Práctica 4 Refuerza del cable de un teleférico

Uriel G. Ernesto A. Marcela O. Ana L. Diego O.

25 de octubre de 2022

Resumen

En esta práctica seguiremos trabajando con los codigos topológico y que sucede cuando lo optimizamos, así mismo se mostrará un ejemplo de cómo se vería un código despues de la optimización.

1. Introducción

La idea de esta practica es desarrollar en el estudiante la capacidad de análisis, implementación y solución del problema propuesto, mediante la optimizacion de codigos para la realizacion de analisis mas detallados en la elaboración de proyectos.[1].

2. Instrucciones

El estudiante implementará un sistema computacional MATLAB para calcular esfuerzos y deformaciones de una geometría basados en el modelado para los cuales la solución analítica simple

3. Nombre y definicion de la forma Geometria

Para el desarrollo la práctica de laboratorio. Con el fin de optimizar su rendimiento, usaremos una figura libre, como la practica anterior que se optimizo un codigo muy similar, utilizando la lógica del código de Matlab para colocación de cargas, apoyos y fuerzas dentro de un espacio de diseño propuesto. Usaremos un letro sobre dos bases.

4. Estado del arte

UEl teleférico es un aparato móvil cuyo principal objetivo es transporter a una persona desde un punto a hasta un punto b. Lo que lo diferencia entre el resto de medios de transporte es que, por lo general, se hace uso de este medio cuando los terrenos son muy difíciles de navegar como lo pueden ser acantilados, montañas o cañones.

En estos casos especiales, un teleférico es de muchísima ayuda para transportar a las personas por el aire sin hacer uso de un vehiculo aereo como un avión o un helicóptero. La cabina o góndola del teleférico se encuentra suspendida en un cable que es por el que se traslada de un punto a otro. Dicho cable esta anclado a dos estaciones de las cuales una se encuentra al tope de la montaña o lugar que se intenta subir y en la base.

Las cabinas de los teleféricos cuelgan de forma en que queden suspendidos de manera vertical. En los sistemas de teleféricos se tienen dos de estos para agilizar un poco el tráfico entre ambos puntos ya que son un medio de transporte seguro, pero lento.

Evidentemente, estos artefactos deben de pasar por muchos filtros de seguridad antes de que sea posible ponerlos en marcha en un punto turístico o de interés. Para eso se puede utilizar también la optimización topológica. Con

este método se puede conseguir un buen diseño para cada una de sus piezas ya sea la cabina las poleas de guía, la polea motriz, el mástil, el cable portante o el cable tractor.

Figura 1: Parte 1 del codigo

Así como en el resto de las prácticas, se estará utilizando el software MATLAB, el cual puede ser utilizado para muchos aspectos de la ingeniería y en esta ocasión, será de gran utilidad para poder mejorar el diseño

5. Propuesta de diseño de la geometria, alcance y limitaciones

Mejorar el diseño del cable de un teleférico, con el objetivo de que sea más Seguro, estable, barato y fácil de producir

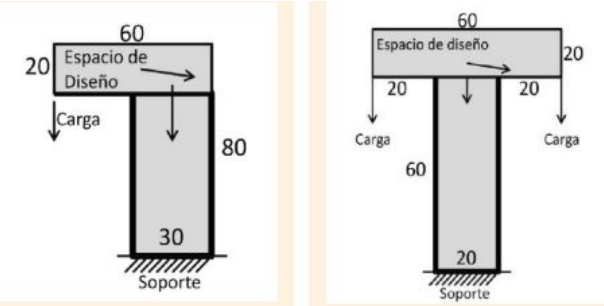

Figura 2: Parte 1 del codigo

En esta práctica, se realizarán dos ejercicios que mejorarán la estructura del teleférico. El primero de diseños tendrá un espacio de diseño superior al espacio de diseño de la segunda estructura que se mejorará.El diseño con menos espacio soportará sólo una carga mientras que el que tiene mayor espacio tendrá más cargas.

6. Codigo de programación

P4.m

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function P4(nelx,nely,volfrac,penal,rmin)
    % INITIALIZE
        x(1:nely,1:nelx) = volfrac;
        %DECLARANDO VACIO
        for ely = 1:nely
            for elx = 1:nelx
                if ely>21
                    if elx<31
                        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
```

Figura 3: Parte 1 del codigo

P4.m

```
            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
            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
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)
```

Figura 4: Parte 2 del codigo

P4.m

```
57          lmid = 0.5*(l2+l1);
58          xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
59          xnew(find(passive)) = 0.001;
60          if sum(sum(xnew)) - volfrac*nelx*nely > 0;
61              l1 = lmid;
62          else
63              l2 = lmid;
64          end
65      end
66  end
67  %%%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%%
68  function [dcn]=check(nelx,nely,rmin,x,dc)
69      dcn=zeros(nely,nelx);
70      for i = 1:nelx
71          for j = 1:nely
72              sum=0.0;
73              for k = max(i-round(rmin),1): min(i+round(rmin),nelx)
74                  for l = max(j-round(rmin),1): min(j+round(rmin), nely)
75                      fac = rmin-sqrt((i-k)^2+(j-l)^2);
76                      sum = sum+max(0,fac);
77                      dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
78                  end
79              end
80              dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
81          end
82      end
83  end
```

Figura 5: Parte 3 del codigo

P4.m

```
%%%%%%%%%% 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;
        end
    end
% DEFINE LOADSAND SUPPORTS(HALF MBB-BEAM)
F(40,1)=-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;
end
%%%%%%%%%% 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];
```

Figura 6: Parte 4 del codigo

```
%%%%%%%%%% 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)];
end
```

Figura 7: Parte 5 del codigo

7. Resultado de la optimización

Para la primera prueba se utilizarán los valores P1(30,10,0.5,3,1.5), los resultados fueron los siguientes.

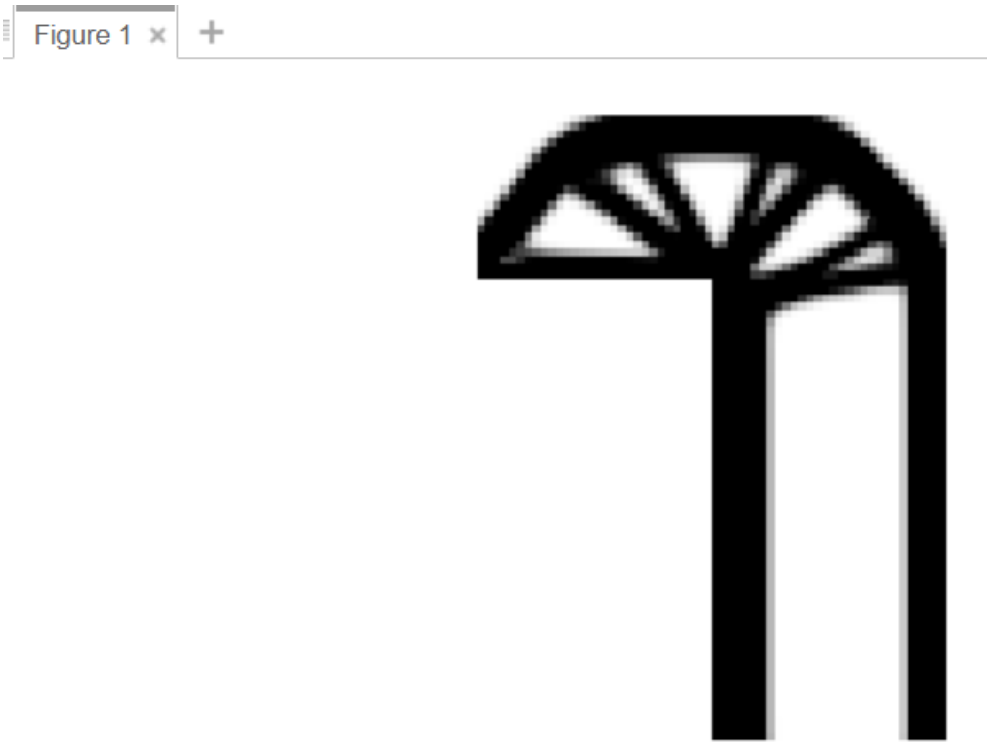

Figura 8: Resultados

8. Codigo de programación

P4_Caso2.m

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function P4_Caso2(nelx,nely,volfrac,penal,rmin)
    % INITIALIZE
        x(1:nely,1:nelx) = volfrac;
        %DECLARANDO VACIO
        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;
```

Figura 9: Parte 1 del codigo

P4_Caso2.m

```
29          c = 0.;
30          for ely = 1:nely
31              for elx = 1:nelx
32                  n1 = (nely+1)*(elx-1)+ely;
33                  n2 = (nely+1)* elx +ely;
34                  dc(ely,elx) = 0.;
35                  for i =1:2
36                      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);
37                      c = c + x(ely,elx)^penal*Ue'*KE*Ue;
38                      dc(ely,elx) = dc(ely,elx)-penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
39                  end
40              end
41          end
42          % FILTERING OF SENSITIVITIES
43          [dc] = check(nelx,nely,rmin,x,dc);
44          % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
45          [x] = OC(nelx,nely,x,volfrac,dc,passive);
46          % PRINT RESULTS
47          change = max(max(abs(x-xold)));
48          disp(['It.:' sprintf('%4i',loop) 'Obj.:' sprintf('%10.4f',c) ...
49          ' Vol.: ' sprintf('%6.3f',sum(sum(x))/(nelx*nely)) ...
50          ' ch.: ' sprintf('%6.3f',change )])
51          % PLOT DENSITIES
52          colormap(gray); imagesc(-x); axis equal; axis tight; axis off;pause(1e-6);
53      end
54  end
55  %%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%
56  function [xnew]=OC(nelx,nely,x,volfrac,dc,passive)
```

Figura 10: Parte 2 del codigo

P4_Caso2.m

```
57        l1 = 0; l2 = 100000; move = 0.2;
58        while (l2-l1 > 1e-4)
59            lmid = 0.5*(l2+l1);
60            xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
61            xnew(find(passive)) = 0.001;
62            if sum(sum(xnew)) - volfrac*nelx*nely > 0;
63                l1 = lmid;
64            else
65                l2 = lmid;
66            end
67        end
68    end
69    %%%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%%
70    function [dcn]=check(nelx,nely,rmin,x,dc)
71        dcn=zeros(nely,nelx);
72        for i = 1:nelx
73            for j = 1:nely
74                sum=0.0;
75                for k = max(i-round(rmin),1): min(i+round(rmin),nelx)
76                    for l = max(j-round(rmin),1): min(j+round(rmin), nely)
77                        fac = rmin-sqrt((i-k)^2+(j-l)^2);
78                        sum = sum+max(0,fac);
79                        dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
80                    end
81                end
82                dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
83            end
84        end
```

Figura 11: Parte 3 del codigo

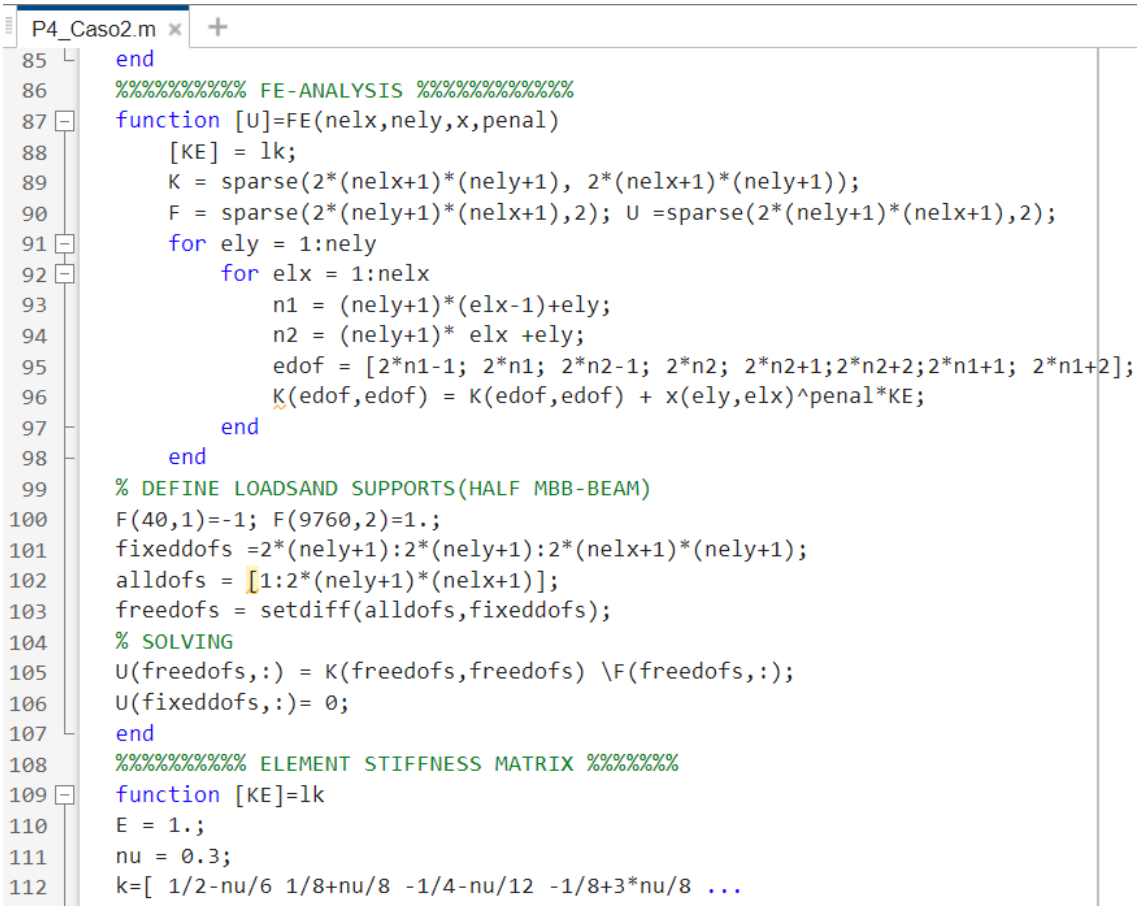

Figura 12: Parte 4 del codigo

```
108 %%%%%%%%%% ELEMENT STIFFNESS MATRIX %%%%%%%
109 function [KE]=lk
110 E = 1.;
111 nu = 0.3;
112 k=[ 1/2-nu/6 1/8+nu/8 -1/4-nu/12 -1/8+3*nu/8 ...
113 -1/4+nu/12 -1/8-nu/8 nu/6 1/8-3*nu/8];
114 KE = E/(1-nu^2)*[ k(1) k(2) k(3) k(4) k(5) k(6) k(7) k(8)
115 k(2) k(1) k(8) k(7) k(6) k(5) k(4) k(3)
116 k(3) k(8) k(1) k(6) k(7) k(4) k(5) k(2)
117 k(4) k(7) k(6) k(1) k(8) k(3) k(2) k(5)
118 k(5) k(6) k(7) k(8) k(1) k(2) k(3) k(4)
119 k(6) k(5) k(4) k(3) k(2) k(1) k(8) k(7)
120 k(7) k(4) k(5) k(2) k(3) k(8) k(1) k(6)
121 k(8) k(3) k(2) k(5) k(4) k(7) k(6) k(1)];
122 end
```

Figura 13: Parte 5 del codigo

9. Resultado de la optimización 2

Para la segunda parte se hara la prueba donde se utilizarán los valores P1(30,10,0.5,3,1.5), los resultados fueron los siguientes.

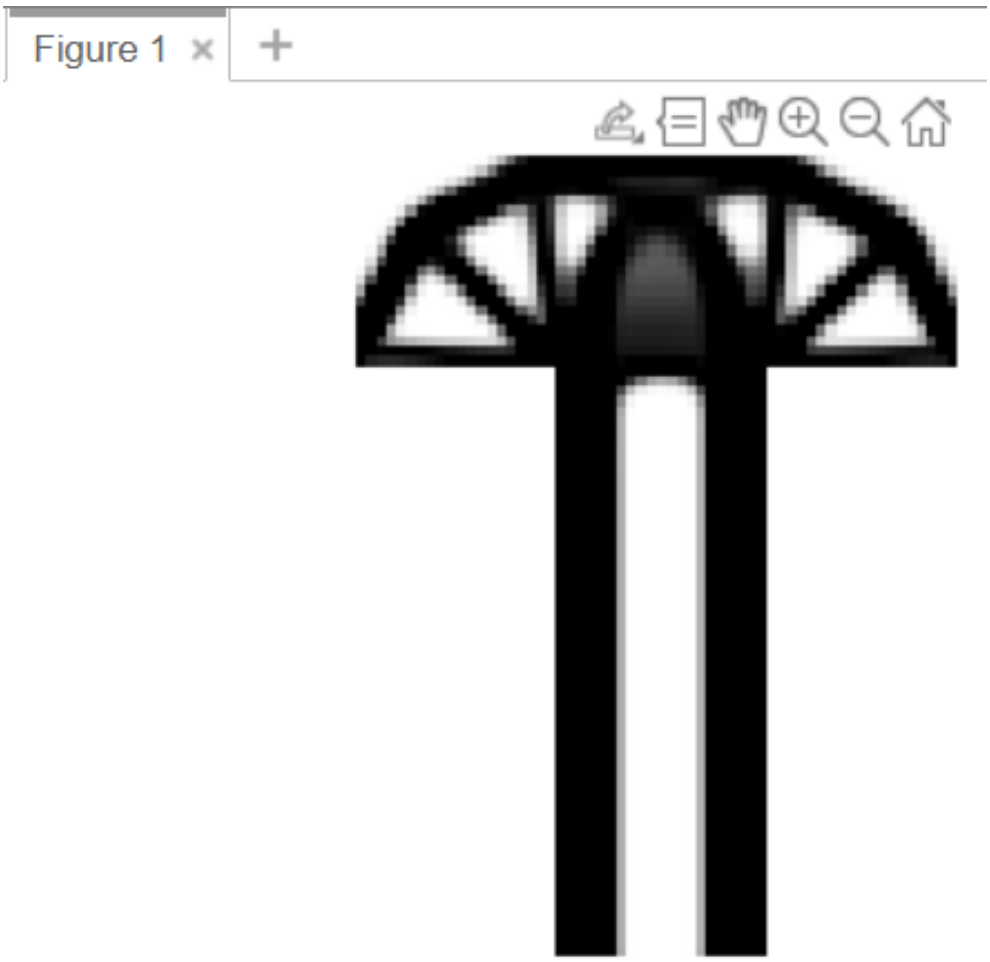

Figura 14: Resultados

10. Conclusiones

- Diego O.

Al término de esta práctica puedo concluir acerca de que aún teniendo de apoyo herramientas muy autonomas como lo es Matlab no significa que siempre será una labor simple manipularla, si bien simplifica demasiado procedimientos de cálculos algebraicos y matemáticos se dan casos como el de esta práctica en la cual su grado de complejidad aumentaba por el hecho de que la precisión de los resultados tenía que ser mayores para cada una de las estructuras requeridas en el problema planteado de la práctica.

- Ana L.

El objetivo de esta práctica número cuatro fue de implementar un sistema computacional para calcular los esfuerzos y deformaciones de una geometría de un cable de un teleférico, para el cual se realizaron dos casos. Esto se realizo por medio del software de modelado y simulacion MATLAB. Gracias a esta practica aprendí como optimizar este tipo de estructura y cómo generar sus resultados por medio de la plataforma ya mencionada

- Uriel G.

En conclusion puedo decir que en esta practica el software MATLAB tomo mas tiempo de lo normal debido a que ahora era necesario el calculo exacto de los esfuerzos que deberían tener cada una de las estructuras que son requeridas en la practica, en cada una de ellas es indispensable el conocer el tipo de carga ademas de su valor ya que esto ayuda a elegir el material indicado para construirlas. La primera estructura como se menciona es para un solo teleférico, es por eso que es un poco menos complicada, en cambio, la segunda estructura es mas estructural debido a que es para la carga de dos teleféricos a la vez, asi generando una tipo T en el diseño.

- Marcela O.

Al concluir con nuestra práctica, nos dimos cuenta de otra aplicación que tiene el software MATLAB, aunque en esta ocasión nos tardamos un poco más, pues necesitamos más cálculos, lo cuál hace un poco más tardado y/o complejo el procedimiento. Al ser un teleférico, se necesita mucha precisión por el manejo y uso que se le da, por lo cuál tenemos que ser más precisos en las operaciones y el diseño. En esta práctica pudimos aprender más sobre las herramientas de MATLAB y sus aplicaciones como en el teleférico.

- Ernesto A.

La práctica se realizó de manera satisfactoria ya que se cumplió con el objetivo, el cual era la optimización por medio del código proporcionado, por supuesto, realizando las modificaciones necesarias. En un principio se dificultó un poco la visualización de los resultados, pero analizando lo obtenido, se le pudo dar una interpretación correcta. La práctica finalizó de manera satisfactoria, con ambas respuestas optenidas de ambas optimizaciones.

Referencias

[1] Anonimo. ¿que es un teleferico?