Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

- *Tarea: Optimización de una prótesis de pie.*

Maestro: Isaac Estrada.

Nombre:	**Matricula:**
Felipe Daniel Zamarripa Valdez	1755483
Miguel Angel Morales Arredondo	1987049
Severo Wenceslao Arreola Platas	1915391
Alvaro Alexis García Garza	1605716
Juan Angel Reyna Medrano	1669008

Materia: Laboratorio de Biomecánica   Hora: Viernes N5

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

Optimización de una Prótesis de Pie

Objetivo:

Proponer el diseño de una prótesis de pie para los diferentes estadios dentro de la marcha humana

Marco Teórico:

La locomoción proviene del fenómeno físico conocido como movimiento. Así, el movimiento siempre significa un cambio de posición en el espacio. Es el movimiento que permite que el sujeto (ya sea una persona o una máquina) se desplace y, además de adquirir otra posición, cambie de lugar. Es una posibilidad que sólo tienen los seres vivos y algunas máquinas o aparatos creados por el ser humano que, de todas maneras, deben contar con algún método de propulsión como motores o energía.

El conocimiento de la locomoción humana normal es la base del tratamiento sistemático y del manejo de la marcha patológica, especialmente cuando se usan prótesis y ortesis.

El caminar o andar de una persona, se define como la repetición de una serie de movimientos simultáneos, que desplazan el cuerpo sobre una línea de progresión deseada. Y al mismo tiempo mantienen una postura estable, soportando el peso corporal.

La movilidad libre de las articulaciones y el trabajo que desempeñan los músculos es importante para el éxito de esta tarea. Estos últimos deben actuar en el momento preciso y con la intensidad necesaria. La falta de ciertas acciones durante la marcha debe ser sustituida por otras, con el fin de mantener la estabilidad y la progresión deseada.

Ciclo de la marcha

El ciclo de la marcha comienza cuando el pie contacta con el suelo y termina con el siguiente contacto con el suelo del mismo pie. Los dos mayores componentes del ciclo de la marcha son: la fase de apoyo y la fase de balanceo (Figura 1). Una pierna está en fase de apoyo cuando está en contacto con el suelo y está en fase de balanceo cuando no contacta con el suelo.

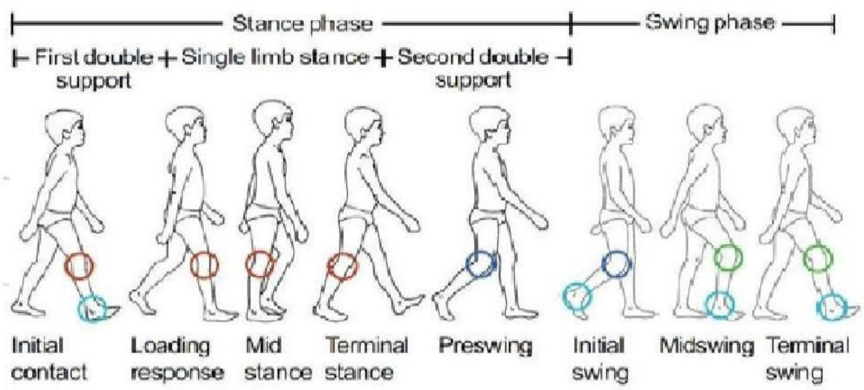


Figura 1. Fases en la Marcha Humana.

La longitud del paso completo es la distancia lineal entre los sucesivos puntos de contacto del talón del mismo pie. Longitud del paso es la distancia lineal en el plano de progresión entre los puntos de contacto de un pie y el otro pie (Figura 2).

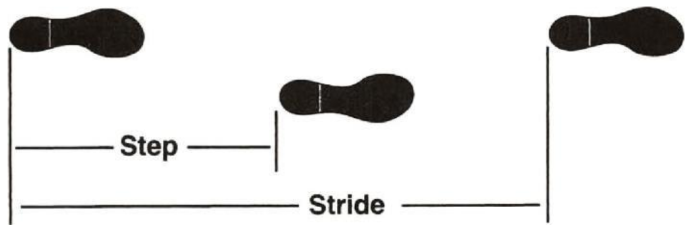
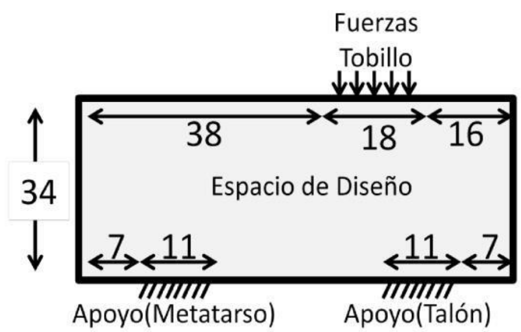


Figura 2. Longitud de Paso.

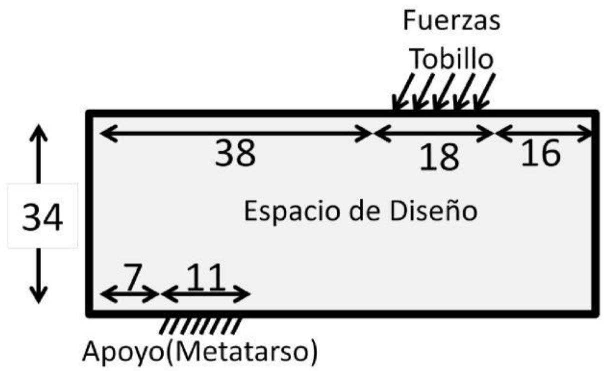
Desarrollo:

Para la realización de esta práctica se analizará el comportamiento de un solo pie dentro de las 3 fases de la marcha humana:

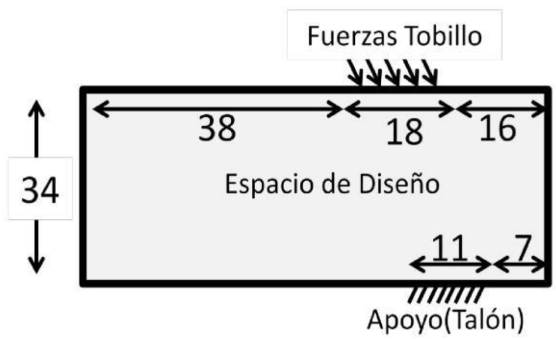
- Normal (El talón y área metatarsial son los apoyos, la fuerza se aplica sobre el tobillo con una fuerza de 500N)



- Despegue (El área metatarsial es el apoyo, la fuerza de 500N se aplica sobre el tobillo con un ángulo de 30º)



- Apoyo (El área del talón es el apoyo, la fuerza de 500N se aplica sobre el tobillo con un ángulo de 60º)

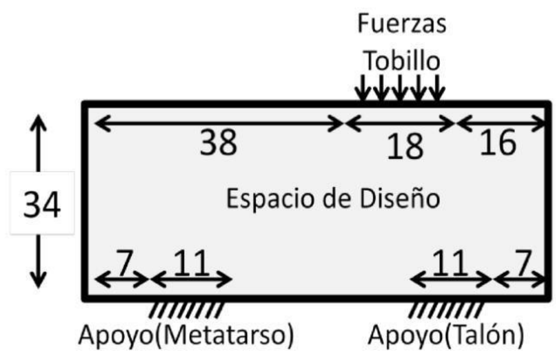
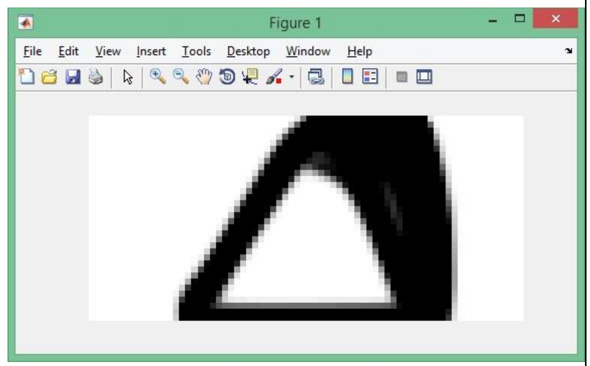
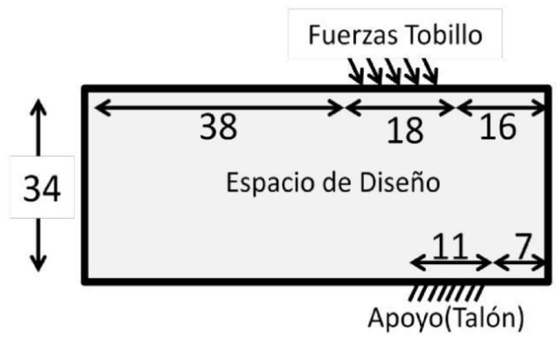
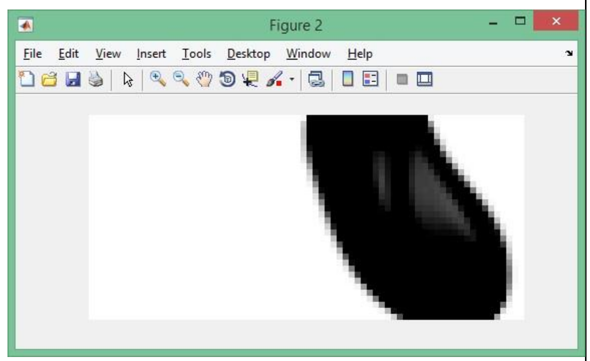
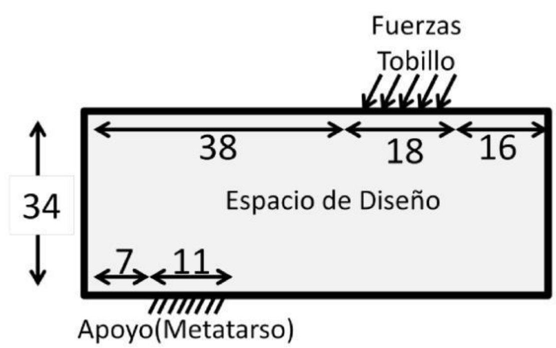
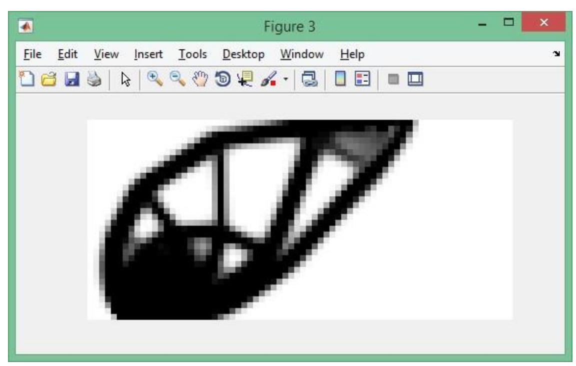


REPORTE

1. Realizar las simulaciones para los tres estados de las marcha y compare los resultados obtenidos.

2. Imprima cada uno de los códigos con sus modificaciones realizadas.

1.

2.

Código del ejercicio 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)) ...

' 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 = 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 del ejercicio 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)];
```

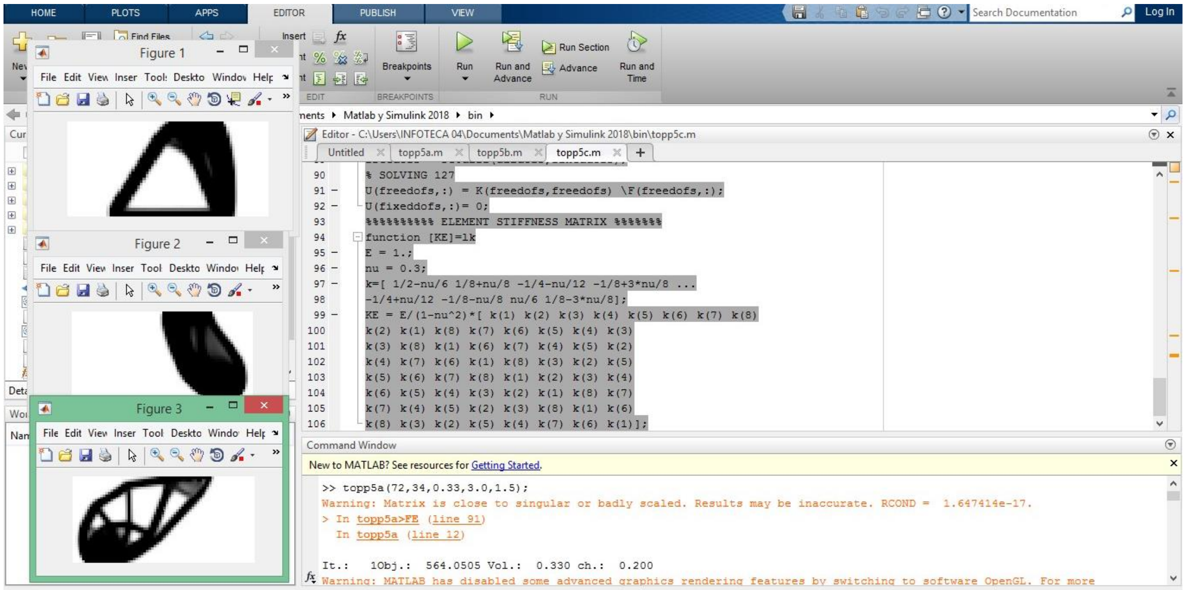
Código del ejercicio 3

```
%%%% 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 =
[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)];
```

Impresión de pantalla



Conclusiones:

Juan Reyna
En esta práctica seguimos utilizando Matlab para modificar y optimizar algún elemento o en este caso, modificarlo creando variantes para diferentes situaciones, esto nos muestra que una cosa puede modificarse las veces que sean necesarias para poder ajustarse a las necesidades de cada persona y los software son la mejor herramienta que tenemos para analizar y llegar al resultado que queremos sin la necesidad de armar un prototipo.

Severo Arreola
En esta práctica se reforzaron los conocimientos en Matlab y a su vez en el trabajo de los espacios funcionales dentro de este mismo software, siendo así que al desarrollar variantes con las cuales se busca tener otros diseños que de igual forma sean óptimos y funcionales y a su vez observar cómo es el comportamiento de los materiales en función de las fórmulas para el desarrollo de esta práctica.

Felipe Zamarripa
Al llevar a cabo esta práctica se aplicó el conocimiento adquirido a través de la carrera, ya que se pudo desarrollar fórmulas con parámetros específicos para poder perfeccionar modelos de diseño en su funcionamiento. Esto se lleva a cabo con un software de programación que se especializa en desarrollo de funciones matemáticas para aplicar en sistemas, lo conocemos como Matlab. Este software facilita la simulación tomando en cuenta el desarrollo matemático para su optimizacion y precisión.

Alexis Garza
Concluyo que al llevar acabo esta práctica, sobre el diseño de una protesis disimulado en cualquier software se hizo interpretar mediante resultados obtenidos a base de una forma adecuada para así llevar acabo em realizar la práctica y tener mejores resultados posibles mediante los parámetros leídos anteriormente, se efectuó conocimiento y así tener en cuenta lo importante que esto del diseño y desarrollo matemático.

Miguel Morales
A lo largo de este reporte pudimos hacer uso de razón y nos percatamos que se puede usar Matlab para generar un análisis de elemento finito para objetos de ámbito simple y que se pueden usar para diferentes casos, además de generar un buen soporte que nos ayudará mucho en este caso.