Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Laboratorio de Biomecánica

Practica 1. Optimización topológica.

1992212	**Jose Eduardo Mendoza Ziga**	**IMTC**
1992298	**Manuel Antonio Ulloa Méndez**	**IMTC**
1733433	**Raúl Alejandro Tamez González**	**IMTC**
1992043	**Eduardo Villarreal Gámez**	**IMTC**
1818785	**Oiram Colunga Bernal**	**IMTC**
1822072	**Brandon Arturo Solano Arias**	**IMTC**
1916426	**Daniel Alberto Acosta Banda**	**IMTC**
Maestro:	**Dra. Yadira Moreno Vera**	
Hora:	**V1**	
Grupo:	**204**	

Fecha de entrega 05 de septiembre, 2022
Cd. Universitaria, San Nicolás de los Garza

Objetivo

Conocer cada una de las secciones que integran un código de optimización topológica, como se debe crear el archivo (.m) en MATLAB y como se ejecuta el análisis.

Estado del arte

Hace aproximadamente cuatro décadas Schmidt [1] propuso una idea revolucionaria que dio origen a una nueva disciplina: los ingenieros, en general, tratan de diseñar objetos o sistemas de coste mínimo que durante su vida útil deben ser capaces de resistir las solicitaciones máximas que se puedan producir; por tanto, los problemas de diseño (óptimo) podrían plantearse de forma sistemática en términos de problemas de minimización con restricciones, y podrían resolverse mediante técnicas de programación no lineal utilizando ordenadores digitales de alta velocidad. Desde entonces, la optimización de formas y dimensiones en ingeniería estructural se ha planteado habitualmente mediante formulaciones de mínimo peso, con restricciones no lineales impuestas con el fin de limitar los valores admisibles de los campos de desplazamientos y tensiones [2] [3] [4].

Sin embargo, desde que Bendsøe y Kikuchi desarrollaron los conceptos básicos [5] en 1988, los problemas de optimización topológica se han planteado tradicionalmente mediante formulaciones de máxima rigidez.

Con este tipo de planteamientos se pretende distribuir una cantidad predeterminada de material en un recinto de forma que se maximice la rigidez (se minimice la energía de deformación) de la pieza resultante para un determinado estado de carga [6]. De esta forma se evita tener que trabajar con numerosas restricciones altamente no lineales, habida cuenta del elevado número de variables de diseño que es consustancial a los problemas de optimización topológica. A cambio, no es posible contemplar múltiples estados de carga, y las formulaciones de máxima rigidez conducen —en principio— a problemas intrínsecamente mal planteados [7, 8], cuyas soluciones oscilan indefinidamente al refinar la discretización.

La formulación SIMP (solid isotropic material with penalty) [7] [8] [9] constituye el método de máxima rigidez más extendido en la actualidad. En esta formulación hay una variable de diseño por elemento (la densidad relativa, o complemento a uno de la porosidad) cuyo valor adimensional debe estar comprendido entre 0 y 1. El objetivo es hallar los valores de las variables de diseño (la cantidad de material que hay que disponer en cada elemento) de forma que se minimice una función objetivo fuertemente no lineal (la energía de deformación) con una sola restricción sencilla (la cantidad total de material está limitada superiormente por un porcentaje predefinido, llamado factor de llenado, del volumen del recinto).

La formulación SIMP es fácil de implementar en el marco de un programa de elementos finitos, y existen procedimientos explícitos de actualización [7] [8] de las densidades relativas, lo que facilita la resolución del problema de minimización. Aparentemente, todo son ventajas. Sin embargo, se producen sistemáticamente inestabilidades numéricas que impiden la convergencia de este tipo de métodos y que dan lugar a configuraciones en damero de carácter oscilante.

Además, una posible solución en la que existan zonas extensas con valores intermedios de la densidad relativa (un material poroso) es considerada —por motivos prácticos y con carácter general— un resultado final indeseable. Por todo ello se imponen comúnmente penalizaciones o restricciones adicionales [7] [8], tanto para estabilizar la solución como para impedir la existencia de valores intermedios de la densidad relativa. Por último, al resultado final se le suele aplicar un filtrado de imagen que favorezca su interpretación.

Las soluciones obtenidas finalmente con estas correcciones semejan estructuras de barras [7]. Sin embargo, este tipo de soluciones son cuestionables, ya que dependen de numerosos parámetros (factor de llenado, grado de discretización, penalizaciones aplicadas, filtrado de imagen, etc.). Además, el diseño final puede no ser aceptable en la práctica, ya que la formulación no tiene en cuenta el estado tensional de la pieza. Y es evidente que el diseño más rígido que se puede construir con una cierta cantidad de material no tiene por qué coincidir con el diseño más económico (en términos de material) que es capaz de resistir unas determinadas solicitaciones. Con el fin de obviar estos inconvenientes se ha desarrollado la formulación MPRT [8] [10] de mínimo peso con restricciones en tensión para la optimización topológica de estructuras mediante el Método de Elementos Finitos.

Marco Teórico

Un código de optimización de topología de 99 líneas escrito en Matlab

La propuesta de análisis de forma de programación es una implementación compacta en MATLAB de un código de optimización topológica para la minimización del cumplimiento de estructuras cargadas estáticamente obtenido de un artículo educacional de parte del autor Ole Sigmund (2001). *En este caso se realizará para vigas (geometría).*

El número total de líneas de entrada de MATLAB para este caso es de 99, incluidos un optimizador y una subrutina de elementos finitos. De tal modo, el código se dispondrá entonces de 36 líneas para el programa principal, 12 líneas para el optimizador basado en criterios, 16 líneas para un filtro de independencia de malla y 35 líneas para el código de elemento finito.

Un número de simplificaciones son introducidas en el código de MATLAB. Primero se asume que el dominio de diseño es rectangular y discretizada por elementos cuadrados finitos. De

esta forma el número de elementos y nodos es simple y la relación de aspecto de la estructura se da por la relación de elementos en las direcciones horizontal (nelx) y vertical (nely)

De forma general, un problema de optimización topológica basado en el enforque de la ley de potencia, donde se busca minimizar el cumplimiento se puede escribir de la siguiente manera:

$$\left.\begin{aligned} \min_{\mathbf{x}}: \quad & c(\mathbf{x}) = \mathbf{U}^T\mathbf{K}\mathbf{U} = \sum_{e=1}^{N} (x_e)^p \, \mathbf{u}_e^T \, \mathbf{k}_0 \, \mathbf{u}_e \\ \text{subject to}: \quad & \frac{V(\mathbf{x})}{V_0} = f \\ : \quad & \mathbf{K}\mathbf{U} = \mathbf{F} \\ : \quad & \mathbf{0} < \mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{1} \end{aligned}\right\},$$

Donde U y F son los vectores globales de desplazamiento y fuerza, respectivamente, K es la matriz de rigidez global, Ue y Ke son el vector de desplazamiento del elemento y la matriz de rigidez , respectivamente, x es el vector de variables de diseño, xmin es un vector de densidades relativas mínimas, N (nelx X nely) es el número de elementos usados para discretizar el domino de diseño, p es el poder de penalización ,V(x) y V0 son el volumen de material y volumen de dominio de diseño, respectivamente y por último f es una fracción de volumen prescrita.

Desarrollo de la práctica

Procedimiento de la programación

El código de MATLAB es constituido como un código de optimización topológica estándar. El programa principal es llamado desde la consola de MATLAB por la siguiente línea:

top(nelx,nely,volfrac,penal,rmin)

Como se mencionó con anterioridad, nelx y nely son el número de elementos en las direcciones vertical y horizontal, respectivamente, voltfrac es una fracción de volumen, penal es el poder de penalización y rmin es el tamaño del filtro (dividido por el tamaño del elemento). Otras variables, así como las condiciones de frontera están también definidas dentro del código (pueden ser editadas). Por cada iteración del ciclo de optimización topológica, el código genera una imagen de la distribución de densidad actual.

Para el código dado, introduciendo la siguiente línea de entrada en MATLAB:

top(60,20,0.5,3.0,1.5)

da como resultado una distribución de densidad que representa una optimización topológica de una viga MBB, esto se aprecia en la Figura 1.

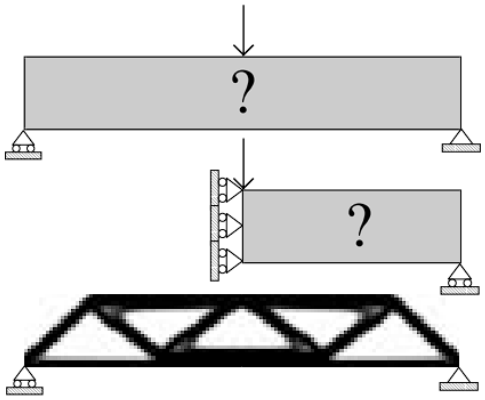

Figura 1: Distribución de densidad que representa una optimización topológica de una viga

La estructura superior representa el domino completo del diseño, la del medio representa medio domino de diseño con condiciones de frontera simétricas y la inferior representa la topología resultante de la viga optimizada (ver figura 1).

Las condiciones de frontera por defecto corresponden a la mitad de la viga MBB (figura 1). La carga es aplicada verticalmente en la esquina superior izquierda, donde hay condiciones simétricas de frontera a lo largo del borde izquierdo y la estructura que es soportada horizontalmente en la esquina inferior derecha.

Implementación o desarrollo de la programación en sus diferentes vistas

Ahora se describirá por secciones y de forma general las líneas del programa, las cuales componen el desarrollo completo de la programación.

- Programa principal (líneas 1-36):

  En esta parte se comienza distribuyendo el material uniformemente en el dominio del diseño (línea 4). Después de ciertas inicializaciones, el ciclo principal inicia con la llamada a la subrutina de elemento finito (línea 12) la cual regresa el vector de desplazamiento U. La subrutina de matriz de rigidez del elemento es solo llamada una vez (línea 14) al tratarse de un material sólido. Después de ello, un ciclo sobre todos los elementos determina la función objetivo y las sensibilidades (líneas 16-24). El análisis de sensibilidad es seguido por una llamada al filtro de independencia de malla (línea 26) y el optimizador de criterios óptimos (línea 28). El cumplimiento actual, así como otros

parámetros son impresos (líneas 30-33) y la distribución de densidad resultante de grafica (línea 35). El ciclo principal se termina si el cambio en las variables de diseño (línea 30) es menos del 1%, de lo contrario los pasos de arriba se vuelven a repetir.

- Optimizador basado en criterios de optimización (líneas 37-48)

El optimizador encuentra las variables de diseño actualizadas (líneas 37-48). Ya que el volumen material (sum(sum(xnew))) es una función monótonamente decreciente del multiplicador de Lagrange (retraso), el valor del multiplicador lagrangiano que satisface la restricción de volumen puede encontrarse por un algoritmo de bisección (líneas 40-48). El algoritmo de bisección se inicializa adivinando unos límites l1 inferior y l2 superior para el multiplicador lagrangiano (línea 39). El intervalo que limita el multiplicador lagrangiano se reduce repetidamente a la mitad hasta que su tamaño es menos que los criterios de convergencia (línea 40).

- Filtrado de independencia de malla (líneas 49-64)

El filtro de independencia de malla funciona al modificar los elementos sensitivos (líneas 49-64).

- Código de elemento finito (líneas 65-99)

El código de elemento finito se escribe en las líneas 65-99. La matriz de rigidez global está formada por un bucle sobre todos los elementos (líneas 70-77). Como sucedía en los principales programas, las variables n1 y n2 indican el número de nodos de elementos de la parte superior izquierda y derecha en números de nodos globales y son usados para insertar la matriz de rigidez del elemento en los lugares correctos en la matriz de rigidez global.

Como se mencionó anteriormente, tanto los nodos como los elementos son columnas numeradas sabiamente de izquierda a derecha. Es más, cada nodo tiene dos grados de libertad (horizontal y vertical), así el comando F (2,1) =-1. (línea 79) aplica una fuerza vertical unitaria en la esquina superior izquierda. Los apoyos se implementan eliminando grados fijos de libertad de las ecuaciones lineales. Matlab lo puede hacer con la línea:

```
84U(freedofs,:) = K(freedofs,freedofs) \
F(freedofs,:);
```

Donde “freedofs” indican, los grados de libertad que no están restringidos. En general, es más fácil definir los grados de libertad que son fijos (fixeddofs) a partir de que los freedofs se encuentran automáticamente usando el operador de Matlab “setdiff” que encuentra los grados de libertad libres como la diferencia entre todos los grados de libertad y los grados de libertad fijos (línea 82).

La matriz de rigidez se calcula en las líneas 86-99. El módulo de Young (E) y el ratio de Poisson (nu) pueden modificarse en las líneas 88 y 89.

Código para la simulación

```
%%% 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;
      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
  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),1); U = zeros(2*(nely+1)*(nelx+1),1);
  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(2,1) = -1;
  fixeddofs   = union([1:2:2*(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)];
```

Al ejecutar el código en MATLAB con la línea de entrada top (60,20,0.5,3.0,1.5) propuesta en un inicio, se obtiene la optimización topológica de la viga (ver figura 2).

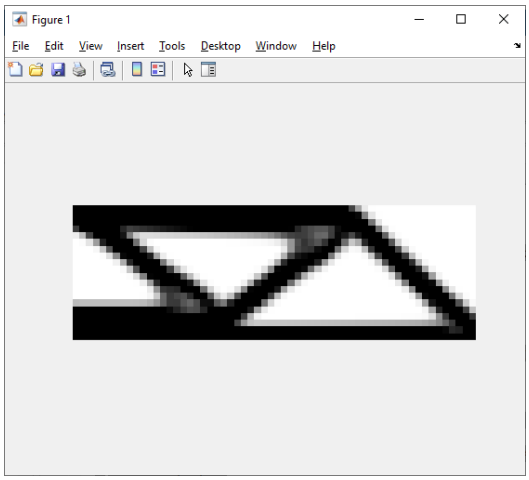

Figura 2: Resultado obtenido después de ejecutar el programa.

El código de Matlab que se proporciona resuelve el problema de optimizar la distribución del material en el MBBbeam de modo que se minimice su cumplimiento.

Es muy sencillo cambiar las condiciones del entorno y las condiciones de soporte para resolver otros problemas de optimización.

Para resolver el ejemplo de voladizo corto que se muestra en la Fig. 2, solo las líneas 79 y 80 deben cambiarse a:

79 F(2*(nelx+1)*(nely+1),1) = -1;

80 grados de libertad fijos = [1:2*(nely+1)];

También es muy simple extender el algoritmo a la cuenta para múltiples casos de carga. De hecho, esto se puede hacer agregando solo tres líneas adicionales y haciendo cambios menores, en el caso de dos casos de carga, fuerza y desplazamiento los vectores deben definirse como vectores de dos columnas que significa que la línea 69 se cambia a:

69 F = escaso(2*(nely+1)*(nelx+1),2);

U = escaso(2*(nely+1)*(nelx+1),2);

La función objetivo es ahora la suma de dos cumplimientos, es decir,

c(x) = 2

yo=1

Utah

yo kui (7)

por lo tanto, las líneas 20–22 se sustituyen por las líneas

19b dc(ely,elx) = 0.;

19c para i = 1:2

20 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);

21c = c + x(ely,elx)^penal*Ue'*KE*Ue;

22 p.a.(ely,elx) = p.a.(ely,elx) -

penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;

22b final

Para resolver el problema de dos cargas indicado en la Fig. 3, una unidad

la carga hacia arriba en la esquina superior derecha se agrega a la línea 79,

que luego se convierte

79 F(2*(nelx+1)*(nely+1),1) = -1.;

F(2*(nelx)*(nely+1)+2,2) = 1.;

En algunos casos, algunos de los elementos pueden ser necesarios para tome el valor de densidad mínima.

Una matriz nely×nelx pasiva con ceros en los elementos libre de cambiar y unos en elementos fijos para ser cero puede ser definido en el programa principal y transferido al OC subrutina (añadiendo pasivo a la llamada en las líneas 28 y 38).

Es cierto que el optimizador basado en criterios de optimización implementado aquí solo es bueno para una sola restricción y se basa en un esquema de actualización de tipo de punto fijo heurístico.

El número total de variables de entrada/salida es 20,incluyendo función objetivo, restricciones, antiguas y nuevas densidades, etc. Implementar el optimizador MMA es bastante simple, pero requiere la definición de varios auxiliares variables Sin embargo,

permite resolver problemas de diseño más complejos con más de una restricción. los El optimizador de Matlab resolverá el problema de optimización de topología estándar utilizando menos iteraciones a costa de un tiempo de CPU ligeramente mayor por iteración.

Las extensiones a tres dimensiones deberían ser sencillas, mientras que los problemas más complejos, como el cumplimiento El diseño del mecanismo (Sigmund 1997) requiere la implementación del optimizador MMA y la definición de restricciones adicionales. La simplicidad de los comandos de Matlab permitir extensiones fáciles de la salida gráfica, entrada interactiva, etc.

Conclusiones

Eduardo Villarreal Gámez

En definitiva, con esta práctica pudimos analizar un código de optimización topológica estándar y compactado de solo 99 líneas. Esto nos permitió de manera sencilla y general, el conocer cada una de las secciones de líneas que componen el código descrito (optimizador, filtro de independencia de malla y elemento finito) y su función específica en la ejecución del análisis topológico. Realmente puedo decir que me sorprendió bastante el cómo es posible realizar un algoritmo de optimización topológica con tan relativamente pocas líneas de código en Matlab. Que, además, bajo la opinión del autor del artículo, dicho algoritmo puede ser fácilmente extendido y/o cambiado para incluir por ejemplo problemas de multicarga y la definición de áreas pasivas, por lo que resulta ser bastante versátil.

Por último, concluyo diciendo que me pareció muy interesante el análisis detrás del desarrollo de estos tipos de algoritmos, sobre todo por su alta aplicabilidad sobre las estructuras reales, por lo que me gustaría seguir informándome al respecto más al futuro.

Daniel Alberto Acosta Banda

En conclusión, se puede observar que el diseño generativo por medio de la optimización topológica es un campo relativamente nuevo, que tiene sus áreas de oportunidad para perfeccionar los diseños generados, sin embargo, sigue siendo un campo prometedor que ya ha sido añadido a varios programas de CAD importante como Fusion360 y demás, también puedo entender el “por qué” de que la mayoría de métodos sea por medio del análisis de elementos, pues se puede observar que es el más “practico”, pero se puede observar un potencial en crear métodos híbridos.

Jose Eduardo Mendoza Ziga

Para esta práctica se utilizó un código de optimización topológica, el se entiendo y analizo desde como abrirlo en matlab hasta su utilización como fue pedido en el objetivo, se aprendió sobre las líneas de código que comprende el problema, este tipo de algoritmos aunque no son muy extensos en comparación a otros, cumplen su función a la hora el análisis topológico y su mejora ya ue este se puede editar o agregar más línea s para un análisis mas detallado.

Oiram Colunga Bernal

En esta actividad aprendimos sobre la topología y optimizar lo en el cual ayudó mucho el Matlab ya que me sorprendió el poder hacer lo en tan pocas líneas a su vez hay que saber que es muy importante tener bien contempladas todas las fuerzas que actúan en la estructura fuera de eso está práctica es muy interesante eh importante para empezar a ver la materia de biomédica, pero en un ámbito práctico

Raul Alejandro Tamez Gonzalez

Para esta actividad número uno de laboratorio de biomecánica nos veamos con una lectura llamada “Optimización topológica estándar y compactado en sólo 99 líneas” esto nos ayudó para entender cuál es la necesidad en el ámbito de la ingeniería para el desarrollo de estructuras con una resistencia estudiada ya que nuestra carrera se relaciona mucho con el área de la planeación de componentes biomecánicos. Es de gran importancia el conocer cada una de Las secciones de las líneas que describe este documento y la función que estas realizan en el análisis topológico. Este algoritmo puede fácilmente ser extendido o cambiado para incluir otro tipo de problemas y análisis que requieran este tipo de estudios con planificación a un tiempo específico y que tenga que ser contempladas múltiples fuerzas que puedan llegar a dañar nuestra estructura. Creo que este es un acercamiento hacia una materia de gran interés e importancia para la carrera de mecatrónica espero poder aprender mucho de esta y que sigas siendo tan interesante como esta primera práctica lo ha sido.

Manuel Antonio Ulloa Méndez. 1992298.

En conclusión, esta práctica cumple con su objetivo, ahora sé que la optimización topológica 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.

Sus inicios empiezan desde hace cuatro décadas cuando Schmidt [1] propuso una idea revolucionaria que dio origen a una nueva disciplina: los ingenieros, en general, tratan de diseñar objetos o sistemas de coste mínimo que durante su vida útil deben ser capaces de resistir las solicitaciones máximas que se puedan producir.

También ahora conozco cada una de las secciones que integran un código de optimización topológica, como se debe crear el archivo (.m) en MATLAB y como se ejecuta el análisis.

Brandon Arturo Solano Arias 1822072 IMTC

En esta práctica vimos cómo se podría optimizar la programación de un análisis topológico de una viga en nuestro caso y como es que funciona a través de la herramienta matemática de programación Matlab en dónde se introdujo este algoritmo y las entradas de éste eran las dimensiones de la viga para ser analizado, esta práctica nos ayudó a poder observar cómo podríamos emplear este algoritmo para lograr resolver el diseño de alguna prótesis o estructura real y así aminorar daños y costos extras en el trabajo que estemos realizando usando la menor cantidad de material y peso para reducir costos de fabricación y material residual.

Bibliografía.

[1] L.A. Schmidt, Structural design by systematic synthesis, Proceedings of the Second ASCE Conference on Electronic Computation, 105–122, Pittsburgh, USA (1960).

[2] S. Hernández, Métodos de Diseño Óptimo de Estructuras, Colegio de Ingenieros de Caminos, Canales y Puertos, Madrid (1990).

[3] F. Navarrina y M. Casteleiro, A general methodologycal analysis for optimum design, Int. J. Num. Meth. Engrg., 31, 85–111 (1991).

[4] F. Navarrina, S. López, I. Colominas, E. Bendito y M. Casteleiro, High order shape design sensitivity: A unified approach. Comp. Meth. Appl. Mech. Engrg., 188, 681–696 (2000).

[5] M. P. Bendsøe y N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, Comp. Meth. Appl. Mech. Engrg., 71, 197–224 (1988).

[6] E. Ramm, S. Schwarz y R. Kemmler, Advances in structural optimization including nonlinear mechanics, Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2000) (CD-ROM, ISBN: 84-89925-70-4), European Community on Computational Methods in Applied Sciences, Barcelona (2000).

[7] M. P. Bendsøe, Optimization of structural topology, shape, and material, Springer-Verlag, Heidelberg (1995).

[8] I. Muiños, Optimización Topológica de Estructuras: Una Formulación de Elementos Finitos para la Minimización del Peso con Restricciones en Tensión, Proyecto Técnico, ETSICCP, Universidad de A Coruña (2001).

[9] M. P. Bendsøe, Variable-topology optimization: status and challenges, Proceedings of the European Conference on Computational Mechanics ECCM'99, W. Wunderlich (Ed.), TUM, Munich (1999).

[10] I. Muiños, I. Colominas, F. Navarrina y M. Casteleiro, Una formulación de mínimo peso con restricciones en tensión para la optimización topológica de estructuras, Métodos Numéricos en Ingeniería y Ciencias Aplicadas (ISBN: 84-89925-91-7), E. Oñate, F. Zárate, G. Ayala, S. Botello y M.A. Moreles (Eds.), CIMNE, Barcelona, 399–408 (2001).