



UANL

# Universidad Autónoma de Nuevo León

## Facultad de Ingeniería Mecánica y Eléctrica



*Práctica #5: Optimización de una prótesis de pie.*

### *Equipo 6*

*Datos del equipo:*

<b>Nombre</b>	<b><u>Matrícula</u></b>	<b><u>Carrera</u></b>
Andrés Anaya Hernández	1914471	IMC
Carlos Eduardo Rivera López	1897545	IMC
César Armando Luna	1844920	IMC
Víctor Adrián Higuera Vázquez	1876474	IMC
Gabryiel Bailon Avila	1869828	IMC

*Catedrático: M.C. Yadira Moreno Vera*

Fecha de entrega: 09 de Noviembre de 2022

Hora: V2

Brigada: 214

Semestre: Agosto - Diciembre 2022

# Contenido

<b>Propuesta de diseño de la geometría, alcances y limitaciones</b>	<b>3</b>
<b>Programación</b>	<b>5</b>
Normal	5
Despegue	7
Apoyo	11
<b>Resultados de la optimización</b>	<b>15</b>
<b>Conclusiones</b>	<b>17</b>
<b>Referencias</b>	<b>18</b>

## Estado del arte.

Para encontrar las primeras piezas protésicas que se conocen, debemos remontarnos a los años 950 y 750 a.C. en el antiguo Egipto. Estas piezas, encontradas en una momia enterrada en Egipto, estaban hechas de madera y cuero y simulaban un dedo gordo del pie. Gracias a esta invención, los egipcios, además de mejorar la estética del pie, podían caminar con normalidad y sin perder el equilibrio.

El siguiente hallazgo más antiguo se encontraba en la ciudad de Capua, en Italia. Data del año 300 a.C. y constituía la prótesis de una pierna realizada en madera y recubierta con hierro y bronce. Lamentablemente, la pieza original se destruyó durante los bombardeos de la Segunda Guerra Mundial.

En la Edad Media, la protésica, al igual que muchas otras ciencias, tuvo un desarrollo prácticamente nulo. Solo destaca el gancho de mano y la pierna de madera y metal, pero era algo más decorativo que funcional. Estas prótesis normalmente solo se utilizaban en las batallas cuando algún soldado perdía una extremidad.

En el año 1800, James Potts creó una pierna de madera que incluía la articulación de la rodilla compuesta de acero y un pie también articulado por tendones de tripa de gato.

## Nombre y definición de la forma geométrica

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.

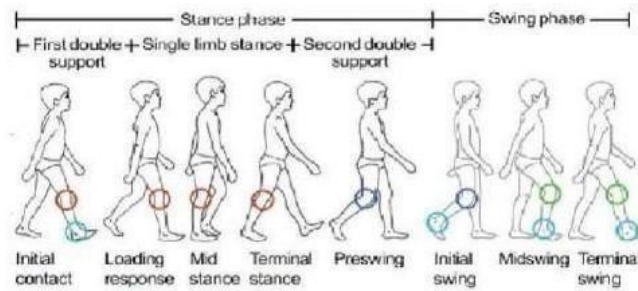
El caminar o andar de una persona, se define como la repetición de una serie de movimientos simultáneos, que desplazan el cuerpo sobre una línea de progresión deseada. Y al mismo tiempo mantienen una postura estable, soportando el peso corporal.

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.

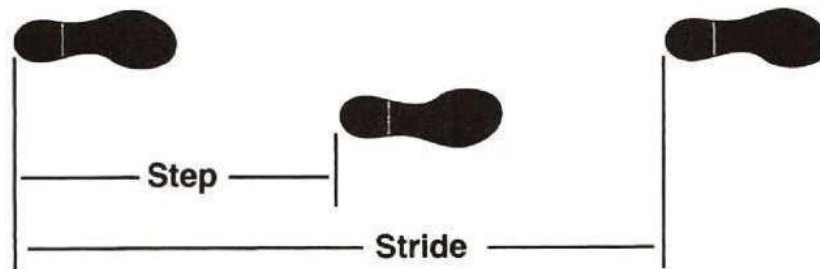
### Ciclo de la marcha

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 5.1).

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



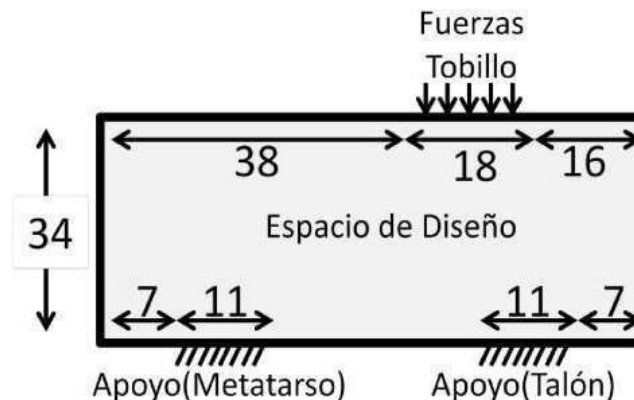
La longitud del paso completo es la distancia lineal entre los sucesivos puntos de contacto del talón del mismo pie. Longitud del paso es la distancia lineal en el plano de progresión entre los puntos de contacto de un pie y el otro pie (Figura 5.2).



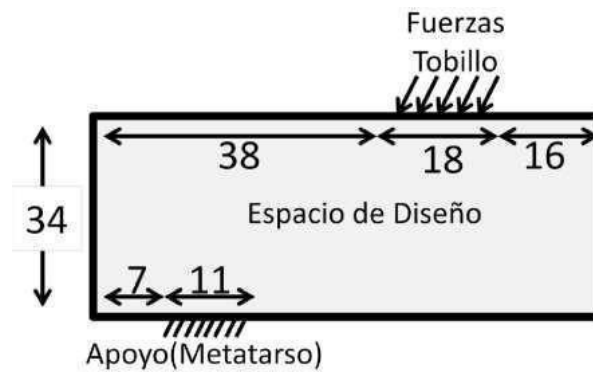
## Propuesta de diseño de la geometría, alcances y limitaciones

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

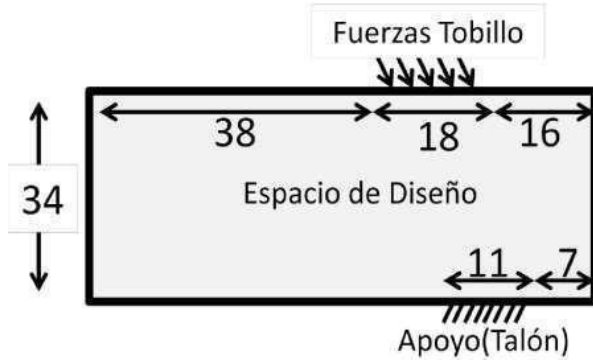
- Normal (El talón y área metatarsial son los apoyos, la fuerza se aplica sobre el tobillo con una fuerza de 500N )



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



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



# Programación Normal

```

%*** 49 LINE TOPOLOGY OPTIMIZATION CODE BY OLESENBERG, OCTOBER 1999 ***
function [topp,nelx,nely,volfrac,penal,xmin] =
% 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] = 1k;
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
sum=0.0;
for k = max(1-round(rmin),1):min(1+round(rmin),nelx)
for l = max(1-round(rmin),1):min(1+round(rmin), nely)
fac = rmin-sqrt((1-k)^2+(1-l)^2);
sum = sum+max(0,fac);
dcn(1,l) = dcn(1,l) + max(0,fac)*x(1,k)*dc(1,l);
end
end
dcn(1,l) = dcn(1,l)/(x(1,l)*sum);
end
%***** FE-ANALYSIS *****
function [U]=FE(nelx,nely,x,penal)
[KE] = 1k;
R = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
R = sparse(2*(nelx+1)*(nelx+1),5);
U=sparse(2*(nelx+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];
R(edof,edof) = R(edof,edof) + x(ely,elx)*penal*KE;
end
end
% DEFINE LOADSAND SUPPORTS (HALF HSB-BEAM)
F(3222,1) = -1;
F(3782,2) = -1;
F(2662,3) = -1;
F(2942,4) = -1;
F(3802,5) = -1;
%***** ELEMENT STIFFNESS MATRIX *****
function [KE]=1k
E = 1.;
nu = 0.3;
k1= 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(5) k(7) k(6) k(5) k(4) k(3)
k(3) k(5) 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)];

```

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

d

% 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;

```

## Despegue

```

%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESGORCHED, OCTOBER 1999
function [x] = OC(nelx,nely,volfrac,penal,rmin);
% INITIALIZE
loop = 0;
change = 0;
% 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
sum = sum(x(ely,elx)^1.5);
dcon(j,i) = dcon(j,i) + max(0,fac)*x(1,k)*dc(1,k);
end
dcon(j,i) = dcon(j,i)/(K(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 HSB-BEAM)
F(322,1) = -1;
F(378,2) = -1;
F(262,3) = -1;
F(294,4) = -1;
F(350,5) = -1;
fixeddofs = [360;2*(nely+1);1260];
alldofs = [1;2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);
% SOLVING I27
U(freedofs,:) = F(freedofs,freedofs) \ F(freedofs,:);
U(fixeddofs,:) = 0;
%***** ELIGIBLE 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)];

```

[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

d

% 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)];

## Apoyo

```

% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESENHED, OCTOBER 1999 %
function [topp] = topop(nelx,nely,volfrac,penal,rmin);
% INITIALIZE
x(l: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],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
% FILTERING OF SENSITIVITIES
[dc] = check(nelx,nely,rmin,x,dc);
% DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
for k = max(1-round(rmin),1):min(1+round(rmin),nelx)
for l = max(2-round(rmin),1):min(2+round(rmin), nely)
fac = rmin-sqrt((1-k)^2+(2-l)^2);
sum = sum+max(0,fac);
dcm(j,l) = dcm(j,l) + max(0,fac)*x(l,k)*dc(l,k);
end
end
dcm(j,l) = dcm(j,l)/(k(j,l)*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 LOADS AND SUPPORTS (HALF HSB-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)+620];
alldofs = [1:2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);
% SOLVING 127
U(freedofs,1) = K(freedofs,freedofs) \ F(freedofs,1);
U(fixeddofs,1) = 0;
% ELIGIBLE 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(5) k(1) k(6)
k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
end

```

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
d
% 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

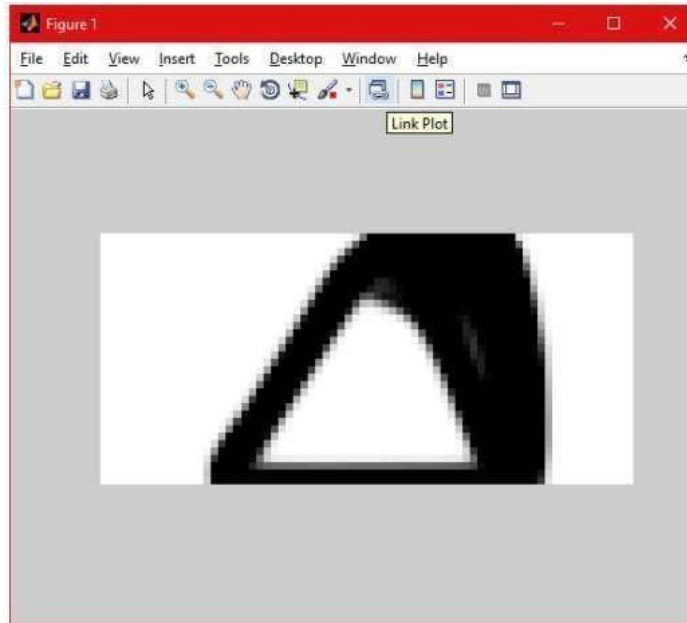
%%%%%%%%%%%% FE-ANALYSIS %%%%%%%%%%%%% function
[U]=FE(nelx,nely,x,penal) [KE] = lk;
K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
F = sparse(2*(nely+1)*(nelx+1),5); U =sparse(2*(nely+1)*(nelx+1),5);
for ely = 1:nely for elx = 1:nelx n1 = (nely+1)*(elx-1)+ely; n2 =
(nely+1)* elx +ely;
edof = [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)

```

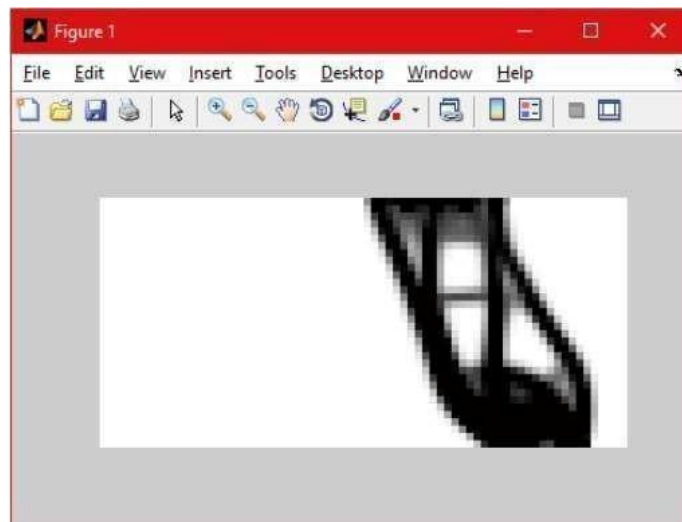
k(7) k(4) k(5) k(2) k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)  
k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4) k(6) k(5) k(4) k(3)  
k(2) k(1) k(8) k(7) k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)  
k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];

## Resultados de la optimización

Normal

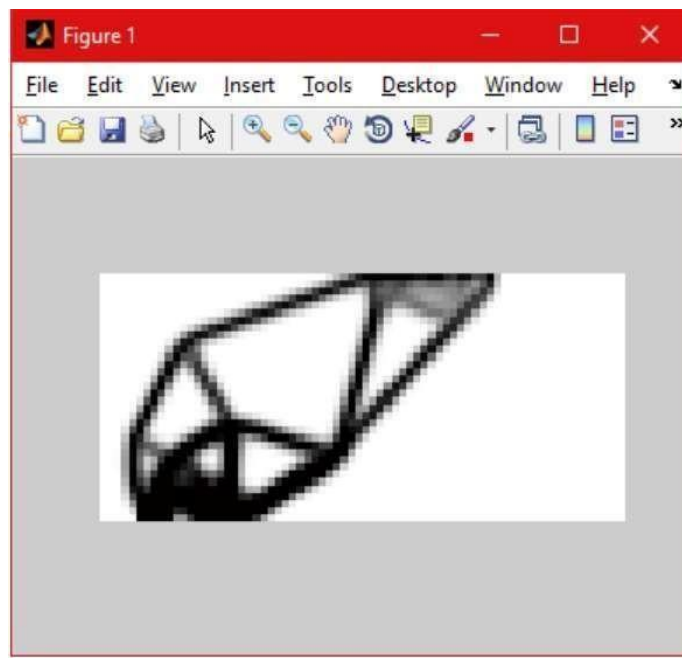


Despegue



Apoyo





## Conclusiones

### **César Armando Luna Zapata 1844920 IMTC**

Como conclusión en base a la comparación me pude dar cuenta que como tienen la misma carga, pero a diferentes ángulos, por lo que la simulación muestra en cuanto esta normal después en despegue y ya en apoyo, ya que las fuerzas se dividen de diferente manera.

### **Andrés Anaya Hernández 1914471 IMTC**

En esta práctica se llevó a cabo una investigación acerca del diseño de piezas destinadas para el uso en prótesis y se llevó a cabo una propuesta y su implementación con código y se expusieron los resultados obtenidos. En cuanto a las pruebas realizadas se observó como es que se comportan las fuerzas que actúan en la pieza y como es que a partir de la carga estás llevan a cabo un comportamiento específico.

### **Carlos Eduardo Rivera López 1897545 IMTC**

Para esta práctica investigamos más a fondo sobre sobre la implementación de prótesis en el mundo actual, así mismo aprendí cómo es que las cargas que se aplican en el apoyo y despegue son iguales solo cambia su ángulo. Esto lo pudimos comprobar en nuestro código de 99 líneas y su ejecución en la simulación.

### **Víctor Adrián Higuera Vázquez 1876474 IMTC**

En esta práctica aprendí que en el apoyo normal se le aplica una carga de 500N, en la de despegue igual se aplica una carga de 500N, pero con un ángulo de 30° y en la de apoyo al igual una carga de 500N, pero a un ángulo de 60°. Con lo anterior aprendí acerca de cómo introducir en Matlab lo anteriormente mencionado modificando el código que siempre utilizamos de 99 líneas. Al igual manera me di cuenta de la simulación de cada una de ellas.

## **Gabryiel Bailon Ávila 1869928 IMTC**

La implementación de prótesis ha mejorado notablemente la vida de personas con amputaciones totales o parciales de miembros por lo cual significa un gran avance tanto medico como mecánico. El uso de este código puede ayudar a visualizar los apoyos diferentes que tiene esta pieza y trabajar en conjunto para realizar una prótesis solo se necesita la implementación en un lugar adecuado para que esté tipo de trabajo puedan trabajar de manera adecuada

## **Referencias**

- 99 Line Topology Optimization Code – O. Sigmund,  
Department of Solid Mechanics, Building 404, Technical University of  
Denmark, DK-2800 Lyngby, Denmark.
- O. Sigmund, Department of Solid Mechanics, Building 404, Technical University of  
Denmark, DK-2800 Lyngby, Denmark. El código puede ser descargado desde la  
página del autor: <http://www.topopt.dtu.dk>