%%%% A Matlab Code for SEMDOT Algorithm %%%%

function SEMDOT(nelx,nely,vol,rmin,nG)

% Author One: Yun-Fei Fu <y.fu@deakin.edu.au>

% Topology Optimization Group

% School of Engineering, Deakin University

% Author Two: Prof. Xiaodong Huang <xhuang@swin.edu.au>

% School of Engineering, Swinburne University of Technology

% Advisors: Dr. Kazem Ghabraie; Prof. Bernard Rolfe; Dr. Yanan Wang;

% Dr. Louis N.S. Chiu

% Date Created: 12/08/18

% Date last modified: 03/01/20

% Description:

% This function SEMDOT implements Smooth-Edged Material Distribution for

% Optimizing Topology (SEMDOT) algorithm

%%% Disclaimer:

%%%% This code is provided for educational or academic purposes and is not

%%%% guaranteed to be free of errors. The authors are not liable for any

%%%% problems caused by the use of this code.

% INPUT:

% nelx: The number of elements in the horizontal direction.

% nely: The number of elements in the vertical direction.

% vol: The prescribed value of the allowable volume.

% rmin: The predefined filter radius.

% nG: The number of grid points assigned to each element.

% OUTPUT:

% It.: The iteration number.

% Obj.: The value of compliance.

% Vol.: The volume fraction.

% ch.: The computational error.

% Topo.: The topological error.

% contourf(fnx, flipud(fny), xg-ls, [0 0]) shows optimized topologies.

%% INITIALIZATION

vx = repmat(vol,nely,nelx); vxPhys = vx; change = 1; loop = 0;

Emin = 0.001; maxloop = 1000; ngrid = nG-1; tolx = 0.001;

E0 = 1; nu = 0.3; penal = 1.5; rnmin = 1;

%% INITIALIZE MMA OPTIMIZER

nele = nely\*nelx; m = 1; n = nely\*nelx; nelm = nely\*nelx;

vxmin = 1e-3\*ones(nelm,1); vxmax = ones(nelm,1);

vxold1 = reshape(vx,nelm,1); vxold2 = vxold1;

a0 = 1; ai = 0; ci = 1000; di = 0; low = ones(nelm,1); upp = 1;

%% INITIALIZE HEAVISIDE SMOOTH FUNCTION

beta = 0.5; ER = 0.5;

%% PREPARE FINITE ELEMENT ANALYSIS

A11 = [12 3 -6 -3; 3 12 3 0; -6 3 12 -3; -3 0 -3 12];

A12 = [-6 -3 0 3; -3 -6 -3 -6; 0 -3 -6 3; 3 -6 3 -6];

B11 = [-4 3 -2 9; 3 -4 -9 4; -2 -9 -4 -3; 9 4 -3 -4];

B12 = [ 2 -3 4 -9; -3 2 9 -2; 4 9 2 3; -9 -2 3 2];

KE = 1/(1-nu^2)/24\*([A11 A12;A12' A11]+nu\*[B11 B12;B12' B11]);

nodenrs = reshape(1:(1+nelx)\*(1+nely),1+nely,1+nelx);

edofVec = reshape(2\*nodenrs(1:end-1,1:end-1)+1,nelx\*nely,1);

edofMat = repmat(edofVec,1,8)+repmat([0 1 2\*nely+[2 3 0 1] -2 -1],nelx\*nely,1);

iK = reshape(kron(edofMat,ones(8,1))',64\*nelx\*nely,1);

jK = reshape(kron(edofMat,ones(1,8))',64\*nelx\*nely,1);

%% ELEMENTAL NODES AND COORDINATES

[nodex,nodey] = meshgrid(0:nelx,0:nely);

[fnx,fny] = meshgrid(0:1/ngrid:nelx,0:1/ngrid:nely);

%% DEFINE LOADS AND SUPPORTS (CANTILEVER BEAM)

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

fixeddofs = (1:2\*(nely+1));

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

alldofs = [1:2\*(nely+1)\*(nelx+1)];

freedofs = setdiff(alldofs,fixeddofs);

%% PREPARE FILTER FOR ELEMENT

iH = ones(nelx\* nely\*(2\*(ceil(rmin)-1)+1)^2,1);

jH = ones(size(iH));

sH = zeros(size(iH));

k = 0;

for i1 = 1:nelx

for j1 = 1:nely

e1 = (i1-1)\*nely+j1;

for i2 = max(i1-(ceil(rmin)-1),1):min(i1+(ceil(rmin)-1),nelx)

for j2 = max(j1-(ceil(rmin)-1),1):min(j1+(ceil(rmin)-1),nely)

e2 = (i2-1)\*nely+j2;

k = k+1;

iH(k) = e1;

jH(k) = e2;

sH(k) = max(0,rmin-sqrt((i1-i2)^2+(j1-j2)^2));

end

end

end

end

H = sparse(iH,jH,sH); Hs = sum(H,2);

%% PREPARE FILTER FOR NODALS

inH = ones((nelx+1)\*(nely+1)\*(2\*(ceil(rnmin)+1))^2,1);

jnH = ones(size(inH)); snH = zeros(size(inH)); k =0;

[elex,eley] = meshgrid(1.5:nelx+0.5,1.5:nely+0.5);

for in1 = 1:nelx+1

for jn1 = 1:nely+1

en1 = (in1-1)\*(nely+1)+jn1;

for in2 = max(in1-ceil(rnmin),1):min(in1+ceil(rnmin)-1,nelx)

for jn2 = max(jn1-ceil(rnmin),1):min(jn1+ceil(rnmin)-1,nely)

en2 = (in2-1)\*nely+jn2; k = k+1; inH(k) = en1;

jnH(k) = en2;

snH(k) = max(0,rnmin-sqrt((in1-elex(jn2,in2))^2+(jn1-eley(jn2,in2))^2));

end

end

end

end

Hn = sparse(inH,jnH,snH); Hns = sum(Hn,2);

%% START ITERATION

while (change > tolx || tol>0.001) && loop < maxloop

loop = loop+1;

%% FE-ANALYSIS

sK = reshape(KE(:)\*(vxPhys(:)'\*E0+(1-vxPhys(:))'\*(Emin^penal\*E0)),64\*nelx\*nely,1);

K = sparse(iK,jK,sK); K = (K+K')/2;

U(freedofs) = K(freedofs,freedofs)\F(freedofs);

%% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

ce = reshape(sum((U(edofMat)\*KE).\*U(edofMat),2),nely,nelx);

c(loop) = sum(sum((vxPhys.\*E0+(1-vxPhys).\*(Emin^penal\*E0)).\*ce));

dc = -penal\*((1-vxPhys)\*Emin.^(penal-1)+vxPhys).\*E0.\*ce;

dv = ones(nely,nelx);

%% FILTERING/MODIFICATION OF SENSITIVITIES

dc(:) = H\*(dc(:)./Hs);

dv(:) = H\*(dv(:)./Hs);

%% UPDATE DESIGN VARIABLES AND FILTERED ELEMENTAL VOLUME FRACTIONS

vxval = reshape(vx,nelm,1);

fval = sum(vxPhys(:))/(vol\*nely\*nelx)-1;

dfdx = dv(:)'/(vol\*nely\*nelx);

f0val = c;

df0dx = dc(:);

[vxmma,~,~,~,~,~,~,~,~,low,upp] = ...

mmasub(m,n,loop,vxval,vxmin,vxmax,vxold1,vxold2,f0val,df0dx,fval,dfdx,low,upp,a0,ai,ci,di);

vxnew = reshape(vxmma,nely,nelx);

vxPhys(:) = (H\*vxnew(:))./Hs;

vxold2 = vxold1(:);

vxold1 = vx(:);

%% ASSIGN FILTERED ELEMENTAL VOLUME FRACTIONS TO NODAL DENSITIES

xn = reshape((Hn\*vxPhys(:)./Hns),nely+1,nelx+1);

%% UPDATE POINT DESNIGY BY A HEAVISIDE SMOOTH/STEP FUNCTION

xg = interp2(nodex,nodey,xn,fnx,fny,'linear');

l1 =0; l2 = 1;

while (l2-l1) > 1.0e-5

ls = (l1+l2)/2.0;

xgnew = max(0.001,(tanh(beta\*ls)+tanh(beta\*(xg-ls)))/(tanh(beta\*ls)+tanh(beta\*(1-ls))));

if sum(sum(xgnew))/((ngrid\*nelx+1)\*(ngrid\*nely+1)) - sum(vxPhys(:))/(nelx\*nely) > 0

l1 = ls;

else

l2 = ls;

end

end

%% ASSEMBLE GRID POINTS TO ELEMENTS

vxPhys(:) = 0;

Terr = 0;

Tm=[];

for i = 1:nelx

for j = 1:nely

e = (i-1)\*nely + j;

for i1 = ngrid\*(i-1)+1:ngrid\*i+1

for j1 = ngrid\*(j-1)+1:ngrid\*j+1

Tm = [Tm;xgnew(j1,i1)];

vxPhys(e) = vxPhys(e)+xgnew(j1,i1);

end

end

if min(Tm)>0.001 && max(Tm)<1

Terr = Terr+1;

end

Tm = [];

end

end

vxPhys = vxPhys/(ngrid+1)^2;

%% CHECK CONVERGENCE

change = sum(abs(vxnew(:)-vx(:)))/(vol\*nely\*nelx);

tol = Terr/nele;

vx = vxnew;

%% PLOT RESULTS

fprintf('It.:%3i Obj.:%8.4f Vol.:%4.3f ch.:%4.5f Topo.:%7.5f\n\n',loop,c(loop),mean(vxPhys(:)),change,tol);

contourf(fnx, flipud(fny), xg-ls, [0 0]);

axis equal; axis tight; axis off; pause(1e-6);

%% UPDATE HEAVISIDE SMOOTH FUNCTION

beta = beta+ER; fprintf('Parameter beta increased to %g.\n',beta);

end