Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Laboratorio de Biomecánica

Práctica 1: Código de optimización topológica.

Docente: Dra. Yadira Moreno Vera

Dia y hora: miércoles N4 Brigada: 309

Integrantes del equipo:

Matriculas	Nombre	Carrera
1806409	Eduardo Antonio Flores Ramírez	IMTC
1884328	Daniel isaac zaragoza Soto	IMTC
1900466	Denisse García Espinoza	IMTC
1897305	Kevin Alberto Flores Martínez	IMTC
1991814	Pedro Hazael Uriegas Peña	IMTC

Semestre agosto-diciembre 2022.

21 de septiembre de 2022

Practica #1

Descripción y uso de código de optimización topológica.

Objetivo

El estudiante conocerá cada una de las secciones que integran el código de optimización topológica, como se debe de crear el archivo (.m) en Matlab y como se ejecuta el análisis.

Marco teórico

La optimización topológica comienza con la creación de un modelo 3D en la fase de borrador, en el que se aplicaran las diferentes cargas o fuerzas para la pieza (una presión sobre las lengüetas de sujeción, por ejemplo).

Un problema clásico de la ingeniería consiste en determinar la configuración geométrica óptima de un cuerpo que minimice o maximice una cierta función objetivo, al mismo tiempo que satisface las restricciones o condiciones de contorno del problema. La solución de este problema puede ser planteada utilizando dos estrategias: como un problema de optimización de forma o de optimización de la topología. La optimización de forma consiste en modificar la geometría del dominio preservando su topología, es decir sin crear huecos o cavidades en su interior, es usualmente conocido como análisis de sensibilidad al cambio de forma y sus bases matemáticas están bien establecidas. El principal inconveniente del análisis de sensibilidad al cambio de forma es que sólo permite cambios en la frontera del dominio, lo que limita su campo de aplicación. Una manera más general de controlar un dominio es mediante modificaciones de su topología, lo que permite obtener la configuración deseada partiendo de una morfología inicial distante de la óptima. Los métodos de homogenización son posiblemente los más utilizados para la optimización topológica. Estos consisten en caracterizar la topología a través de su densidad, es decir, los huecos se identifican con regiones de densidad nula. De esta forma la solución del programa resulta en una distribución ficticia de material. Matlab es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos

hardware.

El código de optimización topológica de 99 líneas en Matlab que se utilizara en este laboratorio se divide en 36 líneas para la programación principal, 12 líneas para los criterios de optimización, 16 líneas para el filtro de mallado y 35 líneas para el código de elemento finito. De hecho, excluyendo las líneas de comentarios y líneas asociadas con la producción y el análisis de elementos finitos, el código resultante es de solo 49 líneas.

Este código fue desarrollado por 99 Line Topology Optimization Code – O. Sigmund, Department of Solid Mechanics, Building 404, Technical University of Denmark, DK-2800 Lyngby, Denmark. El código puede ser descargado desde la página del autor: http://www.topopt.dtu.dk.

Desarrollo

Describir cada una de las partes principales del código de 99 líneas en Matlab.

Programa principal (líneas 1-36)

El programa principal se inicia mediante la distribución del material uniformemente en el domino de diseño (línea 4). Después de algunas inicializaciones, el bucle principal comienza con una llamada a la subrutina de Elementos Finitos, que devuelve el vector de desplazamiento U.

La matriz de rigidez de elemento para el material solido es la misma para todos los elementos, la subrutina solo se llama una vez (línea 14). Después un bucle determinara la función objetiva y las sensibilidade. Las variables n1 y n2 indican los números de nodos de elementos superiores izquierdo y derecho. El análisis de sensibilidad es seguido al llamar al filtro de malla independiente (línea 26) y al optimizador (línea 28).

Otros parámetros son impresos en las lineas 30 a 33 y la distribución de densidad resultante es en la línea 35. El bucle termina si el cambio en las variables de diseño es menor al 1%.

Optimización basada en criterios de optimalidad (líneas 37-48)

Las variables de diseño actualizados se encuentran por el optimizador. Sabiendo que el volumen de material (sum (sum (xNew))) es una función monótonamente decreciente del multiplicador de Lagrange (lag), el valor de la Lagrangian multiplier que satisface la restricción de volumen puede ser encontrada por un algoritmo biseccionar.

Filtro de malla independiente (líneas 49-64)

Líneas 49 a 64 representan la aplicación de Matlab. Tenga en cuenta que no todos los elementos en el dominio de diseño son buscados con el fin de encontrar los elementos que se encuentran dentro la rmin radio, pero sólo aquellos dentro de un cuadrado de lado longitudes dos veces la vuelta (Rmin) alrededor del considerado elemento.

Código de elemento finito (líneas 65-99)

Hay que tener en cuenta que el solucionador hace uso de las escasas opciones en MATLAB. La matriz de rigidez global está formada por un bucle sobre todos los elementos. Como fue el caso en el principal programa, las variables n1 y n2 denotan superior izquierda y derecha números de nodo elemento y son utilizados para insertar el elemento de matriz de rigidez a la derecha de la matriz de rigidez global. Como se mencionó antes, ambos nodos y elementos son numerados en columnas de izquierda a derecha. Además cada nodo tiene dos grados de libertad (horizontal y vertical), por lo que se aplica el comando F(2,1)=-1 (línea 79) una fuerza unitaria vertical en la esquina superior izquierda.

Implementación de la programación

En base a la explicación del procedimiento realizado para la programación de la optimización topología en Matlab, ahora el siguiente paso a realizar es el corroborar que el código propuesto funcione correctamente. Para la implementación de este código lo único necesario es el copiar el siguiente programa en el editor de Matlab.

```
%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%

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-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),1); U =sparse(2*(nely+1)*(nelx+1),1);
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(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 127
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)];
```

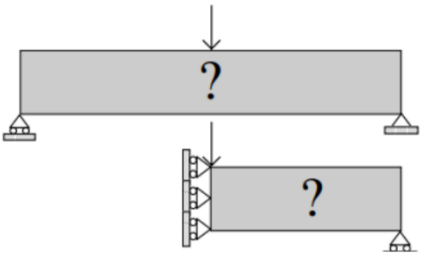
Para correr la simulación de la optimización se deben de llamar con la siguiente línea:

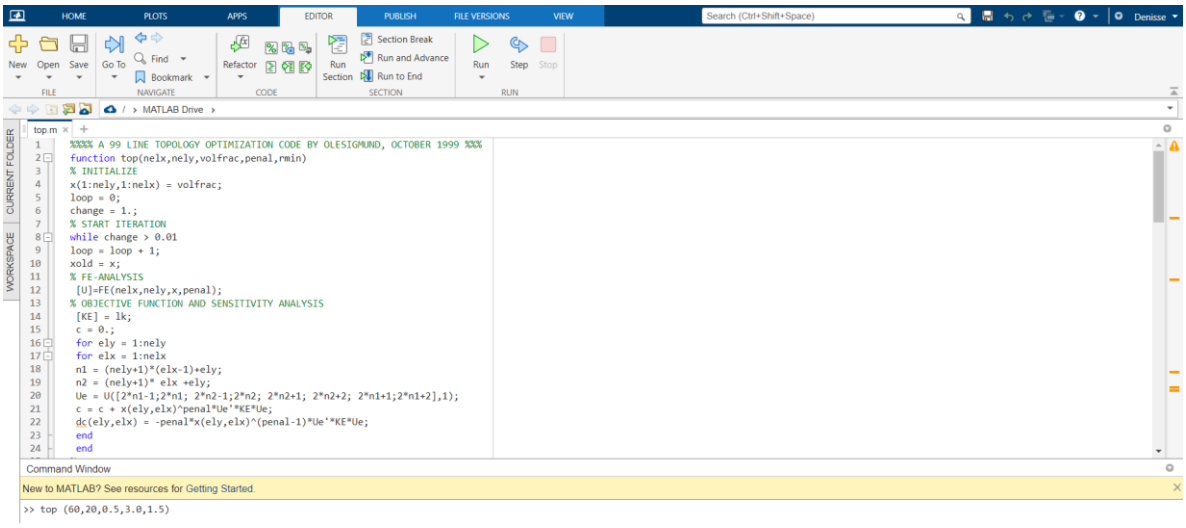
>>top(nelx,nely,volfrac,penal,rmin)

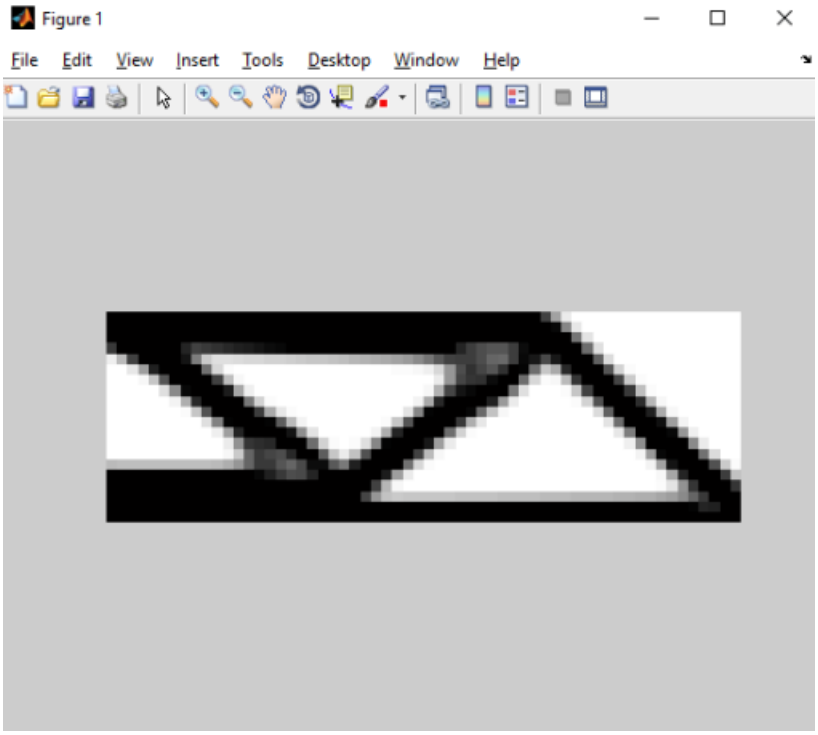
Donde nelx y nely son el número de elementos en el direcciones horizontal y vertical respectivamente, volfrac es la fracción de volumen, penal es el poder de penalización y rmin es el tamaño del filtro (dividido por el tamaño del elemento). Otras variables, así como las condiciones de contorno se definen dentro del propio código de Matlab. Dichas condiciones se pueden modificar en caso de querer ciertas propiedades específicas y de esta manera tener distintas observaciones del comportamiento del código ante distintas entradas. Por lo que por razones prácticas en este caso las condiciones a utilizar son las siguientes:

>> top (30,10,0.5,3.0,1.5)

Una vez definidas las condiciones iniciales de la práctica se proseguirá a realizar la simulación del programa para poder notar el cómo funciona la optimización topológica. La estructura a realizarle la optimización es una barra a la cual se le someten distintas cargas por lo que a continuación se mostraran imágenes de la estructura original y de cómo el mismo programa va optimizando dicha estructura para obtener las mismas o mejores propiedades con menor material y con un acabado más estético.







Conclusiones

Denisse García Espinoza: al termino de este reporte, se cumplió el objetivo, analizamos cada una de las secciones que integran el código de optimización topología, al realizar la practica se logro comprender el concepto de optimización topológica y vemos que es de gran ayuda en la industria y fundamental en el análisis estructural, pues se obtiene un gran ahorro al optimizar la fabricación de estructuras, piezas, etc. Pues cuando se hace el análisis y colocar las fuerzas y todo depende de como se carga la pieza, pues de ahí se empezaría a revisar en donde no se necesita material y eliminarlo, obteniendo como resultado una estructura más liviana, funcional y reduciendo costos.

Pedro Hazael Uriegas Peña: En nuestra primera práctica en el laboratorio de biomecánica, nuestro objetivo es comprender y crear diferentes partes del código optimizado de topología. Para ello, utilizamos el software Matlab, que fue el medio para realiza el código para nuestra estructura elegida. A través de esta práctica, podemos ver cómo es importante entender dichos códigos porque nos ayudan a analizar una estructura e identificar de una forma más rápida sus propiedades. Así que la practica fue realizada con éxito y se cumplieron los objetivos planteados.

Referencias

[1]A. (s. f.). La optimización estructural y sus aplicaciones. Instituto Politécnico Nacional.

Recuperado 4 de septiembre de 2022, de https://www.boletin.upiita.ipn.mx/index.php/ciencia/916-cyt-numero-82/1889-la-optimizacion-estructural-y-sus-aplicaciones

[2] 99 Line Topology Optimization Code – O. Sigmund, Department of Solid Mechanics, Building 404, Technical University of Denmark, DK-2800 Lyngby, Denmark.