UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FIME
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Laboratorio De Biomecánica.

PRÁCTICA 03.

Diseño de la Estructura de un Panorámico.

Equipo 06-N5.

Andrik David Salas Carranza
Juan Carlos Saldaña González
Jeiddy Michel Martinez Navéjar
Ana Sofía Limón González
Joel Zuñiga Olvera
Yuliana Lizbeth Bravo Salazar
Fred Raúl Peña Mata
Raúl Alexandro Vega López
Jesús Alberto Medina González

3 de noviembre de 2022

Índice

1. OBJETIVO. **3**

2. INTRODUCCIÓN. **3**

3. MARCO TEÓRICO. **3**

4. DESARROLLO. **5**

4.1. Estado del arte 6

4.2. Procedimiento de la programación. 7

5. CONCLUSIONES. **18**

6. REFERENCIAS. **20**

1. OBJETIVO.

El estudiante deberá presentar un reporte con la solución numeral a computacional del problema de la simulación del Desempeño mecánico de componentes mecánicos por medio del método de MATLAB, esto para desarrollar en el estudiante la capacidad de análisis, implementación y solución de un problema propuesto.

2. INTRODUCCIÓN.

Las estructuras panorámicas son soportes en donde pueden ir ubicados diferentes tipos de anuncios publicitarios. La ventaja de estas estructuras es que pueden ir ubicadas en diferentes paisajes urbanos con el fin de promocionar un producto o servicio.Las estructuras pueden ser desde desde 4 x 6 metros, hasta 8 x 6 metros, ubicadas al ras del piso. Lo más común es que estas sean vistas desde cualquier parte de una zona y estén al alcance de todos. Algunas medidas habituales de los espectaculares pueden ser entre 320x200cm, 400×300 cm, 800×300 cm y 1200×400 cm.

3. MARCO TEÓRICO.

Una estructura panorámica es el soporte sobre el cual se posicionará un anuncio publicitario, ya sea de una cara o de tres caras.

Estas estructuras usualmente se encuentran en medio de diversos paisajes urbanos y sostienen diseños publicitarios con el objetivo de promocionar un producto, servicio o transmitir un mensaje.

Cada país tiene ciertas normativas en cuanto a dónde es apropiado o no colocar estos soportes para anuncios publicitarios. En algunos no está permitido que se construyan estructuras panorámicas a los lados de autopistas porque estos pueden distraer a los conductores.

Los panorámicos se exponen a altas ráfagas de viento, por lo que su estructura ocupa ser muy rígida para soportar estas fuerzas.

En la figura 1 se muestra el panorámico que será el espacio de diseño a evaluar, este será de 2 dimensiones, con cargas y apoyos como se muestra a continuación:

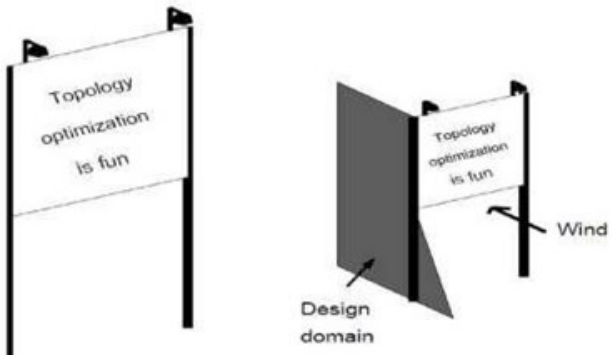


Figura 1: Imagen del Panorámico.

Existen diversos materiales que las agencias especializadas usan en la creación y diseño para estas estructuras. Usualmente, los postes panorámicos están construidos de metal y acero para que sean lo suficientemente resistentes al clima, lluvias y cualquier otro fenómeno de la naturaleza.

Por otra parte, el panorámico en sí mismo son hechos de lona, vallas de PVC, plástico, tela, metal o acrílico. También existen espectaculares digitales o electrónicos que tienen luces, pantallas eléctricas y música.

En la figura 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.

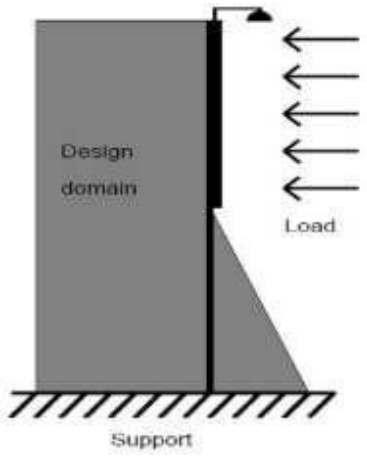


Figura 2: Espacio de Diseño.

4. DESARROLLO.

Se tomarán 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:

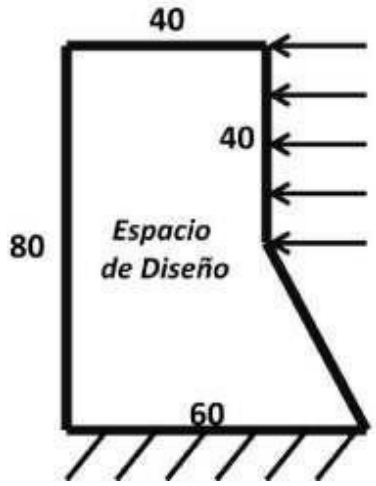


Figura 3: Espacio de Diseño

4.1. Estado del arte

Título Del Documento	A 99 line topology optimization code written in Matlab
Fuente Bibliográfica	99 Line Topology Optimization Code – O. Sigmund, Department of Solid Mechanics, Building 404, Technical University of Denmark, DK-2800 Lyngby, Denmark.
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 al análisis de elementos finitos, se observa que solo se necesitan 49 líneas 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.
Palabras, Clave	Optimización Topológica, criterios de optimización, web mundial, código Matlab
Conclusión	En este artículo investigado sobre la optimización topológica y su programación nos habla de cómo está constituido dicho código de 99 líneas y cuál es la idea principal que este debe cumplir y que es lo que debe demostrar al ejecutarlo.

Cuadro 1: Tabla Estado del Arte.

4.2. Procedimiento de la programación.

Usaremos el código de la práctica 1 dándole unos cambios necesarios para poder implementar de manera exitosa la nueva implementación, la cual en este caso es el panorámico.

Código Original.

```
%Práctica # Laboratorio Biomecánica Equipo 7

function toppract(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;
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

%%%%%%%%%% 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),1);
```

```
76 -    U = sparse(2*(nely+1)*(nelx+1),1);
77 -    for ely = 1:nely
78 -    for elx = 1:nelx
79 -    n1 = (nely+1)*(elx-1)+ely;
80 -    n2 = (nely+1)*elx+ely;
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 LOADSAND 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
94      %%%%%%%%%% ELEMENT STIFFNESS MATRIX %%%%%%%
95      function [KE]=lk
96 -    E = 1.;
97 -    nu = 0.3;
98 -    k=[ 1/2-nu/6 1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
99      -1/4+nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8];
100 -   KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
```

```
101     k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
102     k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
103     k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
104     k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
105     k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
106     k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
107     k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
```

Figura 4: Código Original.

Cambios para Fuerzas Múltiples.

Se tiene que editar el script para poder ingresar las fuerzas necesarias, observando vemos que tenemos 5 y para cambiar el anclaje del espacio de diseño a otra posición se tiene que cambiar con la instrucción fixeddofs modificando ciertas líneas.

```
71      %%%%%%%%%% FE-ANALYSIS %%%%%%%%%%%%
72      function [U]=FE(nelx,nely,x,penal)
73 -      [KE] = lk;
74 -      K = sparse(2*(nelx+1)*(nely+1),2*(nelx+1)*(nely+1));
75 -      F = sparse(2*(nely+1)*(nelx+1),1);
```

Figura 5: Fragmento del Código Original

```
86      %%%%%%%%%% FE-ANALYSIS %%%%%%%%%%%%
87      function [U]=FE(nelx,nely,x,penal)
88 -      [KE] = lk;
89 -      K = sparse(2*(nelx+1)*(nely+1),2*(nelx+1)*(nely+1));
90 -      F = sparse(2*(nely+1)*(nelx+1),5);
91 -      U = sparse(2*(nely+1)*(nelx+1),5);
```

Figura 6: Fragmento del código con líneas modificadas.

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

Figura 7: Fragmento del código Original.

```
28 -  for ely = 1:nely
29 -  for elx = 1:nelx
30 -    n1 = (nely+1)*(elx-1)+ely;
31 -    n2 = (nely+1)* elx +ely;
32 -    dc(ely,elx) = 0.;
33 -  for i = 1:5
34 -    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);
35 -    c = c + x(ely,elx)^penal*Ue'*KE*Ue;
36 -    dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
37 -  end
38 -  end
39 -  end
```

Figura 8: Fragmento del código con líneas modificadas.

```
85      % DEFINE LOADSAND 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)];
```

Figura 9: Fragmento del código Original.

```
100     % DEFINE LOADSAND SUPPORTS(HALF MBB-BEAM)
101 -   F(2*(nelx)*(nely+1)+2,1) = 1;
102 -   F(2*(nelx)*(nely+1)+(nely/4),2) = 1;
103 -   F(2*(nelx)*(nely+1)+(nely/2),3) = 1;
104 -   F(2*(nelx)*(nely+1)+(nely),4) = 1;
105 -   F(2*(nelx)*(nely+1)+(nely*1.2),5) = 1;
106 -   fixeddofs = 2*(nely+1):2*(nely+1):2*(nelx+1)*(nely+1);
107 -   alldofs = [1:2*(nely+1)*(nelx+1)];
108 -   freedofs = setdiff(alldofs,fixeddofs);
```

Figura 10: Fragmento del código con líneas modificadas.

Empotramiento Diagonal

Para nosotros poder crear el empotramiento diagonal o también conocido como espacio en blanco en la parte inferior derecha, necesitamos modificar el código original para poder recrear el espacio conocido con los siguientes cambios en la codificación para llegar al resultado del panorámico.

```
3     function toppract(nelx,nely,volfrac,penal,rmin)
4     % INITIALIZE
5 -   x(1:nely,1:nelx) = volfrac;
6 -   loop = 0;
7 -   change = 1.;
```

Figura 11: Fragmento del código original.

```
3     function toppract(nelx,nely,volfrac,penal,rmin)
4     % INITIALIZE
5 -   x(1:nely,1:nelx) = volfrac;
6 -   loop = 0;
7     %DECLARACION DE VACIO
8 -   for ely = 1:nely
9 -       for elx = 1:nelx
10 -          if (((ely-(nely*0.5)<(2*elx)-(1.36*nelx)) |(ely<(1+nely*0.5)))&(elx >(1+nelx)*0.6666))
11 -              passive(ely,elx)=1;
12 -          else
13 -              passive(ely,elx) = 0;
14 -          end
15 -      end
16 -  end
17 -  x(find(passive))=0.001;
18 -  change = 1.;
```

Figura 12: Fragmento del código con líneas modificadas.

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

Figura 13: Fragmento del código Original.

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

Figura 14: Fragmento del código con líneas modificadas.

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

Figura 15: Fragmento del código original.

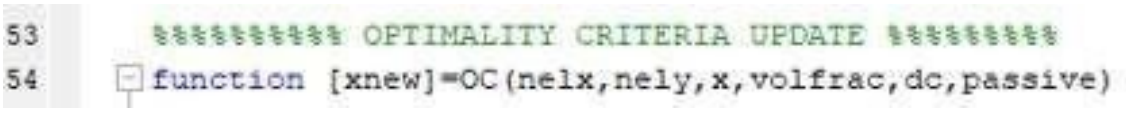


Figura 16: Fragmento del código con líneas modificadas.

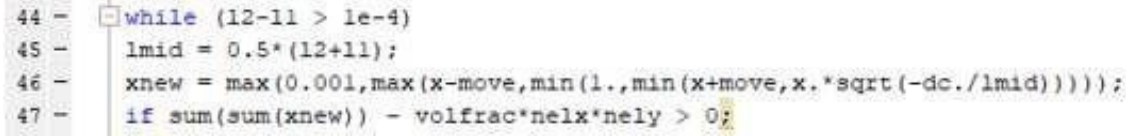


Figura 17: Fragmento del código original.

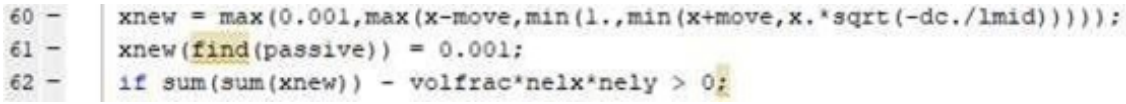


Figura 18: Fragmento del código con líneas modificadas.

Código Final.

```
%Práctica #3 Laboratorio Biomecánica Equipo 7

function toppract(nelx,nely,volfrac,penal,rmin)
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
%DECLARACION DE 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))=0.001;
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;
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) = -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,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
```

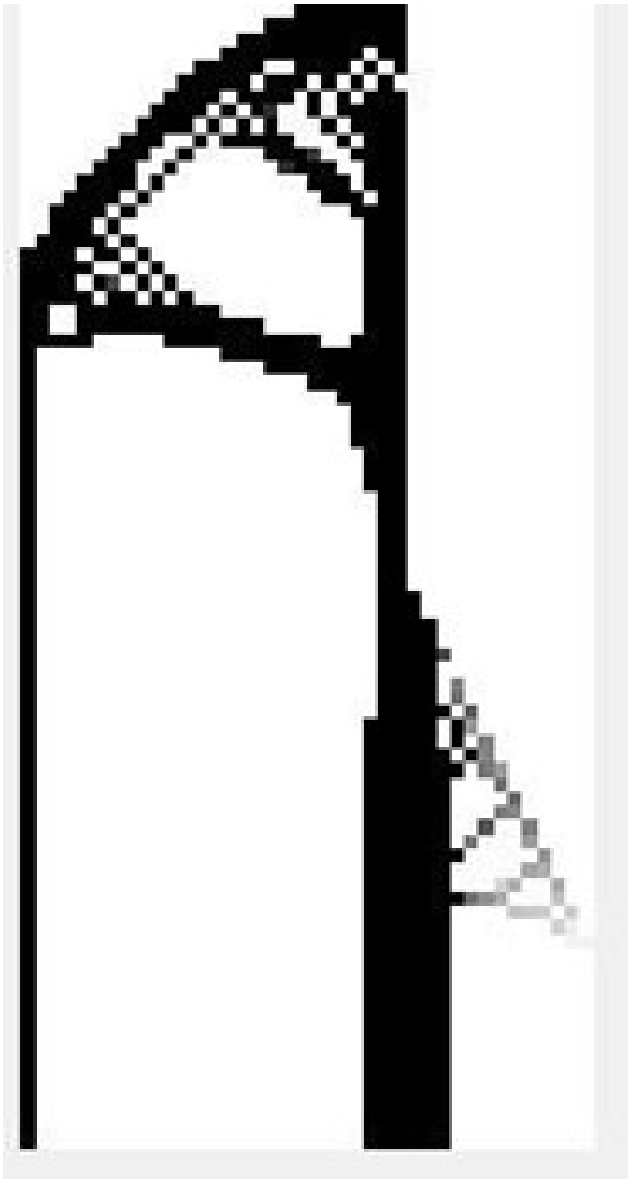
```
52
53      %%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%
54      function [xnew]=OC(nelx,nely,x,volfrac,dc,passive)
55 -    l1 = 0;
56 -    l2 = 100000;
57 -    move = 0.2;
58 -    while (l2-l1 > 1e-4)
59 -    lmid = 0.5*(l2+l1);
60 -    xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
61 -    xnew(find(passive)) = 0.001;
62 -    if sum(sum(xnew)) - volfrac*nelx*nely > 0;
63 -    l1 = lmid;
64 -    else
65 -    l2 = lmid;
66 -    end
67 -    end
68
69      %%%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%%%
70      function [dcn]=check(nelx,nely,rmin,x,dc)
71 -    dcn=zeros(nely,nelx);
72 -    for i = 1:nelx
73 -    for j = 1:nely
74 -    sum=0.0;
75 -    for k = max(i-round(rmin),1):min(i+round(rmin),nelx)
76 -    for l = max(j-round(rmin),1):min(j+round(rmin), nely)
77 -    fac = rmin-sqrt((i-k)^2+(j-l)^2);
```

```
78 -    sum = sum+max(0,fac);
79 -    dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
80 -    end
81 -    end
82 -    dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
83 -    end
84 -    end
85
86      %%%%%%%%%% FE-ANALYSIS %%%%%%%%%%%%
87      function [U]=FE(nelx,nely,x,penal)
88 -    [KE] = lk;
89 -    K = sparse(2*(nelx+1)*(nely+1),2*(nelx+1)*(nely+1));
90 -    F = sparse(2*(nely+1)*(nelx+1),5);
91 -    U = sparse(2*(nely+1)*(nelx+1),5);
92 -    for ely = 1:nely
93 -    for elx = 1:nelx
94 -    n1 = (nely+1)*(elx-1)+ely;
95 -    n2 = (nely+1)*elx+ely;
96 -    edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
97 -    K(edof,edof) = K(edof,edof)+x(ely,elx)^penal*KE;
98 -    end
99 -    end
100     % DEFINE LOADSAND SUPPORTS(HALF MBB-BEAM)
101 -   F(2*(nelx)*(nely+1)+2,1) = 1;
102 -   F(2*(nelx)*(nely+1)+(nely/4),2) = 1;
103 -   F(2*(nelx)*(nely+1)+(nely/2),3) = 1;
```

```
104 -    F(2*(nelx)*(nely+1)+(nely),4) = 1;
105 -    F(2*(nelx)*(nely+1)+(nely*1.2),5) = 1;
106 -    fixeddofs = 2*(nely+1):2*(nely+1):2*(nelx+1)*(nely+1);
107 -    alldofs = [1:2*(nely+1)*(nelx+1)];
108 -    freedofs = setdiff(alldofs,fixeddofs);
109      % SOLVING
110 -    U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
111 -    U(fixeddofs,:)= 0;
112
113      %%%%%%%%%% ELEMENT STIFFNESS MATRIX %%%%%%%%
114      function [KE]=lk
115 -    E = 1.;
116 -    nu = 0.3;
117 -    k=[ 1/2-nu/6 1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
118      -1/4+nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8];
119 -    KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
120      k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
121      k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
122      k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
123      k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
124      k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
125      k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
126      k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
```

Después de tener los cambios necesarios en el código se optó por darle un valor a toppract de toppract(40,80,0.2,3.0,0.5) en la ventana de comando de Matlab. Llegando a nuestra respuesta final la cual es esta:

Figura 19: Fig.16 Ventana de comando con el inicio de implementación del diseño con los parámetros dados [toppract(40,80,0.2,3.0,0.5)]



5. CONCLUSIONES.

Raúl Alexandro Vega López. 1853087.

Gracias a esta práctica aprendí un poco más sobre la estabilidad en estructuras estáticas, como lo fue el diseño de un panorámico a base de código en MatLAB. Después de varias modificaciones se consiguió una estructura estable.

Jesús Alberto Medina González. 1822858.

Para esta práctica se implementó el uso del software matlab ingresamos datos con los códigos y programación, con el cual resolvimos el problema planteado en el objetivo de la investigación, se resolvió el análisis y el diseño conforme a lo establecido en los criterios de evaluación, con esto aprendimos la importancia y el rol de una estructura de panorámico.

Yuliana Lizbeth Bravo Salazar. 1921580.

En esta práctica se trabajó con el método de MATLAB para encontrar una solución numérica computacional, en donde se analizaron diversos aspectos como análisis, implementación y la solución de un problema.

Jeiddy Michel Martínez Navéjar. 1851073.

Para esta práctica podemos concluir que el objetivo planteado, el cual, es presentar un reporte con la solución numérica computacional del problema de la simulación del desempeño mecánico de componentes mecánicos por medio del método de MATLAB, esto para desarrollar en el estudiante la capacidad de análisis, implementación y solución de un problema propuesto fue cumplido con éxito, ya que apoyando nos con herramientas de simulación logramos generar la optimización deseada así como hacer la observación necesaria de las diferencias que esta conlleva. Con esto podemos ampliar nuestros conocimientos con respecto a este laboratorio, así como de la utilización de simuladores como lo es MATLAB. Nuevamente realizando configuraciones en el código que se nos dio para trabajar en conjunto como equipo.

Andrik David Salas Carranza. 1992390.

Para esta actividad se realizó la observación del comportamiento estático de un panorámico para posteriormente realizar modificaciones que alteren el código, posteriormente a esto muestra una representación diferente al panorámico, con esto entendí un poco más la realización del código y como al hacer algunas modificaciones respecto a los valores dados se puede obtener un resultado totalmente distinto.

Juan Carlos Saldaña González. 1869591.

Para poder empezar y terminar esta actividad, primeramente tuvimos que contemplar un panorámico, el cual se encontraba estático, en el cual manipulamos y jugamos un poco con sus valores, llámense parámetros, lo cual nos dio la conclusión que con solo ase te una pequeña modificación podemos obtener un resultado totalmente distinto. Lo cual podíamos analizar y checar como se comportaba en diferentes condiciones

Joel Zúñiga Olvera. 1857780.

En esta actividad se observó el comportamiento estático de un panorámico y al cambiar ciertas partes del código, esta muestra ahora una representación del ya mencionado panorámico, con esto se analiza más a fondo dicho código y como con solo cambiar algunos valores o datos puede crear algo nuevo

Ana Sofía Limón González 1904075

En esta práctica se buscaba mediante el programa MATLAB mostrar la solución numérica computacional de la estructura que elegimos, en este caso una estructura panorámica. Al realizar modificaciones en los parámetros al código con el que se contaba fue posible lograr el objetivo que se tenía planteado.

Fred Raúl Peña Mata. 1866587.

Concluimos esta práctica adquiriendo un nuevo conocimiento acerca del comportamiento estático de un panorámico y descubrimos que pequeños detalles o modificaciones en su diseño pueden generar un gran cambio en el resultado final, así que después de probar la modificación del diseño con Matlab logramos conseguir una estructura bien estabilizada

6. REFERENCIAS.

[Pérez, F. (2022)] Pérez, F. Diseño y construcción de anuncios panorámicos, julio 2022.

[Hernández, M. (2022)] Hernández, M. ¿Cuánto cuesta construir la estructura de un panorámico?, Marzo 2022.