

# o León

## 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:

Nombre	<u>Matrícula</u>	<u>Carrera</u>
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: <u>09 de Noviembre de 2022</u>

Hora: V2 Brigada: 214

Semestre: <u>Agosto - Diciembre 2022</u>

### Contenido

Propuesta de diseño de la geometría, alcances y limitaciones	
Programación	į
Normal	5
Despegue	7
Apoyo	11
Resultados de la optimización	15
Conclusiones	17
Referencias	18

#### 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. Databa 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.

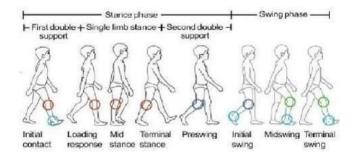
El caminar o andar de una persona, se define como la repetición de una serie de movimientos simultáneos, que desplazan el cuerpo sobre una línea de progresión deseada. Y al mismo tiempo mantienen una postura estable, soportando el peso corporal.

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.

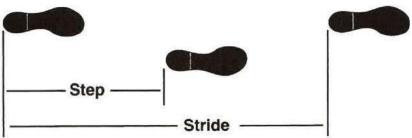
#### Ciclo de la marcha

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



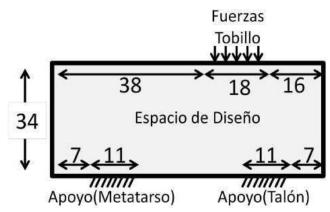
La longitud del paso completo es la distancia lineal entre los sucesivos puntos de contacto del talón del mismo pie. Longitud del paso es la distancia lineal en el plano de progresión entre los puntos de contacto de un pie y el otro pie (Figura 5.2).



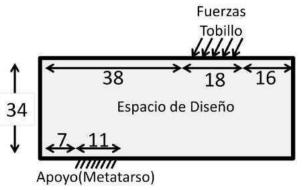
## Propuesta de diseño de la geometría, alcances y limitaciones

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

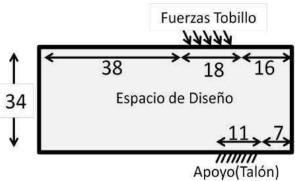
 Normal (El talón y área metatarsial son los apoyos, la fuerza se aplica sobre el tobillo con una fuerza de 500N)



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



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



#### **Programación**

#### Normal

```
FIGHTIATION CODE BY CLESIGNORD, OCTOBER 1989 %%
                                                                                                                                                                         % FILTERING OF SENSITIVITIES
[dc] = check(nelx,nely,rmin,x,do);
% DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
function topp (nelw, nely, volfrac, penal, zmin) ;
                                                                                                                                                                    x(linely,linelx) = volfrec;
| dop = 0;
| change = 1;
| t START ITERATION
| while change > 0.01
| dop = loop + 1;
| xold = x;
| t FE-RHALVSIS
                                                                                                                                                                   % FRITE RESULTS
change = nax(max(abs(k-xold)));
disp(['Ic.i' sprintf('%41',loop) 'Obj.i' sprintf('%10.4f',c) ...
' Vol.i' sprintf('%4.3f',aum(sum(x))/(nelk'nely)) ...
' ch.i' sprintf('%4.3f',change )])
% FLOT DEBSTITES
                                                                                                                                                                       colornap(gray); imagesc(-x); axis equal; axis tight; axis off:pause(le-6); and
                                                                                                                                                                         ******* OPTIMALITY CRITERIA UPDATE ********
         [KE] = 1k;
                                                                                                                                                                        Tunction (News)=OC(nek), nely,x,volfrec,do)

11 = 0; 12 = 100000; nove = 0.2;

|while (13-11) = 10-4;

|Inid = 0.5*(12*11);

|xnew = max(0.001,max(x-move,min(i.,min(x-move,x.*sqrt(-dc./Inid)))));

|f sum(sum(xnew)) = volfrac*nelx*nely > 0;

11 = inid;
    e = 0.;
for ely = 1:hely
          function [dom]"effect(neix,nely,rmin,x,dc)
dom*reros(nely,nelx);
for i = linelx
for j = linely
sun=0.07
          end
 A FILTERING OF SENSITIVITIES
[dc] = check(nelx,nely,smin,x,do);
A DESIGN UPDATE BY THE OFFINALITY CRITERIA METHOD
swn=010;
 aum=0f0;
for k = max(1-round(rmin),1):min(1-round(rmin),nelx)
for i = max(3-round(rmin),1):min(3-round(rmin), nely)
fac = rmin-sqrt((i-k)^2+(3-1)^2);
                                                                                                                                                             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;
 sum = sum+max(0,fac);
 den(j,i) = den(j,i) + max(0,fac)*x(l,k)*de(l,k);
                                                                                                                                                              A DEFINE LOADSAND SUPPORTS (HALF HBB-BEAM)
                                                                                                                                                            A DEFIDE LOADGRAD SUPFORTS (MALF HIB-BEAM)
F(3782,2) = -1;
F(3782,2) = -1;
F(2662,4) = -1;
F(3802,5) = -1;
F(3802,5) = -1;
fixeddofs = union([560:2*(nely+1):1260],[3820:2*(nely+1):4620]);
fixeddofs = fiz2*(nely+1)*(helx+1);
fixedofs = secdiff(alidofs,fixeddofs);
* Solv/Hus [360]
 don(3, 4) = don(3, 4) / (x(3, 4) * sum);
end
[HZ] = 1K:
R = 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 elx = 1:nelx
for elx = 1:nelx
n = (nely+1)*(elx+1)+ely;
n2 = (nely+1)*(elx+1)+ely;
n3 = (nely+1)*(elx-1)=2*n2+1,2*n2+3;2*n2+3;2*n1+1; 2*n1+2];
K(edof = (2*n1-1; 2*n1; 2*n2+1); 2*n2+3;2*n2+3;2*n1+1; 2*n1+2];
K(edof, edof) = K(edof, edof) + X(ely, elx)*pensl*KE;
                                                                                                                                                         ke[ 1/2-nu/8 - 1/4-nu/8 - 1/4-nu/2 - 1/8-3*nu/8 - - - - 1/4-nu/8 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8-1 - 1/8
 * DEFINE LOADSAND SUPPORTS (HALF HEB-BEAM)
   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;
   en d en d en
   % 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(['lt.:'
   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) I1
= 0; I2 = 100000; move = 0.2;
while (12-11 > 1e-4) Imid
= 0.5*(12+11);
xnew = max(0.001, max(x-move, min(1., min(x+move, x.*sqrt(-dc./lmid)))));
if sum(sum(xnew)) - volfrac*nelx*nely > 0; I1 = Imid; else
12 = Imid;
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

```
* DESIGN OFFDRIE 85 THE OFFTRALITE
[x] = OC(nelx,nely,x,volfrac,do);
* PRINT RESULTS
channe
  ANNA A SS LINE TOPOLOGY OFFINITATION CODE BY OLESIGMUND, OCTOBER 1899 NAM
function topp (nels, nely, volfrac, penal, zmin);
                                                                                                                                                                              change = max(max(abs(x-xold)));
 % HINTIALIZE

g(A:mely,kinelx) = volfrac;

loop = 0;

change = 1;

% START HINATION

While change > 0.01

loop = loop + 1;

gold = x;
                                                                                                                                                                              colormap(gray); imagesc(-K); axis equal; axis right; axis off;pause(le-
end annual colormap of melan, mely, x, volfrac, do)

11 = 0; 12 = 100000; move = 0.2;

while (12-1) > 1e-0;

imid = 0.5*(12-11);
xnew = max(0.901,max(x-move,min(1.,min(x+move,x.*sqrt(-do./lmid))));
if sun(sum(xmow)) = volfrac*nelx*nely > 0;

11 = lmid;
else
12 = lmid;
end
                                                                                                                                                                                        olormap(gray); imagesc(-x); axis equal; axis tight; axis off;pause(le-6);
           [U] =FE(nelx,nely,x,penal);
                                                                ITIVITY ANALYSIS
   c = 0.;
for ely = linely
for elx = linelx
nl = (nely+1)^(elx-1)+ely;
nl = (nely+1) * elx +ely;
                    n2 = (ncly=1)* clx +cly;

dc(cly,=lx)=(c,:
for l=1)5

Ue = U({2*ni-1:2*ni; 2*nz-1:2*nz; 2*nz+1; 2*nz+2; 2*ni+1:2*ni+2},1);

c = c + x(cly,=lx)*penal*Ue**XE**Ue;

dc(cly,elx) = dc(cly,elx)*penal*Xe**(cly,elx)*(penal-1)* Ue**XE**Ue;
                                                                                                                                                                                 function [don]=check(nelx,nely,rmin,x,dc)
don=zeros(nely,nelx);
                                                                                                                                                                        demwestor(nelly,nelk);
for i = linelx
for j = linelx
for j = linely
sum=0.0;
for k = nex(i-round(rmin),l):min(i=round(rmin),nelk)
for l = nex(j-round(rmin),l):min(j=round(rmin), nelk)
edof = [2*nl-1; 2*nl; 2*nl-1; 2*nl; 2*nl+1; 2*nl+2];
g(edof,edof) = g(edof,edof) + g(ely,elx)*penal*gE;
end
  [dc] = check(nelk, nely, rmin, k, dc);
  don(j,1) = don(j,1) + max(0,fac)*x(1,K)*dc(1,K);
  den(3,1) = den(3,1)/(x(3,1)*sum);
                                                                                                                                                                           DEFINE LOADSAND SUPPORTS (HALF HBB-BEAM)
                                                                                                                                                                         a Define LOADSAND SUPPORTS (HALF HDB-BEA
f(3222.1) = -1;
f(3762.2) = -1;
f(2662.3) = -1;
f(2662.4) - -1;
f(3602.5) - -1;
fixediofs = [56012*(nely+1):1260];
alidofs = [1:2*(nely+1)*(nelx+1)];
freedofs = serdiff(alidofs,fixeddofs);
accurate (12.2)
  function [U]=FE(nelx,nely,x,penal)
[RE] = lk;
E = 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 = linely
for elx = linelx
                                                                                                                                                                        % SOLUTION 127
U(freedofs,) = W(freedofs,freedofs) \F(freedofs,i):
U(freedofs,i) = 0;
U(freedofs,i) = 0;
Elementon [RZ]=1k
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  1/2-3*nu/8 !...
-1/4-nu/12 -1/8-nu/8 nu/6  1/2-3*nu/8 !...
-1/4-nu/12 -1/8-nu/8 nu/6  1/2-3*nu/8 |
RZ = F(1-nu^2): (x(1) k(2) k(3) k(4) k(6) k(6) k(7) k(8)
-1/2 | x(1) k(8) k(7) f(6) k(5) f(6) k(7) k(8)
  end

**neffile Loadsand Supports(HALF Mem-BRAN)

**F(322,1) = -1;

**F(3782,2) = -1;

**F(2682,3) = -1;

**F(2682,3) = -1;

**F(3502,5) = -1;

**fixedors = [5602*(nely+1)*11260];

**elidors = [120*(nely+1)*(nelx+1)];

*freedors = setdiff(alldofs,fixeddofs);

**Accurate Accurate Acc
                                                                                                                                                                        [U]=FE(nelx,nely,x,penal);
   % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
   [KE] = Ik; 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;
   en d en d en
   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(['lt.:'
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) I1
= 0; I2 = 100000; move = 0.2;
while (12-11 > 1e-4) Imid
= 0.5*(I2+I1);
xnew = max(0.001, max(x-move, min(1., min(x+move, x.*sqrt(-dc./lmid)))));
if sum(sum(xnew)) - volfrac*nelx*nely > 0; I1 = Imid;
else
12 = Imid:
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
I = 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 = \text{sparse}(2^*(\text{nelx+1})^*(\text{nely+1}), 2^*(\text{nelx+1})^*(\text{nely+1}));
F = \text{sparse}(2^{*}(\text{nely+1})^{*}(\text{nelx+1}),5); U = \text{sparse}(2^{*}(\text{nely+1})^{*}(\text{nelx+1}),5);
for ely = 1:nely for elx = 1:nelx n1 = (nely+1)*(elx-1)+ely; n2 = (nely+1)*(elx-1)+ely
(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 -1/4
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**

```
State A 95 LINE TOPOLOGY SPITHIZATION CODE BY

function topp (nelk, nely, volfrac, penal, min) (
                                                                                                                                                                                                                                                                                           % PRINT RESULTS
dasp(['fit.i' sprintf('%i'.loop) 'Obj.i' sprintf('%iO.4f'.o) ...
'Vol.i' sprintf('%i'.loop')
'ch.i' sprintf(
     x(linely,linels) = volfrac;
    x(1:nely,:nelx) = v
loop = 0;
change = 1,;
% START ITERATION
While change > 0.01
loop = loop + 1;
xold = x;
% START VALUE

                                                                                                                                                                                                                                                                                             ******* OPTIMALITY CRITERIA UPDATE *******
                                                                                                                                                                                                                                                                                  (U) =FE(nelx,nely,x,penal);
                                                         CTION AND SENSITIVITY AMALYATS
       % GRECTIVE TUNCTION AND SENSITIVITY ANALYSIS
(RE) = 18;
c = 0.;
for ely = linely
for elx = linelx
    nl = (nely+1)*(elx-1)+ely;
    n2 = (nely+1)* elx +ely;
    dc(ely,elx)=0.;
for i=15
    ue = U((2*nl-1;2*nl; 2*n2+1;2*n2; 2*n2+1; 2*n2+2; 2*n1+1;2*n1+2],1);
                                                                                                                                                                                                                                                                                            else
12 = lmid;
                                                                                                                                                                                                                                                                                             ******* HESH-INDEPENDENCY FILTER ********
                                                                                                                                                                                                                                                                                      dom-wheel DECPENDENCY FILTER excession of function (dom-wheek (nelx, nely, min, x, do) dom-weres(nely, nelx);

For i = inelx | for j = inely | sum-0.0;

For k = max(1-round(rmin),1):min(i=round(min), nelx) | for k = max(1-round(min),1):min(j=round(min), nely) | for c = min-seqt((i=x)-2+(j-1)-2);

sum = sum-max(0,fac);
                                               c = c + x(ely,elx) penal*Ue'*XE*Ue;
dc(ely,elx) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)* Ue'*ME*Ue;
     end

* FILTERENG OF SENSITIVITIES

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

* DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
   | for k = max(i-round(rmin),1);min(i+round(rmin),nelx)
| for l = max(3-round(rmin),1);min(3-round(rmin), nely)
                                                                                                                                                                                                                                                                         edof = {2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1;2*n2+2;2*n1+1; 2*n1+2}; 

£(edof,edof) = £(edof,edof) + x(e1y,e1x)*pena1*£;
    fac = rmin-sqrt((1-k)"2+(2-1)"2);

sum = sum-max(0,fac);

dcn(j,1) = dcn(j,1) + max(0,fac)"x(1,k)"dc(1,k);
                                                                                                                                                                                                                                                                         end
& DEFINE LOADSAND SUPPORTS (HALF MBB-BEAM)
                                                                                                                                                                                                                                                                        A DETIME COADSAIN SUPPORTS (HALF MES-BEF F(3222,1) = -1; F(3762,2) = -1; F(2662,3) = -1; F(2662,3) = -1; F(3502,5) = -1; F(3502,5) = -1; F(3502,5) = -1; f(3602,5) = -1; f(360
    end
end
den(j,1) = den(j,1)/(x(j,1)*sum):
     ******** FE-ANALYSIS ********
     function [U]=FE(nels,nely,x,pensi)
     (HE) - 1%;
   [RE] - 1kr
K = sparse(2*(nelx+1)*(nelx+1), 2*(nelx+1)*(nelx+1));
F = sparse(2*(nely+1)*(nelx+1),5);
U =sparse(2*(nely+1)*(nelx+1),5);
for ely = 1:nely
[for ely = 1:nely
                                                                                                                                                                                                                                                                          % SOLVING 127
U(freedofs,i) = K(freedofs,freedofs) \F(freedofs,i);
U(fixeddofs,i) = 0;
%%%%%%%% ELHKNT SIFFNESS HATRIX *******
                                                                                                                                                                                                                                                                   function [KE]=lk

E = 1.;

nu = 0.3;
    nl = (nely+1)*(elx-1)+ely;
nl = (nely+1)* elx +ely;
edof = (2*ni-1; 2*ni 2*n2+1; 2*n2+3*n2+1;2*n2+2;2*ni+1; 2*ni+2];
K(edof,=sof) = K(edof,=sof) + x(ely,elx)*penai*KE;
                                                                                                                                                                                                                                                                         k=[ 1/2-nu/6 1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
                                                                                                                                                                                                                                                                      end

b DEFINE LOADSAUD SUPPORTS (HALE NOBE-BEAM)

F(3222,1) = -1;
F(3762,2) = -1;
F(3662,3) = -1;
F(3961,4) = -1;
F(3903,5) = -1;
       >> top5_2(72,54,.20,5.0,1.5)
Weening: Matrix is close to
                           40b3.: 290,6244 Vol.: 0.200 db.: 0.200
                          Sobj: 226,9078 Vol.: 0.200 ch:: 0,200

- Nacria is diome to election of bedly scaled, Results may be imacourate. RCORD = 5.115055=18.
                          Sobj.: 174,9473 Vol.: 0,200 ch.: 0,135
Sobj.: 174,9473 Vol.: 0,200 ch.: 0,135
       while change > 0.01
       loop = loop +
       1: xold = x:
       % FE-ANALYSIS
       [U]=FE(nelx,nely,x,penal);
       % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
       [KE] = Ik; c = 0.; for ely =
        1:nely for elx = 1:nelx n1
       = (nely+1)*(elx-1)+ely; n2
```

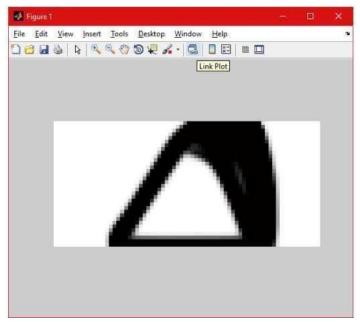
```
= (nely+1)^* elx + ely;
dc(ely,elx)=0.; for i=1:5
Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1; 2*n2+2;
2*n1+1;2*n1+2],i);
c = c + x(ely,elx)^penal^*Ue'^*KE^*Ue; dc(ely,elx)
= dc(ely,elx)-penal*x(ely,elx)^(penal-
1)* Ue'*KE*Ue;
en d en d en
d
% FILTERING OF SENSITIVITIES
[dc] = check(nelx,nely,rmin,x,dc);
% DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
[x] = OC(nelx,nely,x,volfrac,dc);
% PRINT RESULTS
change = max(max(abs(x-xold))); disp(['lt.:'
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)
11 = 0; 12 = 100000; move = 0.2; while (12-
11 > 1e-4
Imid = 0.5*(I2+I1);
xnew = max(0.001, max(x-move, min(1., min(x+move, x.*sqrt(-dc./lmid))))); if
sum(sum(xnew)) - volfrac*nelx*nely > 0;
I1 = Imid; else
```

```
12 = Imid;
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
I = 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] = Ik;
K = \text{sparse}(2^*(\text{nelx+1})^*(\text{nely+1}), 2^*(\text{nelx+1})^*(\text{nely+1}));
F = \text{sparse}(2^{*}(\text{nely+1})^{*}(\text{nelx+1}),5); U = \text{sparse}(2^{*}(\text{nely+1})^{*}(\text{nelx+1}),5);
for ely = 1:nely for elx = 1:nelx n1 = (nely+1)*(elx-1)+ely; n2 = (nely+1)*(elx-1)+ely
(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)
```

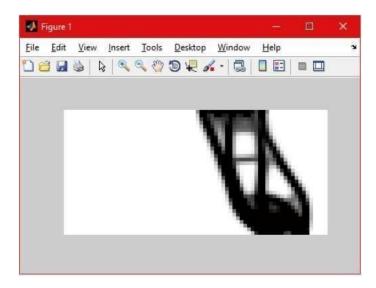
- k(7) k(4) k(5) k(2) k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
- k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4) k(6) k(5) k(4) k(3)
- k(2) k(1) k(8) k(7) k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
- k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];

#### Resultados de la optimización

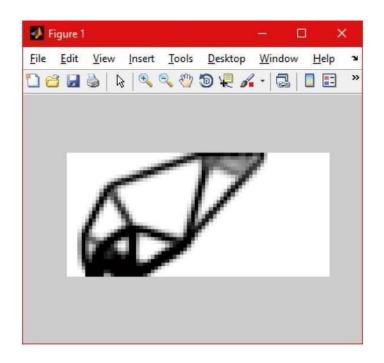
Normal



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