Appendix 1

Appendix 1 contains the MATLAB code for the multiscale concurrent optimization algorithm. The MATLAB code is divided into seven codes which are listed the following:

Appendix 1-1 contain the main program i.e., Concurrent MOO.m

Appendix 1-2 has the auxiliary function get_Initials.m

Appendix 1-3 has the function Homogenization_full.m

Appendix 1-4 has the function Concurrent_solver.m

Appendix 1-5 has the function elementMatVec.m

Appendix 1-6 has the function Objective_Calculator.m

Appendix 1-7 has the function OC_2D.m

To run the program, make each subsection as a dedicated code in a separate MATLAB file (as shown in Fig. A1) and run the main program Concurrent MOO.m. The code is sectioned to give the reader the freedom to be fully customized.

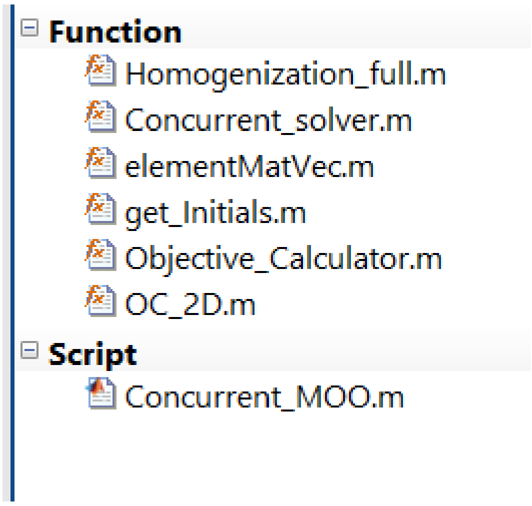

Fig. A1 The MATLAB code and its sub-functions

Appendix 1-1 the main program (Concurrent_MOO.m)

```
%% Concurrent Multiphysics Multiscale Topology Optimization
%% Material input
%% Microscale input
Micro_nelx=100;Micro_nely=100;Micro_volfrac=.5;penal=3;Micro_rmin=1.5;
% Macroscale input
Macro_nelx=200;Macro_nely=90;Macro_volfrac=.5;%Secound Example Macro_nelx=150;Macro_nely=150
Macro_rmin=1.5;dx = 1/Macro_nelx; dy = 1/Macro_nely;
% Optimization input
maxIter=500;eta=0.5;
%% Mechanical FEM
Macro_nodenrs_mech = reshape(1:(1+Macro_nelx)*(1+Macro_nely),1+Macro_nely,1+Macro_nelx);
Macro_edofVec_mech = reshape(2*Macro_nodenrs_mech(1:end-1,1:end-1)+1,Macro_nelx*Macro_nely,1);
Macro_edofMat_mech = repmat(Macro_edofVec_mech,1,8)+repmat([0 1 2*Macro_nely+[2 3 0 1] -2 -1], ...
Macro_nelx*Macro_nely,1);
iK_mech = reshape(kron(Macro_edofMat_mech,ones(8,1))',64*Macro_nelx*Macro_nely,1);
jK_mech = reshape(kron(Macro_edofMat_mech,ones(1,8))',64*Macro_nelx*Macro_nely,1);
U = zeros(2*(Macro_nely+1)*(Macro_nelx+1),1);
F = zeros(2*(Macro_nely+1)*(Macro_nelx+1),1);
F(2*(Macro_nelx+1)*(Macro_nely+1),1)=1 ;
fixeddofs_mech=union([2:2*(Macro_nely+1)],[1]);
alldofs_mech = [1:2*(Macro_nely+1)*(Macro_nelx+1)];
freedofs_mech = setdiff(alldofs_mech,fixeddofs_mech);
%% Heat FEM
Macro_nodenrs = reshape(1:(1+Macro_nelx)*(1+Macro_nely),1+Macro_nely,1+Macro_nelx);
Macro_edofVec = reshape(Macro_nodenrs(1:end-1,1:end-1)+1,Macro_nelx*Macro_nely,1);
Macro_edofMat_Heat = repmat(Macro_edofVec,1,4)+repmat([0 Macro_nely+[1 0] -1],Macro_nelx* ...
Macro_nely,1);
iK = reshape(kron(Macro_edofMat_Heat,ones(4,1))',16*Macro_nelx*Macro_nely,1);
jK = reshape(kron(Macro_edofMat_Heat,ones(1,4))',16*Macro_nelx*Macro_nely,1);
P = sparse((Macro_nely+1)*(Macro_nelx+1),1); P(:,1) = .01;
fixeddofs_Heat = [Macro_nely/2-20:Macro_nely/2+20];
T = zeros((Macro_nely+1)*(Macro_nelx+1),1);
alldofs_Heat = [1:(Macro_nely+1)*(Macro_nelx+1)];
freedofs_Heat = setdiff(alldofs_Heat,fixeddofs_Heat);
%% Mechanical  Compliance **
[Macro_x,Micro_x]=get_Initials(Macro_volfrac,Micro_volfrac,Micro_nelx,Micro_nely, ...
Macro_nelx,Macro_nely);
Micro_xPhys = Micro_x;Macro_xPhys=Macro_x;
[QM, ~] = Homogenization_full(Micro_xPhys,1);
[~,~,~,~,KE_Homogenized_Mech] = elementMatVec(dx/2, dy/2,QM,1);
[N_macro_Mech]=Objective_Calculator(iK_mech,jK_mech,Macro_nely,Macro_nelx,freedofs_mech, ...
Macro_edofMat_mech,Macro_xPhys,penal,KE_Homogenized_Mech,F,1);
[U_macro_Mech,~]=Concurrent_solver(Micro_nelx,Micro_nely,Macro_nelx,Macro_nely, ...
Macro_edofMat_mech,iK_mech,jK_mech,F,U,freedofs_mech,freedofs_Heat,Macro_edofMat_Heat, ...
T,P,iK,jK,Macro_volfrac,Micro_volfrac,Micro_rmin,Macro_rmin,dx,dy,penal,maxIter,1);
Diffrence_Nadir_Utopia_Mech=N_macro_Mech-U_macro_Mech;
clear Macro_x Micro_x
%% Heat Compliance **
[Macro_x,Micro_x]=get_Initials(Macro_volfrac,Micro_volfrac,Micro_nelx,Micro_nely,Macro_nelx,Macro_nely);
Micro_xPhys = Micro_x;Macro_xPhys=Macro_x;
[Q,~]=Homogenization_full(Micro_xPhys,2);
[~,~,~,~,KE_Homogenized_Heat] = elementMatVec(dx/2, dy/2,Q,2);
[N_macro_Heat]=Objective_Calculator(iK,jK,Macro_nely,Macro_nelx,freedofs_Heat,Macro_edofMat_Heat, ...
Macro_xPhys,penal,KE_Homogenized_Heat,P,2);
[U_macro_Heat,~]=Concurrent_solver(Micro_nelx,Micro_nely,Macro_nelx,Macro_nely,Macro_edofMat_mech, ...
iK_mech,jK_mech,F,U,freedofs_mech,freedofs_Heat,Macro_edofMat_Heat,T,P,iK,jK,Macro_volfrac, ...
Micro_volfrac,Micro_rmin,Macro_rmin,dx,dy,penal,maxIter,2);
Diffrence_Nadir_Utopia_Heat=N_macro_Heat-U_macro_Heat;
%% MOO optimization loop
clear Macro_x Micro_x
[Macro_x,Micro_x]=get_Initials(Macro_volfrac,Micro_volfrac,Micro_nelx,Micro_nely,Macro_nelx,Macro_nely);
```

```
Micro_xPhys = Micro_x;Macro_xPhys=Macro_x;loop = 0;
while loop < maxIter;
loop = loop+1;
%% Mechanical Part
[QM,dQM]=Homogenization_full(Micro_xPhys,1);
[~,~,~,~,KE_Macro_Homogenized_mech] = elementMatVec(dx/2, dy/2,QM,1);
sK_mech = reshape(KE_Macro_Homogenized_mech(:)*(1e-6+Macro_xPhys(:)'.^penal*(1-1e-6)),64*Macro_nelx*Macro_nely,1);
K_mech = sparse(iK_mech,jK_mech,sK_mech); K_mech = (K_mech+K_mech')/2;
U(freedofs_mech,:) = K_mech(freedofs_mech,freedofs_mech)\F(freedofs_mech,:);
ce_mech = reshape(sum((U(Macro_edofMat_mech)*KE_Macro_Homogenized_mech).*U(Macro_edofMat_mech),2),Macro_nely,Macro_nelx);
c= sum(sum((1e-6+Macro_xPhys.^penal*(1-1e-6)).*ce_mech));
Macro_dc_mech = -penal*(1-1e-6)*Macro_xPhys.^(penal-1).*ce_mech;
Micro_dc_mech = zeros(Micro_nely, Micro_nelx);
for i = 1:Micro_nelx*Micro_nely
dQe_mech = [dQM{1,1}(i) dQM{1,2}(i) dQM{1,3}(i);
            dQM{2,1}(i) dQM{2,2}(i) dQM{2,3}(i);
            dQM{3,1}(i) dQM{3,2}(i) dQM{3,3}(i)];
[~,~,~,~,dKE_mech] = elementMatVec(dx/2, dy/2,dQe_mech,1);
dce_mech = reshape(sum((U(Macro_edofMat_mech)*dKE_mech).*U(Macro_edofMat_mech),2),Macro_nely,Macro_nelx);
Micro_dc_mech(i) = -sum(sum((1e-6+Macro_xPhys.^penal*(1-1e-6)).*dce_mech));
end
%% Heat Part
[Q,dQ]=Homogenization_full(Micro_xPhys,2);
[~,~,~,~,KE_Homogenized_Heat] = elementMatVec(dx/2, dy/2,Q,2);
T = zeros((Macro_nely+1)*(Macro_nelx+1),1);
sK = reshape(KE_Homogenized_Heat(:)*(1e-6+Macro_xPhys(:)'.^penal*(1-1e-6)),16*Macro_nelx*Macro_nely,1);
KH = sparse(iK,jK,sK); KH = (KH+KH')/2;
T(freedofs_Heat,:) = KH(freedofs_Heat,freedofs_Heat)\P(freedofs_Heat,:);
ce_Heat = reshape(sum((T(Macro_edofMat_Heat)*KE_Homogenized_Heat).*T(Macro_edofMat_Heat),2),Macro_nely,Macro_nelx);
c_Heat = sum(sum((1e-6+Macro_xPhys.^penal*(1-1e-6)).*ce_Heat))
Macro_dcH = -penal*(1-1e-6)*Macro_xPhys.^(penal-1).*ce_Heat;
Macro_dv = ones(Macro_nely, Macro_nelx);
Micro_dcH = zeros(Micro_nely, Micro_nelx);
for i = 1:Micro_nelx*Micro_nely;
dQe = [dQ{1,1}(i) dQ{1,2}(i);
dQ{2,1}(i) dQ{2,2}(i)];
[~,~,~,~,dKE] = elementMatVec(dx/2, dy/2,dQe,2);
dce = reshape(sum((T(Macro_edofMat_Heat)*dKE).*T(Macro_edofMat_Heat),2),Macro_nely,Macro_nelx);
Micro_dcH(i) = -sum(sum((1e-6+Macro_xPhys.^penal*(1-1e-6)).*dce));
end
Micro_dv = ones(Micro_nely, Micro_nelx);
%% MMO objective function
Macro_dc= eta*Macro_dcH/(Diffrence_Nadir_Utopia_Heat)+(1-eta)*Macro_dc_mech/(Diffrence_Nadir_Utopia_Mech);
Micro_dc=eta*Micro_dcH/(Diffrence_Nadir_Utopia_Heat)+(1-eta)*Micro_dc_mech/(Diffrence_Nadir_Utopia_Mech);
%% Optimality Criteria Update for Macro and Micro Element Densities
[Macro_x, Macro_xPhys, Macro_change] = OC_2D(Macro_x, Macro_dc, Macro_volfrac,Macro_rmin, Macro_nelx,Macro_nely);
[Micro_x, Micro_xPhys, Micro_change] = OC_2D(Micro_x, Micro_dc, Micro_volfrac,Micro_rmin, Micro_nelx,Micro_nely);
Macro_xPhys = reshape(Macro_xPhys, Macro_nely, Macro_nelx); Micro_xPhys = reshape(Micro_xPhys, Micro_nely, Micro_nelx);
%% Printing the Results
fprintf(' It.:%5i Obj.:%11.4f Macro_Vol.:%7.3f Micro_Vol.:%7.3f Macro_ch.:%7.3f Micro_ch.:%7.3f4\n',loop,mean(Macro_xPhys(:)),...
mean(Micro_xPhys(:)), Macro_change, Micro_change);
hold on;
figure (6);clf;colormap(gray); imagesc(1-Micro_x); caxis([0 1]);
axis equal; axis off; t1=title('Microscale Design');t1.Color ='#FE8402';drawnow;
```

```
figure (7);clf;colormap(gray); imagesc(1-Macro_x); caxis([0 1]);
axis equal; axis off; t2=title('Macroscale Design');t2.Color ='#0053ED';drawnow;
end;
disp('********* Concurrent Multiphysics Multiscale Topology Optimization is finished ***************');
```

Appendix 1-2 the function (get_Initials.m)

```
function [Macro_x,Micro_x]=get_Initials(Macro_volfrac,Micro_volfrac,Micro_nelx,...
Micro_nely,Macro_nelx,Macro_nely)
Macro_x = repmat(Macro_volfrac,Macro_nely,Macro_nelx);
Micro_x = repmat(Micro_volfrac,Micro_nely,Micro_nelx);
for i = 1:Micro_nelx;for j = 1:Micro_nely;
if sqrt((i-Micro_nelx/2)^2+(j-Micro_nely/2)^2)< min(Micro_nelx,Micro_nely)/4;
Micro_x(j,i) = Micro_volfrac/4;
end;end;end;end
```

Appendix 1-3 the function (Homogenization_full.m)

```
function [CH,dCH]= Homogenization_full(Micro,select_case)
%% Initialize input data
[nelx,nely]=size(Micro);
for i=1:nelx
for j=1:nely
if Micro(i,j)<.1;x(i,j)=2;else;x(i,j)=1;
end;end;end
lx=1;ly=1;lambda=[1 0];mu=[1 0];
dx = lx/nelx; dy = ly/nely;
nel = nelx*nely;
Q=ones(3,3);
[keLambda, keMu, feLambda, feMu,~] = elementMatVec(dx/2, dy/2,Q,select_case);
nodenrs = reshape(1:(1+nelx)*(1+nely),1+nely,1+nelx);
edofVec = reshape(2*nodenrs(1:end-1,1:end-1)+1,nel,1);
edofMat = repmat(edofVec,1,8)+repmat([0 1 2*nely+[2 3 0 1] -2 -1],nel,1);
switch(select_case);case(1);
case(2)
keMu(1:2:end,1:2:end) = keMu(1:2:end,1:2:end)+keMu(2:2:end, 2:2:end);end
%% Impose Periodic Boundary Conditions
nn = (nelx+1)*(nely+1);
nnP = (nelx)*(nely);
nnPArray = reshape(1:nnP, nely, nelx);
nnPArray(end+1,:) = nnPArray(1,:);
nnPArray(:,end+1) = nnPArray(:,1);
dofVector = zeros(2*nn, 1);
dofVector(1:2:end) = 2*nnPArray(:)-1;
dofVector(2:2:end) = 2*nnPArray(:);
edofMat = dofVector(edofMat);
ndof = 2*nnP;
%% Assemble Stiffness Matrix
iK = kron(edofMat,ones(8,1))';
jK = kron(edofMat,ones(1,8))';
lambda = lambda(1)*(x==1) + lambda(2)*(x==2);
mu = mu(1)*(x==1) + mu(2)*(x==2);
for i=1 nely;for j=1;
lambda(i,j)=lambda(i,j);
mu(i,j)=mu(i,j);end;end
sK = keLambda(:)*lambda(:).' + keMu(:)*mu(:).';
K = sparse(iK(:), jK(:), sK(:), ndof, ndof);
%% Load Vectors and Solution
sF = feLambda(:)*lambda(:).'+feMu(:)*mu(:).';
iF = repmat(edofMat',3,1);
jF = [ones(8,nel); 2*ones(8,nel); 3*ones(8,nel)];
F = sparse(iF(:), jF(:), sF(:), ndof, 3);
% Solve
activedofs=edofMat(x==1,:);
activedofs=sort(unique(activedofs(:)));
switch(select_case);
case(1)
Xi=zeros(ndof,3);
Xi(activedofs(3:end),:)
=K(activedofs(3:end),activedofs(3:end))\F(activedofs(3:end),:);
case(2)
Xi = zeros(ndof,2);
Xi(activedofs(3:2:end),:) = K(activedofs(3:2:end),activedofs(3:2:end))\...
[F(activedofs(3:2:end),1) F(activedofs(4:2:end),2)];end
%% Homogenization
Xi0 = zeros(nel, 8, 3);Xi0_e = zeros(8, 3);
ke = keMu + keLambda;fe = feMu + feLambda;
Loop=0;
switch(select_case);
case(1)
Loop=3;
Xi0_e([3 5:end],:) = ke([3 5:end],[3 5:end])\fe([3 5:end],:);
case(2)
Loop=2;
Xi0_e(3:2:end,1:2) = keMu(3:2:end,3:2:end)\[feMu(3:2:end,1) feMu(4:2:end,2)]; end
Xi0(:,:,1) = kron(Xi0_e(:,1)', ones(nel,1));
Xi0(:,:,2) = kron(Xi0_e(:,2)', ones(nel,1));
Xi0(:,:,3) = kron(Xi0_e(:,3)', ones(nel,1));
CH = zeros(3);
cellVolume = lx*ly;
for i = 1:Loop
```

```
for j = 1:Loop
sumLambda = ((Xi0(:,:,i) - Xi(edofMat+(i-1)*ndof))*keLambda).*...
(Xi0(:,:,j) - Xi(edofMat+(j-1)*ndof));
sumMu = ((Xi0(:,:,i) - Xi(edofMat+(i-1)*ndof))*keMu).*...
(Xi0(:,:,j) - Xi(edofMat+(j-1)*ndof));
sumLambda = reshape(sum(sumLambda,2), nely, nelx);
sumMu = reshape(sum(sumMu,2), nely, nelx);
qe=lambda.*sumLambda + mu.*sumMu;
CH(i,j) =  1/cellVolume*sum(sum(qe.*(1e-6+Micro.^(3)*(1-1e-6))));
dCHq = 1/cellVolume*(qe.*(3*Micro.^(3-1)*(1-0.0001)).'); dCH{i,j}=dCHq;
end;end
```

Appendix 1-4 the function (Concurrent_solver.m)

```
function [C_concurrent,CH]=Concurrent_solver(Micro_nelx,Micro_nely,Macro_nelx,Macro_nely,Macro_edofMat_mech,iK_mech,jK_mech,F,U,freedofs_mech, ...
freedofs_Heat,Macro_edofMat_Heat,T,P,iK,jK,Macro_volfrac,Micro_volfrac,Micro_rmin,Macro_rmin,dx,dy,penal,maxIter,option)
%% Preparing Design Variables
clear Macro_x Micro_x
[Macro_x,Micro_x]=get_Initials(Macro_volfrac,Micro_volfrac,Micro_nelx,Micro_nely,Macro_nelx,Macro_nely);
Micro_xPhys = Micro_x;Macro_xPhys=Macro_x;
loop = 0;
while loop < maxIter;
loop = loop+1;
switch(option)
    case(1);
[Q,dQ]=Homogenization_full(Micro_xPhys,1);
[~,~,~,~,KE_Macro_Homogenized_mech] = elementMatVec(dx/2, dy/2,Q,1);
sK_mech = reshape(KE_Macro_Homogenized_mech(:)*(1e-6+Macro_xPhys(:)'.^penal*(1-1e-6)),64*Macro_nelx*Macro_nely,1);
K_mech = sparse(iK_mech,jK_mech,sK_mech); K_mech = (K_mech+K_mech')/2;
U(freedofs_mech,:) = K_mech(freedofs_mech,freedofs_mech)\F(freedofs_mech,:);
ce_mech = reshape(sum((U(Macro_edofMat_mech)*KE_Macro_Homogenized_mech).*U(Macro_edofMat_mech),2),Macro_nely,Macro_nelx);
c= sum(sum((1e-6+Macro_xPhys.^penal*(1-1e-6)).*ce_mech));
Macro_dc_mech = -penal*(1-1e-6)*Macro_xPhys.^(penal-1).*ce_mech;
Micro_dc_mech = zeros(Micro_nely, Micro_nelx);
for i = 1:Micro_nelx*Micro_nely
dQe_mech = [dQ{1,1}(i) dQ{1,2}(i) dQ{1,3}(i);
            dQ{2,1}(i) dQ{2,2}(i) dQ{2,3}(i);
            dQ{3,1}(i) dQ{3,2}(i) dQ{3,3}(i)];
[~,~,~,~,dKE_mech] = elementMatVec(dx/2, dy/2,dQe_mech,1);
dce_mech = reshape(sum((U(Macro_edofMat_mech)*dKE_mech).*U(Macro_edofMat_mech),2),Macro_nely,Macro_nelx);
Micro_dc_mech(i) = -sum(sum((1e-6+Macro_xPhys.^penal*(1-1e-6)).*dce_mech));
end
[Macro_x, Macro_xPhys, Macro_change] = OC_2D(Macro_x, Macro_dc_mech, Macro_volfrac,Macro_rmin, Macro_nelx,Macro_nely);
[Micro_x, Micro_xPhys, Micro_change] = OC_2D(Micro_x, Micro_dc_mech, Micro_volfrac,Micro_rmin, Micro_nelx,Micro_nely);

case(2)
[Q,dQ]=Homogenization_full(Micro_xPhys,2);
[~,~,~,~,KE_Homogenized_Heat] = elementMatVec(dx/2, dy/2,Q,2);
sK = reshape(KE_Homogenized_Heat(:)*(1e-6+Macro_xPhys(:)'.^penal*(1-1e-6)),16*Macro_nelx*Macro_nely,1);
K = sparse(iK,jK,sK); K = (K+K')/2;
T(freedofs_Heat,:) = K(freedofs_Heat,freedofs_Heat)\P(freedofs_Heat,:);
ce = reshape(sum((T(Macro_edofMat_Heat)*KE_Homogenized_Heat).*T(Macro_edofMat_Heat),2),Macro_nely,Macro_nelx);
c= sum(sum((1e-6+Macro_xPhys.^penal*(1-1e-6)).*ce));
Macro_dcH =-penal*(1-1e-6)*Macro_xPhys.^(penal-1).*ce;
Macro_dv = ones(Macro_nely, Macro_nelx);
Micro_dcH = zeros(Micro_nely, Micro_nelx);
for i = 1:Micro_nelx*Micro_nely;
dQe = [dQ{1,1}(i) dQ{1,2}(i);
dQ{2,1}(i) dQ{2,2}(i)];
[~,~,~,~,dKE] = elementMatVec(dx/2, dy/2,dQe,2);
dce = reshape(sum((T(Macro_edofMat_Heat)*dKE).*T(Macro_edofMat_Heat),2),Macro_nely,Macro_nelx);
Micro_dcH(i) = -sum(sum((1e-6+Macro_xPhys.^penal*(1-1e-6)).*dce));
end
[Macro_x, Macro_xPhys, Macro_change] = OC_2D(Macro_x, Macro_dcH, Macro_volfrac,Macro_rmin, Macro_nelx,Macro_nely);
[Micro_x, Micro_xPhys, Micro_change] = OC_2D(Micro_x, Micro_dcH, Micro_volfrac,Micro_rmin, Micro_nelx,Micro_nely);

end
```

```
Macro_xPhys = reshape(Macro_xPhys, Macro_nely, Macro_nelx); Micro_xPhys =
reshape(Micro_xPhys, Micro_nely, Micro_nelx);
%% To be deleted when submitted the Program
clf;
%% PRINT RESULTS
fprintf(' It.:%5i Obj.:%11.4f Macro_Vol.:%7.3f Micro_Vol.:%7.3f Macro_ch.:%7.3f
Micro_ch.:%7.3f\n',loop,mean(Macro_xPhys(:))...
,mean(Micro_xPhys(:)), Macro_change, Micro_change);
hold on;
switch(option)
case(1);
figure (1);clf;colormap(gray); imagesc(1-Micro_x); caxis([0 1]);
axis equal; axis off; t1=title('Microscale Design Mechanical');t1.Color ='#E50112
';drawnow;
figure (2);clf;colormap(gray); imagesc(1-Macro_x); caxis([0 1]);
axis equal; axis off; t2=title('Macroscale Design Mechanical');t2.Color
='#0C95D1';drawnow;
case(2);
figure (4);clf;colormap(gray); imagesc(1-Micro_x); caxis([0 1]);
axis equal; axis off; t1=title('Microscale Design Heat');t1.Color ='#E600E2';drawnow;
figure (5);clf;colormap(gray); imagesc(1-Macro_x); caxis([0 1]);
axis equal; axis off; t2=title('Macroscale Design Heat');t2.Color ='#09D61F';drawnow;
end;
end;
C_concurrent=c;CH=Q;
disp('********End Concurrent Solver case *********\n');
end
```

Appendix 1-5 the function (elementMatVec.m)

```
function [keLambda, keMu, feLambda, feMu,KE] = elementMatVec(a, b,Q,select_case)
switch(select_case)
case(1)
DH=Q;
CMu = diag([2 2 1]); CLambda = zeros(3); CLambda(1:2,1:2) = 1;
LM=1;Bele= (1/2/LM)*[-1 0 1 0 1 0 -1 0; 0 -1 0 -1 0 1 0 1;-1 -1 -1 1 1 1 1 -1];
KE=zeros(8,8);
case(2)
DH(1:2,1:2)=Q(1:2,1:2);
CMu = diag([1 1 0]); CLambda = zeros(3);
Bele=0.5*[1,-1,-1,1;1,1,-1,-1];
KE=zeros(4,4);end;
xx=[-1/sqrt(3), 1/sqrt(3)]; yy = xx;
ww=[1,1];
keLambda = zeros(8,8); keMu = zeros(8,8);
feLambda = zeros(8,3); feMu = zeros(8,3);
L = [1 0 0 0; 0 0 0 1; 0 1 1 0];
for ii=1:2
for jj=1:2
x = xx(ii); y = yy(jj);
dNx = 1/4*[-(1-y) (1-y) (1+y) -(1+y)];
dNy = 1/4*[-(1-x) -(1+x) (1+x) (1-x)];
J = [dNx; dNy]*[ -a  a  a  -a;  -b  -b  b  b]';
detJ = J(1,1)*J(2,2) - J(1,2)*J(2,1);
invJ = 1/detJ*[J(2,2) -J(1,2); -J(2,1) J(1,1)];
weight = ww(ii)*ww(jj)*detJ;
G = [invJ zeros(2); zeros(2) invJ];
dN = zeros(4,8);
dN(1,1:2:8) = dNx;
dN(2,1:2:8) = dNy;
dN(3,2:2:8) = dNx;
dN(4,2:2:8) = dNy;
B = L*G*dN;
KE = KE + ww(ii)*ww(jj)*detJ*Bele'*DH*Bele;
keLambda = keLambda + weight*(B' * CLambda * B);
keMu = keMu + weight*(B' * CMu * B);
feLambda = feLambda + weight*(B' * CLambda * diag([1 1 1]));
feMu = feMu + weight*(B' * CMu * diag([1 1 1]));
end;end;end
```

Appendix 1-6 the function (Objective_Calculator.m)

```
function [c]=Objective_Calculator(iK,jK,nely,nelx,freedofs,edofMat,x,penal,KE,F,option)
switch(option)
case(1) %Mechanical
U = zeros(2*(nely+1)*(nelx+1),1);
sK = reshape(KE(:)*(1e-6+x(:)'.^penal*(1-1e-6)),64*nely*nelx,1);K = sparse(iK,jK,sK); K = (K+K')/2;
U(freedofs,:) = K(freedofs,freedofs)\F(freedofs,:);
ce= reshape(sum((U(edofMat)*KE).*U(edofMat),2),nely,nelx);
c= sum(sum((1e-6+x.^penal*(1-1e-6)).*ce ));
case(2) %Thermal
U = zeros((nely+1)*(nelx+1),1);
sK = reshape(KE(:)*(1e-6+x(:)'.^penal*(1-1e-6)),16*nelx*nely,1);K = sparse(iK,jK,sK); K = (K+K')/2;
U(freedofs,:) = K(freedofs,freedofs)\F(freedofs,:);
ce = reshape(sum((U(edofMat)*KE).*U(edofMat),2),nely,nelx);
c = sum(sum((1e-6+x.^penal*(1-1e-6)).*ce));
end
```

Appendix 1-7 the function (OC_2D.m)

```
function [x, xPhys, change] = OC_2D(x, dc,volfrac,rmin, nelx,nely)
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);dv = ones(nely,nelx);
%% Filtering
dc(:) = H*(dc(:)./Hs);
dv(:) = H*(dv(:)./Hs);
%% Optimality Criteria Update of Design Variables & Physical Densities
l1 = 0; l2 = 1e9; move = 0.009;
while (l2-l1)/(l1+l2) > 1e-6
lmid = 0.5*(l2+l1);
xnew = max(0,max(x-move,min(1,min(x+move,x.*sqrt(abs(dc)./dv/lmid)))));
xPhys(:) = (H*xnew(:))./Hs;
if sum(xPhys(:)) > volfrac*nelx*nely, l1 = lmid; else l2 = lmid; end
end;  change = max(abs(xnew(:)-x(:)));x = xnew;end
```