PA7 Report

Problem (a)

The following figures (fig.1 to fig. 3) show the same optimized structure but with different . Since the is used to do the filtering and tells how big your element is. If there is no filter and each element will not be affected or averaged by adjacent elements. That is what fig. 1 shows, a ill-posed structure. When , there is a filter now and this will let each element take adjacent three to four elements into consideration and average their design variable change rate, each element size is estimated 1.5. When , the elements's size will be too large. And the averaged number will be between 0 and 1, that is why there is some gray area in the fig. 3.

Fig. 1 Optimized structure with $r_{min} = 1.0$

Fig. 2 Optimized structure with $r_{min} = 1.5$

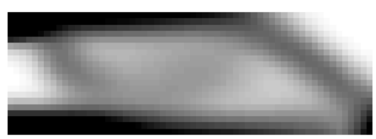

Fig. 3 Optimized structure with $r_{min} = 5.0$

Problem (b)

The following figures(fig. 4 to fig. 6) show the same optimized structure but with different p. P is penalization factor and sufficiently big "p" will penalize internediate densities, resulting in "black and white" designs. P is a numerical method, which makes our design more clear. That is why fig. 6 has the most clear image when compared to fig. 4 and fig. 5.

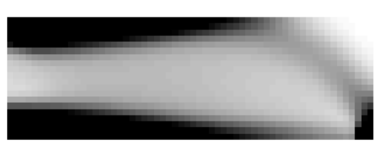

Fig. 4 Optimized structure with p = 1.0

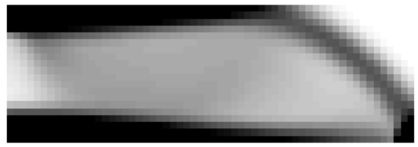

Fig. 5 Optimized structure with p = 2.0

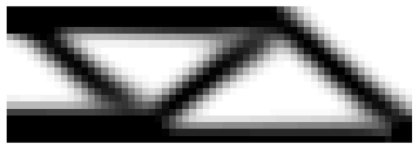

Fig. 6 Optimized structure with p = 3.0

Problem (c)

Fig. 7 shows a topology optimized cantilever beam. It has two load cases, one on the top and one on the bottom. There is a hole in it and the left edge is clamped to a wall.

Fig. 7 Topology optimization of a cantilever beam with a hole under 2 different load cases

Modified code:

```
%%%% Modified 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND%%%

function PA7(nelx,nely,volfrac,penal,rmin);

% INITIALIZE

x(1:nely,1:nelx) = volfrac;

% ADD A HOLE

for ely = 1:nely
```

```
  for elx = 1:nelx
    if sqrt((elx-nelx*2/3.)^2+(ely-nely/2.)^2)<nely/3.
        passive(ely,elx) = 1;
        x(ely,elx) = 0.001;
    else
        passive(ely,elx) = 0;
    end
  end
end
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;
      dc(ely,elx) = 0.;
      for i = 1:2
      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);
      c = c + x (ely,elx)^penal*Ue'*KE*Ue;
      dc(ely,elx) = dc(ely,elx) - penal*x (ely,elx)^(penal-1)*Ue'*KE*Ue;
      end
    end
```

```
  end
% FILTERING OF SENSITIVITIES
  [dc]   = check(nelx,nely,rmin,x,dc);
  % passive(1:nely,1:nelx) = 0;
% 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 ('%4 i',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 > 1e-4)
  lmid = 0.5*(l2+l1);
  xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
  xnew(find(passive)) = 0.001;
  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),2); U = zeros(2*(nely+1)*(nelx+1),2);
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)
% DEFINE LOAD 1
for elx = (nelx/2+1):(nelx+1)
    node = (nely+1)*(elx-1)+1;
```

```
    F(2*node,1) = 10;
end
F(2*((nely+1)*nelx/2+1),1) = 5;
F(2*((nely+1)*nelx+1),1) = 5;
% DEFINE LOAD 2
for elx = (nelx/2+1):(nelx+1)
    node = (nely+1)*(elx-1)+nely+1;
    F(2*node,2) = -10;
end
F(2*((nely+1)*nelx/2+nely+1),2) = -5;
F(2*((nely+1)*nelx+nely+1),2) = -5;
fixeddofs   = [1:2*(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)];
```