Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

- *Tarea: Diseño de la estructura de un Panorámico.*

Maestro: Isaac Estrada.

Nombre:	**Matricula:**
Felipe Daniel Zamarripa Valdez	1755483
Miguel Angel Morales Arredondo	1987049
Severo Wenceslao Arreola Platas	1915391
Alvaro Alexis García Garza	1605176
Juan Angel Reyna Medrano	1669008

Materia: Laboratorio de Biomecánica Hora: Viernes N5

Cd. Universitaria, San Nicolás de los Garza, N.L.

Diseño de la estructura de un Panorámico

Objetivo:

Desarrollar en el estudiante la capacidad de análisis, implementación y solución de un problema propuesto.

Marco Teórico:

Los panorámicos se exponen a altas ráfagas de viento, por lo que su estructura ocupa ser muy rígida para soportar estas fuerzas. El espacio de diseño a evaluar será de 2 dimensiones, las cargas y los apoyos de observan en la figura 3.1

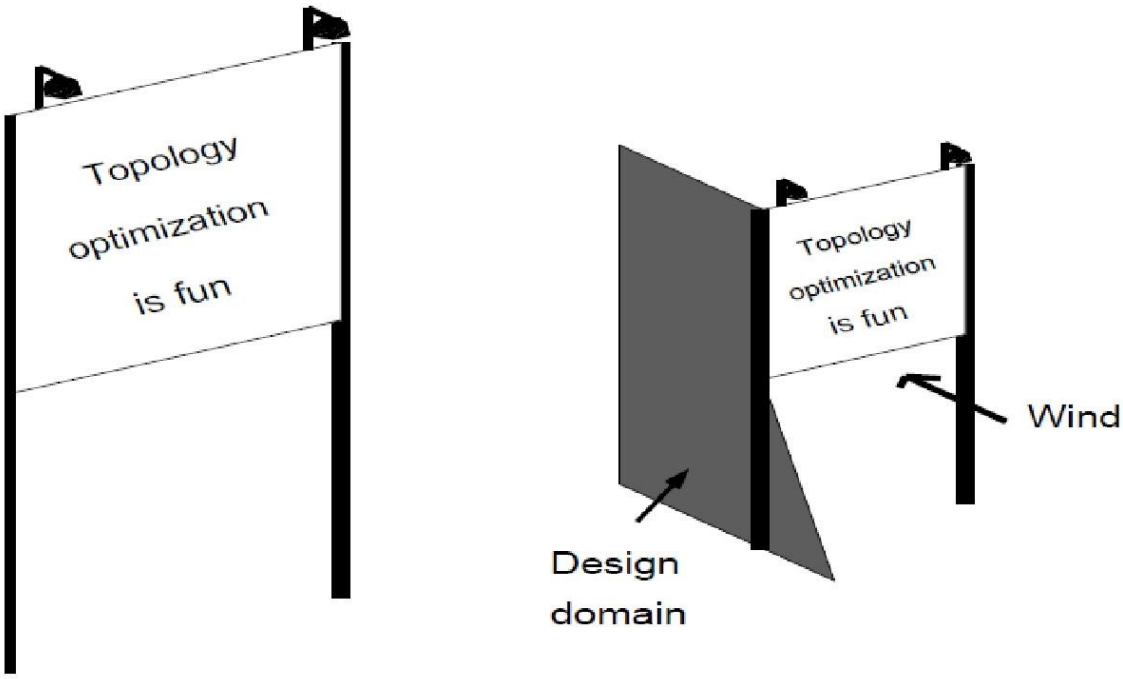


Figura 3.1 Imagen del Panorámico

En la figura 3.2 se puede ver el espacio de diseño para esta práctica. Se espera una fracción volumétrica aproximada de 0.20% del espacio de diseño. Supongamos que el panorámico es muy rígido 1, y sus patas son del mismo material que el marco.

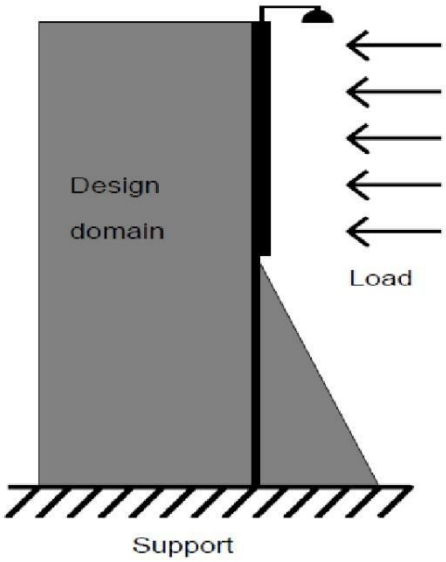
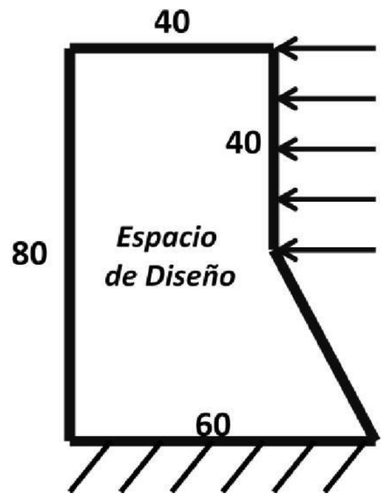


Figura 3.2 Espacio de diseño

Desarrollo:

Se tomaran ciertas consideraciones para la solución de esta práctica: 5 cargas, los apoyos tendrán restricciones en “X”,”Y” y el espacio de diseño para esta práctica será de:



Reporte:

- ❖ Realizar el ejercicio con el espacio de diseño propuesto

Fuerzas múltiples

Para empezar tenemos que editar nuestro script topp, se tiene guardado como topp3, para poder ingresar las fuerzas que requerimos, si observamos nos encontramos con 5 y para cambiar el anclaje del espacio de diseño a otra posición se tiene que cambiar la línea con la instrucción <u>*fixeddofs*</u>, para esto se modificaran las siguientes líneas:

```
%%%%%%%%%% 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);
```

Figura 1. Código original

```
%%%%%%%%%%% 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);
```

Figura 2. Líneas modificadas

```
16    for ely = 1:nely
17      for elx = 1:nelx
18        n1 = (nely+1)*(elx-1)+ely;
19        n2 = (nely+1)* elx   +ely;
20        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);
21        c = c + x(ely,elx)^penal*Ue'*KE*Ue;
22        dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
23      end
24    end
```

Figura 3. Código original

```
32 -      for i= 1:5
33 -      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);
34 -      c = c + x(ely,elx)^penal*Ue'*KE*Ue;
35 -      dc(ely,elx) = dc(ely,elx) - penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
36 -    end
```

Figura 4. Líneas modificadas

```
78    % DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
79    F(2,1) = -1;
80    fixeddofs   = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
81    alldofs     = [1:2*(nely+1)*(nelx+1)];
```

Figura 5. Código original

```
94    % DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
95 -  F(2*(nelx)*(nely+1)+2,1)=1;
96 -  F(2*(nelx)*(nely+1)+20,1)=1;
97 -  F(2*(nelx)*(nely+1)+40,1)=1;
98 -  F(2*(nelx)*(nely+1)+60,1)=1;
99 -  F(2*(nelx)*(nely+1)+80,1)=1;
100 - fixeddofs   = 2*(nely+1):2*(nely+1):2*(nelx+1)*(nely+1);
```

Figura 6. Líneas modificadas

Empotramiento diagonal (elementos pasivos)

Para crear el empotramiento diagonal, o crear el espacio en blanco para recrear el empotramiento en la parte inferior derecha; en el archivo del uso del código de 99 líneas existe una sección donde se habla de elementos pasivos el cual sirve de

ayuda para determinar un espacio en blanco, en el ejemplo del archivo viene como hacer un circulo, y nosotros necesitamos un rectángulo y un triángulo para esto se modificaron y/o agregaron las siguientes líneas:

```
1     %%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
2     function top(nelx,nely,volfrac,penal,rmin);
3     % INITIALIZE
4     x(1:nely,1:nelx) = volfrac;
5     loop = 0;
6     change = 1.;
7     % START ITERATION
```

Figura 7. Código original

```
5 -   for ely = 1:nely
6 -       for elx = 41:nelx
7 -           if elx - 20 < (ely/2)
8 -               passive(ely,elx)=0;
9 -           else
10 -              passive(ely,elx)=1;
11 -          end
12 -      end
13 -  end
14 -  x(find(passive))= 0.001;
```

Figura 8. Líneas modificadas

```
27    % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
28      [x]     = OC(nelx,nely,x,volfrac,dc);

37    %%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%
38    function [xnew]=OC(nelx,nely,x,volfrac,dc)
```

Figura 9. Código original

```
41    % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
42 -    [x]     = OC(nelx,nely,x,volfrac,dc,passive);

51    %%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%%%%%%%%
52    function [xnew]=OC(nelx,nely,x,volfrac,dc,passive)
```

Figura 10. Líneas modificadas

```
40    while (l2-l1 > 1e-4)
41      lmid = 0.5*(l2+l1);
42      xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
43      if sum(sum(xnew)) - volfrac*nelx*nely > 0;
```

Figura 11. Código original

```
56 -    xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
57 -    xnew(find(passive)) = 0.001;
58 -    if sum(sum(xnew)) - volfrac*nelx*nely > 0;
```

Figura 12. Líneas modificadas

Resultado de la optimización

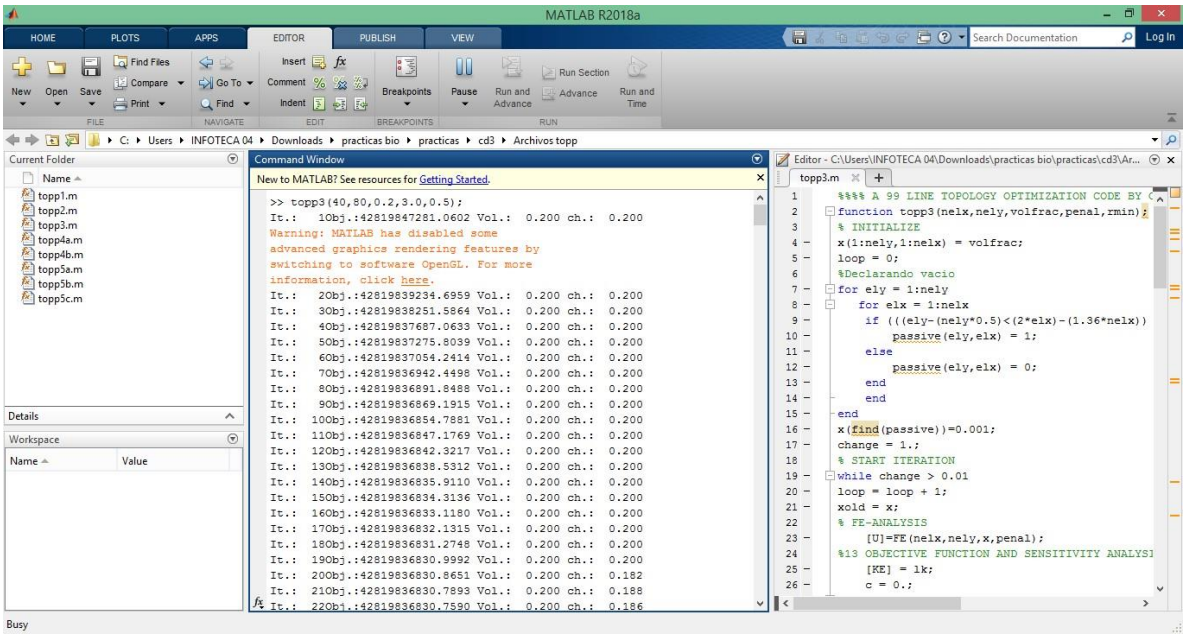


Figura 13. Resultado en la ventana de comando y editor

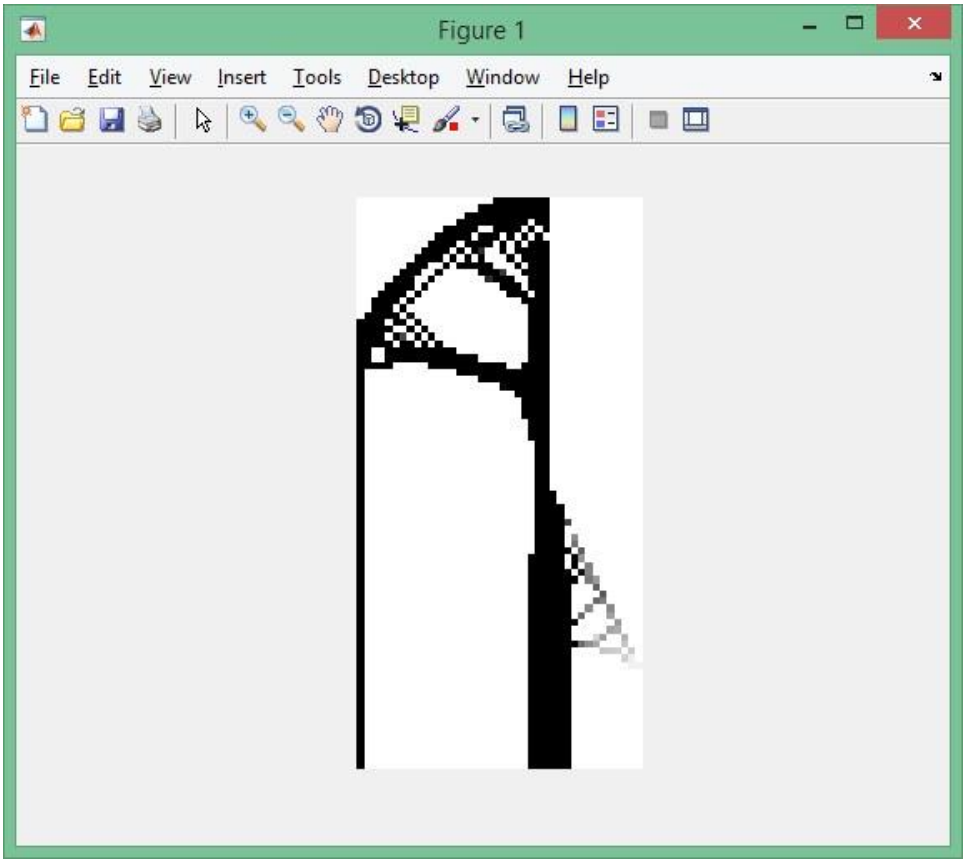


Figura 14. Diseño resultante para el problema planteado

Código completo con las modificaciones finales

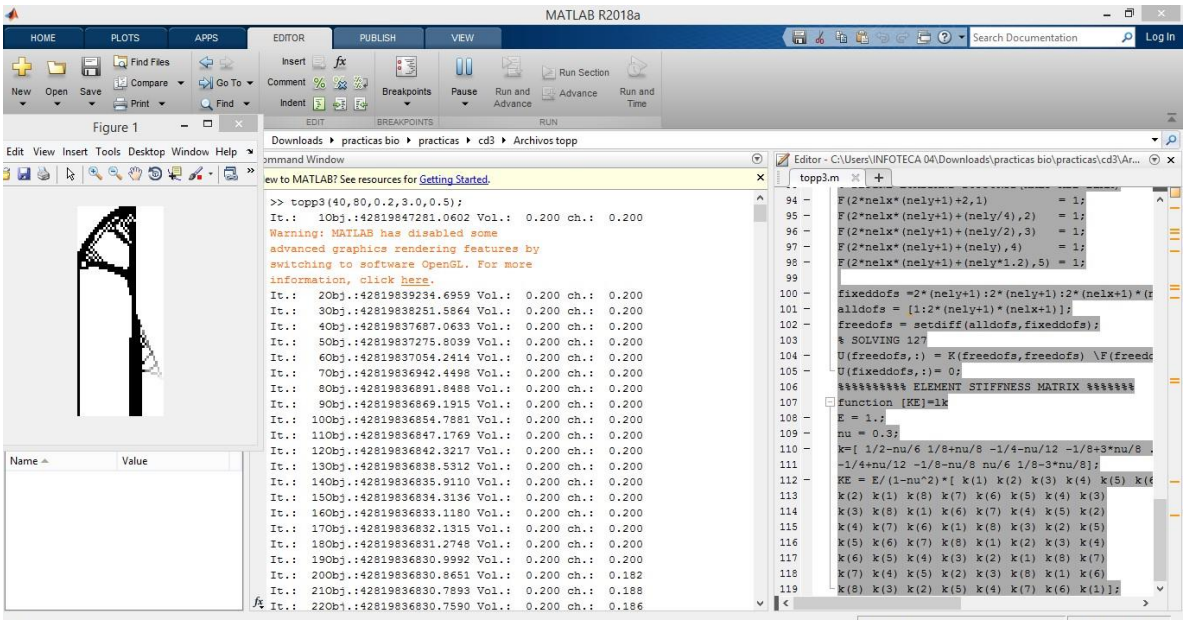
```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function topp3(nelx,nely,volfrac,penal,rmin);
% INITIALIZE
x(1:nely,1:nelx) = volfrac; loop = 0; %Declarando vacio for ely = 1:nely
for elx = 1:nelx       if (((ely-(nely*0.5)<(2*elx)-(1.36*nelx)) |(ely
<(1+nely*0.5))) &(elx
>(1+nelx)*0.6666))
passive(ely,elx)     =     1;
else
passive(ely,elx) = 0;     end
end end
x(find(passive))=0.00
1;   change   =   1.;   %
START ITERATION while
change > 0.01 loop =
loop + 1; xold = x;
% FE-ANALYSIS
    [U]=FE(nelx,nely,x,penal);
%13 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;    %19
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],1);
        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
%25 FILTERING OF SENSITIVITIES
[dc] = check(nelx,nely,rmin,x,dc);
%27 DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
[x] = OC(nelx,nely,x,volfrac,dc,passive);
%29 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
%40 %%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%% function
[xnew]=OC(nelx,nely,x,volfrac,dc,passive)  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))))); xnew(find(passive)) = 0.001; 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
%65 %%%%%%%%% 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
=zeros(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(2*nelx*(nely+1)+2,1)
= 1;
F(2*nelx*(nely+1)+(nely/4),2) = 1;
F(2*nelx*(nely+1)+(nely/2),3) = 1;
F(2*nelx*(nely+1)+(nely),4) = 1;
F(2*nelx*(nely+1)+(nely*1.2),5) = 1;


fixeddofs
=2*(nely+1):2*(nely+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)];
```

Impresión de pantalla



Conclusiones:

Juan Reyna
En esta practica aprendimos a usar MATLAB para realizar análisis de elemento finito para objetos de ámbito simple, dicho análisis nos ayuda mucho a optimizar piezas para hacerlas más seguras, resistentes y al menor costo siendo prueba de que el hacer uso de la tecnología puede facilitarnos el trabajo y realizarlo con una precisión mayor.

Alexis García
Esta practica fue algo sencilla ya que teniamos que saber las medidas o mas bien sacar las medidas exactas mediante el uso de MATLAB y asi llevar acabo el procedimiento adecuado para poder realiza la estructura de un panoramico.

Severo Arreola
Para la realización de esta práctica se hizo uso de Matlab, este software nos ayudó para sacar unos cálculos exactos y necesarios con los cuales este software nos apoya y nos da valores más precisos en dónde nos dice también los costos y los valores de seguridad.

Daniel Zamarripa
En esta reporte de laboratorio mostramos lo que se realizó a través de la práctica, esto usando Matlab, observamos que el tiempo para la realización de ésta fue mayor a la anteriores por el proceso que tuvo que llevar el software para optimizar los esfuerzos, además de ver los espacios en blanco que son elementos pasivos que necesitan ser tomados en cuenta para el diagrama.

Miguel Morales
A partir de lo que hicimos podemos concluir que aunque se crea que algo no se toma en cuenta dentro de un sistema de esfuerzos por ser un espacio en blanco, esto no debe ser así, debemos darle la importancia para el diseño óptimo del diagrama.