

**Universidad Autónoma de Nuevo
León**

**Facultad de Ingeniería Mecánica y
Eléctrica**

PRÁCTICA #5:

Optimización de una
prótesis de pie

Asignatura: Lab.

Biomecánica

Maestro(a): Yadira

Moreno Vera

Grupo/Brigada: 408

Nombre	Matricula
Victor Alan Cavazos Ramírez	1902881
Arturo Mariscal Picón	1806989
José Francisco Juárez Segundo	1992319
Jorge Eduardo Ortiz Cruz	1992029

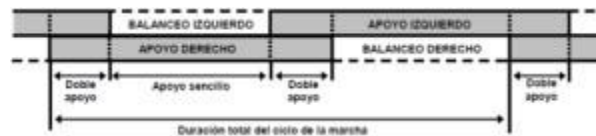


Objetivo

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).

Geometría

El pie es el encargado de proporcionar estabilidad y equilibrio en el ciclo de la marcha, esto mientras absorbe los impactos, por lo cual para poder analizar la geometría y la biomecánica de este se debe conocer la anatomía de este.



1 - Ciclo de marcha




Para soportar el peso del cuerpo, el pie funciona de forma análoga a un arco y viga. El arco es un elemento estructural curvo el cual permite soportar cargas por encima de un vacío. Cuando el arco longitudinal se aplanata hasta un grado apreciable, aumenta la presión en las presiones dorsales de los huesos del tarso. En un pie normal los ligamentos y músculos impiden el aplanamiento del arco

Estado del arte

La amputación es la separación o el corte de alguna extremidad del cuerpo en la mayoría de los casos por medio de una intervención quirúrgica. Cuando una persona sufre de alguna puede comenzar a cambiar su estilo de vida, y una forma de "facilitar" el proceso de adaptación es usando alguna prótesis capaz de cumplir con los movimientos básicos que realizaba la extremidad que se perdió.

Nombre	Tipo	Función	Ventajas	Limitaciones
Sach		Es el más simple, rígido y no puede doblarse. El talón es una cuña de goma que se comprime bajo el peso del usuario y permite que el talón se mueva un poco al comenzar la fase de apoyo de la marcha.	Por su practicidad y bajo costo puede ser mas accesible a las personas y puede funcionar como una solución temporal.	Es incómodo para el paciente al tener limitaciones de movimiento y puede llegar a ser peligroso.
Pie monoaxial		El pie tiene un eje, el cual cuenta con un 35% de eficiencia (la relación entre la energía liberada y la almacenada).	Es mas cómodo y tiene una mejor funcionalidad y su costo es mayor pero sigue considerable	Sigue teniendo limitaciones y es mas costoso que un sech.
Pie multiaxial		Este pie permite movimientos en los 3 planos, lo que facilita la adaptación a terrenos irregulares y la absorción de movimientos de torsión que eliminan las fuerzas cortantes en el pie.	Es mas seguro, confiable y hay mas libertad de movimiento y actividades, suele ser de materiales mas ligeros y resistentes.	Este modelo ya tiene un mayor costo y al estar compuesto por materiales de mayor calidad y por su mantenimiento.

Tip flex		Se diseña pensando en las necesidades específicas de cada tipo de usuario, es la que suelen usar los deportistas.	Es el diseño mas optimo para el paciente, seguro y confiable.	Es el mas costoso de todos y su tiempo de fabricación y adaptación es mayor debido al diseño especifico de cada persona.
----------	---	--	---	--

El principal objetivo de una prótesis es sustituir una parte del cuerpo que haya sido perdida por una amputación o que no exista a causa de agenesia, cumpliendo las mismas funciones que la parte faltante.

La principal dificultad de la protetización consiste en adaptar de forma relativamente confortable el encaje, de manera que permita amortiguar el peso corporal y pueda transmitir las fuerzas dinámicas que se producen durante la marcha.

De esta forma, deben de ser considerados los siguientes principios biomecánicos:

- *Principio del alargamiento mecánico del pie:* Establece que se debe sustituir la longitud perdida del pie para compensar la afectación funcional que la amputación supone al quedar disminuida la longitud de la palanca del pie. Este alargamiento puede obtenerse mediante una suela rígida dispuesta en balancín.
- *Principio de la disposición anatómica del muñón:* Llamado también de carga en talo o colocación en gancho para el retropié, consiste en disponer el muñón dentro del calzado o prótesis de tal forma que se encuentre en la misma posición que esa parte del pie tendría, caso de que no hubiere amputación.
- *Principio de la escuadra interna:* La escuadra interna se materializa por un refuerzo o lengüeta anterior que hace cuerpo con el pie protésico. No debe ser demasiado corta, pues si no la pierna, apoyándose en el borde superior de la lengüeta durante el desarrollo, tiende a girar hacia atrás con lo que por un lado hace daño el borde sobre el que se clava la pierna, duele en el apoyo anterior y por otro lado, la presión concentrada en el talón, que tiende a girar hacia atrás y arriba, puede llegar a lesionarlo y/o descalzarlo.
- *Principio de la descarga proximal:* Consiste en repartir directamente a la pierna cargas y esfuerzos que normalmente toma a su cargo el pie, para descargar un muñón que tenga disminuida o haya perdido su capacidad de carga.

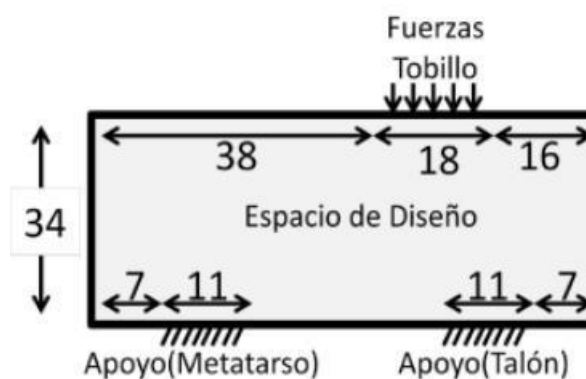
Diseño de geometría, alcances y limitaciones.

Descripción del ambiente sintético: Según lo solicitado en esta práctica, se utilizara una modificación del código de optimización de 99 líneas para analizar el diseño de una prótesis de pie en sus diferentes fases del ciclo de marcha.



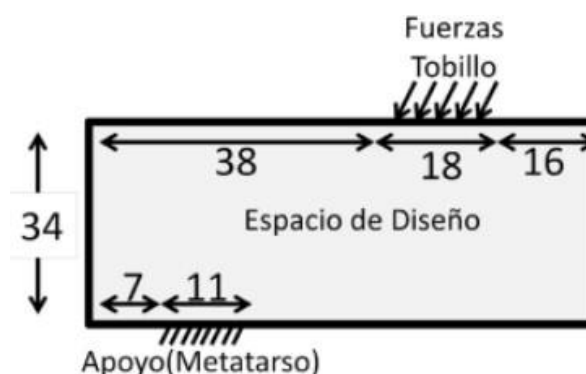
Parada normal

En esta fase el talón y área metatarsial son los apoyos, aquí se le aplica una fuerza sobre el tobillo de 500 N.



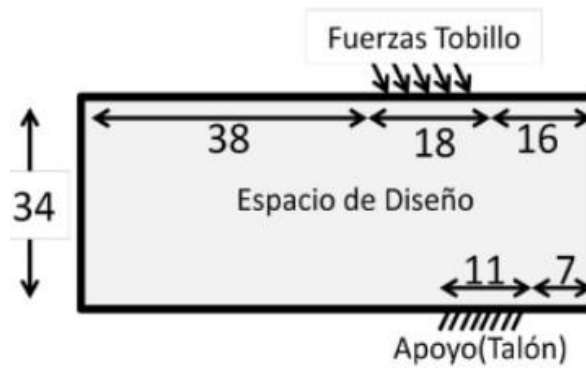
Despegue

Aquí el apoyo se aplica en el área metatarsial, mientras que la fuerza de 500 N se aplica al tobillo en unos 30 grados.



Apoyo

El área de apoyo esta sobre el talón, mientras que la fuerza de 500 N se aplica sobre el tobillo a un ángulo de 60 grados.



Metodología de solución: Parada normal

```
% 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]);
```

2 - Definir cargas que interactúan cuando el pie se encuentra parado de forma normal.

Metodología de solución: Despegue

```
% 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];
```

3 - Cargas que actúan cuando ocurre el despegue.

Metodología de solución: Apoyo

```
% 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 = [560:2*(nely+1):1260];
```

4 - Definir las cargas para el apoyo.

Desarrollo de programación

Programación para el parado normal.

```
1 %%% A 70 LINE TOPOLOGY OPTIMIZATION CODE BY OLKREIMUND, OCTOBER 1999 %%%
2 %%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE NIKHUND %%%
3 %%% Authors: IS12877, Alexander IS16379, Raul LARSEN, Mauricio IS11100 %%%
4 function [x] = AAMP_70(nlx,nly,volfrac,penal,rmin)
5 % INITIALIZE
6 x(l:nly,1:nlx) = volfrac;
7 loop = 0;
8 change = 1.;
9 % START ITERATION
10 while change > 0.01
11 loop = loop + 1;
12 xold = x;
13 % FE-ANALYSIS
14 [U]=FE(nlx,nly,x,penal);
15 % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
16 [ME] = 1k;
17 c = 0.;
18 for nly = 1:nly
19 for nlx = 1:nlx
20 nl = (nly+1)*(nlx-1)+nly;
21 n2 = (nly+1)* nlx +nly;
22 do(nly,nlx)=0.;
23 for i=1:3
24 de = U((2*n1-1)*n1; 2*n2-1)*n2; 2*n2+1; 2*n2+2; 2*n3+1;2*n3+2;1);
25 c = c + x(nly,nlx)*penal*de*FE*De;
26 do(nly,nlx) = do(nly,nlx)-penal*x(nly,nlx)*(penal-1)* U*FE*De;
27 end
28 end
29 end
30 % FILTERING OF SENSITIVITIES
31 [dc] = check(nlx,nly,rmin,x,dc);
32 % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
33 [x] = OC(nlx,nly,x,volfrac,dc);
34 % PRINT RESULTS
35 change = max(max(abs(x-xold)));
36 disp(['In: ' sprintf('%d',loop) 'Out: ' sprintf('%d-%d',c),...
37 ' Vol: ' sprintf('%d',sum(sum(x))/(nlx*nly)) ...
38 ' ch: ' sprintf('%d',change) ])
39 % POST SENSITIVE
40 colormap(gray); imagesc(-x); axis equal; axis tight; axis off;pause(1e-6);
41 end
42 %%%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%%%
43 function [xnew]=OC(nlx,nly,x,volfrac,dc)
44 l1 = 0; l2 = 10000; move = 0.2;
45 while (l2-l1 > 1e-4)
46 lmid = 0.5*(l2+l1);
47 xnew = max(0.001,max(x-move,min(1,min(x+move,x.*sqrt(-dc./lmid)))));
48 if sum(sum(xnew)) - volfrac*nlx*nly > 0;
49 l1 = lmid;
50 else
51 l2 = lmid;
52 end
53 end
54 %%%%%%%%%%% NEIGH-DEPENDENCY FILTER %%%%%%%%%%%
55 function [dcn]=check(nlx,nly,rmin,x,dc)
56 dcn=zeros(nly,nlx);
57 for i = 1:nlx
58 for j = 1:nly
59 sum=0.0;
60 for k = max(i-round(rmin),1):min(i+round(rmin),nlx)
61 for l = max(j-round(rmin),1):min(j+round(rmin), nly)
62 fac = rmin-sqrt((l-k)^2+(j-l)^2);
63 sum = sum+max(0,fac);
64 dcn(j,l) = dcn(j,l) + max(0,fac)*x(l,k)*dc(l,k);
65 end
66 end
67 dcn(j,l) = dcn(j,l)/(x(j,l)*sum);
68 end
69 end
70 %%%%%%%%%%% FE-ANALYSIS %%%%%%%%%%%
71 function [U]=FE(nlx,nly,x,penal)
72 [ME] = 1k;
73 K = sparse(2*(nlx+1)*(nly+1), 2*(nlx+1)*(nly+1));
74 F = sparse(2*(nly+1)*(nlx+1),5); U =sparse(2*(nly+1)*(nlx+1),5);
75 for nly = 1:nly
76 for nlx = 1:nlx
77 nl = (nly+1)*(nlx-1)+nly;
78 n2 = (nly+1)* nlx +nly;
79 edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1;2*n2+2;2*n3+1; 2*n3+2];
80 [edof,edof] = K(edof,edof) + x(nly,nlx)*penal*ME;
81 end
82 end
83 % DEFINE LOADAND SUPPORTS(HALF NEIGH-NEIGH)
84 F(322,1) = -1;
85 F(3782,2) = -1;
86 F(2642,3) = -1;
87 F(2842,4) = -1;
88 F(3502,5) = -1;
89 fixeddofs = union([560;2*(nly+1);1240],[3920;2*(nly+1);4620]);
90 alldofs = 1:2*(nly+1)*(nlx+1);
```

```

91 - freedofs = setdiff(alldofs,fixeddofs);
92 - % SOLVING 127
93 - U(freedofs,:) = K(freedofs,freedofs) \F(freedofs,:);
94 - U(fixeddofs,:)= 0;
95 - %***** ELEMENT STIFFNESS MATRIX *****
96 - function [KE]=lk
97 - E = 1.;
98 - nu = 0.3;
99 - k=[ 1/2-nu/6 1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
100 -1/4+nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8];
101 - KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
102 - k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
103 - k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
104 - k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
105 - k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
106 - k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
107 - k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
108 - k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];

```

Programación para el apoyo.

```

1 %*** A 2D LINE TOPOLOGY OPTIMIZATION CODE BY HANSEN, OCTOBER 1999 ***
2 %*** CODE MODIFIED FOR INCREASED SPEED, September 2001, BY GLE STOKED ***
3 %*** ANDRÉAS LILLY, Alexandre LILLY, Mail 100503, Nantes 44000 ***
4 function [xnew]=topo(mels,nely,volfrac,penal,smin)
5 % INITIALIZE
6 x(nels,nely)= volfrac;
7 loop = 0;
8 change = 1.;
9 % START ITERATION
10 while change > 0.01
11 loop = loop + 1;
12 xold = x;
13 % FE ANALYSIS
14 [U]=FE(mels,nely,x,penal);
15 % COLLECTIVE FUNCTION AND SENSITIVITY ANALYSIS
16 [KE] = lk;
17 c = 0.;
18 for nly = 1:nely
19 for nlx = 1:nels
20 nl = (nely+1)*(nlx-1)+nly;
21 n2 = (nely+1)* nlx +nly;
22 %c(nly,nlx)=0.;
23 for i=1:5
24 Ue = U([2*n1-1:2*n1+2*n2-1:2*n2+2 2*n2+2 2*n1+1:2*n1+3],i);
25 c = c + n(nly,nlx)*penal*Ue'*KE*Ue;
26 %c(nly,nlx) = c(nly,nlx)-penal*x(nly,nlx)*(penal-1)* Ue'*KE*Ue;
27 end
28 end
29 % end
30 % FILTERING OF SENSITIVITIES
31 [dc] = check(mels,nely,smin,x,dc);
32 % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
33 [x] = OC(mels,nely,x,volfrac,dc);
34 % PRINT RESULTS
35 change = max(abs(x-xold));
36 disp(['Iteration ',loop,' : ',sprintf('%10.4f',c)...
37 ' Vol. : ',sprintf('%4.2f',sum(x)/(mels*nely)) ...
38 ' ch. : ',sprintf('%4.2f',change)]);
39 % PLOT DENSITIES
40 colormap(gray); imagesc(-X); axis equal; axis tight; axis off; pause(5e-6);
41 end
42 %***** OPTIMALITY CRITERIA UPDATE *****
43 function [xnew]=OC(mels,nely,x,volfrac,dc)
44 i1 = 0; i2 = 10000; move = 0.2;
45 while (i2-i1 > 1e-6)
46 imid = 0.5*(i2+i1);
47 xnew = max(0.001,max(x-move,min(1,min(x+move,x.*sqrt(-dc./imid)))));
48 if sum(xnew) - volfrac*mels*nely > 0;
49 i1 = imid;
50 else
51 i2 = imid;
52 end
53 end
54 %***** HESO-INDEPENDENCY FILTER *****
55 function [dcn]=check(mels,nely,smin,x,dc)
56 dcn=zeros(nely,nels);
57 for i = 1:nels
58 for j = 1:nely
59 sum=0.0;
60 for k = max(1-round(smin),1):min(1+round(smin),nels)

```



```

61 - for l = max(j-round(xmin),1):min(j+round(xmin), nely)
62 - fac = xmin-sqrt((1-K)^2+(j-l)^2);
63 - sum = sum+max(0,fac);
64 - dcn(j,l) = dcn(j,l) + max(0,fac)*x(l,w)*dc(l,k);
65 - end
66 - end
67 - dcn(j,l) = dcn(j,l)/(w(l)*sum);
68 - end
69 - end
70 - %%%%%%%%%%% FE-ANALYSIS %%%%%%%%%%%%%%%
71 - function [U]=FE(nels,nely,x,penal)
72 - [KE] = lk;
73 - K = sparse(2*(nels+1)*(nely+1), 2*(nels+1)*(nely+1));
74 - F = sparse(2*(nely+1)*(nels+1),5); U = sparse(2*(nely+1)*(nels+1),5);
75 - for ely = 1:nely
76 - for elx = 1:nels
77 - n1 = (nely+1)*(elx-1)+ely;
78 - n2 = (nely+1)* elx +ely;
79 - edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1;2*n2+2;2*n1+1; 2*n1+2];
80 - K(edof,edof) = K(edof,edof) + x(ely,elx)*penal*KE;
81 - end
82 - end
83 - % DEFINE LOADS AND SUPPORTS (HALF WING-BEAM)
84 - F(922,1) = -1;
85 - F(378,2) = -1;
86 - F(264,3) = -1;
87 - F(294,4) = -1;
88 - F(350,5) = -1;
89 - fixeddofs = [3920:2*(nely+1):4620];
90 - alldofs = [1:2*(nely+1)*(nels+1)];

91 - freedofs = setdiff(alldofs,fixeddofs);
92 - % SOLVING I2?
93 - U(freedofs,:) = K(freedofs,freedofs) \F(freedofs,:);
94 - U(fixeddofs,:) = 0;
95 - %%%%%%%%%%% ELEMENT STIFFNESS MATRIX %%%%%%%%%%%
96 - function [KE]=lk
97 - I = 1.;
98 - nu = 0.3;
99 - k=[ 1/2-nu/6 1/8-nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
100 -1/4-nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8];
101 - KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
102 - k(2) k(1) k(5) k(7) k(6) k(3) k(4) k(3)
103 - k(3) k(5) k(1) k(6) k(7) k(4) k(5) k(2)
104 - k(4) k(7) k(6) k(1) k(5) k(3) k(2) k(5)
105 - k(5) k(6) k(7) k(5) k(1) k(2) k(3) k(4)
106 - k(6) k(5) k(4) k(3) k(2) k(1) k(5) k(7)
107 - k(7) k(4) k(5) k(2) k(3) k(5) k(1) k(6)
108 - k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)]';

```

Programación para el despegue.

```

1 - %%% A 90 LINE SUBROUTINE OPTIMIZATION CODE BY ALEXANDER, OCTOBER 1999 %%%
2 - %%% CODE MODIFIED FOR INCREASED SPEED, September 2003, BY DR S. J. DUNN %%%
3 - %%% Revision 101277, Alexander ILYIN, Nov 199903, Waterloo 101100 %%%
4 - function [ASM,fc,nels,nely,volfrac,penal,minit]
5 - % INITIALISE
6 - x(linel,lnels) = volfrac;
7 - loop = 0;
8 - change = 1.;
9 - % START ITERATION
10 - while change > 0.01
11 - loop = loop + 1;
12 - end = 0;
13 - % FE-ANALYSIS
14 - [U]=FE(nels,nely,x,penal);
15 - % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
16 - [KE] = lk;
17 - c = 0.;
18 - for ely = 1:nely
19 - for elx = 1:nels
20 - n1 = (nely+1)*(elx-1)+ely;
21 - n2 = (nely+1)* elx +ely;
22 - de(nly,ely)=0.;
23 - for i=1:5
24 - de = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1; 2*n2+2; 2*n1+1;2*n1+2],i);
25 - c = c + x(ely,elx)*penal*(de'*KE*de);
26 - de(ely,elx) = dc(ely,elx)-penal*x(ely,elx)*(penal-1)* de'*KE*de;
27 - end
28 - end
29 - % FILTERING OF SENSITIVITIES

```

```

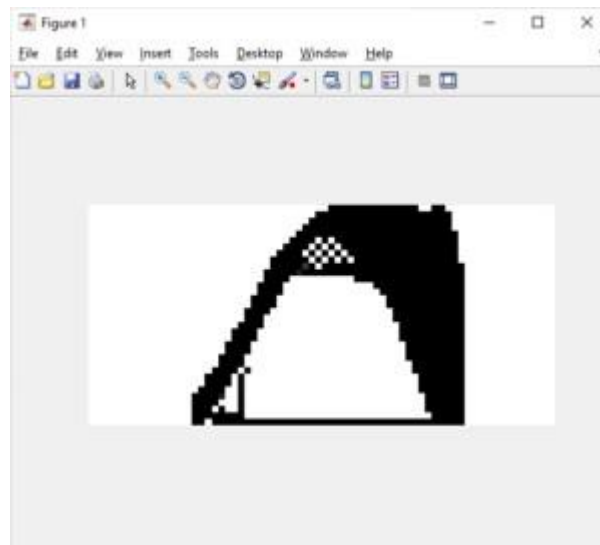
31 - [dc] = check(nelx,nely,rmin,x,dc);
32 % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
33 [x] = OC(nelx,nely,x,volfrac,dc);
34 % PRINT RESULTS
35 change = max(max(abs(x-xold)));
36 disp(['Iteration ',loop,' loop: ',x,' ',sprintf('%10.4f',dc) ...
37 ' Vol: ', sprintf('%6.3f',sum(sum(x)/(nelx*nely)) ...
38 ' ch: ', sprintf('%6.3f',change)])
39 % PLOT RESULTS
40 colormap(gray); imagesc(x); axis equal; axis tight; axis off; pause(1e-6);
41 end
42 %***** OPTIMALITY CRITERIA UPDATE *****
43 function [xnew]=OC(nelx,nely,x,volfrac,dc)
44 l1 = 0; l2 = 10000; move = 0.2;
45 while (l2-l1 > 1e-4)
46     lmid = 0.5*(l2+l1);
47     xnew = max(0.001,max(x-move,min(l1,min(x+move,x.*sqrt(-dc./lmid)))));
48     if sum(sum(xnew)) - volfrac*nelx*nely > 0;
49         l1 = lmid;
50     else
51         l2 = lmid;
52     end
53 end
54 %***** MASS-INDEPENDENCY FILTER *****
55 function [dcm]=check(nelx,nely,rmin,x,dc)
56 dcm=zeros(nely,nelx);
57 for i = 1:nely
58     for j = 1:nelx
59         sum=0;
60         for k = max(j-round(rmin),1):min(j+round(rmin),nelx)
61             for l = max(j-round(rmin),1):min(j+round(rmin),nely)
62                 fac = rmin*sqrt((i-k)^2+(j-l)^2);
63                 sum = sum+max(0,fac);
64                 dcm(j,l) = dcm(j,l) + max(0,fac)*x(l,k)*dc(l,k);
65             end
66         end
67         dcm(j,l) = dcm(j,l)/(k(j,l)*sum);
68     end
69 end
70 %***** FE-ANALYSIS *****
71 function [U]=FE(nelx,nely,x,penal)
72 [KE] = lk;
73 K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
74 F = sparse(2*(nely+1)*(nelx+1),5); U = sparse(2*(nely+1)*(nelx+1),5);
75 for ely = 1:nely
76     for elx = 1:nelx
77         n1 = (nely+1)*(elx-1)+ely;
78         n2 = (nely+1)*elx+ely;
79         edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1; 2*n2+2; 2*n1+1; 2*n1+2];
80         K(edof,edof) = K(edof,edof) + x(ely,elx)*penal*KE;
81     end
82 end
83 % DEFINE LOADS AND SUPPORTS(HALF HSB-BEAM)
84 F(322,1) = -1;
85 F(370,2) = -1;
86 F(442,3) = -1;
87 F(492,4) = -1;
88 F(502,5) = -1;
89 fixeddofs = [540;2*(nely+1):1260];
90 alldofs = [1:2*(nely+1)*(nelx+1)];

91 freedofs = setdiff(alldofs,fixeddofs);
92 % SOLVING 127
93 U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
94 U(fixeddofs,:) = 0;
95 %***** ELEMENT STIFFNESS MATRIX *****
96 function [KE]=lk
97 E = 1;
98 nu = 0.3;
99 k=[ 1/2-nu/6 1/8+nu/8 -1/6-nu/12 -1/8+3*nu/8 ...
100 -1/4+nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8];
101 KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
102 k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
103 k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
104 k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
105 k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
106 k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
107 k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
108 k(8) k(3) k(2) k(5) k(7) k(6) k(1) k(4)];

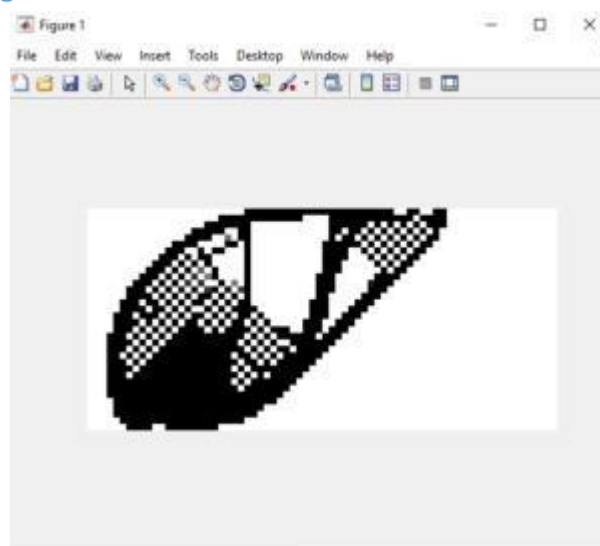
```

Resultados de la optimización

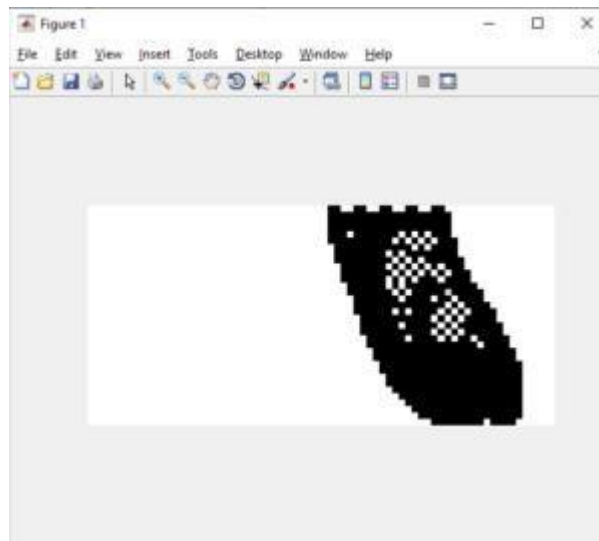
Resultados de parado normal



Resultados del despegue



Resultados del apoyo



Conclusiones

Durante esta práctica se estudió la geometría de una prótesis para un pie humano, junto con los aspectos que deben de tomarse en cuenta aspectos que afectan el como este interactúa con las fuerzas como lo es el ciclo de marcha, pues dependiendo del acomodo que tenga el pie en el talón será la inclinación en la cual estará actuando la carga. En esta ocasión la programación se enfocó en tres fases: pie parado normal, pie cuando este apoyado y el pie cuando ocurre el despegue., se utilizo Matlab para generar un análisis de elemento finito para objetos de ámbito simple y que se pueden usar paradiferentes casos, además de generar un buen soporte que nos ayudará mucho en este caso. A partir de lo que aprendimos, nos damos cuenta que los software de hoy en día nos apoyan mucho con cálculos e impresiones que nos facilitan el poder generar nuevas ideas e ir más rápido en nuestras investigaciones.

Victor Alan Cavazos Ramírez

En este caso, se puede ver que se puede diseñar para piezas en diferentes etapas, que en su conjunto crearían una pieza útil para cualquiera de estos casos. Para esto nada más se tiene que analizar la pieza considerando los diferentes soportes de este.

Jorge Eduardo Ortiz Cruz

Con los resultados de esta última práctica podemos concluir con una serie de ejercicios donde optimizamos diferentes diseños, donde en esta última tenemos el más complicado y avanzado que hubo, donde se modificó el código tres veces para obtener resultados en cada parte del pie. Ahora podemos ver reflejada la biomecánica en estas simulaciones y cómo gracias al código de 99 líneas podemos modificarlo a gusto para poder analizar en Matlab estos diseños que se vieron durante el semestre

Referencias

- Vesga, S. A. (2018). Fabricación de un prototipo de una prótesis de miembro inferior transtibial mediante tecnologías aditivas de acuerdo con las medidas antropométricas del paciente. *UNIVERSIDAD SANTO TOMÁS*.

- Gutiérrez García, W. A. (2017). *Diseño y simulación prótesis de pie* (Bachelor's thesis, Universidad Piloto de Colombia).
- Rozo Pinzon, Y. (2007). Rediseño, construcción y optimización del sistema de control para pie de prótesis transfemoral semiactiva.