UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FIME
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Laboratorio de Biomecánica

Práctica 2

INSTRUCTOR@: YADIRA MORENO VERA

MARTES V1 BRIGADA 204

Equipo #2:

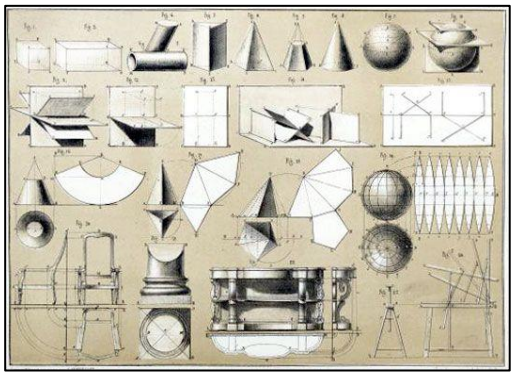
Matricula	Nombre	Carrera
1991908	Covarrubias Becerril Brian Eduardo	IMTC
1991966	Luis Javier Rodríguez Vizcarra	IMTC
1925324	Rubén Cantú Espinoza	IMTC
1926098	Ángel Fernando Mexquitic Rodríguez	IMTC
1895460	Elmer Javier Delgadillo García	IMTC
1847932	Francisco Javier Velazco Rivas	IMTC

12 de septiembre del 2022, San Nicolás de los Garza, Nuevo León

Objetivo

El estudiante deberá presentar una propuesta de análisis de formas y de la programación para la ejecución de la optimización de características de trabajo específicas que presenta la(s) ventaja(s).

Marco Teórico



El análisis de formas es la disciplina que se ocupa del procesamiento de las figuras geométricas para posibilitar su caracterización. Está relacionado con el análisis estadístico de formas, y su propósito es posibilitar la determinación de la coincidencia entre formas y facilitar su reconocimiento. Se aplica exclusivamente a la geometría de un objeto, pero no a su análisis estructural.

El análisis de formas permite la clasificación de formas geométricas, por ejemplo, usando un ordenador para detectar objetos de forma parecida a partir de una base de datos o partes que encajan entre sí. Para que una máquina analice y procese automáticamente formas geométricas, los objetos deben representarse en forma digital. Más comúnmente, se utilizan representaciones de contorno para describir el objeto por sus límites. Sin embargo, otras representaciones basadas en el volumen o representaciones basadas en puntos se pueden usar para representar las formas.

Una vez recibidas las representaciones de los objetos en forma numérica, ya sea mediante modelado, escaneado o extrayendo la forma de imágenes 2D o 3D, se deben simplificar antes de poder ser comparadas. La representación simplificada a menudo se denomina descriptor de forma. Estas representaciones simplificadas pretenden almacenar la mayor parte de la información relevante, a la vez que son más fáciles de manejar, almacenar y comparar que las propias formas directamente.

En el pasado, muchos de los procesos de diseño fueron realizados por la experiencia e intuición del diseñador en vez de una aplicación intensiva de la teoría de optimización. Recientemente esta forma de pensar ha cambiado debido a la importancia que ha tomado el campo de la optimización estructural en el diseño, ya que mediante su aplicación se logra reducir costos, materiales y tiempo en los procesos de diseño realizados por los ingenieros.

El propósito de aplicar los conceptos de diseño óptimo a la ingeniería estructural es el de obtener una solución a un problema de ingeniería que cumpla con todas las limitaciones y restricciones impuestas, y que a la vez resulte ser la mejor en cuanto a uno o varios criterios de diseño previamente establecidos.

El incremento dramático en la capacidad y velocidad de proceso de los computadores modernos en las últimas décadas y los avances realizados en el desarrollo de nuevos métodos numéricos para el análisis de medios continuos en conjunción con los algoritmos tradicionales de optimización, tales como programación cuadrática secuencial, programación no lineal han hecho posible automatizar el diseño y el proceso de optimización de formas.
En este tipo de diseños las técnicas de optimización numérica son usadas en un intento de obtener la mejor forma y, adicionalmente, el uso más efectivo de los materiales para una determinada estructura bajo las restricciones impuestas por las condiciones de diseño.

Definición de la programación y ejemplo de la geometría

El éxito de cualquier metodología de optimización descansa sobre su habilidad para tratar problemas complejos, como es el caso del diseño de optimización de formas. Resolver eficientemente problemas de tipo no lineal es desafiante. Por ello, se ha hecho práctica común que los métodos sean modificados, combinados y extendidos a fin de construir algoritmos que combinen las mejores características de varios de ellos en una propuesta que de cierta manera se adapte al problema en particular y permita resolver el problema en un contexto general que de otra manera cada método por separado no podría resolver por sí solo.

Por estas razones, la metodología de optimización de formas presentada se basa en el fundamento de varios métodos su combinación y la escogencia más apropiada de ellos.

En este trabajo, entre las numerosas técnicas evolucionarias existentes, se han seleccionados los Algoritmos Genéticos debido al hecho de que teórica y experimentalmente se ha demostrado que proveen una búsqueda robusta en espacios complejos de diseño. Estos algoritmos tienen muchas ventajas sobre los métodos tradicionales de búsqueda. Entre otras consideraciones, no necesitan de información especializada de la función a optimizar si no los valores de la función objetivo o de mérito.

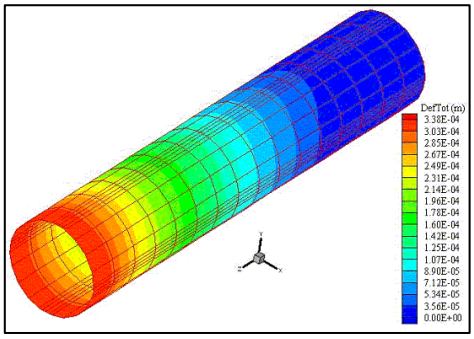
Los algoritmos genéticos son algoritmos de búsqueda basados en los mecanismos de la selección natural y la genética, fueron desarrollados en un intento de simular algunos de los procesos observados en la evolución natural, como la selección de

parejas, los procesos de recombinación de información y mutación, con el fin de generar mecanismos de búsqueda que convergen rápidamente a la estructura óptima o casi-óptima del problema con un esfuerzo mínimo y probando solo una pequeña fracción del espacio de diseño.

Estado del arte

Recientemente, los Algoritmos Evolutivos (AE) se han usado extensivamente para encontrar soluciones casi-óptimas u óptimas al problema de optimización estructural de formas debido a su robustez, eficiencia y eficacia. Entre sus ventajas se encuentra el hecho de no requerir información especializada para obtener el óptimo buscado además de no necesitar de un análisis de sensibilidades durante este proceso. Todas estas ventajas hacen a AE un buen candidato para resolver problemas de optimización en general, sin embargo algunas de sus desventajas, tales como la carencia de diversidad en la población durante el proceso de evolución y su excesivo tiempo de cálculo debido al gran número de funciones de evaluación, han llamado la atención de la comunidad especializada a fin de solventar estos inconvenientes.

En este sentido, el objetivo de este artículo es proponer una metodología de optimización basada en un Algoritmo Genético Distribuido de carácter flexible que permita la optimización estructural de la forma de modelos de elementos finitos o elementos de contorno modelados con elementos de diseño geométrico. El carácter distribuido y el manejo flexible de las variables y parámetros permiten al algoritmo solventar los inconvenientes antes mencionados. Finalmente, un ejemplo numérico es presentado y discutido a fin de mostrar la habilidad de la metodología propuesta para optimizar este tipo de problemas.

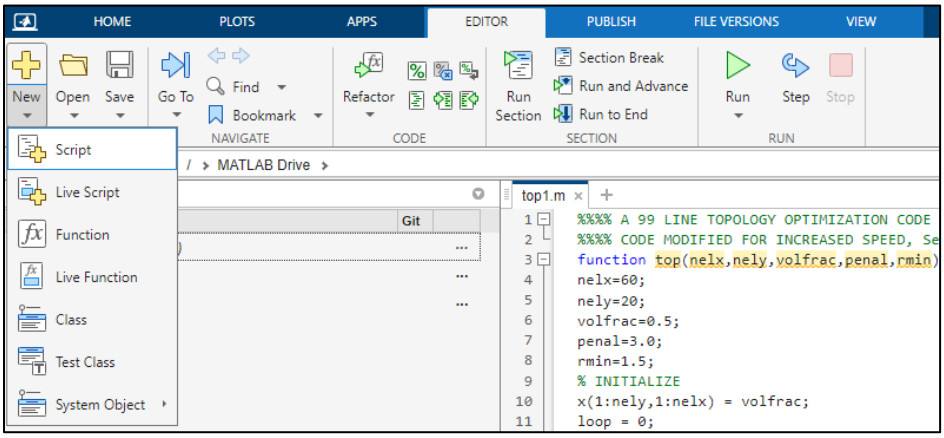


Procedimiento de la programación

Para simular en Matlab el código propuesto de optimización topológica, debemos llevar a cabo los siguientes pasos:

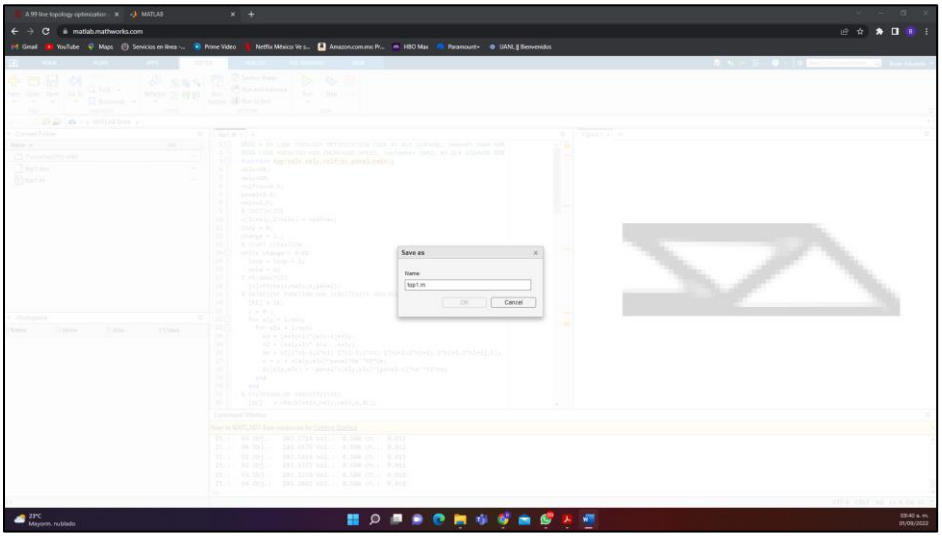
1) Abrir Matlab, ya sea la aplicación de escritorio o directamente desde la página oficial usando su repositorio online.

2) Crear un nuevo script, para ello seleccionamos la opción en la barra de herramientas.



3) Una vez hecho esto, solo tenemos que copiar el código proporcionado que se localiza en la página indicada por el instructor.

4) Ya con el código copiado (99 Line Topology Optimization Code), tenemos que guardar primero el documento, en este caso como se utilizó Matlab online, se guardará en la nube de la cuenta que esta iniciada.



5) Antes de ejecutar la simulación, es necesario definir los valores de las variables de entrada (nelx, nely, volfrac, penal, rmin), ya sea sustituyendo directamente los valores dentro de la línea de código donde se define o asignándolos cada uno.

6) Se deberá realizar una serie de modificaciones en el código base, en las siguientes líneas, para efectuar una primera optimización:

- **80** `F (2,1)=1;`
- **81** `fixeddofs = 2 * nelx * (nely + 1) + 1:2 * (nelx + 1) * (nely + 1);`
- `top(20,20,0.33,3.0,1.5)`

7) Posteriormente se debe modificar el mismo código de la siguiente manera para concretar con una segunda optimización:

- **80** `F (2,1)=1;`
- **81** `fixeddofs = 2 * nelx * (nely + 1) + 1:2 * (nelx + 1) * (nely + 1);`
- `top(20,20,0.33,3.0,1.5)`
- 41 `while ((l2-l1)/l2 > 1e-4)`
- Agregar las siguientes líneas al código entre la línea 5 y 6 para hacer esto:

```
for ely = 1:nely
      for elx = 1:nelx
            if ((elx)^2+(ely-nely)^2) < (0.65*nelx)^2
                  passive(ely,elx) = 1;
            else
                  passive(ely,elx) = 0;
            end
      end
end
x(find(passive))=0.001;
```

- También tenemos que actualizar la línea 29 y 40 e insertar una línea adicional entre 43 y 44:

```
29 [x] = OC(nelx,nely,x,volfrac,dc,passive);
40 function [xnew]=OC(nelx,nely,x,volfrac,dc,passive)
43b xnew(find(passive)) = 0.001;
```

- Realiza estos cambios y ejecuta con `top(20,20,0.33,3,1.5)`

8) Procedemos a correr la simulación y observamos los resultados mostrados.

Implementación o desarrollo de la programación

El código implementado para la simulación del programa el siguiente:

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLESIGMUND, OCTOBER 1999 %%%
function topp1(nelx,nely,volfrac,penal,rmin)
% INITIALIZE
x(1:nely,1:nelx) = volfrac;
loop = 0;
for ely = 1:nely
for elx = 1:nelx
if ((elx)^2+(ely-nely)^2) < (0.65*nelx)^2
passive(ely,elx) = 1;
else
passive(ely,elx) = 0;
end
end
end
x(find(passive))=0.001;
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,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)/l2 > 1e-4)
lmid = 0.5*(l2+l1);
xnew(find(passive))=0.001
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 =2*nelx*(nely+1)+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)];
```

Conclusiones

- **Brian Eduardo Covarrubias Becerril:** esta práctica consistió en la elaboración de un marco de bicicleta, donde realizamos un análisis de formas y de la programación, para ello modificamos un poco el código de optimización topológica de la práctica pasada cambiando los valores de ciertos parámetros. Una vez que lo ejecutamos, nos dimos cuenta del resultado obtenido para que cumpla con las características deseadas. Esto puede realizarse también para diversos problemas.

- **Elmer Javier Delgadillo García:** En la presente práctica se pudo apreciar la importancia y labor que ejecuta la modificación del código base de una forma al cambiar sus valores en los parámetros de carga, apoyo y fuerza. De esta manera se plasma que el análisis de formas y la programación de la ejecución de optimización de características requeridas permite realizar una gran variedad de diseños dentro del mismo código.

- **Luis Javier Rodriguez Vizcarra:** En conclusión, en esta práctica se aplicó lo de la práctica anterior que fue lo de optimizar el modelo y mediante esta práctica se reforzaron esos conocimientos dejándolo más claro, otros puntos que trabajamos fueron las interacciones, los grados de libertad y las restricciones para la generalización de la pieza, también se observó la estructura que tiene el código cuando se va a colocar un valor de carga, apoyo y fuerza dentro de un espacio de diseño que son los aspectos que se deben tener en cuenta cuando se quiere construir una pieza a la que estará expuesta a estos factores.

- **Francisco Javier Velazco Rivas:** Para esta práctica se evalúa un problema que se podría aplicar a distintas cosas como lo sería la mejora continua en el caso de la programación fue cuestión de ir viendo que sería mejor modificar o quitar para hacerlo mejor. El aspecto del análisis fue fundamental ya que cada uno de nosotros lo tomamos de diferente manera tomando en cuenta que todas fueron opciones y las juntamos para ver que sería mejor como en la forma o el mismo código no fuera tan largo si era algo que también se buscaba reducir.

- **Ángel Fernando Mexquitic Rodriguez:** Con esta práctica pude ver lo importante que resulta la optimización a la hora de realizar la programación, similar a lo realizado en la practica anterior, analizando y eliminando partes para hacer de este uno que resultase mas sencillo para de esa manera sea mas comprensible, comprendiendo a su vez el termino de análisis de formas y lo fundamental que llega a resultar la comprensión de este a la hora de clasificar formas geométricas.

- **Rubén Cantú Espinoza:** Con la elaboración de esta segunda práctica se aclararon ciertos puntos que se vieron en la práctica anterior, ya que para ello se utilizaron diferentes concepto como el Algoritmo Evolutivo, Algoritmo Genérico Distribuido, con el objetivo de optimizar la estructura de una pieza de la forma de elementos finitos (en este caso se diseñó un marco de bicicleta). Para poder cumplir con este objetivo nos sirvió mucho el código generado en la anterior práctica ya que solo definimos los valores de las variables de entrada y modificamos ciertas líneas del código base, para o posteriormente observar las actualizaciones de las optimizaciones en la pieza.

Referencias

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

- IMME. (2007). *Una metodología para la optimización estructural de formas usando principios de evolución flexible distribuida*. Scientific Electronic Library Online. http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S0376-723X2007000100002