Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Laboratorio de Biomecánica

Práctica 4

Refuerza del cable de un teleférico

1904701	Bernardo Canul Aguilar	IMTC
1904820	Sylaid Pérez Oviedo	IMTC
1910351	Daniel Tudón González	IMTC
1991843	Javier Rangel Elizondo	IMTC
1992120	Francisco Adrián Castillo Herrera	IMTC

Hora: N5

Brigada: 509

Fecha: 27 octubre 2022

Ciudad Universitaria, San Nicolás de la Garza, N.L

Práctica #4: Refuerza del cable de un teleférico

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).

Marco Teórico

El teleférico es un sistema de transporte no tripulado aéreo constituido por cabinas colgadas de una serie de cables que se encargan de hacer avanzar a las unidades a través de las estaciones. Cuando las cabinas van por tierra se denomina funicular.

El sistema de cada teleférico está compuesto por uno o más cables, el primer cable está fijo y sirve para sostener las cabinas, el segundo está conectado a un motor ubicado en la estación, el cual hace mover las cabinas.

Algunos teleféricos usan dos cabinas por tramo a fin de crear un contrapeso. Otros sistemas más complejos tienen varias cabinas suspendidas simultáneamente en cada dirección. Este transporte se usa en zonas con grandes diferencias de altura, donde el acceso por carretera o ferrocarril resulta difícil.

Un motor eléctrico impulsa al teleférico. Este mismo se encuentra en el centro de la estación y actúa directamente en la polea del cable impulsando así el cable y los vehículos. Los frenos, engranajes y motores no son necesarios en los vehículos individuales, ya que estos únicamente son transportados, acelerados y frenados mediante el cable, el cual es el componente que diferencia los funiculares y teleféricos sobre otros tipos de transporte.

El diámetro del cable, los cordones del cable, la longitud del empalme, el sistema de sujeción de los extremos del cable son datos que se calculan en el proceso de diseño por ingenieros donde la precisión es muy importante por los costes elevador que tienen al errar en la fabricación de un cable.

Estado del arte

Título del documento	“Refuerza de un cable de un teleférico”
Fuente Bibliográfica	Leitner. Elementos de un teleférico
Objetivo	Con el código de optimización se obtiene un mejor cable para un

	teleférico, con mejores características
Contenido	Ejemplo de código de programa de 99 líneas ejecutado en matlab, se busca mejorar las especificaciones. Se presenta código principal, optimizador y código del método de elementos finitos.
Palabras clave	Refuerzo de cable de teleférico, Matlab, análisis de formas, optimización.
Conclusión	Este documento muestra una propuesta con el ejemplo de la optimización, utilizando un código en Matlab.

Procedimiento de la programación

Igualmente, como en las prácticas pasadas, se utiliza el mismo código de la práctica #1, con ajustes específicos para la solución del problema.

Código original:

```
%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function plabbiom(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;
    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) = -penal*x(ely,elx)^(penal-1)*Ue *KE*Ue;
    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
```

```
49    end
50    %%%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%%
51    function [dcn]=check(nelx,nely,rmin,x,dc)
52    dcn=zeros(nely,nelx);
53    for i = 1:nelx
54       for j = 1:nely
55       sum=0.0;
56       for k = max(i-round(rmin),1):min(i+round(rmin),nelx)
57         for l = max(j-round(rmin),1):min(j+round(rmin), nely)
58             fac = rmin-sqrt((i-k)^2+(j-l)^2);
59             sum = sum+max(0,fac);
60             dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
61         end
62       end
63       dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
64       end
65    end
66    %%%%%%%%%% FE-ANALYSIS %%%%%%%%%%%%
67    function [U]=FE(nelx,nely,x,penal)
68    [KE] = lk;
69    K = sparse(2*(nelx+1)*(nely+1), 2*(nelx+1)*(nely+1));
70    F = sparse(2*(nely+1)*(nelx+1),1); U =sparse(2*(nely+1)*(nelx+1),1);
71    for ely = 1:nely
72        for elx = 1:nelx
```

```
73            n1 = (nely+1)*(elx-1)+ely;
74            n2 = (nely+1)* elx +ely;
75            edof = [2*n1-1; 2*n1; 2*n2-1; 2*n2; 2*n2+1;2*n2+2;2*n1+1; 2*n1+2];
76            K(edof,edof) = K(edof,edof) + x(ely,elx)^penal*KE;
77        end
78    end
79    % DEFINE LOADSAND SUPPORTS(HALF MBB-BEAM)
80    F(2,1) = -1;
81    fixeddofs = union((1:2:2*(nely+1)),(2*(nelx+1)*(nely+1)));
82    alldofs = (1:2*(nely+1)*(nelx+1));
83    freedofs = setdiff(alldofs,fixeddofs);
84    % SOLVING
85    U(freedofs,:) = K(freedofs,freedofs) \F(freedofs,:);
86    U(fixeddofs,:)= 0;
87    %%%%%%%%%% ELEMENT STIFFNESS MATRIX %%%%%%%
88    function [KE]=lk
89    E = 1.;
90    nu = 0.3;
91    k=[ 1/2-nu/6 1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
92    -1/4+nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8];
93    KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
94        k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
95        k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
96        k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
```

```
97        k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
98        k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
99        k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
100       k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
```

Código final con las modificaciones:

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, OCTOBER 1999 %%%
function pruebap4(nelx,nely,volfrac,penal,rmin);
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
for ely = 1:nely
for elx = 1:nelx
if ely>21
if elx<21
passive(ely,elx) = 1;
elseif elx>41
passive(ely,elx)=1;
else
passive(ely,elx) = 0;
end
end
end
end
x(find(passive))=0.001;
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=1:2
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
6
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,passive);
% 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,passive)
    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)))));
    xnew(find(passive))=0.001;
    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),2); U = sparse(2*(nely+1)*(nelx+1),2);
    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;
    7
    end
    end
```

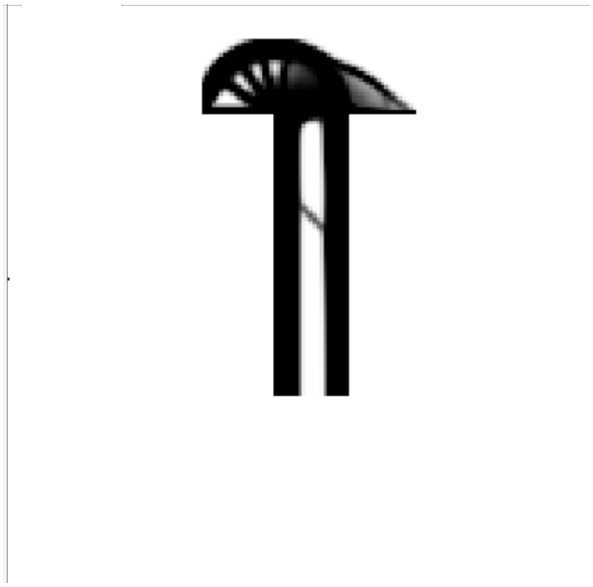
```
% DEFINE LOADSAND SUPPORTS(HALF MBB-BEAM)
F(40,1) = -1.; F(9760,2)=1.;
fixeddofs = 2*(nely+1):2*(nely+1):2*(nelx+1)*(nely+1);
alldofs = [1:2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);
% SOLVING
U(freedofs,:) = K(freedofs,freedofs) \F(freedofs,:);
U(fixeddofs,:)= 0;
%%%%%%%%%% ELEMENT 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)];
```

Implementación del programa

Se le dieron los siguientes valores al programa de (60,100,0.33,3,1.5) en la ventana de comando de Matlab, se observa:

```
Command Window
>> pruebap4(60,100,0.33,3,1.5)
```

Se presentó la siguiente figura como resultado final:



Conclusiones

-Sylaid Pérez Oviedo

En conclusión una vez que se hizo el trabajo de laboratorio, se mostró el trabajo el cual se hizo por medio de la codificación de 99 líneas, esto utilizando Matlab; se observamos que la época para la ejecución de ésta ha sido superior a la anteriores por el proceso que tuvo que llevar el programa para optimizar los esfuerzos, además de ver los espacios en blanco que son recursos pasivos que requieren ser tomados presente para el diagrama. Comenzando con lo cual hicimos tenemos la posibilidad de concluir que aunque se crea que algo no se toma en cuenta dentro de un sistema de esfuerzos por ser un espacio en blanco, esto no debería ser de esta forma, debemos darle el valor para el diseño óptimo del diagrama.

-Francisco Adrián Castillo Herrera

En esta práctica se volvió a hacer uso del código de 99 líneas para generar una optimización topológica. En este caso se escogió un elemento relacionado con refuerzos de cables, para lo cual se requiere restringir una zona pasiva. Esto significa que la densidad relativa se aproxima a cero, por lo que no debería haber material. Esta función se usa especialmente para dejar espacio a otros componentes (tubos por ejemplo), siendo este caso en específico para el cable.

-Daniel Tudón González

En esta práctica se realizó un refuerzo del cable de un teleférico, con el cual se puede observar el cambio en la geometría con una menor masa, con lo que se puede observar la utilidad del software para poder realizar operaciones de una

manera más rápida que la realización de las operaciones que se realizaría por medio de de operaciones hechas a mano, con lo que se puede hacer tanto en 2 dimensiones como en 3 dimensiones. Esto tiene mucha aplicación dentro del área de la mecánica, ya que con eso se puede conocer los esfuerzos de von Mises máximos y compararlos con el esfuerzo de cedencia del material en cuestión. Esto sin tener que llegar a obtener una ecuación diferencial que describa el comportamiento de las fuerzas y la flexión en función de las coordenadas por las cuales se genera la geométrica.

-Javier Rangel Elizondo

En esta práctica, se vieron muchos elementos que se aplican en la optimización. Un teleférico es un sistema de transporte no tripulado aéreo constituido por cabinas colgadas de cables que ayudan a avanzar a la cabina a través de distintas estaciones. En cualquier objeto se puede hacer una optimización y es mucho más fácil cuando ya existen codificaciones como la de las 99 líneas. Existen elementos pasivos que ayudan a que se hagan estructuras pero con diferentes partes. El código se hizo en Matlab; cada vez es más fácil el saber como cambiar el código y poner uno adecuado.

-Bernardo Canul Aguilar

Los teleféricos son medios de transporte que, al menos en la ciudad de monterrey, no son vistos de forma común, por lo cual fue interesante poder investigar más acerca de este transporte y sobre su funcionamiento, añadido a esto, tuvimos la oportunidad de optimizar el cable del teleférico, de forma que sea más resistente y pueda proteger de mejor manera a las personas que lo utilicen. Al igual que en el resto de prácticas, se ha seguido utilizando el código para la optimización de un objeto, lo cual es un claro ejemplo de cómo la optimización puede ser aplicada a cualquier objeto, desde algo muy común hasta objetos muy específicos, siempre y cuando se apliquen los parámetros de forma correcta.