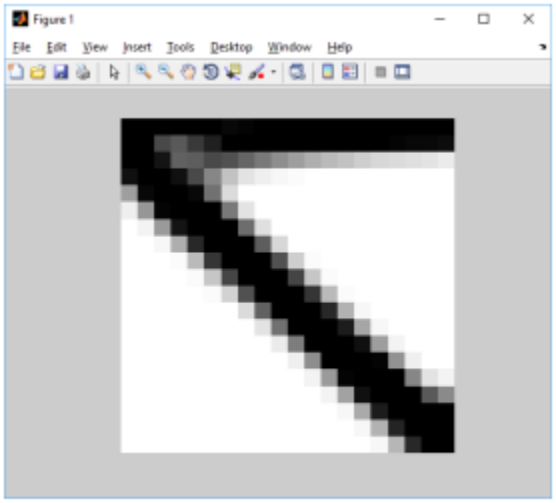
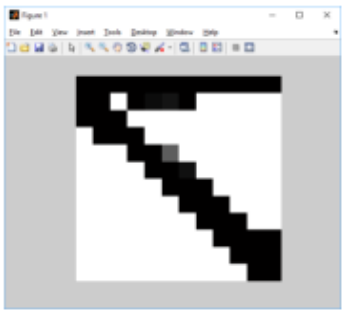


Figura 2.4 : Resultado de la segunda optimización

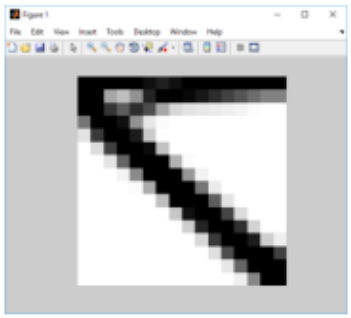
Ahora vuelva a realizar el ejercicio anterior y cambie los valores de los parámetros para responder a las siguientes preguntas:

1. ¿El diseño final depende del tamaño de mallado? Compare el resultado con los siguientes comandos:

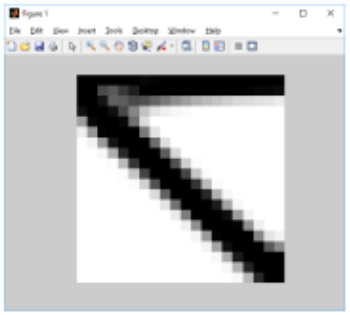
top(12,12,0.33,3.0,0.9);



top(16,16,0.33,3.0,1.2);

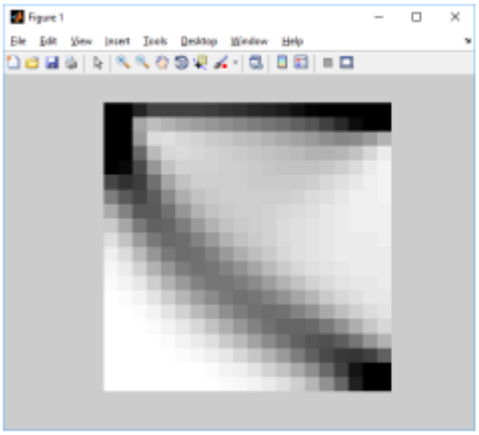


top(20,20,0.33,3.0,1.5);

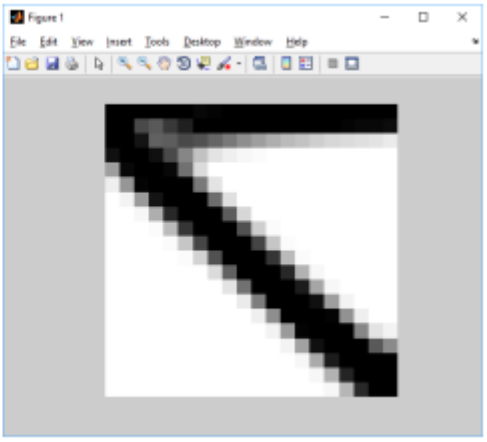


2. ¿Cuán mejor es el resultado del diseño si no exigimos que sea en blanco y negro? Comprobar el cumplimiento de diseño final de los siguientes casos:

top(20,20,0.33,1.0,1.5);



top(20,20,0.33,3.0,1.5);

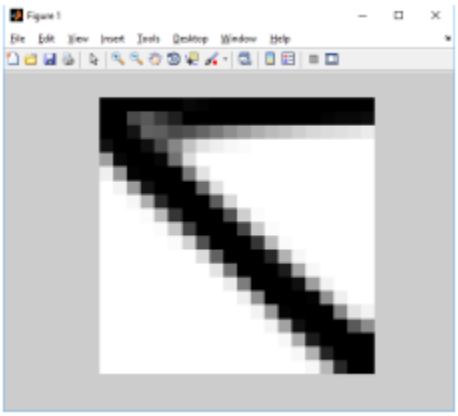


3. Ahora vamos a estudiar el filtro de mallado. El filtro se desactiva eligiendo un rmin menor que 1 o poniendo la línea 27 como comentario para que el programa ignore la instrucción.

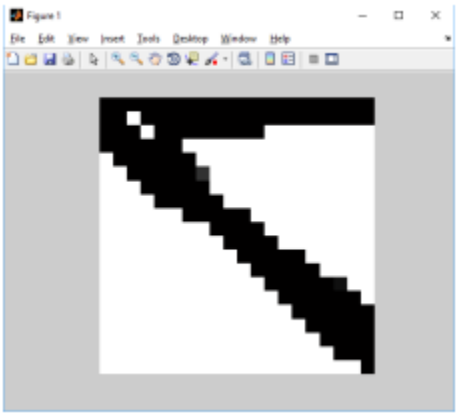
27 % [dc] = check(nelx,nely,rmin,x,dc);

top(20,20,0.33,3.0,1.5);

Con filtro



Sin filtro



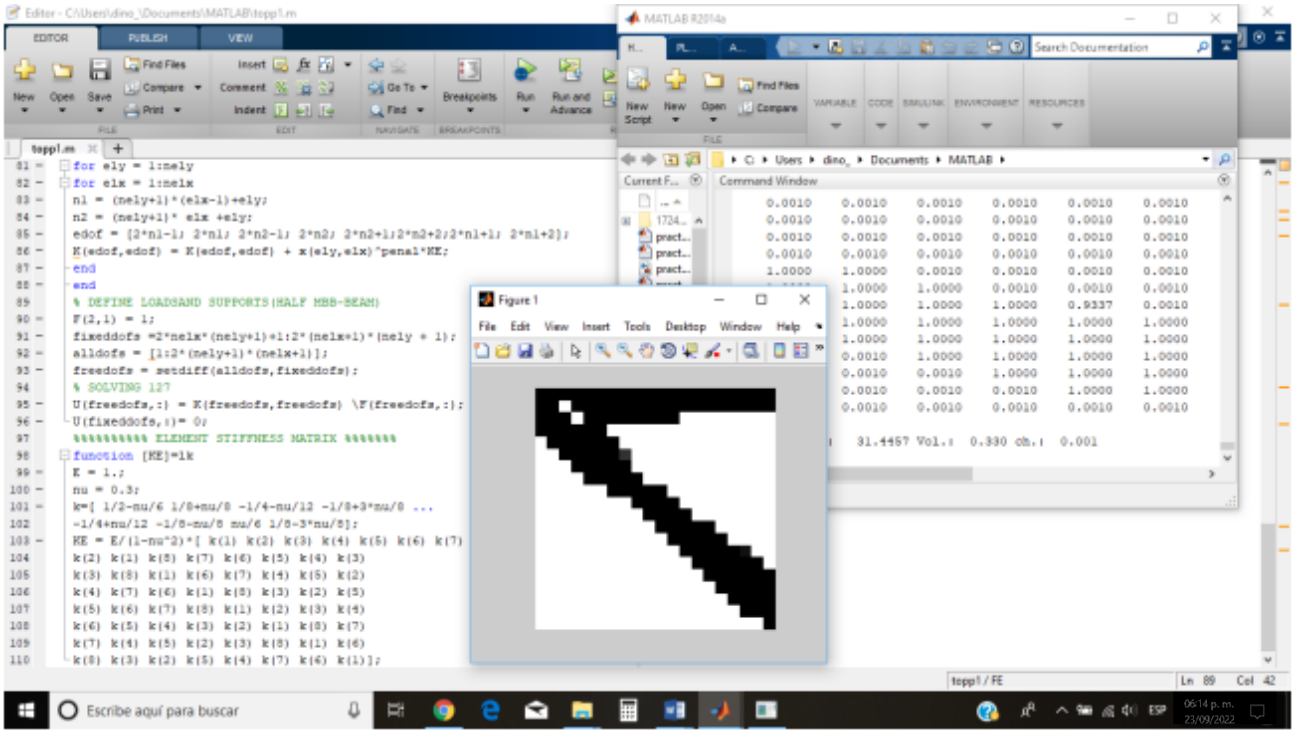
Código del programa

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function topp1(nelx,nely,volfrac,penal,rmin)
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
for ely = 1:nely
for elx = 1:nelx
 if ((elx)^2+(ely-nely)^2) < (0.65*nelx)^2
 passive(ely,elx) = 1;
 else
 passive(ely,elx) = 0;
 end
end
end
x(find(passive))=0.001;
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,passive);
% 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,passive)
l1 = 0; l2 = 100000; move = 0.2;
while ((l2-l1)/l2 > 1e-4)
lmid = 0.5*(l2+l1);
xnew(find(passive))=0.001
```

```
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),1); U =sparse(2*(nely+1)*(nelx+1),1);
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(2,1) = 1;
fixeddofs =2*nelx*(nely+1)+1:2*(nelx+1)*(nely + 1);
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)];
```

Captura de pantalla



Conclusiones

Miguel Angel Morales Arredondo

Para empezar en la primera instrucción que nos dio la práctica somos capaces de observar que efectivamente el diseño final depende mucho del tamaño del mallado. Al exigir que el resultado no sea en blanco y negro podemos ver que se pierde la nitidez del diseño y se ve muy diferente. Por último, al aplicarle un filtro podemos suavizar los bordes del resultado final de diseño para poder obtener una imagen de una manera limpia comparándola con la anterior. Con esto podemos ver que podemos hacer varios diseños con las diferentes funciones dentro del código.

Alvaro Alexis García Garza

Concluyo que en esta práctica fue fácilmente hacer el diseño sobre el tamaño del mallado, así como sus medidas para poder realizarla mediante el Matlab y comparar resultados precisos, y conllevar de una u otra forma una mejor explicación a lo que se estaba ejerciendo dentro de la practica más bien de lo que te pedía la dicha práctica, en si mediante las imágenes se puede mostrar el diseño y los resultados finales.

Felipe Daniel Zamarripa Valdez

Al llevar a cabo esta práctica, se pudo comprender la importancia que tiene considerar los problemas que puede tener un diseño al momento de ser usado, es por esto que se implementan métodos dentro de los diseños, ecuaciones, y parámetros dentro de la programación que puede analizar estos casos, proyectando en un resultado previo a ejecutar el proyecto por completo.

Juan Angel Reyna Medrano

En esta práctica comprobamos que el tamaño del mallado es muy importante para que el diseño de nuestra pieza tenga una mejor nitidez o traducido al maquinado, para evitar bordes irregulares. De esta manera podemos optimizar piezas para reducir material innecesario y ahorrar costos en producción.

Severo Wenceslao Arreola Platas

En base a esta práctica, se tiene en cuenta lo que es la división de el modelado de la pieza, a esta división se le puede llamar, malla, la cual entre más puntos de división tenga nos puede ayudar a reducir o encontrar ciertos cortes y disminuir bordes con los cuales podemos trabajar y hacer cortes de una mejor manera, también se analizó que en esta práctica se pueden sacar y visualizar el resultado de la minimización de la pieza en masa, asimismo el desarrollo de los esfuerzos aplicados a la pieza antes y después de su nueva modelación.