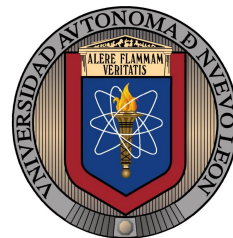




Facultad de Ingeniería Mecánica y Eléctrica

UANL

MECATRÓNICA



Laboratorio de Biomecánica

Grupo:309

Optimización de una prótesis de pie Práctica 5

FECHA DE ENTREGA: 16 NOVIEMBRE 2022

Profesor:

Dra. Yadira Moreno Vera

Nombre	Matricula	Carrera
Betsaida Alejandra Ruedas Vázquez	1730437	IMTC
Miguel Rodrigo Aguilar Moreno	1801380	IMTC
Erick Daniel Esquivel Arguelles	1826021	IMTC
Omar Isaí Moreno Cruz	1849630	IMTC
Saul Moises Mendoza Cida	1942534	IMTC

Índice

1. Objetivo:	3
2. Nombre y definición de la gemoetría:	3
3. Estado del arte:	4
4. Marco teórico:	5
5. Desarrollo:	6
5.1. Caso: 1	6
5.2. Caso: 2 y 3	9
6. Conclusiones	21

1. 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 las ventajas.

2. Nombre y definición de la gemoetría:

Prótesis de pierna

Este dispositivo está diseñado para adaptarse a todo tipo de necesidades y cuerpos que han sufrido una amputación parcial o total de los miembros inferiores, y el cual brinda funciones tanto estéticas como mecánicas para el movimiento del cuerpo.

Partes prótesis de pierna

Los siguientes son los componentes en que consiste una prótesis de pierna arriba de rodilla:

- **Socket.** Es el componente de prótesis para pierna que va en contacto con el muñón.
- **Rodilla.** Es la que permite el movimiento de flexión extensión de la rodilla de prótesis para pierna.
- **Pie.** Es la parte de la prótesis para pierna que es el que contacta con el suelo para dar el siguiente paso.

Los siguientes son los componentes de una prótesis para pierna por debajo de la rodilla:

- **Socket.** Esta en contacto con el muñón del paciente amputado.
- **Pie.** Contemplado para realizar el mismo objetivo de ser el primer contacto con el suelo.

El **socket** es un componente fundamental para prótesis de pierna arriba de rodilla como para la prótesis de pierna abajo de rodilla que estará en contacto directo con el muñón del paciente amputado y permitirán realizar la contención de interface entre la prótesis de pierna que adquiera el paciente y el muñón y este socket será distinto dependiendo de la necesidad de cada paciente.

La **rodilla protésica** que le permita al paciente tener mayor movilidad al caminar, en donde entre mayor sea la inversión, mejor será el movimiento de la marcha con la prótesis, notando una gran diferencia con una prótesis básica y con una prótesis de alta gama, la rodilla de la prótesis de pierna básica le permite al paciente realizar movimiento a la distancia que él se proponga, sin ningún inconveniente, pero generará un desgaste energético mayor, que se puede ver traducido en mucho cansancio, cabe mencionar que el cansancio o desgaste físico que tiene el paciente amputado

es mayor que el de una persona con ambas piernas.

El **pie de prótesis** de pierna básica es el componente que me va a permitir amortiguar el primer contacto con el talón, para posteriormente trasladar el peso del cuerpo a la punta del pie de la prótesis para pierna y despegar el peso del cuerpo del suelo.

3. Estado del arte:

En la actualidad existen prótesis biónicas o robóticas, las cuales utilizan sistemas computacionales e inteligencia artificial para aprender sobre los patrones de movimiento del paciente portador y predecir sus movimientos, adelantándose y permitiéndole a las personas caminar, correr, subir escaleras, sentarse o cualquier movimiento necesario.

Las prótesis de piernas modernas también cada vez están más enfocadas en la estética, por lo que existen modelos que se han recubierto de materiales como silicón quirúrgico que asemeja los tejidos de la piel, con colores y detalles que las hacen pasar desapercibidas a simple vista, lo que es importante para la seguridad y salud psicológica de las personas que sufren una amputación.

Una prótesis de pierna biónica o robótica funciona de igual manera que las prótesis tradicionales, pero la diferencia es que implementa la más moderna tecnología para que la fuerza y los movimientos sean independientes, predictivos y lo más acertados a los deseos del portador.

Estas prótesis de piernas modernas vienen implementadas con una serie de motores, articulaciones, sensores y procesadores computacionales que recogen e interpretan los impulsos nerviosos del cerebro y los traduce en movimientos que permiten a la persona realizar las tareas de su vida diaria.

Exo-Prosthetic leg

Es una alternativa a la prótesis robótica tradicional, que nace de la conjunción de las tecnologías de escaneado, modelado e impresión 3D, creando un exoesqueleto de titanio que replica la forma exacta del miembro amputado. Es un ejemplo de las prótesis de pierna que existen en la actualidad.

El proceso de creación Su pierna es el resultado del escaneo, impresión y modelado 3D, Root piensa que todo el proceso puede ser automatizado para crear un producto estético, ajustable y sobre todo costeable (en mexicano bueno, bonito y barato)

En cuanto al proceso de fabricación, el muñón del paciente junto con el miembro intacto son escaneados en primer lugar para crear un modelo virtual de alta precisión, permitiendo que la anatomía creada se asemeje a la existente con una diferencia de apenas unas fracciones de milímetro.

Durante el proceso, la tecnología FitSocket, desarrollada por el laboratorio de Biomecatrónica

del MIT, también captura las propiedades del tejido de la pierna permitiendo que quede mejor y que sea más cómoda.

Los escaneos de la pierna intacta, el muñón y mecanismos prefabricados, son combinados en un modelo 3D, para crear la base de la prótesis.



Figura 1: Prótesis hecha con tecnología 3D.

4. Marco teórico:

¿Cómo funciona una Prótesis de Pierna?

Una prótesis estándar se fabrica con diversos materiales como titanio, aluminio, u otros polímeros, etc. y sus piezas están conectadas a un encaje que se ajusta al paciente. Para ello, se coloca una capa o funda, que funciona como barrera entre la piel y el encaje para mayor comodidad y ajuste., el cual puede ser de succión, de vacío y sistema de anclaje por lanzadera.

Las opciones van, desde una apariencia solo funcional, en cuanto a las piezas mecánicas, hasta una funda estética. Cada día surgen nuevas oportunidades para poder brindarte nuevas tecnologías en:

- Prótesis de piernas para diabéticos.
- Prótesis de piernas para deportistas.

- Prótesis de piernas para niños.

Los **materiales protésicos**, aunque esencialmente son los mismos, han mejorado gracias a la innovación tecnológica, por lo cual se manejan varias aleaciones y compuestos. También, los dispositivos atractivos tienen alta demanda, a base de espumas de poliuretano, polietileno, nylon, látex y más. Contamos con prótesis de pierna con los siguientes materiales:

- Prótesis de Pierna de acero inoxidable.
- '' de Pierna de titanio.
- '' de Pierna de aleaciones de titanio.
- '' de Pierna de Fibra de carbono pre impregnado.
- '' de Pierna con biocerámicas.
- '' de Pierna con óxido de aluminio.
- '' de Pierna de biopolímeros.
- '' de Pierna con biomateriales compuestos.

Costos de las prótesis de pierna

Por lo general los precios de las prótesis de pierna rondan los 20,000 a los 30,000 pesos mexicanos, aunque en el caso de las prótesis especializadas para deportistas o prótesis robóticas, el precio puede elevarse a más de 100,000MXN.

5. Desarrollo:

Desarrollo de código

5.1. Caso: 1

1) El primer cambio se realiza a partir de la línea 7, es necesario eliminar todo lo que se encuentra dentro del primer ciclo for, desde la línea 8 a la 19, quedando de la siguiente forma.

```

4      % INITIALIZE
5      x(1:nely,1:nelx) = volfrac;
6      loop = 0;
7      change = 1.;
8      % START ITERATION
9      while change > 0.01
10         loop = loop + 1;
11         xold = x;
12      % FE-ANALYSIS

```

Figura 2: Lineas del código 8-19 modificadas

2) El segundo cambio se realiza en la sección del diseño actualizado por el método de optimización de criterios, eliminando la variable “passive” de la función X

```

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

```

Figura 3: Linea de código 32 modificada

3) Para el tercer cambio, es necesario eliminar una vez más la variable “passive” de la función “Xnew” y borrar la línea 47 del código, quedando de la siguiente manera.

```

41      %%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%%
42      function [xnew]=OC(nelx,nely,x,volfrac,dc)
43      l1 = 0; l2 = 100000; move = 0.2;
44      while (l2-l1 > 1e-4)
45          lmid = 0.5*(l2+l1);
46          xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
47          if sum(sum(xnew)) - volfrac*nelx*nely > 0
48              l1 = lmid;

```

Figura 4: Lineas del código 42 y 47 modificadas

4) Para el cuarto cambio corregimos el valor de las cargas, siendo 5 en este caso, esto se hace cambiando el 2 por un 5 en la línea 22 y de igual forma se realiza esta operación en las líneas 72 y 73, quedando de la siguiente forma.

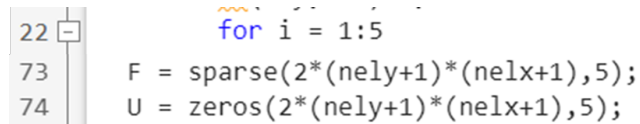
```

22 for i = 1:5

```

```
72 F = sparse(2*(nely+1)*(nelx+1),5);
```

```
73 U = sparse(2*(nely+1)*(nelx+1),5);
```



```

22  for i = 1:5
73  F = sparse(2*(nely+1)*(nelx+1),5);
74  U = zeros(2*(nely+1)*(nelx+1),5);

```

Figura 5: Lineas del código 72 y 73 modificadas

5) Finalmente, para el quinto y ultimo cambio es necesario definir las fuerzas que acabamos de corregir, para esto se añaden en la sección del código donde se definen las cargas y los soportes, aparir de la línea 84, quedando de la siguiente manera.

```
84 F(3222,1)=-1;
```

```
85 F(3782,2)=-1;
```

```
86 F(2662,3)=-1;
```

```
87 F(2942,4)=-1;
```

```
88 F(3502,5)=-1;
```

```
89 fixeddofs =union([560:2*(nely+1):1260],[3920:2*(nely+1):460])
```



```

70 function [U]=FE(nelx,nely,x,penal)
71 [KE] = lk;
72 K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
73 F = sparse(2*(nely+1)*(nelx+1),5);
74 U = sparse(2*(nely+1)*(nelx+1),5);
75 for elx = 1:nelx
76     for ely = 1:nely
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 MBB-BEAM)
84 F(3222,1)=-1;
85 F(3782,2)=-1;
86 F(2662,3)=-1;
87 F(2942,4)=-1;
88 F(3502,5)=-1;
89 fixeddofs = union([560:2*(nely+1):1260],[3920:2*(nely+1):460])

```

Figura 6: Líneas del código 84 a 89 modificadas

Una vez realizados estos cambios podemos obtener la topología de la prótesis de pie, para esto es necesario correr el código especificando las variables que se muestran a continuación, basta con escribir la línea de código en la ventana de comandos.

```
top(72,34,0.33,3.0,1.5)
```

Después de que el código se haya ejecutado, la topología estará lista, obteniendo la siguiente imagen.

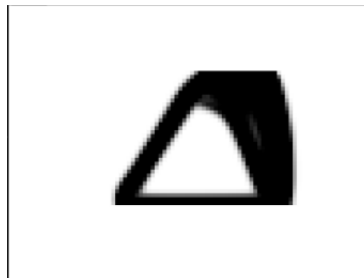


Figura 7: Prótesis caso 1 optimizada

5.2. Caso: 2 y 3

Para el caso 2 y 3 es necesario cambiar la función de “allfixeddoffs”, la cual se encuentra en la sección de definición de cargas y soportes, en la línea 89 quedando de la siguiente forma.

94 fixeddofs =[3920:2*(nely+1):460]

```

70 function [U]=FE(nelx,nely,x,penal)
71 [KE] = lk;
72 K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
73 F = sparse(2*(nely+1)*(nelx+1),5);
74 U = sparse(2*(nely+1)*(nelx+1),5);
75 for elx = 1:nelx
76     for ely = 1:nely
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 MBB-BEAM)
84 F(3222,1)=-1;
85 F(3782,2)=-1;
86 F(2662,3)=-1;
87 F(2942,4)=-1;
88 F(3502,5)=-1;
89 fixeddofs = union([560:2*(nely+1):1260],[3920:2*(nely+1):460])

```

Figura 8: Lineas del código 89 modificada

94 fixeddofs =[560:2*(nely+1):1260]

```

70 function [U]=FE(nelx,nely,x,penal)
71 [KE] = lk;
72 K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
73 F = sparse(2*(nely+1)*(nelx+1),5);
74 U = sparse(2*(nely+1)*(nelx+1),5);
75 for elx = 1:nelx
76     for ely = 1:nely
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 MBB-BEAM)
84 F(3222,1)=-1;
85 F(3782,2)=-1;
86 F(2662,3)=-1;
87 F(2942,4)=-1;
88 F(3502,5)=-1;
89 fixeddofs = union([560:2*(nely+1):1260],[3920:2*(nely+1):460])

```

Figura 9: Lineas del código 89 modificada

Una vez realizados estos cambios podemos obtener la topología de la prótesis de pie, para esto es necesario correr el código especificando las variables que se muestran a continuación, basta con escribir la línea de código en la ventana de comandos.

```
top(72,34,0.33,3.0,1.5)
```

Después de que el código se haya ejecutado, la topología estará lista, obteniendo la siguiente imagen.



Figura 10: Prótesis caso 2 optimizado



Figura 11: Prótesis caso 3 optimizado

Código completo Caso 1

```
%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%  
%%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%  
function top(nelx,nely,volfrac,penal,rmin);  
% INITIALIZE  
x(1:nely,1:nelx) = volfrac;  
loop = 0;  
change = 1.;  
% START ITERATION  
while change > 0.01  
    loop = loop + 1;  
    xold = x;  
% FE-ANALYSIS  
    [U]=FE(nelx,nely,x,penal);  
% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
```

```

[KE] = lk;
c = 0.;
for ely = 1:nely
    for elx = 1:nelx
        n1 = (nely+1)*(elx-1)+ely;
        n2 = (nely+1)* elx +ely;
        dc(ely,elx)=0;
        for i = 1:5
            Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2],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
[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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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 elx = 1:nelx
    for ely = 1:nely
        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):460])
alldofs     = [1:2*(nely+1)*(nelx+1)];
freedofs    = setdiff(alldofs,fixeddofs);
% SOLVING
U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
U(fixeddofs,:)= 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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)];
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This Matlab code was written by Ole Sigmund, Department of Solid      %
% Mechanics, Technical University of Denmark, DK-2800 Lyngby, Denmark.  %
% Please sent your comments to the author: sigmund@fam.dtu.dk          %
%                                                                       %
% The code is intended for educational purposes and theoretical details  %
% are discussed in the paper                                           %
% "A 99 line topology optimization code written in Matlab"            %
% by Ole Sigmund (2001), Structural and Multidisciplinary Optimization, %
% Vol 21, pp. 120--127.                                              %
%                                                                       %
% The code as well as a postscript version of the paper can be        %
% downloaded from the web-site: http://www.topopt.dtu.dk %
%                                                                       %
% Disclaimer:                                                         %
% The author reserves all rights but does not guaranty that the code is %
% free from errors. Furthermore, he shall not be liable in any event   %
% caused by the use of the program.                                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

Caso 2

```

%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
%%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%
function top(nelx,nely,volfrac,penal,rmin);
% INITIALIZE
x(1:nely,1:nelx) = volfrac;

```

```

loop = 0;
change = 1.;
% START ITERATION
while change > 0.01
    loop = loop + 1;
    xold = x;
% FE-ANALYSIS
    [U]=FE(nelx,nely,x,penal);
% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
    [KE] = lk;
    c = 0.;
    for ely = 1:nely
        for elx = 1:nelx
            n1 = (nely+1)*(elx-1)+ely;
            n2 = (nely+1)* elx +ely;
            dc(ely,elx)=0;
            for i = 1:5
                Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2],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
    [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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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 elx = 1:nelx
    for ely = 1:nely
        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 =[560:2*(nely+1):1260]
alldofs    = [1:2*(nely+1)*(nelx+1)];
freedofs   = setdiff(alldofs,fixeddofs);
% SOLVING
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)];
%
%%%%%%%%%%
% This Matlab code was written by Ole Sigmund, Department of Solid
% Mechanics, Technical University of Denmark, DK-2800 Lyngby, Denmark.
% Please sent your comments to the author: sigmund@fam.dtu.dk
%
% The code is intended for educational purposes and theoretical details
% are discussed in the paper
% "A 99 line topology optimization code written in Matlab"
% by Ole Sigmund (2001), Structural and Multidisciplinary Optimization,
% Vol 21, pp. 120--127.
%
% The code as well as a postscript version of the paper can be
% downloaded from the web-site: http://www.topopt.dtu.dk
%
% Disclaimer:
% The author reserves all rights but does not guaranty that the code is
% free from errors. Furthermore, he shall not be liable in any event
% caused by the use of the program.
%%%%%%%%%%

```

Caso 3

```

%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
%%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%
function top(nelx,nely,volfrac,penal,rmin);
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
change = 1.;
% START ITERATION
while change > 0.01
    loop = loop + 1;
    xold = x;
% FE-ANALYSIS
    [U]=FE(nelx,nely,x,penal);
% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
    [KE] = lk;
    c = 0.;
    for ely = 1:nely
        for elx = 1:nelx
            n1 = (nely+1)*(elx-1)+ely;
            n2 = (nely+1)* elx +ely;
            dc(ely,elx)=0;
            for i = 1:5
                Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2; 2*n1+1;2*n1+2],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
    [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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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 elx = 1:nelx
    for ely = 1:nely
        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

```

```

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 = [560:2*(nely+1):1260]
alldofs    = [1:2*(nely+1)*(nelx+1)];
freedofs    = setdiff(alldofs,fixeddofs);
% SOLVING
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)];
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This Matlab code was written by Ole Sigmund, Department of Solid      %
% Mechanics, Technical University of Denmark, DK-2800 Lyngby, Denmark. %
% Please sent your comments to the author: sigmund@fam.dtu.dk          %
%                                                                       %
% The code is intended for educational purposes and theoretical details %
% are discussed in the paper                                           %
% "A 99 line topology optimization code written in Matlab"            %
% by Ole Sigmund (2001), Structural and Multidisciplinary Optimization, %
% Vol 21, pp. 120--127.                                               %
%                                                                       %
% The code as well as a postscript version of the paper can be        %

```

```
% downloaded from the web-site: http://www.topopt.dtu.dk
%
% Disclaimer:
% The author reserves all rights but does not guaranty that the code is
% free from errors. Furthermore, he shall not be liable in any event
% caused by the use of the program.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

6. Conclusiones

Betsaida Alejandra Ruedas Vázquez: La optimización de esta actividad fue la de un diseño estructural enfocado a la parte biomecánica, o sea a una prótesis de pierna. El peso de la aplicación en esta práctica recayó de nuevo en las cargas que caían sobre los soportes. Gracias a este código, nos fue posible llevar la optimización a un nivel más complejo de análisis a nivel estático y dinámico.

Miguel Rodrigo Aguilar Moreno: El desarrollo de la práctica fue bastante interesante ya que esta vez tocó la realización de una prótesis de pierna. Podemos ver que la optimización juega un rol importante a la hora de elegir un diseño final para su fabricación.

Erick Daniel Esquivel Arguelles: Es difícil asimilar como funciona el código del programa al principio, pero a medida que avanzó la práctica entendí que es mejor analizar el código por partes. Las optimizaciones en la estructura no se hacen de forma simultánea, sino que existe todo un proceso que se compone funciones que analizan los parámetros de entrada que resulta en un análisis de la estructura y en una propuesta de optimización de la misma.

Omar Isaí Moreno Cruz: En esta práctica tuvimos que optimizar una prótesis de pie. Para entender la optimización se tuvo que analizar el código por partes, desde el análisis de fuerzas definiendo las cargas hasta que se obtuvo la topología del periférico.

Saul Moises Mendoza Cida: Esta práctica nuevamente fue sencilla, ya que tuvimos que modificar el código anteriormente utilizado. Solo que esta vez realizamos cambios que se muestran en el desarrollo de la práctica. En resumen fueron los siguientes: ciclo lopp, ciclo for, cambiar número de cargas y redefinirlas. Así pudimos optimizar la prótesis.

Referencias

- [1] Campos, T. (2014). Esta prótesis de pierna impresa en 3D, tiene las 3B. Recuperado de: <https://www.xataka.com.mx/investigacion/esta-protesis-de-pierna-impresa-en-3d-tiene-las-3b>. Fecha de consulta: 16/11/2022.
- [Cuevas] Cuevas, E. Vuelve a caminar con una Prótesis de Pierna. Recuperado de: <https://miprotesisdepierna.mx/>, year = "2020", note = Fecha de consulta: 16/11/2022.
- [3] Medina, S. (2020). Elementos De Una Prótesis Para Pierna Básicas. ¿La Mejor Opción? Recuperado de: <https://mediprax.mx/elementos-de-una-protesis-para-pierna-basicas-la-mejor-opcion/>. Fecha de consulta: 16/11/2022.
- [4] Varela, J. (2020). Prótesis de Pierna. Recuperado de: <https://protesisdepierna.mx/>. Fecha de consulta: 16/11/2022.

Los créditos de las fotografías pertenecen a sus respectivos autores.