Practica 1 Biomecánica

Eimie Carolina Pereda Sanchez 1915035
Aarón Lozano Aguilar 1844469
Montserrat Granados Salinas 1817165
Eunice Carolina Méndez Sosa 1851345
Aida Mata Moreno 1734743

4 de September del 2022

1. Objetivo

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

2. Marco teórico

La optimización topológica es una técnica que se engloba dentro del campo de análisis mecánico de un componente o estructura y su objetivo principal es el aligeramiento estructural, pero manteniendo las funcionalidades mecánicas del componente objetivo. Nos ofrece un concepto nuevo de diseño estructural enfocado hacia aquellas aplicaciones donde el peso del componente es parte crucial, un ejemplo muy claro es la industria aeroespacial.

La optimización hoy en día tiene un gran avance gracias a los métodos computacionales que salen día con día, ya que con esto lo lleva a un nivel más complejo del análisis a un nivel estático, dinámico, plástico, modal o de impacto, los cuales se consideran durante todo el proceso de optimizació.

Tiene un amplio campo de aplicación para las tecnologías de fabricación aditiva, un ejemplo es la fabricación SLM (Selective Laser Melting), ya que tiene grandes posibilidades en términos de diseño.

Podemos tomar de ejemplo una geometría que se usa día con día, ya sea en construcción, así como en los grandes puertos de México ya sean marítimos o aéreos, un gancho.

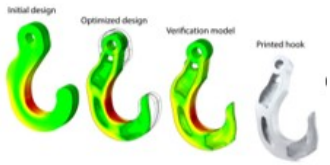


Figura 1: Ejemplo 1

Por medio de la optamización se va analizando y descartando las zonas que menor esfuerzo presenta a comparación de las zonas donde los esfuerzos son los más considerables, repitiendo el análisis las veces necesarias hasta llegar al mejor resultado de optimización.

El código del cual haremos uso durante el curso de laboratorio fue hecho en MATLAB y está destinado al campo

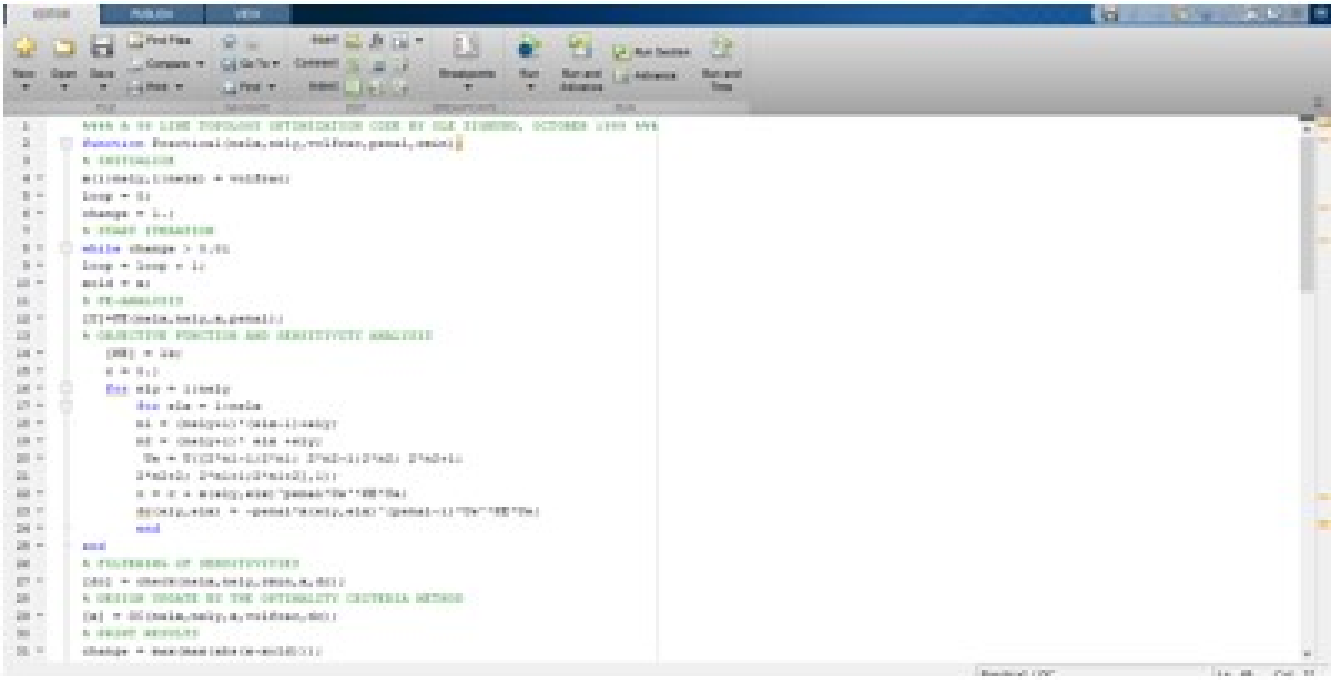
de la optimización de topología y se puede usar para hacer extensiones como múltiples casos de carga, esquemas alternativos de independencia de malla, áreas pasivas, etc.

El código en uso consta de 99 líneas, las cuales 36 son para la programación principal, 12 para los criterios de optimización, 16 para el filtro de mallando y 35 líneas para el código de elemento finito.

Este código fue desarrollado por 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.

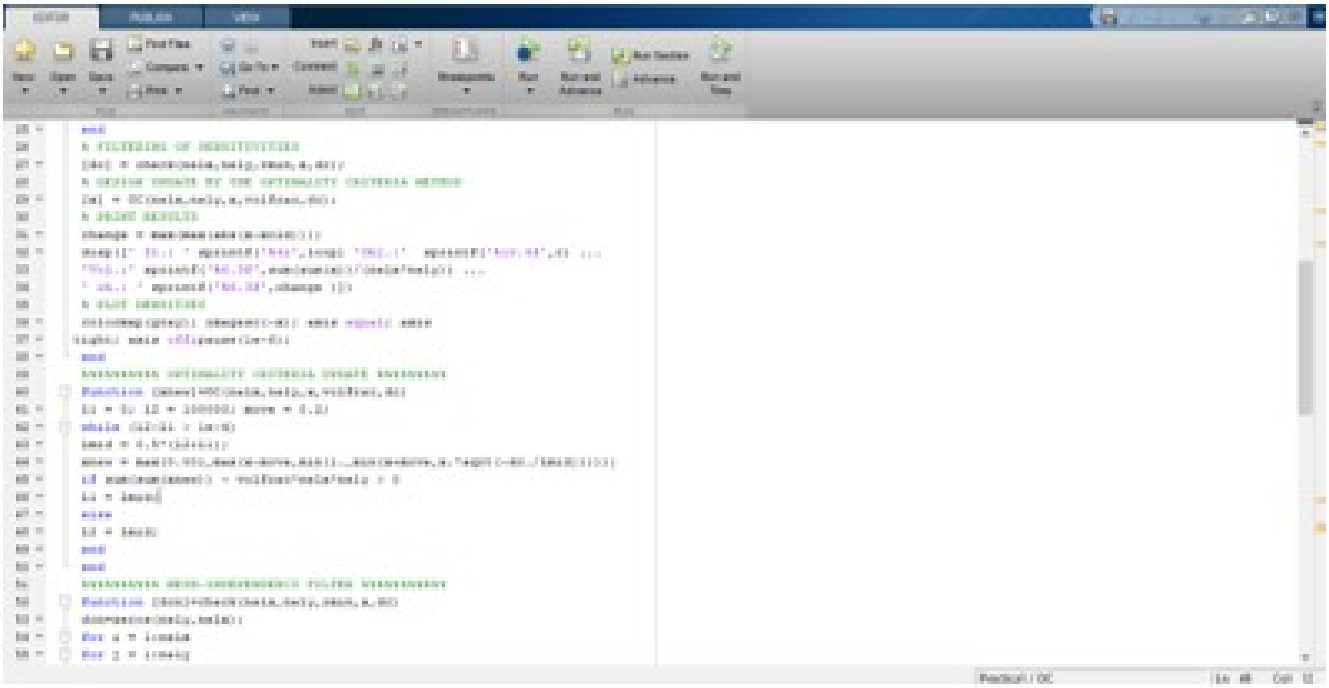
3. Desarrollo

1. Empezamos abriendo el software MATLAB o en su caso algún similar como Scilab y empezamos un nuevo script.
2. Escribimos el código de manera correcta corrigiendo sus errores.

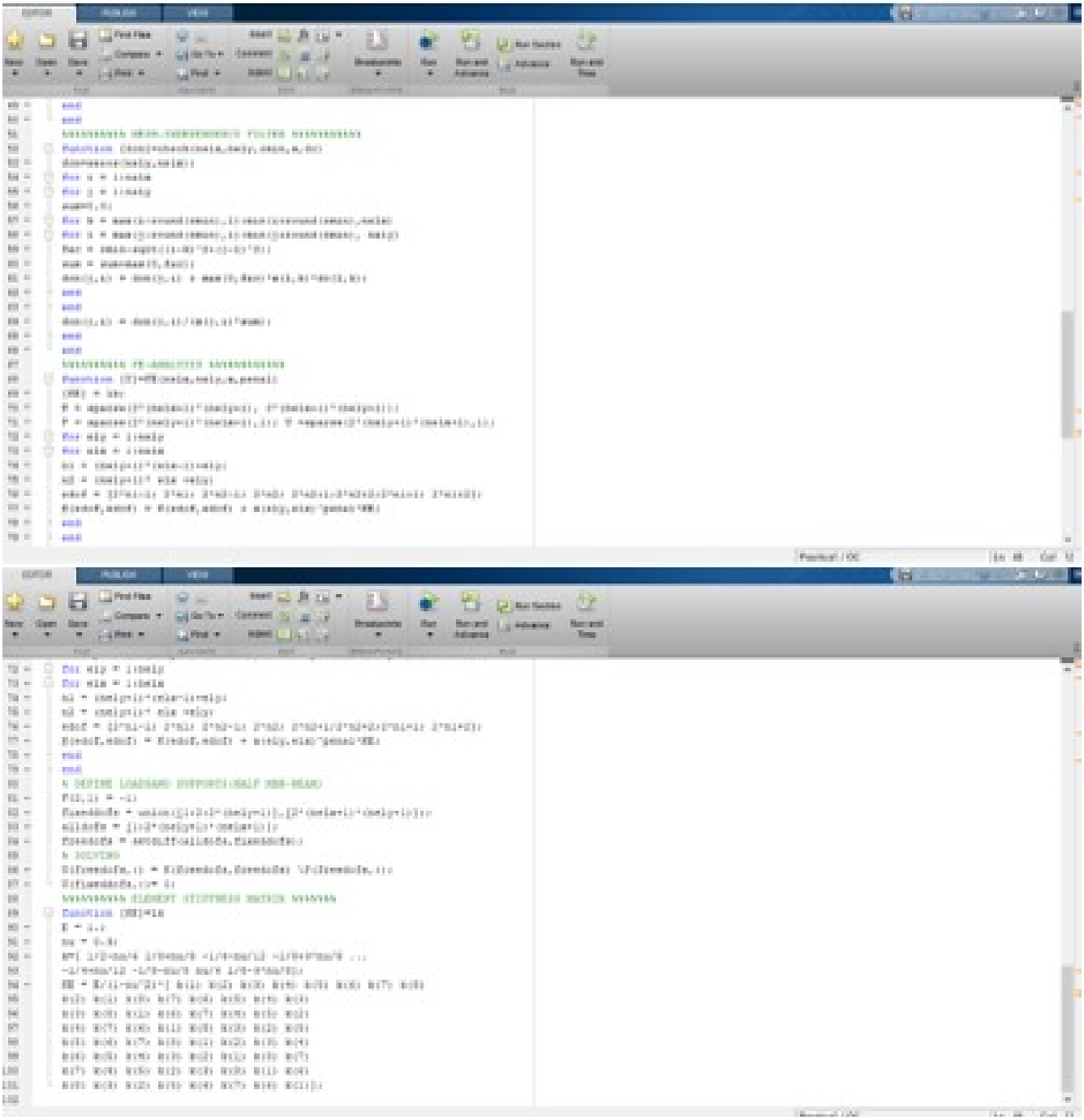


[h]

[h]

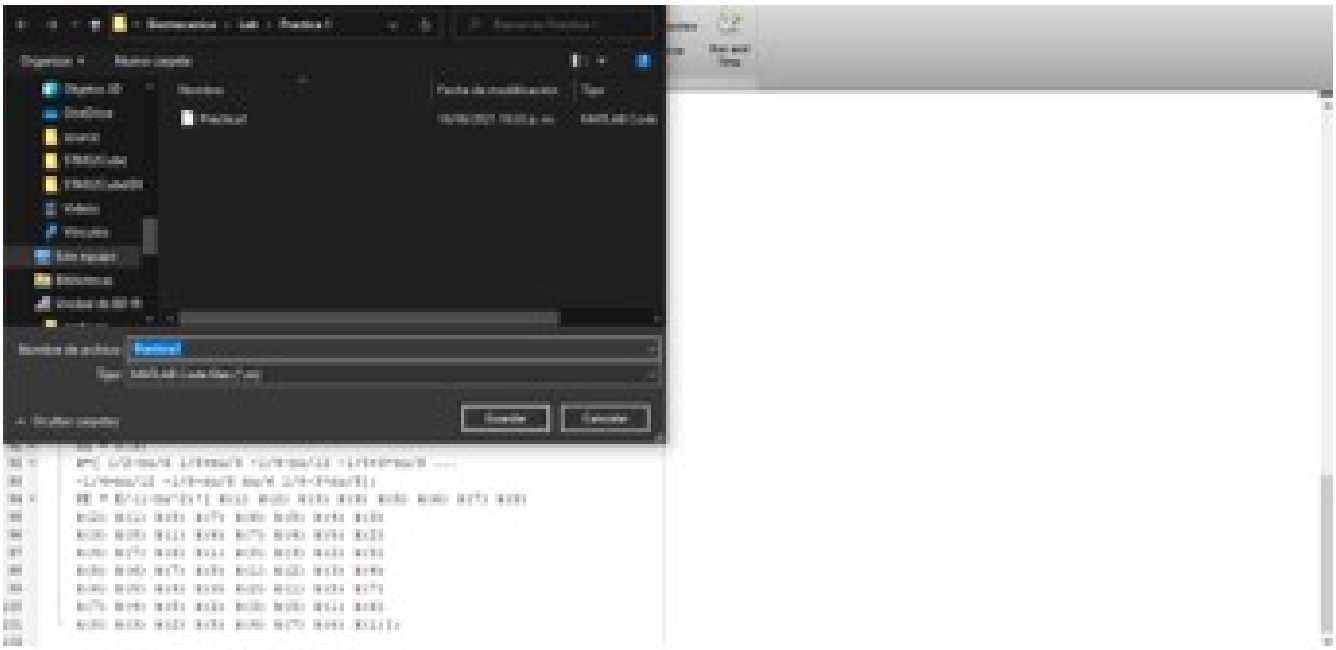


[h]



3. Guardamos el trabajo antes de correrlo y ver su resultado en la ventana de comando.

[h]



Código de 99 líneas

function Practica1(nelx,nely,volfrac,penal,rmin); x(1:nely,1:nelx) = volfrac; loop = 0; change = 1.; while change ¿0.01 loop = loop + 1; xold = x; [U]=FE(nelx,nely,x,penal); [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)^{p}enal$*Ue'*KE*Ue$; $dc(ely, elx) = -penal*x(ely, elx)^(penal-1)*Ue'*KE*Ue$; $endend[dc] = check(nelx, nely, rmin, x, dc); [x] = OC(nelx, nely, x, volfrac, dc); change = max(max(abs(x - xold))); disp(['It. :' sprintf("Vol. :' sprintf("ch. :' sprintf('colormap(gray); imagesc(-x); axisequal; axistight; axisoff; pause 6); end$

$function[xnew] = OC(nelx, nely, x, volfrac, dc)l1 = 0; l2 = 100000; move = 0{,}2; while(l2-l1 > 1e-4)lmid = 0{,}5*(l2+l1); xnew = max(0{,}001, max(x - move, min(1., min(x + move, x. * sqrt(-dc./lmid))))); if sum(sum(xnew)) - volfrac * nelx * nely > 0l1 = lmid; elsel2 = lmid; endend$

$function[dcn] = check(nelx, nely, rmin, x, dc)dcn = zeros(nely, nelx); fori = 1 : nelxforj = 1 : nelysum = 0{,}0; fork = max(i - round(rmin), 1) : min(i + round(rmin), nelx)forl = max(j - round(rmin), 1) : min(j + round(rmin), nely)fac = rmin - sqrt((i - k)^2 + (j - l)^2); sum = sum + max(0, fac); dcn(j, i) = dcn(j, i) + max(0, fac) * x(l, k) * dc(l, k); endenddcn(j, i) = dcn(j, i)/(x(j, i) * sum); endend$

$function[U] = FE(nelx, nely, x, penal)[KE] = lk; K = sparse(2*(nelx+1)(nely+1), 2(nelx+1)*(nely+1)); F = sparse(2 * (nely + 1)(nelx + 1), 1); U = sparse(2(nely + 1) * (nelx + 1), 1); forely = 1 : nelyforelx = 1 : nelxn1 = (nely + 1) * (elx - 1) + ely; n2 = (nely + 1) * elx + ely; edof = [2 * n1 - 1; 2 * n1; 2 * n2 - 1; 2 * n2; 2 * n2 + 1; 2 * n2 + 2; 2 * n1 + 1; 2 * n1 + 2]; K(edof, edof) = K(edof, edof) + x(ely, elx)^penal * KE; endend$

$F(2, 1) = -1; fixeddofs = union([1 : 2 : 2 * (nely + 1)], [2 * (nelx + 1) * (nely + 1)]); alldofs = [1 : 2 * (nely + 1) * (nelx + 1)]; freedofs = setdiff(alldofs, fixeddofs); U(freedofs, :) = K(freedofs, freedofs)(freedofs, :); U(fixeddofs, :) = 0;$

$function[KE] = lkE = 1.; nu = 0{,}3; k = [1/2-nu/61/8+nu/8-1/4-nu/12-1/8+3*nu/8...-1/4+nu/12-1/8-nu/8nu/61/8-3*nu/8]; KE = E/(1-nu^2)*[k(1)k(2)k(3)k(4)k(5)k(6)k(7)k(8)k(2)k(1)k(8)k(7)k(6)k(5)k(4)k(3)k(3)k(8)k(1)k(6)$

4. Conclusión

Fue una práctica sencilla debido a que solamente explicamos el funcionamiento del código de manera que las practicas que siguen lo pondremos en práctica de manera eficaz en todo lo que se nos aborde aquí en el laboratorio al igual en clase. Personalmente se me hizo muy fácil ya que simplemente era explicar el funcionamiento del código, y aunque no soy muy bueno en Matlab pude reconocer las funciones que se hacían en el código y con ayuda de mis compañeros pude comprender mejor lo que se hacía.

5. Referencia Blibliografica

Referencias

[1] Technical University of Denmark. 99 line topology optimization code. *Department of Solid Mechanics*.