UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FIME
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Laboratorio de Biomecánica

Práctica 5: Optimización de una prótesis de pie

INSTRUCTOR@: YADIRA MORENO VERA

MARTES V1 BRIGADA 204

Equipo #2:

Matricula	Nombre	Carrera
1991908	Covarrubias Becerril Brian Eduardo	IMTC
1991966	Luis Javier Rodríguez Vizcarra	IMTC
1925324	Rubén Cantú Espinoza	IMTC
1926098	Ángel Fernando Mexquitic Rodríguez	IMTC
1895460	Elmer Javier Delgadillo García	IMTC
1847932	Francisco Javier Velazco Rivas	IMTC

23 de octubre del 2022, San Nicolás de los Garza, Nuevo León

Objetivo

El estudiante deberá presentar una propuesta de análisis de formas y de la programación para la ejecución de la optimización (descripción funcional) de características de trabajo específicas que presenta la(s) ventaja(s) (mencionar ventajas).

Nombre y definición de la forma geometría

Marco Teórico

El pie posee un conjunto de articulaciones que le permiten el movimiento en los 3 planos del espacio. Estos movimientos son de flexión-extensión, rotación interna (aducción)-rotación externa (abducción) y pronación-supinación.

1. Articulaciones de acomodación, que tienen como misión amortiguar el choque del pie con el suelo y adaptarlo a las irregularidades del terreno. Son las articulaciones del tarso y tarsometatarsianas.

2. Articulaciones de movimiento. Su función es principalmente dinámica y son fundamentales para la marcha. Son la del tobillo y las de los dedos.

Durante el movimiento de supinación, se involucra el primer estabilizador de la

articulación subastragalina (ligamento colateral lateral-CLL) y cuando se acentúa

este movimiento, el interóseo calcáneo-astragalino en su porción lateral se ve

involucrado, si este movimiento es completado por flexión plantar para lograr

inversión, el segundo estabilizador (peroneo-astragalino anterior) se activa. Esta

estabilización doble es esencial y ofrece una explicación clínica de la asociación

o disociación de las articulaciones talo crural y sub talar.

Durante la flexión plantar, el astrágalo lleva a cabo rotación medial, y durante

la flexión dorsal, se realiza una rotación lateral. El rango de movimiento rotacional es producido en su mayor parte entre la posición neutra a flexión dorsal,

documentado por varios autores de 5 a 6°, 12 y 10°.4

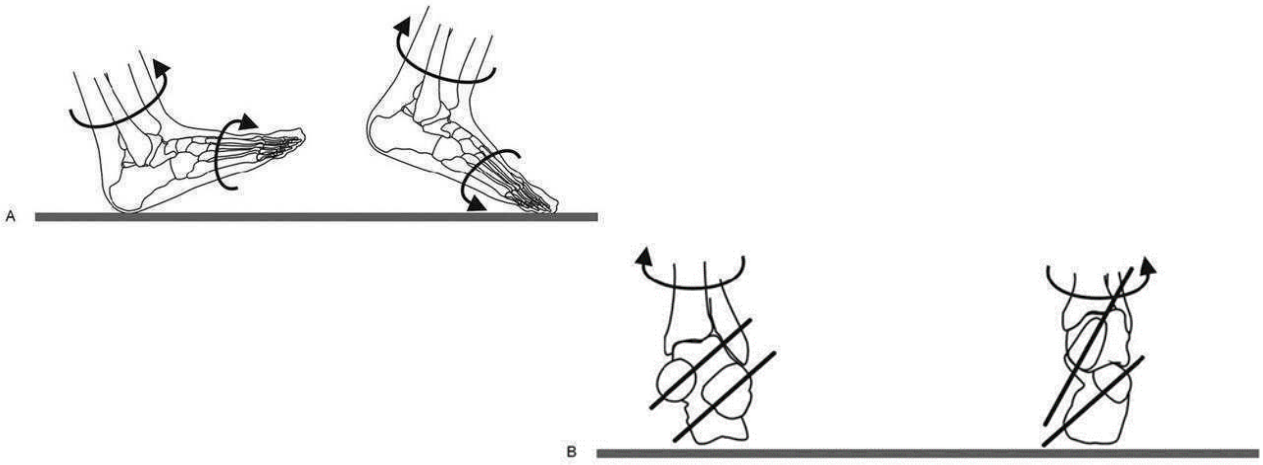
Cuando la carga se adiciona al tobillo, reporta un movimiento rotacional del

astrágalo sin flexión dorsal asociada: con rotación medial de la tibia, el astrágalo

realiza un movimiento lateral. Actualmente, los autores consideran que el astrágalo realiza un movimiento de rodaje combinando la flexión dorsal con deslizamiento horizontal en la abducción y aducción.

El tobillo permite principalmente la flexión plantar y la dorsiflexión del pie. La articulación subastragalina y los otros huesos del tarso crean muchas articulaciones sinérgicas, lo que permite un amplio rango de movimiento—flexión plantar, dorsiflexión, eversión, inversión, abducción y aducción.

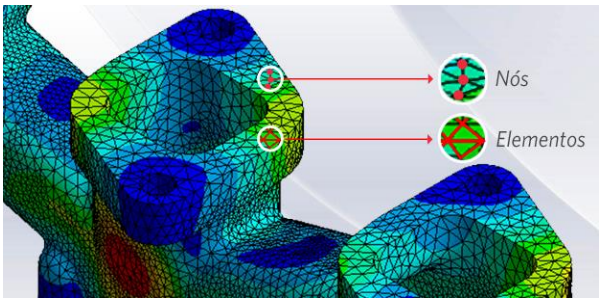
Los movimientos del pie y del tobillo, incluyen la flexión plantar, flexión dorsal, inversión y eversión. Los cuales son utilizadas para realizar la marcha y para dirigir la zona distal de los miembros inferiores hacia otras direcciones.



Estado del arte

El artículo proporcionado por el instructor de laboratorio presenta una implementación de un código de implementación 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, por último, 35 líneas para el código de elementos finitos.

Como paréntesis, el método de elementos finitos consiste en proponer que un número infinito de variables desconocidas, sean sustituidas por un número limitado de elementos de comportamiento bien definido. Esas divisiones pueden tener diferentes formas, tales como triangular, cuadrangular, entre otros, dependiendo del tipo y tamaño del problema. Como el número de elementos es limitado, son llamados de “elementos finitos” – palabra que da nombre al método. Los elementos finitos están conectados entre sí por puntos, que se llaman nodos o puntos nodales. Al conjunto de todos estos ítems – elementos y nodos – se lo denomina malla. Debido a las subdivisiones de la geometría, las ecuaciones matemáticas que rigen el comportamiento físico no se resolverán de una manera exacta, sino aproximada por este método numérico. La precisión de los Métodos dos Elementos Finitos depende de la cantidad de nodos y elementos, del tamaño y de los tipos de elementos de la malla. Por lo tanto, cuanto menor sea el tamaño y mayor el número de elementos en una malla, más precisos serán los resultados del análisis.

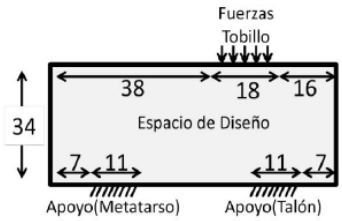


El código Matlab presentado en la página https://www.topopt.mek.dtu.dk/apps-and-software/a-99-line-topology-optimization-code-written-in-matlab está destinado a la educación en ingeniería. Los estudiantes y los recién llegados al campo de la optimización topológica pueden encontrar el código aquí y descargarlo. El código se puede utilizar en cursos de optimización estructural en los que se puede asignar a los estudiantes la realización de extensiones, como casos de carga múltiples, esquemas alternativos de independencia de malla, áreas pasivas, etc. Esta versión del código está optimizada con respecto a la velocidad en comparación con la versión original y fue lanzada el 28 de mayo de 2002.

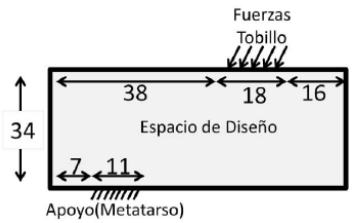
Propuesta de diseño de la geometría, alcances y limitaciones

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).

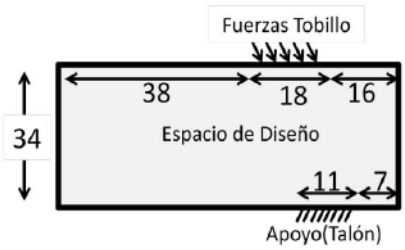


- Despegue (El área metatarsial es el apoyo, la fuerza de 500N se aplica sobre el tobillo con un ángulo de 30º)



- Apoyo (El área del talón es el apoyo, la fuerza de 500N se aplica sobre

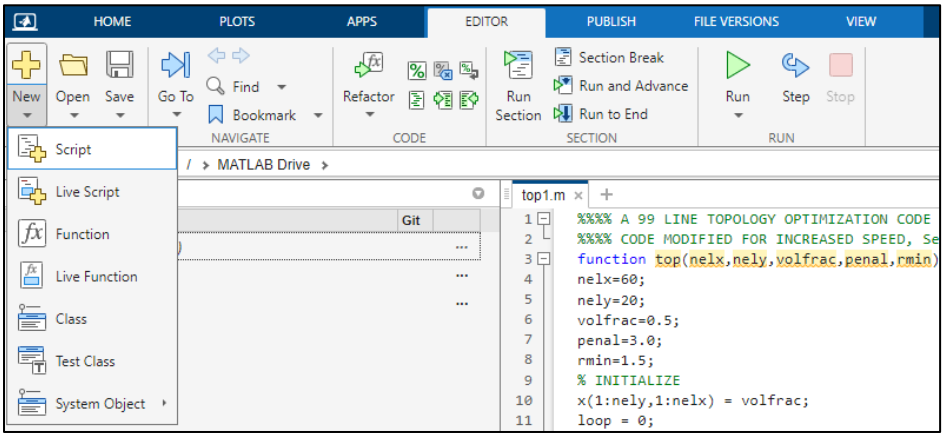
el tobillo con un ángulo de 60º).



Pasos del desarrollo de la programación

Para simular en Matlab el código propuesto de optimización topológica, debemos llevar a cabo los siguientes pasos:

1) Abrir Matlab, ya sea la aplicación de escritorio o directamente desde la página oficial usando su repositorio online.
2) Crear un nuevo script, para ello seleccionamos la opción en la barra de herramientas.



3) Una vez hecho esto, solo tenemos que copiar el código proporcionado que se localiza en la página indicada por el instructor.

4) Ya con el código copiado (99 Line Topology Optimization Code), tenemos que guardar primero el documento, en este caso como se utilizó Matlab online, se guardará en la nube de la cuenta que esta iniciada.
5) Antes de ejecutar la simulación, es necesario modificar ciertas líneas del código, el cuál será modificado para los 3 casos.

Para el caso 1:

```
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
change=1.;
```

```
% FILTERING OF SENSITIVITIES
  [dc]   = check(nelx,nely,rmin,x,dc);
% DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
  [x]    = OC(nelx,nely,x,volfrac,dc);
```

```
%%%%%%%%%% 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
```

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

Para el caso 2:

```
%%%%%%%%%% 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
```

```
%%%%%%%%%% 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;
```

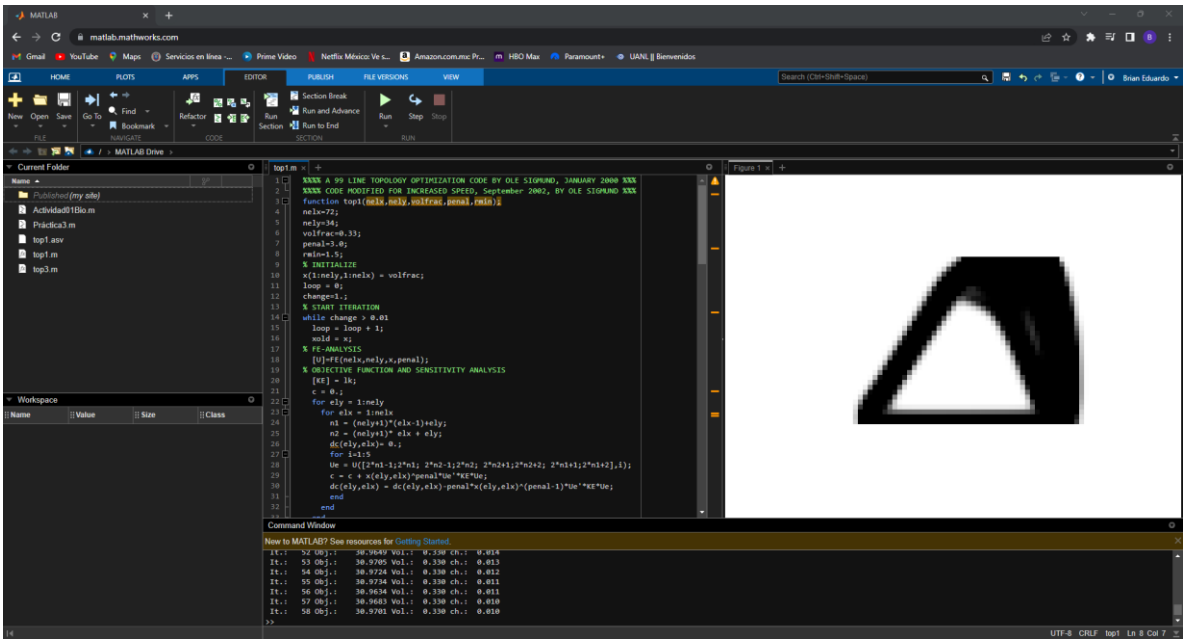
Para el caso 3:

```
%%%%%%%%%% 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
```

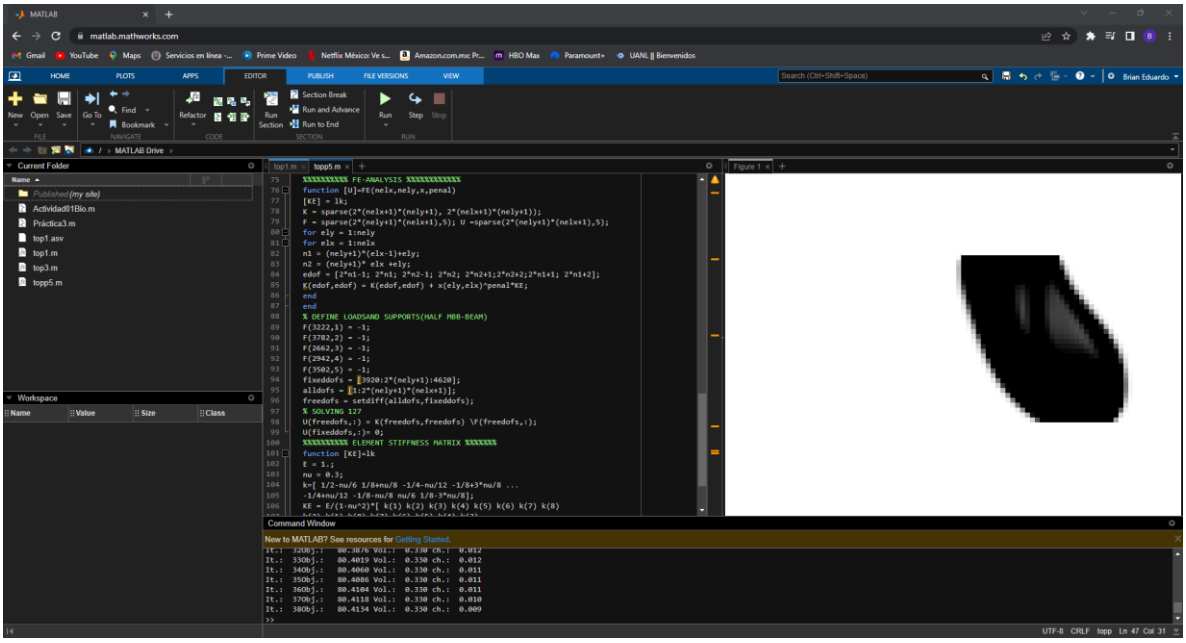
```
%%%%%%%%%% FE-ANALYSIS %%%%%%%%%%%%
function [U]=FE(nelx,nely,x,penal)
[KE] = lk;
K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
F = sparse(2*(nely+1)*(nelx+1),5); U =sparse(2*(nely+1)*(nelx+1),5);
for ely = 1:nely
for elx = 1:nelx
n1 = (nely+1)*(elx-1)+ely;
n2 = (nely+1)* elx +ely;
edof =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

Resultados de la optimización

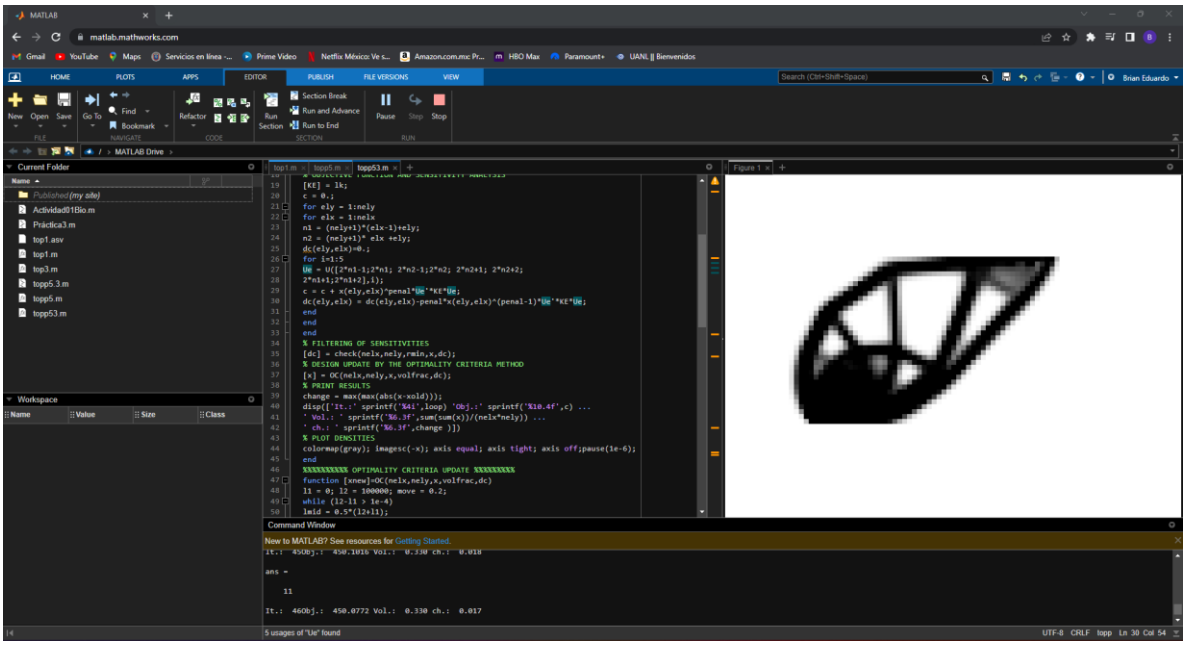
Caso 1:



Caso 2:



Caso 3:



Código utilizado

-Caso 1:

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
%%%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%
function top1(nelx,nely,volfrac,penal,rmin);
nelx=72;
nely=34;
volfrac=0.33;
penal=3.0;
rmin=1.5;
% 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-floor(rmin),1):min(i+floor(rmin),nelx)
      for l = max(j-floor(rmin),1):min(j+floor(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 LOADS AND 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);
%%%%%%%%% 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)];
```

-Caso 2:

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
%%%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%
function topp(nelx,nely,volfrac,penal,rmin);
nelx=72;
nely=34;
volfrac=0.33;
penal=3.0;
rmin=1.5;
% 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)];
```

-Caso 3:

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
%%%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%
function topp(nelx,nely,volfrac,penal,rmin);
nelx=72;
nely=34;
volfrac=0.33;
penal=3.0;
rmin=1.5;
% 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)
11
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)];
```

Conclusiones

- **Covarrubias Becerril Brian Eduardo:** para la realización de esta práctica se nos solicitó presentar una propuesta de análisis de formas y de la programación para la ejecución de la optimización de una prótesis de pie. Se tomaron 3 casos debido a los tres estados de la marcha que realiza comúnmente cualquier persona, para así comparar los resultados obtenidos. El código proporcionado por el profesor al comienzo del laboratorio tuvo que ser modificado en secciones específicas para definir las cargas, variables y el análisis que se llevaría a cabo en cada uno de los 3 casos presentados. Al final observamos los resultados de la optimización para así darnos cuenta de las diversas aplicaciones con las que cuenta el código proporcionado.

- **Velazco Rivas Francisco Javier:** para esta práctica como las anteriores se ocupó como base el código ia utilizado sin embargo en este caso se nos pedía que analizaremos una prótesis de pie es por eso que analizando los casos de forma y los movimientos que este mismo puede hacer o producir nos basamos para modificar el código ver las cargas definidas y así mismo poder analizar y observar que podríamos cambiar mejorar ya que algunas prótesis ya existen sin embargo siempre se busca optimizar lo que estamos haciendo es por eso que en la investigación vimos que el pie también podría incluirse la parte del tobillo para poder hacer movimientos sin embargo nos centramos más en el movimiento del pie que este mismo pueda ser al pisar.

- **Delgadillo Garcia Elmer Javier:** en la realización de la presente práctica se tuvo como objetivo una optimización a una prótesis de pie por el método de elementos finitos, puesto que este permite una mayor comprensión de la mecánica de la ingeniería y es sumamente adecuado para la resolución de problemas sobre geometrías complicadas y permitirá efectuar una optimización topológica a la pieza por medio del análisis del mallado, y finalmente se dio su respectiva legalidad a la efectividad de la pieza al ser sometida a distintos parámetros físicos en una serie de casos sugeridos.

- **Rodriguez Vizcarra Luis Javier.**

  En esta ultima practica se hizo uso de la optimización topológica sobre una prótesis, la cual es una técnica englobada dentro del campo de análisis estructural. Se basa en el análisis mecánico de un componente o estructura. Su principal objetivo es el aligeramiento estructural manteniendo las funcionalidades mecánicas del componente objetivo. A diferencia de otros tipos de optimización, la optimización topológica ofrece un nuevo concepto

de diseño estructural enfocado a aquellas aplicaciones donde el peso del componente es crucial, el objetivo se logró ya que se optimizo la prótesis correctamente.

- **Mexquitic Rodriguez Angel Fernando:** Con esta práctica se llevó a cabo la optimización de una prótesis de pie mediante una programación y análisis de formas, parecido a las practicas realizadas anteriormente, con esto dicho, se pudo resolver el objetivo de esta misma, gracias a lo aprendido con anterioridad, lo que nos llevó a realizar todo de manera más sencilla y con un camino más claro que seguir, y los resultados de la optimización fue bastante satisfactorio, logrando implementarlo en el problema planteado. Siendo de mucha utilidad este tipo de optimización topológica.
- **Cantú Espinoza Rubén:** en la realización de esta práctica se llevaron a cabo los conocimientos adquiridos y empleados en las prácticas anteriores. Primeramente, definimos nuestra forma geométrica, la cual vendría siendo la de un pie (donde tenemos 3 tipos de movimiento: flexión, tensión, rotación); posteriormente procedemos con el análisis de formas y propuestas de optimización por medio de análisis de elemento finito. Para llevar a cabo esta práctica tomamos en cuenta tres casos, por los movimientos que realiza un individuo comúnmente, con el objetivo de comparar los resultados. Para ello modificamos el código en diferentes secciones y comparamos nuestros resultados.

Referencias

- 99 Line Topology Optimization Code – O. Sigmund, Department of Solid Mechanics, Building 404, Technical University of Denmark, DK-2800 Lyngby, Denmark.

- IMME. (2007). *Una metodología para la optimización estructural de formas usando principios de evolución flexible distribuida*. Scientific Electronic Library Online. http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S0376-723X2007000100002