```
classdef Optimizer
    % Optimizer - class contains only the code that actually
 optimizes.
    %

    properties
%         ExxInterp=1;
%         EyyInterp=1;
%         thetaInterp=1;
%         rhoInterp=1;
    end

    methods
        % -------------------------------
        % TOPOOLOGY, SIMP METHOD
        % -------------------------------
        function DV = OptimizeTopology(obj,DV, config,
 matProp,masterloop)
            DV = DV.CalculateTopologySensitivity(config, matProp,
 masterloop);
            % normalize the sensitivies  by dividing by their max
 values.
            if (config.w1 ~= 1) % if we are using the heat objective
                temp1Max =-1* min(min(DV.sensitivityElastic));
                DV.sensitivityElastic = DV.sensitivityElastic/
temp1Max;
                temp2Max = -1* min(min(DV.sensitivityHeat));
                DV.sensitivityHeat = DV.sensitivityHeat/temp2Max;
                DV.dc =1000* (config.w1*DV.sensitivityElastic
+config.w2*DV.sensitivityHeat); % add the two sensitivies together
 using their weights
            else
                DV.dc = config.w1*DV.sensitivityElastic;
            end



            % FILTERING OF SENSITIVITIES
            [DV.dc]   = DV.check( config.nelx,
 config.nely,config.rmin,DV.x,DV.dc);
            % DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
            moveLimit=0.1;
            [DV.x]    = OC( config.nelx,
 config.nely,DV.x,config.totalVolume,DV.dc, DV, config,moveLimit);

            DV.x=DV.ApplyLoadSpecificEmptyRegions(config,DV.x);

        end

        % ----------------------------------
        % VOLUME FRACTION OPTIMIZATION
        % ----------------------------------
        function DV = OptimizeVolumeFraction(obj,DV,config, matProp,
 masterloop)
            DV = DV.CalculateMaterialGradientSensitivity(config,
 matProp, masterloop);

            DV =  DV.CalculateVolumeFractions(config,matProp);

            totalVolLocal = DV.currentVol1Fraction+
 DV.currentVol2Fraction;
            fractionCurrent_V1Local = DV.currentVol1Fraction/
totalVolLocal;
            targetFraction_v1 = config.v1/(config.v1+config.v2);

            % Normalize the sensitives.
            if (config.w1 ~= 1) % if we are using the heat objective
                temp1Max = max(max(abs(DV.sensitivityElastic)));
                DV.sensitivityElastic = DV.sensitivityElastic/
temp1Max;
                temp2Max = max(max(abs(DV.sensitivityHeat)));
                DV.sensitivityHeat = DV.sensitivityHeat/temp2Max;

                g1 = config.w1*DV.sensitivityElastic
+config.w2*DV.sensitivityHeat; % Calculate the weighted volume
 fraction change sensitivity.
            else
                g1 = config.w1*DV.sensitivityElastic;
            end

            % Filter the g1 sensitivies
            [g1]   = DV.check( config.nelx,
 config.nely,config.rmin,DV.x,g1);

            if(config.volFractionOptiizationMethod==1)
                G1 = g1 - DV.lambda1 +1/(DV.mu1)*( targetFraction_v1-
fractionCurrent_V1Local); % add in the lagrangian
                DV.w = DV.w+config.timestep*G1; % update the volume
 fraction.
                DV.w = max(min( DV.w,1),0);    % Don't allow the
 vol fraction to go above 1 or below 0
                DV.lambda1 =  DV.lambda1 -1/
(DV.mu1)*(targetFraction_v1-
fractionCurrent_V1Local)*config.volFractionDamping;
%                 DV.lambda1
            else
                largest=1e9;
                l1 = 0; l2 = largest;% move = 0.2;
                %             sumDensity =0;
                totalMaterial = sum(sum(DV.x));
                wProposed = DV.w;
                g1Max = max(max(g1));
                g1Min = min(min(g1));
                %                 if(g1Max>0)
                %                 g1=g1-g1Max;
                %                 end
                if(g1Min<0)
                    g1 = -g1Min+g1;
                end
                moveLimit = 0.1;

                targetRatioMethod =1;


                   targetRatio = config.v1/config.v2;

                while (l2-l1 > 1e-4)
                    lambda1 = 0.5*(l2+l1);
                    %
 wProposed=min(max(max(0,min(1,DV.w.*(sqrt(-g1Min+g1/lambda1)))),DV.w-
moveLimit),DV.w+moveLimit);
                    wProposed=min(max(max(0,min(1,DV.w.*(sqrt(g1/
lambda1)))),DV.w-moveLimit),DV.w+moveLimit);

                    %                   totalMat1
 =sum(sum( DV.x.*DV.w*matProp.E_material1));
                    %                 totalMat2 =sum(sum( DV.x.*(1-
DV.w)*matProp.E_material2));
                    % obj.actualAverageE=
 obj.currentVol1Fraction*matProp.E_material1+  obj.
 currentVol2Fraction*matProp.E_material2;
                    %                 obj.actualAverageE=
 (totalMat1+totalMat2)/totalMaterial;

                    %                 obj.   currentVol2Fraction
 =sum(sum( obj.x.*(1-obj.w)))/ne;
                    %                     fractionCurrent_V1Local =
 currentVol1Fraction/totalVolLocal;


                    if(1==0)
                         currentVol1Fraction
 =sum(sum( DV.x.*wProposed))/totalMaterial;
                        if(targetRatioMethod==1)
                             % ------------------------
                            % Target ratio v1/(v1+v2))
                            % ------------------------

                            if targetFraction_v1-
 currentVol1Fraction<0
                                l1 = lambda1;
                            else
                                l2 = lambda1;
                            end
                        elseif(  targetRatioMethod==2)
                             % ------------------------
                            % Target ratio v1/v2
                            % ------------------------

                            currentV1 = sum(sum( DV.x.*wProposed));
                            currentV2 = sum(sum((DV.x).*(1-
wProposed)));
                            currentRatio = currentV1/currentV2;
                             if 100*targetRatio- 100*currentRatio<0;
                                l1 = lambda1;
                                %                           l2 =
 lambda1;
                            else
                                l2 = lambda1;
                                %                           l1 =
 lambda1;
                             end

                        elseif(  targetRatioMethod==3)

                            % ------------------------
                            % Target v1 only
                            % ------------------------
%                               totalMaterial = sum(sum(DV.x));

                            currentV1 = sum(sum( DV.x.*wProposed));

                            v1RatioToTotal=currentV1/
(config.nelx*config.nely);

                            targetRatio=   config.v1;

                             if targetRatio- v1RatioToTotal<0;
                                l1 = lambda1;
                                %                           l2 =
 lambda1;
                            else
                                l2 = lambda1;
                                %                           l1 =
 lambda1;
                             end

                        end



                    else
                        % ------------------------
                        % Target an Elastic Modulus
                        % ------------------------
                        totalMat1
 =sum(sum( DV.x.*wProposed*matProp.E_material1));
                        totalMat2 =sum(sum( DV.x.*(1-
wProposed)*matProp.E_material2));
                        % obj.actualAverageE=
 obj.currentVol1Fraction*matProp.E_material1+  obj.
 currentVol2Fraction*matProp.E_material2;
                        averageElasticLocal= (totalMat1+totalMat2)/
totalMaterial;
```

```
                       %               averageElasticLocal =
(sum(sum(EyyNew.*Xtemp))+sum(sum(ExxNew.*Xtemp)))/neSolid;
                       %
averageElasticLocal=averageElasticLocal/2; % Becuse Eyy and Exx are
from one element, so to get the average divide by 2
                       E_target=config.targetAvgExxEyy;
                       if E_target- averageElasticLocal<0
                           l1 = lambda1;
                       else
                           l2 = lambda1;
                       end
                   end
               end
               DV.w=wProposed;
           end
       end

       % ----------------------------------
       % ORTHO DISTRIBUTION OPTIMIZATION
       % ----------------------------------
       %         function [] =
OptimizeOrthoDistribution(obj,DV,config, matProp, masterloop)
       %             DV =
DV.CalculateOthogonalDistributionSensitivity(config, matProp,
masterloop);
       %             DV.sensitivityElastic = check( config.nelx,
config.nely,config.rmin,DV.x,DV.sensitivityElastic);
       %             % move= 0.1* 20/(20+masterloop);
       %             move = config.orthDistMoveLimit;
       %             config.orthDistMoveLimit=
config.orthDistMoveLimit* 10/(10+masterloop);
       %             %-----------------------
       %             %
       %             % Update design var.
       %             %-----------------------
       %             for ely = 1:config.nely
       %                 for elx = 1:config.nelx
       %                     if(DV.sensitivityElastic(ely,elx)<0.05)
       %                         DV.d(ely,elx) =  max(
 DV.d(ely,elx)-move,config.minDorth);
       %                     end
       %
       %                     if(DV.sensitivityElastic(ely,elx)>0.05)
       %                         DV.d(ely,elx) =  min(
 DV.d(ely,elx)+ move,config.maxDorth);
       %                     end
       %
       %                 end
       %             end
       %         end

       % ----------------------------------
       % ROTATION OPTIMIZATION
       % ----------------------------------
       function DV = OptimizeRotation(obj,DV,config, matProp,
masterloop)
           %                 move= 0.1* 20/(20+masterloop);
           % allow multiple loading cases.
           [~, t2] = size(config.loadingCase);

           epsilon = pi/180; % 1 DEGREES ACCURACY
           elementsInRow = config.nelx+1;

           for ely = 1:config.nely
               rowMultiplier = ely-1;
               for elx = 1:config.nelx
                   rhoSIMP =  DV.x(ely,elx);
                   if(rhoSIMP>config.noNewMesoDesignDensityCutOff)

                       % -------------------
                       % STEP 1, GET THE DISPLACEMENT FOR THIS NODE
                       % -------------------
                       nodes1=[rowMultiplier*elementsInRow+elx;
                           rowMultiplier*elementsInRow+elx+1;
                           (rowMultiplier +1)*elementsInRow+elx+1;
                           (rowMultiplier +1)*elementsInRow+elx];

                       xNodes = nodes1*2-1;
                       yNodes = nodes1*2;
                       NodeNumbers = [xNodes(1) yNodes(1) xNodes(2)
yNodes(2) xNodes(3) yNodes(3) xNodes(4) yNodes(4)];
                       UallCaseForElement = DV.U(1:t2,NodeNumbers);
                       U = UallCaseForElement;

                       % -------------------
                       % STEP 2, SET UP GOLDEN RATIO METHOD TO FIND
                       % OPTIMAL THETA FOR ROTATION
                       % -------------------

                       n = 0;
                       x0 = config.minRotation; %lower_bracket;
                       x3 = config.maxRotation;% higher_bracket;
                       leng = x3-x0;
                       grleng = leng*config.gr ; % golden ratio lenth
                       x1 = x3 - grleng;
                       x2 = x0 + grleng;
                       rhoSIMP =  DV.x(ely,elx);
                       mat1Frac  =[];% DV.w(ely,elx);
                       Exx = DV.Exx(ely,elx);
                       Eyy = DV.Eyy(ely,elx);

                       thetaSubSystem = DV.thetaSub(ely,elx);
                       penaltyValue=DV.penaltyTheta(ely,elx);
                       lagraMultiplier=DV.lambdaTheta(ely,elx);

                       %                          orthD =
DV.d(ely,elx);
```

```
                       %fx1 = obj.EvaluteARotation(U,rhoSIMP,
mat1Frac,Exx,Eyy,x1,thetaSubSystem,penaltyValue,lagraMultiplier,matProp,
config,DV.maxElemStraniEnergy);


                       fx1= obj.EvaluteARotation(U,rhoSIMP,
mat1Frac,Exx,Eyy,x1,thetaSubSystem,penaltyValue,lagraMultiplier,matProp,
config,DV.maxElemStraniEnergy);
                       fx2 = obj.EvaluteARotation(U,rhoSIMP,
mat1Frac,Exx,Eyy,x2,thetaSubSystem,penaltyValue,lagraMultiplier,matProp,
config,DV.maxElemStraniEnergy);

                       %                         if(masterloop>5)
                       %                             debug = 1;
                       %                         else
                       debug=0;
                       %                         end
                       verbosity = 0;

                       if(   debug == 1)
                           xtemp = x0:pi/180:x3;
                           ytemp = zeros(1, size(xtemp,2));
                           count = 1;
                           for thetaTemp = xtemp
                               ytemp(count)=
obj.EvaluteARotation(U,rhoSIMP,
mat1Frac,Exx,Eyy,thetaTemp,thetaSubSystem,penaltyValue,lagraMultiplier,matProp,
config,DV.maxElemStraniEnergy);
                               count = count+1;
                           end
                           figure(2)
                           subSysXvalus = [x0 DV.thetaSub(ely,elx)
x3];
                           subSysYvalus = [min(ytemp) max(ytemp)
max(ytemp)];
                           plot(xtemp,ytemp);
                           hold on
                           stairs(subSysXvalus,subSysYvalus)
                           hold off
                           title(sprintf('Lagrangian Function for
Element x = %i, y = %i',elx,ely));
                           nothin = 1;
                       end


                       while(1 == 1)
                           if(debug == 1 && verbosity ==1)
                               str = sprintf('loop# = %d, x0 = %f,
x1 = %f, x2 = %f, x3 = %f, fx1 = %f, fx2 = %f\n', n, x0, x1, x2, x3,
fx1, fx2); display(str);
                           end

                           if(fx1<=fx2) % less than or equal
                               % x0 = x0; % x0 stays the same
                               x3 = x2; % the old x2 is now x3
                               x2 = x1; % the old x1 is now x2
                               fx2 = fx1;
                               leng = x3 - x0; % find the length of
the interval
                               x1 = x3 - leng*config.gr; % find
golden ratio of length, subtract it from the x3 value
                               fx1 = obj.EvaluteARotation(U,rhoSIMP,
mat1Frac,Exx,Eyy,x1,thetaSubSystem,penaltyValue,lagraMultiplier,matProp,
config,DV.maxElemStraniEnergy);% calculate the fx

                           elseif(fx1>fx2) % greater than
                               x0 = x1; % the old x1 is now x0
                               x1 = x2; % the old x2 is now the new
x1
                               fx1 = fx2;
                               % x3 = x3; % x3 stays the same.

                               leng = (x3 - x0); % find the length of
the interval
                               x2 = x0 + leng*config.gr; % find
golden ratio of length, subtract it from the x3 value
                               fx2 = obj.EvaluteARotation(U,rhoSIMP,
mat1Frac,Exx,Eyy,x2,thetaSubSystem,penaltyValue,lagraMultiplier,matProp,
config,DV.maxElemStraniEnergy);  % calculate the fx
                           end

                           % check to see if we are as close as we
want
                           if(leng < epsilon || n>100)
                               break;
                           end
                           n = n +1; % increment

                       end

                       % -------------------
                       % STEP 3, RECORD THE OPTIMAL THETA
                       % -------------------
                       minTvalue = (x2 + x3)/2;
                       moveLimit = config.rotationMoveLimit;

                       % max move limit =  half the diff to optimal
                       diffT = abs(minTvalue-DV.t(ely,elx));
                       moveLimit=min(moveLimit,diffT*0.1);

                       tOld = DV.t(ely,elx);
                       if(minTvalue>DV.t(ely,elx)+moveLimit)
                           DV.t(ely,elx)= DV.t(ely,elx)+moveLimit;
                       elseif(minTvalue<DV.t(ely,elx)-moveLimit)
                           DV.t(ely,elx)= DV.t(ely,elx)-moveLimit;
                       else
                           DV.t(ely,elx)=minTvalue;
                       end
```

```
                        % Damp the changes
                        if(tOld>0 &&   DV.t(ely,elx) >0)
                            DV.t(ely,elx) = tOld*sqrt( DV.t(ely,elx)/
tOld);
                        end
                    end
                end
            end
        end

        % --------------------------
        % EVALUTE THE OBJECTIVE FUNCTION FOR A ROTATION
        %---------------------------
        function lagrangianValue = EvaluteARotation(~,U,topDensity,
 material1Fraction,Exx,Eyy,thetaSys,thetaSubSystem,penaltyValue,lagraMultiplier,ma
 config,maxElemStraniEnergy)
            K =
 matProp.getKMatrixTopExxYyyRotVars(config,topDensity,Exx,
 Eyy,thetaSys,material1Fraction, 1, 1);
            % LOOP OVER LOADING CASES.
            % U'S ROWS ARE UNIQUE LOADING CASES
            % EACH ROW CONTAINS 8 VALUES FOR THE 8 DOF OF THE ELEMENT
            % allow multiple loading cases.
            [~, t2] = size(config.loadingCase);
            term1=0;
            for i = 1:t2
                Ucase = U(i,:)';
                term1= term1+Ucase'*K*Ucase;
            end
            term1=-term1;
            %             term1=-term1/maxElemStraniEnergy;

            %             term2 = penaltyValue/2*(thetaSys-
thetaSubSystem)^2;
            %             term2 = penaltyValue*(thetaSys-
thetaSubSystem)^2;
            term2 = penaltyValue*abs(thetaSys-thetaSubSystem);
            %             term3 = lagraMultiplier*(thetaSys-
thetaSubSystem);
            %             lagrangianValue=term1+term2+term3;
            %             normalizer=penaltyValue/2*(pi/4)^2;
            %                term2=term2/normalizer;

            %                lagrangianValue=term1+term2+term3;
            lagrangianValue=term1+term2;


        end


        % ---------------------------------
        % E_xx and E_yy  OPTIMIZATION
        % ---------------------------------
        function [DV] = OptimizeExxEyy(obj,DV,config, matProp,
 masterloop)
            %             if(config.useTargetMesoDensity==1)
            DV= OptimizeExxEyy_V3(obj,DV,config, matProp, masterloop);
            %             else
            %                 DV= OptimizeExxEyy_V2(obj,DV,config,
 matProp, masterloop);
            %             end
        end


        % ----------------------------------
        % E_xx and E_yy  OPTIMIZATION
        %
        %   Version 3
        % TARGET AVG MESO DENSITY AS CONSTRAINT.
        % ----------------------------------
        function [DV] = OptimizeExxEyy_V3(obj,DV,config, matProp,
 masterloop)
            DV = DV.CalculateExxEyySensitivity(config, matProp,
 masterloop);
            DV.sensitivityElastic = DV.check( config.nelx,
 config.nely,config.rminExxEyy,DV.x,DV.sensitivityElastic);
            DV.sensitivityElasticPart2 = DV.check( config.nelx,
 config.nely,config.rminExxEyy,DV.x,DV.sensitivityElasticPart2);


            testingIsoTropicRecution=1;
            if(testingIsoTropicRecution==1)
                combinedSensitivity = DV.sensitivityElastic+
 DV.sensitivityElasticPart2;
                 DV.sensitivityElasticPart2=combinedSensitivity;
                 DV.sensitivityElastic=combinedSensitivity;
            end

            % if(config.macro_meso_iteration>=2 &&
 mod(masterloop,3)==1)
            if(config.macro_meso_iteration>=2 )
                deltaT=0.2;
                diffExx = DV.ExxSub-DV.Exx;
                diffEyy = DV.EyySub-DV.Eyy  ;
                %
                DV.lambdaExx=max( min(DV.lambdaExx+deltaT
 *diffExx,matProp.E_material1),-matProp.E_material1);
                DV.lambdaEyy= max( min( DV.lambdaEyy
+deltaT*diffEyy,matProp.E_material1),-matProp.E_material1);
                %  DV.lambdaExx=DV.lambdaExx+deltaT *diffExx;
                %   DV.lambdaEyy= DV.lambdaEyy+deltaT*diffEyy;
                disp('Updated Lambda Values Exx Eyy')
            end


            %-----------------------
            %
```

```
           % Update design var.
           %-----------------------
           largest=1e9;

           move = matProp.E_material1*0.05;
           minimum =config.minEallowed;

           % ----------------
           % Exx
           % ----------------
           ExxNew = DV.Exx;
           EyyNew = DV.Eyy;

           totalMaterial = sum(sum(DV.x));

           term1Exx = DV.sensitivityElastic;
           term1Eyy= DV.sensitivityElasticPart2;

           smallestLambdExx = min(min(DV.lambdaExx));
           smallestLambdEyy = min(min(DV.lambdaEyy));
           smallestOfTwo = min(smallestLambdExx,smallestLambdEyy)-1;

           term2Exx =( DV.lambdaExx-smallestOfTwo).*DV.penaltyExx;
           term2Eyy = (DV.lambdaEyy-smallestOfTwo).*DV.penaltyEyy;

           w1 = 1;
           w2 = 0;
           %    if( config.macro_meso_iteration>=2 ) % Weight toward
satifying consistency constraint.
           %    w2=min( ( config.macro_meso_iteration-2)*0.2,1); %
staring iteration 3, start relaxing the meso density constraint.
           %      w1 = 1-w2;
           %     end

           theta = DV.t;

           % ----------------------------------------------------
           %
           % TARGET AVG MESO DENSITY AS CONSTRAINT.
           % Update 3 (idea)
           %
           % ----------------------------------------------------
           l1 = 0; l2 = largest;% move = 0.2;
           sumDensity =0;
           while (l2-l1 > 1e-5)
               lambda1 = 0.5*(l2+l1);

               if(config.useTargetMesoDensity==1)
                   ExxInput =ExxNew/matProp.E_material1; % % MOVED to
the function scale down by the simp density, since the actual rho is
a function of what is SIMP density and Exx or Eyy
                   EyyInput = EyyNew/matProp.E_material1;
                   [dDensityEyy, dDensityExx,~] =
obj.CalculateDensitySensitivityandRho(ExxInput,EyyInput,theta,DV.x,DV.ResponseSur
                    dDensityEyy = DV.check( config.nelx,
 config.nely,config.rminExxEyy,DV.x,dDensityEyy);
                    dDensityExx = DV.check( config.nelx,
 config.nely,config.rminExxEyy,DV.x,dDensityExx);
                else
                    dDensityExx=ones(size(term1Exx));
                    dDensityEyy=ones(size(term1Exx));
                end




                combinedTermsExx=(term1Exx+term2Exx)./
(lambda1*dDensityExx);
                combinedTermsEyy=(term1Eyy+term2Eyy)./
(lambda1*dDensityEyy);

                %                 targetExx =
 ExxNew.*combinedTermsExx;
                %                 targetEyy =
 EyyNew.*combinedTermsEyy;
                targetExx = DV.Exx.*combinedTermsExx;
                targetEyy = DV.Eyy.*combinedTermsEyy;
                ExxNew = max(0.1,max( minimum - EyyNew,  max(DV.Exx-
move ,  min(  min(targetExx,DV.Exx+move ),matProp.E_material1))));
                EyyNew = max(0.1,max(minimum -  ExxNew,  max(DV.Eyy-
move ,  min(  min(targetEyy,DV.Eyy+move ),matProp.E_material1))));
                %                  logicTest1 =
 mesoDensity<config.minMesoDensityInOptimizer;
                %                   logicTest2
 =DV.x>config.voidMaterialDensityCutOff;
                %
 logicTest=(logicTest1+logicTest2)>1.1;
                %                  minE_allowed =
 ones(size(targetExx));
                %
 minE_allowed(logicTest)=ExxNew(logicTest);
                %                  ExxNew =
 max(minE_allowed,max( minimum - EyyNew,  max(DV.Exx-move ,  min(
  min(targetExx,DV.Exx+move ),matProp.E_material1))));
                %
                %
 minE_allowed(logicTest)=EyyNew(logicTest);
                %                 EyyNew =
 max(minE_allowed,max(minimum -  ExxNew,  max(DV.Eyy-move ,  min(
  min(targetEyy,DV.Eyy+move ),matProp.E_material1))));

                %                 sumDensity = sumDensity/
(config.nelx*config.nely*config.totalVolume);
                ExxSysAndSubDiffSummed=sum(sum(abs(DV.x.*(ExxNew-
DV.ExxSub)))); %
                EyySysAndSubDiffSummed=sum(sum(abs(DV.x.*(EyyNew-
DV.EyySub))));%
```

```
                ConsistConstraintMag = ExxSysAndSubDiffSummed
+EyySysAndSubDiffSummed;
                %                 ConsistConstraintMag=-
ConsistConstraintMag;
                ConsistConstraintMag = ConsistConstraintMag/
(matProp.E_material1*totalMaterial);

                if(config.useTargetMesoDensity==1)
                    [~, ~,rhoValue] =
 obj.CalculateDensitySensitivityandRho(ExxNew/
matProp.E_material1,EyyNew/
matProp.E_material1,theta,DV.x,DV.ResponseSurfaceCoefficents,config,matProp,DV.den
                    rhoValue=max(0,min(rhoValue,1));
                    temp2 = sum(sum(rhoValue));
                    sumDensity=temp2/
(config.nelx*config.nely*config.totalVolume);

                    % Determine if the consistency constraint is being
 under
                    % valued


                    terms= w1*(config.targetExxEyyDensity-
 sumDensity)+w2*(ConsistConstraintMag);
                    if (terms)<0
                        l1 = lambda1;
                    else
                        l2 = lambda1;
                    end

                else
                    totalExx =DV.x.*ExxNew;
                    totalEyy = DV.x.* EyyNew;
                    avgE = (totalExx+totalEyy)/2;
                    averageElasticLocal= sum(sum(avgE))/totalMaterial;
                    %               averageElasticLocal =
 (sum(sum(EyyNew.*Xtemp))+sum(sum(ExxNew.*Xtemp)))/neSolid;
                    %
 averageElasticLocal=averageElasticLocal/2; % Becuse Eyy and Exx are
 from one element, so to get the average divide by 2
                    E_target=config.targetAvgExxEyy;
                    if E_target- averageElasticLocal<0;
                        l1 = lambda1;
                    else
                        l2 = lambda1;
                    end
                end
            end
            multiplier= 10000;
            text1 =    sprintf('\nExxNew\t\t\t\t\t[1,1   5,1
 5,5],  %f ,%f %f',ExxNew(1,1) ,ExxNew(5,1),ExxNew(5,5));
            text2 =    sprintf('ExxSub\t\t\t\t\t[1,1   5,1      5,5],
 %f ,%f %f',DV.ExxSub(1,1),DV.ExxSub(5,1),DV.ExxSub(5,5));
```

```
            text3=    sprintf('Exx\\t\t\t\t\t\t[1,1   5,1      5,5],
 %f ,%f %f',DV.Exx(1,1),DV.Exx(5,1),DV.Exx(5,5));
            text4 =    sprintf('DiffXXNew \t\t\t\t[1,1   5,1
   5,5],  %f ,%f %f',ExxNew(1,1) - DV.ExxSub(1,1),ExxNew(5,1) -
 DV.ExxSub(5,1),ExxNew(5,5) - DV.ExxSub(5,5));
            text5 =    sprintf('combinedTermsExx\t\t[1,1   5,1
 5,5],  %f ,%f
 %f',combinedTermsExx(1,1),combinedTermsExx(5,1),combinedTermsExx(5,5));
            text6 =    sprintf('penaltyExx*10000\t\t[1,1   5,1
 5,5],  %f ,%f
 %f',DV.penaltyExx(1,1)*multiplier,DV.penaltyExx(5,1)*multiplier,DV.penaltyExx(5,5
            text7 =    sprintf('lambdaExx\t\t\t\t[1,1   5,1      5,5],
  %f ,%f %f',DV.lambdaExx(1,1),DV.lambdaExx(5,1),DV.lambdaExx(5,5));
            text75 =    sprintf('dDensityExx
\t\t\t\t[1,1   5,1      5,5],  %f ,%f
 %f',dDensityExx(1,1),dDensityExx(5,1),dDensityExx(5,5));
            text8 =    sprintf('lambda1 and density\t\t[  %f ,
%f',lambda1,sumDensity);
            text9 =    sprintf('ConsistConstraintMag and w2 \t\t[
  %f ,%f\n',ConsistConstraintMag,w2);

            disp(text1)
            disp(text2)
            disp(text3)
            disp(text4)
            disp(text5)
            disp(text6)
            disp(text7)
            disp(text75)
            disp(text8)
            disp(text9)


            debug = 0;
            if(debug ==1)
                figure(2)
                p = plotResults;
                xplots=3;
                yplots =3;
                plotNum=1;

                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(dDensityEyy,'dDensityEyy');
                plotNum=plotNum+1;

                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(dDensityExx,'dDensityExx');
                plotNum=plotNum+1;

                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(numeratorExx,'completeExx');
                plotNum=plotNum+1;

                subplot(xplots,yplots,plotNum);
```

```
        p. PlotArrayGeneric(numeratorEyy,'completeEyy');
        plotNum=plotNum+1;

        subplot(xplots,yplots,plotNum);
        p. PlotArrayGeneric(combinedTermsExx/
lmid,'combinedTermsExx/lmid');
        plotNum=plotNum+1;

        subplot(xplots,yplots,plotNum);
        p. PlotArrayGeneric(combinedTermsEyy/
lmid,'combinedTermsEyy/lmid');
        plotNum=plotNum+1;

        subplot(xplots,yplots,plotNum);
        p. PlotArrayGeneric(DV.t-DV.thetaSub,'theta diff');
        plotNum=plotNum+1;

        subplot(xplots,yplots,plotNum);
        p. PlotArrayGeneric(DV.Eyy - DV.EyySub,'Eyy diff');
        plotNum=plotNum+1;

        subplot(xplots,yplots,plotNum);
        p. PlotArrayGeneric(DV.Exx-DV.ExxSub,'Exx diff');
        plotNum=plotNum+1;
      end

      % -----------------------
      % Set the valeus.
      % -----------------------
      DV.Exx =DV.Exx.*sqrt( ExxNew./  DV.Exx);
      DV.Eyy = DV.Eyy.*sqrt( EyyNew./  DV.Eyy );

       if(testingIsoTropicRecution==1)
            E_combined =( DV.Exx+DV.Eyy)./2;
              DV.Exx =E_combined;
              DV.Eyy =E_combined;
       end


    end


    function [DV] =FindStartingExxEyy_V3(obj,DV,config, matProp,
 masterloop)
      % --------------------
      %
      %   SCALE starting Exx Eyy values
      %
      % --------------------
      if(config.macro_meso_iteration==1 )
          if (49<config.mode && config.mode <100  )
              l1 = 0; l2 = 100000;% move = 0.2;
              %              sumDensity =0;
              o=Optimizer;
```

```
                    if(config.useTargetMesoDensity==1)
                        target=config.targetExxEyyDensity;
                        theta=DV.t;
                    else
                        target=config.targetAvgExxEyy;
                        totalMaterial= sum(sum(DV.x));
                    end

                    fprintf('try scaling the starting values\n');

                    while (l2-l1 > 1e-6)
                        lambda1 = 0.5*(l2+l1);
                        ExxNew=DV.Exx*lambda1;
                        EyyNew=DV.Eyy*lambda1;

                        if(config.useTargetMesoDensity==1)
                            [~, ~,rhoValue] =
 o.CalculateDensitySensitivityandRho(ExxNew/
matProp.E_material1,EyyNew/
matProp.E_material1,theta,DV.x,DV.ResponseSurfaceCoefficents,config,matProp,0);
                            rhoValue=max(0,min(rhoValue,1));
                            temp2 = sum(sum(rhoValue));
                            sumDensity=temp2/
(config.nelx*config.nely*config.totalVolume);
                            currentValue=sumDensity;
                        else


                            totalExx =DV.x.*ExxNew;
                            totalEyy = DV.x.* EyyNew;
                            avgE = (totalExx+totalEyy)/2;
                            averageElasticLocal= sum(sum(avgE))/
totalMaterial;

                            currentValue=averageElasticLocal;
                        end


                        fprintf('Target %f and current %f
\n',target,currentValue);
                        if target- currentValue<0;
                            l2 = lambda1;
                        else
                            l1 = lambda1;
                        end
                    end

                    DV.Exx=    DV.Exx*lambda1;
                    DV.Eyy=      DV.Eyy*lambda1;

                    fprintf('Final Lambda = %f with final value of %f
\n',lambda1,currentValue);
                end
            end
```

```
       end

       % -------------------------------------
       % Calculate the density and sensitivity of the Exx,Eyy,theta
       % values.
       %
       % USe a response surface, ANN, or interpolation depending on
       % settings.
       % -------------------------------------
       function [EyySensitivty, ExxSensitivity,rhoValue] =
CalculateDensitySensitivityandRho(obj,Exx,Eyy,theta,xSimp,Coefficents,config,matP

           [EyySensitivty, ExxSensitivity,rhoValue] =
CalculateDensitySensitivityandRho_OLD(obj,Exx,Eyy,theta,xSimp,Coefficents,config,
           %
rhoValue=max(config.MesoMinimumDensity,min(rhoValue,1));
           rhoValue=rhoValue+OffSet;
       end

       function [EyySensitivty, ExxSensitivity,rhoValue] =
CalculateDensitySensitivityandRho_OLD(obj,Exx,Eyy,theta,xSimp,
Coefficents,config,matProp)
           co = Coefficents;



           Exx=Exx.*(xSimp.^config.penal);
           Eyy=Eyy.*(xSimp.^config.penal);

           ExxOriginal = Exx;
           EyyOriginal= Eyy;
           thetaOriginal = theta;

           if(config.useANN==1)

               % Make the inputs be so taht Exx > Eyy
               % Rather than a strict theta, use the distance from
pi/4, since the problem
               % is symmetric arround pi/4

               temp = Exx;
               logic = Eyy>Exx;
               Exx(logic)=Eyy(logic);
               Eyy(logic) =temp(logic);
               %
               % min(thetaArray)
               % max(thetaArray)
               % thetaArray=((pi/4)^2+thetaArray.^2).^(1/2);
               %                 temp2 = theta;
               logic1 = theta<0;
               theta(logic1) = -theta(logic1);
               logic2 = theta>pi/4;
               logic3 = theta<pi/4;
```

```
                theta(logic2)=theta(logic2)-pi/4;
                theta(logic3)=pi/4-theta(logic3);


                Exx=Exx*matProp.E_material1;
                Eyy=Eyy*matProp.E_material1;

                [t1,t2]=size(Exx);

                Exx=reshape(Exx,1,[]);
                Eyy=reshape(Eyy,1,[]);
                theta=reshape(theta,1,[]);

                X=[Exx;Eyy;theta];

                if(config.UseLookUpTableForPsuedoStrain==1)
                    if config.mesoDesignInitalConditions==3
                        %                         [rhoValue,~,~] =
 annOutput_LookUpTable(X,[],[]);
                        [rhoValue,~,~] =
 annOutput_lookupTable_withFmincon(X,[],[]);
                    elseif(config.mesoDesignInitalConditions==1)
                        [rhoValue,~,~] =
 annOutput_RandomMesoInitialLookUpTable(X,[],[]);
                    end
                else
                    if(config.mesoVolumeUpdateMethod==2)
                        [rhoValue,~,~] = annOutput_matUpdateV2(X,[],
[]);
                    else
                        [rhoValue,~,~] = annOutput_matUpdateV1(X,[],
[]);
                    end
                end

                deltaT=1;
                XCopy = X;
                XCopy(1,:)=XCopy(1,:)+deltaT;

                if(config.UseLookUpTableForPsuedoStrain==1)
                    if config.mesoDesignInitalConditions==3
                        %                         [rhoValue,~,~] =
 annOutput_LookUpTable(XCopy,[],[]);
                        [rhoValue,~,~] =
 annOutput_lookupTable_withFmincon(XCopy,[],[]);
                    elseif(config.mesoDesignInitalConditions==1)
                        [rhoValue,~,~] =
 annOutput_RandomMesoInitialLookUpTable(XCopy,[],[]);
                    end
                else
                    if(config.mesoVolumeUpdateMethod==2)
                        [rhoValueXShift,~,~] =
 annOutput_matUpdateV2(XCopy,[],[]);
                    else
```

```
                       [rhoValueXShift,~,~] =
annOutput_matUpdateV1(XCopy,[],[]);

                   end
               end

               ExxSensitivity=(rhoValueXShift-rhoValue)/deltaT;

               XCopy = X;
               XCopy(2,:)=XCopy(2,:)+deltaT;

               if(config.UseLookUpTableForPsuedoStrain==1)
                   if config.mesoDesignInitalConditions==3
                       %                          [rhoValue,~,~] =
annOutput_LookUpTable(XCopy,[],[]);
                       [rhoValue,~,~] =
annOutput_lookupTable_withFmincon(XCopy,[],[]);
                   elseif(config.mesoDesignInitalConditions==1)
                       [rhoValue,~,~] =
annOutput_RandomMesoInitialLookUpTable(XCopy,[],[]);
                   end
               else
                   if(config.mesoVolumeUpdateMethod==2)
                       [rhoValueYShift,~,~] =
annOutput_matUpdateV2(XCopy,[],[]);
                   else
                       [rhoValueYShift,~,~] =
annOutput_matUpdateV1(XCopy,[],[]);
                   end
               end

               EyySensitivty=(rhoValueYShift-rhoValue)/deltaT;

               % REshape to the orginal shape
               rhoValue=reshape(rhoValue,t1,t2);
               scaleUpValue=1000;

ExxSensitivity=reshape(ExxSensitivity,t1,t2)*scaleUpValue;

EyySensitivty=reshape(EyySensitivty,t1,t2)*scaleUpValue;

               rhoValue(rhoValue>1)=1;
               rhoValue(rhoValue<0)=0;

               %
ExxSensitivity(ExxSensitivity<0)=0.000001;
               %
EyySensitivty(EyySensitivty<0)=0.000001;

           else
               if(config.useThetaInSurfaceFit==1)

                   % make it so that Exx is always larger
                   temp1=Eyy;
```

```
                    valueConditionTrue = Eyy>Exx;
                    Eyy(valueConditionTrue)=Exx(valueConditionTrue);
                    Exx(valueConditionTrue)=temp1(valueConditionTrue);
                    %                 if(Eyy>Exx)
                    %                     Exx=Eyy;
                    %                     Eyy=Exx;
                    %                 end
                    % rhoValue= x(1)  + x(2)* exp(E_xx)  + x(3)*
 exp(E_yy)+x(4) *exp(theta) +x(5)*E_xx  + x(6)* E_yy +x(7)*theta+
 x(8)*E_xx.*E_yy;
                    % ExxSensitivity=  x(2)* exp(E_xx)  +x(5) +
 x(8)*E_yy;
                    % EyySensitivty= x(3)* exp(E_yy) + x(6)+
 x(8)*E_xx;
                    rhoValue= co(1)+co(2)*Exx+co(3)*Eyy+co(4)*theta
+co(5)*Exx.^2+co(6)* Eyy.^2+co(7)*theta.^2+co(8)*Exx.*Eyy
+co(9)*Eyy.*theta+co(10)*Exx.*theta;
                    ExxSensitivity =co(2)+2*co(5)*Exx+co(8)*Eyy
+co(10)*theta;
                    EyySensitivty = co(3)+2*co(6)* Eyy+co(8)*Exx
+co(9)*theta;

                    % Scale Up
                    %                 rhoValue=rhoValue*scaleUp;
                    %
 ExxSensitivity=ExxSensitivity*scaleUp;
                    %
 EyySensitivty=EyySensitivty*scaleUp;
                    %                 rhoValue(rhoValue>1)=1;
                    %                 rhoValue(rhoValue<0)=1;
                else
                    % obj.
 ResponseSurfaceCoefficents=[ 1.0000000000463e-05 9.99988184437107e-06
 9.9998491550433e-06 -3.40115537230351e-11 -5.52110060132392e-12
 -3.81038581303971e-11];
                    if(config.useAnnForDensityNotDerivative==1)
                        minAllowed = 0.01;
                        x = ExxOriginal;
                        y = EyyOriginal;
                        %                     rhoValue= co(1)  +
 co(2) *x +  co(3) *y + co(4)*x^2 + co(5)*x*y + co(6)*y^2 + co(7)*x^3
 + co(8)*x^2*y + co(9)*x*y^2 + co(10)*y^3;

                        EyySensitivty= max(  co(3) *1 + co(5)*x*1 +
 2*co(6)*y  + co(8)*x.^2*1 + 2*co(9)*x.*y +3* co(10)*y.^2,minAllowed);
                        ExxSensitivity=max( co(2) *1 +  2* co(4)*x +
 co(5)*1*y + 3*co(7)*x.^2 + 2*co(8)*x.*y + co(9)*1*y.^2 ,minAllowed);



                        temp = Exx;
                        logic = Eyy>Exx;
                        Exx(logic)=Eyy(logic);
                        Eyy(logic) =temp(logic);
```

```
                       %
                       % min(thetaArray)
                       % max(thetaArray)
                       % thetaArray=((pi/4)^2+thetaArray.^2).^(1/2);
                       %                 temp2 = theta;
                       logic1 = theta<0;
                       theta(logic1) = -theta(logic1);
                       logic2 = theta>pi/4;
                       logic3 = theta<pi/4;
                       theta(logic2)=theta(logic2)-pi/4;
                       theta(logic3)=pi/4-theta(logic3);


                       Exx=Exx*matProp.E_material1;
                       Eyy=Eyy*matProp.E_material1;

                       [t1,t2]=size(Exx);

                       Exx=reshape(Exx,1,[]);
                       Eyy=reshape(Eyy,1,[]);
                       theta=reshape(theta,1,[]);

                       X=[Exx;Eyy;theta];

                       if(config.UseLookUpTableForPsuedoStrain==1)
                           if config.mesoDesignInitalConditions==3
                               %
[rhoValue,~,~] = annOutput_LookUpTable(X,[],[]);
                               [rhoValue,~,~] =
annOutput_lookupTable_withFmincon(X,[],[]);

 elseif(config.mesoDesignInitalConditions==1)
                               [rhoValue,~,~] =
annOutput_RandomMesoInitialLookUpTable(X,[],[]);
                           end
                       else
                           if(config.mesoVolumeUpdateMethod==2)
                               [rhoValue,~,~] =
annOutput_matUpdateV2(X,[],[]);
                           else
                               [rhoValue,~,~] =
annOutput_matUpdateV1(X,[],[]);
                           end
                       end
                       rhoValue=reshape(rhoValue,t1,t2);


                       return
                   end
                   minAllowed = 0.01;
                   % funciton from the values that I came up with as
my
                   % first esimate of best fit.
                   %
 EyySensitivty=max(co(3)+co(5).*Exx+2*co(6).*Eyy,minAllowed);
                   %                     ExxSensitivity=max(co(2)+
 2*co(4).*Exx+co(5).*Eyy,minAllowed);
                   %                     rhoValue=   co(1) +
 co(2)*Exx + co(3)*Eyy + co(4)*Exx.^2 + co(5)*Exx.*Eyy + co(6)*Eyy.^2;

                   x = ExxOriginal;
                   y = EyyOriginal;
                   rhoValue= co(1)  +  co(2) *x +  co(3) *y +
 co(4)*x^2 + co(5)*x*y + co(6)*y^2 + co(7)*x^3 + co(8)*x^2*y +
 co(9)*x*y^2 + co(10)*y^3;

                   EyySensitivty= max(  co(3) *1 + co(5)*x*1 +
 2*co(6)*y  + co(8)*x^2*1 + 2*co(9)*x*y +3* co(10)*y^2,minAllowed);
                   ExxSensitivity=max( co(2) *1 +  2* co(4)*x +
 co(5)*1*y + 3*co(7)*x^2 + 2*co(8)*x*y + co(9)*1*y^2 ,minAllowed);
               end
           end

       end

       %         function
 [obj]=GenerateInterpolateANN(obj,Coefficents,config,matProp)
       %             if(config.useTargetMesoDensity==1)
       %
       %
       %                 outname = sprintf('./out%i/
ANN_interp_E_xx.csv',0);
       %                 obj.ExxInterp=csvread(outname);
       %                 outname = sprintf('./out%i/
ANN_interp_E_yy.csv',0);
       %                 obj.EyyInterp=csvread(outname);
       %                 outname = sprintf('./out%i/
ANN_interp_Theta.csv',0);
       %                 obj.thetaInterp=csvread(outname);
       %                 outname = sprintf('./out%i/
ANN_interp_Rho.csv',0);
       %                 obj.rhoInterp=csvread(outname);
       %
       %
 obj.ExxInterp=reshape(obj.ExxInterp,21,21,21);
       %
 obj.EyyInterp=reshape(obj.EyyInterp,21,21,21);
       %
 obj.thetaInterp=reshape(obj.thetaInterp,21,21,21);
       %
 obj.rhoInterp=reshape(obj.rhoInterp,21,21,21);
       %                 %                valuesPerDir=15;
       %                 %                ExxRange = 0:
(matProp.E_material1)/valuesPerDir:matProp.E_material1;
       %                 %
 EyyRange=0:matProp.E_material1/valuesPerDir:matProp.E_material1;
```

```
        %                 %                 thetaRange = 0:(pi/2)/
valuesPerDir:pi/2;
        %                 %
        %                 %                 [Exx,Eyy,theta] =
 ndgrid(ExxRange,EyyRange,thetaRange);
        %                 %                 % Needs to be reshaped
        %                 %                 [~, ~,rhoValue]=
 CalculateDensitySensitivityandRho_OLD(obj,Exx,Eyy,theta,Coefficents,config,matPro
        %                 %
        %                 %                 obj.ExxInterp=Exx;
        %                 %                 obj.EyyInterp=Eyy;
        %                 %                 obj. thetaInterp=theta;
        %                 %                 obj.rhoInterp=rhoValue;
        %             end
        %
        %         end
        %
        %         %----------------------------------
        %         % Meso Optimization
        %         %----------------------------------
        %         function [DVmeso] =
 MesoDensityOptimization(~,mesoConfig,
 DVmeso,old_muMatrix,penaltyValue,macroElemProps)
        %             ne = mesoConfig.nelx*mesoConfig.nely; % number
 of elements
        %             %               dH_total=[DVmeso.d11;
        %             %                     DVmeso.d12;
        %             %                     DVmeso.d22;
        %             %                     DVmeso.d33];
        %             Diff_Sys_Sub =  (macroElemProps.D_subSys-
 macroElemProps.D_sys);
        %             localD = zeros(3,3);
        %             for e = 1:ne
        %
        %                 [x,y]= DVmeso.GivenNodeNumberGetXY(e);
        %                 xx=DVmeso.x(y,x); % =min(optimalEta,
 designVars.x+move)
        %                 %                  term1 = 10*xx^9;
        %                 %                  power = 1/4;
        %                 %                  term1 =
 power*xx^(power-1);
        %                 term1=2*xx;
        %
        %
        %
        %                 rowIndex = [1,1,2,3];
        %                 columnIndex = [1,2,2,3];
        %
        %                 dH = zeros(3,3);
        %                 dH(1,1) = DVmeso.d11(y,x);
        %                 dH(1,2) = DVmeso.d12(y,x);
        %                 dH(2,2) = DVmeso.d22(y,x);
        %                 dH(3,3) = DVmeso.d33(y,x);
        %
        %                 localD(1,1) = DVmeso.De11(y,x);
        %                 localD(1,2) = DVmeso.De11(y,x);
        %                 localD(2,2) = DVmeso.De11(y,x);
        %                 localD(3,3) = DVmeso.De11(y,x);
        %
        %                 Diff_Sys_Sub =  (localD-
 macroElemProps.D_sys);
        %
        %                 constraintCount = 0;
        %                 term2=0;
        %                 %                 term1=0;
        %                 for k = [1 2 3 ]
        %                     %                     term1=  dH(1,1)+
 dH(1,2)+  dH(2,2)+  dH(3,3);
        %                     i = rowIndex(k);
        %                     j = columnIndex(k);
        %                     Ctemp = dH(i,j)*(-old_muMatrix(i,j)-
penaltyValue*Diff_Sys_Sub(i,j));
        %                     term2 =term2 +Ctemp;
        %                     constraintCount=constraintCount+1;
        %                 end
        %
        %                 dL = term1+term2;
        %                 delta = 0.1;
        %                 optimalEta=xx+delta*dL;
        %                 move = 0.02;
        %                 DVmeso.x(y,x)=  max(0.01,max(xx-
move,min(1.,min(xx+move,optimalEta))));
        %
        %                 DVmeso.x([10:13],[10:13])=1;
        %             end
        %         end

        % ---------------------------------
        % Optimize ANISOTROPIC Material
        %
        % TARGET AVG MESO DENSITY AS CONSTRAINT.
        % ---------------------------------
        function [DV] = OptimizeAnisotropicMaterial(obj,DV,config,
 matProp, masterloop)
            DV = DV.CalculateANISOTROPICSensitivity(config, matProp,
 masterloop);
            DV.sensitivityElastic = DV.check( config.nelx,
 config.nely,config.rminExxEyy,DV.x,DV.sensitivityElastic);
            DV.sensitivityElasticPart2 = DV.check( config.nelx,
 config.nely,config.rminExxEyy,DV.x,DV.sensitivityElasticPart2);
            DV.sensitivityElasticE12 = DV.check( config.nelx,
 config.nely,config.rminExxEyy,DV.x,DV.sensitivityElasticE12);
            DV.sensitivityElasticE33 = DV.check( config.nelx,
 config.nely,config.rminExxEyy,DV.x,DV.sensitivityElasticE33);


            % if(config.macro_meso_iteration>=2 &&
 mod(masterloop,3)==1)
```

```
            if(config.macro_meso_iteration>=2 )
                deltaT=0.2;
                diffExx = DV.ExxSub-DV.Exx;
                diffEyy = DV.EyySub-DV.Eyy  ;
                %
                DV.lambdaExx=max( min(DV.lambdaExx+deltaT
 *diffExx,matProp.E_material1),-matProp.E_material1);
                DV.lambdaEyy= max( min( DV.lambdaEyy
+deltaT*diffEyy,matProp.E_material1),-matProp.E_material1);
                %  DV.lambdaExx=DV.lambdaExx+deltaT *diffExx;
                %   DV.lambdaEyy= DV.lambdaEyy+deltaT*diffEyy;
                disp('Updated Lambda Values Exx Eyy')
            end


            %-----------------------
            %
            % Update design var.
            %-----------------------
            largest=1e8;

            E_target=config.targetAvgExxEyy;

            move = matProp.E_material1*0.05;
            minimum = matProp.E_material2*0.25;

            % ----------------
            % Exx
            % ----------------
            ExxNew = DV.Exx;
            EyyNew = DV.Eyy;
            E12New = DV.E12;
            E33New = DV.E33;

            totalMaterial = sum(sum(DV.x));

            term1Exx = DV.sensitivityElastic;
            term1Eyy= DV.sensitivityElasticPart2;
            term1E12 = DV.sensitivityElasticE12;
            term1E33= DV.sensitivityElasticE33;

            smallestLambdExx = min(min(DV.lambdaExx));
            smallestLambdEyy = min(min(DV.lambdaEyy));
            smallestOfTwo = min(smallestLambdExx,smallestLambdEyy);

            % TODO !!!! Add term2 for E12 E 13
            term2Exx =( DV.lambdaExx-smallestOfTwo).*DV.penaltyExx;
            term2Eyy = (DV.lambdaEyy-smallestOfTwo).*DV.penaltyEyy;

            w1 = 1;
            w2 = 0;
            %    if( config.macro_meso_iteration>=2 ) % Weight toward
 satifying consistency constraint.
```

```
            %    w2=min( ( config.macro_meso_iteration-2)*0.2,1); %
 staring iteration 3, start relaxing the meso density constraint.
            %      w1 = 1-w2;
            %     end

            %             theta = DV.t;

            % ---------------------------------------------------
            %
            % TARGET AVG MESO DENSITY AS CONSTRAINT.
            % Update 3 (idea)
            %
            % ---------------------------------------------------
            l1 = 0; l2 = largest;% move = 0.2;
            sumDensity =0;
            theta=ExxNew*0;
            while (l2-l1 > 1e-5)
                lambda1 = 0.5*(l2+l1);
                %                 ExxInput =ExxNew/
matProp.E_material1; % % MOVED to the function scale down by the simp
 density, since the actual rho is a function of what is SIMP density
 and Exx or Eyy
                %                 EyyInput = EyyNew/
matProp.E_material1;
                %                 [dDensityEyy, dDensityExx,~] =
 obj.CalculateDensitySensitivityandRho(ExxInput,EyyInput,theta,DV.x,DV.ResponseSur

                % testing. Set equal to one for now.
                dDensityExx=ones(size(term1Exx));
                dDensityEyy=ones(size(term1Exx));
                dDensityE12=ones(size(term1Exx));
                dDensityE33=ones(size(term1Exx));

                % TODO !!!! Add term2 for E12 E 13
                combinedTermsExx=(term1Exx+term2Exx)./
(lambda1*dDensityExx);
                combinedTermsEyy=(term1Eyy+term2Eyy)./
(lambda1*dDensityEyy);
                combinedTermsE12=(term1E12+term2Eyy)./
(lambda1*dDensityE12);
                combinedTermsE33=(term1E33+term2Eyy)./
(lambda1*dDensityE33);

                %                 targetExx =
 ExxNew.*combinedTermsExx;
                %                 targetEyy =
 EyyNew.*combinedTermsEyy;
                targetExx = DV.Exx.*combinedTermsExx;
                targetEyy = DV.Eyy.*combinedTermsEyy;
                targetE12 = DV.E12.*combinedTermsE12;
                targetE33 = DV.E33.*combinedTermsE33;

                ExxNew = max(0.1,max( minimum - EyyNew,  max(DV.Exx-
move ,  min(  min(targetExx,DV.Exx+move ),matProp.E_material1))));
```

```
                EyyNew = max(0.1,max(minimum -  ExxNew,  max(DV.Eyy-
move ,  min(  min(targetEyy,DV.Eyy+move ),matProp.E_material1))));

                E12New = max(0.1,max(DV.E12-move ,  min(
  min(targetE12,DV.E12+move ),matProp.E_material1)));
                E33New = max(0.1, max(DV.E33-move ,  min(
  min(targetE33,DV.E33+move ),matProp.E_material1)));

                %                 sumDensity = sumDensity/
(config.nelx*config.nely*config.totalVolume);
                % Determine if the consistency constraint is being
 under
                % valued
                % TODO, add consistency contraint for E12 and E33
 values.
                ExxSysAndSubDiffSummed=sum(sum(DV.x.*(ExxNew-
DV.ExxSub))); %
                EyySysAndSubDiffSummed=sum(sum(DV.x.*(EyyNew-
DV.EyySub)));%
                ConsistConstraintMag = ExxSysAndSubDiffSummed
+EyySysAndSubDiffSummed;
                ConsistConstraintMag=-ConsistConstraintMag;
                ConsistConstraintMag = ConsistConstraintMag/
(matProp.E_material1*totalMaterial);

                if(config.useTargetMesoDensity==1)
                    [~, ~,rhoValue] =
 obj.CalculateDensitySensitivityandRho(ExxNew/
matProp.E_material1,EyyNew/
matProp.E_material1,theta,DV.x,DV.ResponseSurfaceCoefficents,config,matProp,DV.den
                    rhoValue=max(0,min(rhoValue,1));
                    temp2 = sum(sum(rhoValue));
                    sumDensity=temp2/
(config.nelx*config.nely*config.totalVolume);
                    terms= w1*(config.targetExxEyyDensity-
 sumDensity)+w2*(ConsistConstraintMag);
                else

                    totalExx =DV.x.*ExxNew;
                    totalEyy = DV.x.* EyyNew;
                    avgE = (totalExx+totalEyy)/2;
                    averageElasticLocal= sum(sum(avgE))/totalMaterial;
                    %               averageElasticLocal =
 (sum(sum(EyyNew.*Xtemp))+sum(sum(ExxNew.*Xtemp)))/neSolid;
                    %
 averageElasticLocal=averageElasticLocal/2; % Becuse Eyy and Exx are
 from one element, so to get the average divide by 2
                    terms =  E_target- averageElasticLocal;

                end


                if (terms)<0
                    l1 = lambda1;
                else
                    l2 = lambda1;
                end
            end
            multiplier= 10000;
            text1 =    sprintf('\nExxNew\t\t\t\t\t[1,1   5,1
 5,5],  %f ,%f %f',ExxNew(1,1) ,ExxNew(5,1),ExxNew(5,5));
            text2 =    sprintf('ExxSub\t\t\t\t\t[1,1   5,1      5,5],
 %f ,%f %f',DV.ExxSub(1,1),DV.ExxSub(5,1),DV.ExxSub(5,5));
            text3=    sprintf('Exx\\t\t\t\t\t\t[1,1   5,1      5,5],
 %f ,%f %f',DV.Exx(1,1),DV.Exx(5,1),DV.Exx(5,5));
            text4 =    sprintf('DiffXXNew \t\t\t\t[1,1   5,1
   5,5],  %f ,%f %f',ExxNew(1,1) - DV.ExxSub(1,1),ExxNew(5,1) -
 DV.ExxSub(5,1),ExxNew(5,5) - DV.ExxSub(5,5));
            text5 =    sprintf('combinedTermsExx\t\t[1,1   5,1
 5,5],  %f ,%f
 %f',combinedTermsExx(1,1),combinedTermsExx(5,1),combinedTermsExx(5,5));
            text6 =    sprintf('penaltyExx*10000\t\t[1,1   5,1
 5,5],  %f ,%f
 %f',DV.penaltyExx(1,1)*multiplier,DV.penaltyExx(5,1)*multiplier,DV.penaltyExx(5,5
            text7 =    sprintf('lambdaExx\t\t\t\t[1,1   5,1      5,5],
  %f ,%f %f',DV.lambdaExx(1,1),DV.lambdaExx(5,1),DV.lambdaExx(5,5));
            text75 =    sprintf('dDensityExx
\t\t\t\t[1,1   5,1      5,5],  %f ,%f
 %f',dDensityExx(1,1),dDensityExx(5,1),dDensityExx(5,5));
            text8 =    sprintf('lambda1 and density\t\t[  %f ,
%f',lambda1,sumDensity);
            text9 =    sprintf('ConsistConstraintMag and w2 \t\t[
  %f ,%f\n',ConsistConstraintMag,w2);

            disp(text1)
            disp(text2)
            disp(text3)
            disp(text4)
            disp(text5)
            disp(text6)
            disp(text7)
            disp(text75)
            disp(text8)
            disp(text9)


            debug = 0;
            if(debug ==1)
                figure(2)
                p = plotResults;
                xplots=3;
                yplots =3;
                plotNum=1;

                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(dDensityEyy,'dDensityEyy');
                plotNum=plotNum+1;
                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(dDensityExx,'dDensityExx');
                plotNum=plotNum+1;

                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(numeratorExx,'completeExx');
                plotNum=plotNum+1;

                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(numeratorEyy,'completeEyy');
                plotNum=plotNum+1;

                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(combinedTermsExx/
lmid,'combinedTermsExx/lmid');
                plotNum=plotNum+1;

                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(combinedTermsEyy/
lmid,'combinedTermsEyy/lmid');
                plotNum=plotNum+1;

                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(DV.t-DV.thetaSub,'theta diff');
                plotNum=plotNum+1;

                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(DV.Eyy - DV.EyySub,'Eyy diff');
                plotNum=plotNum+1;

                subplot(xplots,yplots,plotNum);
                p. PlotArrayGeneric(DV.Exx-DV.ExxSub,'Exx diff');
                plotNum=plotNum+1;
            end

            % -----------------------
            % Set the valeus.
            % -----------------------
            DV.Exx =DV.Exx.*sqrt( ExxNew./  DV.Exx);
            DV.Eyy = DV.Eyy.*sqrt( EyyNew./  DV.Eyy );
            DV.E12 = DV.Eyy.*sqrt( E12New./  DV.E12 );
            DV.E33 = DV.E33.*sqrt( E33New./  DV.E33 );



        end

                % ----------------------------------
        % Version 2
        % ----------------------------------
%         function [DV] = OptimizeExxEyy_V2(obj,DV,config, matProp,
 masterloop)
%             DV = DV.CalculateExxEyySensitivity(config, matProp,
 masterloop);
```

```
%             DV.sensitivityElastic = DV.check( config.nelx,
 config.nely,config.rminExxEyy,DV.x,DV.sensitivityElastic);
%             DV.sensitivityElasticPart2 = DV.check( config.nelx,
 config.nely,config.rminExxEyy,DV.x,DV.sensitivityElasticPart2);
%
%             if(config.macro_meso_iteration>=2 )
%                 deltaT=0.2;
%                 diffExx = DV.ExxSub-DV.Exx;
%                 diffEyy = DV.EyySub-DV.Eyy  ;
%                 %
%                 DV.lambdaExx=max( min(DV.lambdaExx+deltaT
 *diffExx,matProp.E_material1),-matProp.E_material1);
%                 DV.lambdaEyy= max( min( DV.lambdaEyy
+deltaT*diffEyy,matProp.E_material1),-matProp.E_material1);
%                 %                   DV.lambdaExx=DV.lambdaExx+deltaT
 *diffExx;
%                 %                 DV.lambdaEyy= DV.lambdaEyy
+deltaT*diffEyy;
%                 disp('Updated Lambda Values Exx Eyy')
%
%             end
%
%             %             if(config.testingVerGradMaterail ==1)
%             %                 avgSensitivy =
 0.5*( DV.sensitivityElastic+  DV.sensitivityElasticPart2);
%             %                 DV.sensitivityElastic =avgSensitivy;
%             %                 DV.sensitivityElasticPart2
 =avgSensitivy;
%             %             end
%
%             %-----------------------
%             %
%             % Update design var.
%             %-----------------------
%             largest=1e8;
%             move = matProp.E_material1*0.05;
%             minimum =config.minEallowed;
%
%             %              E_target
 =(config.v1*matProp.E_material1+config.v2*matProp.E_material2)/
(config.v1+config.v2);
%             %             DV.targetAverageE = E_target;
%             E_target=config.targetAvgExxEyy;
%
%             % ----------------
%             % Exx
%             % ----------------
%             ExxNew = DV.Exx;
%             EyyNew = DV.Eyy;
%
%             totalMaterial = sum(sum(DV.x));
%
%             term1Exx = DV.sensitivityElastic;
%             term1Eyy= DV.sensitivityElasticPart2;
```

```
%
%             smallestLambdExx = min(min(DV.lambdaExx));
%             smallestLambdEyy = min(min(DV.lambdaEyy));
%             smallestOfTwo = min(smallestLambdExx,smallestLambdEyy);
%
%             term2Exx =( DV.lambdaExx-smallestOfTwo).*DV.penaltyExx;
%             term2Eyy = (DV.lambdaEyy-smallestOfTwo).*DV.penaltyEyy;
%
%             % -----------------------------------------------
%             %
%             % TARGET AVG MESO DENSITY AS CONSTRAINT.
%             % Update 3 (idea)
%             %
%             % -----------------------------------------------
%             l1 = 0; l2 = largest;% move = 0.2;
%             %             sumDensity =0;
%             while (l2-l1 > 1e-4)
%                 lambda1 = 0.5*(l2+l1);
%                 %                 ExxInput =ExxNew/
matProp.E_material1.*((DV.x).^config.penal); % scale down by the simp
 density, since the actual rho is a function of what is SIMP density
 and Exx or Eyy
%                 %                 EyyInput = EyyNew/
matProp.E_material1.*((DV.x).^config.penal);
%                 %                 [dDensityEyy,
 dDensityExx,rhoValue] =
 obj.CalculateDensitySensitivityandRho(ExxInput,EyyInput,DV.t,DV.ResponseSurfaceCo
%                 % testing. Set equal to one for now.
%                 %                 dDensityExx=ones(size(term1Exx));
%                 %                 dDensityEyy=ones(size(term1Exx));
%                 combinedTermsExx=(term1Exx+term2Exx)./(lambda1*1);
%                 combinedTermsEyy=(term1Eyy+term2Eyy)./(lambda1*1);
%
%                 targetExx = DV.Exx.*combinedTermsExx;
%                 targetEyy = DV.Eyy.*combinedTermsEyy;
%                 ExxNew = max(0.1,max( minimum - EyyNew,  max(DV.Exx-
move ,  min(  min(targetExx,DV.Exx+move ),matProp.E_material1))));
%                 EyyNew = max(0.1,max(minimum -  ExxNew,  max(DV.Eyy-
move ,  min(  min(targetEyy,DV.Eyy+move ),matProp.E_material1))));
%
%                 %                 for i = 1:config.nelx
%                 %                     for j = 1:config.nely
%                 %                         % scale down the X and Y
%                 %                         x=ExxNew(j,i)/
matProp.E_material1;
%                 %                         y=EyyNew(j,i)/
matProp.E_material1;
%                 %                         theta=DV.t(j,i);
%                 %
       [~, ~,estimateElementDensity] =
 obj.CalculateDensitySensitivityandRho(x,y,theta,DV.ResponseSurfaceCoefficents,con
%                 %
%                 %                         estimateElementDensity=
 min(max(estimateElementDensity,0.05),1);%1 is max, 0.5 is min
```

```
%                 %                         eleDensity =
 DV.x(j,i)*estimateElementDensity;
%                 %                         sumDensity =sumDensity
+eleDensity;
%                 %
%                 %
%                 %                     end
%                 %                 end
%                 %                 sumDensity = sumDensity/
(config.nelx*config.nely*config.totalVolume);
%
%                 totalExx =DV.x.*ExxNew;
%                 totalEyy = DV.x.* EyyNew;
%                 avgE = (totalExx+totalEyy)/2;
%                 averageElasticLocal= sum(sum(avgE))/totalMaterial;
%                 %                averageElasticLocal =
 (sum(sum(EyyNew.*Xtemp))+sum(sum(ExxNew.*Xtemp)))/neSolid;
%                 %
 averageElasticLocal=averageElasticLocal/2; % Becuse Eyy and Exx are
 from one element, so to get the average divide by 2
%                 if E_target- averageElasticLocal<0;
%                     l1 = lambda1;
%                 else
%                     l2 = lambda1;
%                 end
%             end
%
%
%             text1 =    sprintf('\nExxNew\t\t\t\t\t[1,1   5,1
 5,5],  %f ,%f %f',ExxNew(1,1) ,ExxNew(5,1),ExxNew(5,5));
%             text2 =    sprintf('ExxSub\t\t\t\t\t[1,1   5,1
 5,5],  %f ,%f %f',DV.ExxSub(1,1),DV.ExxSub(5,1),DV.ExxSub(5,5));
%             text3=    sprintf('Exx\t\t\t\t\t\t[1,1   5,1      5,5],
 %f ,%f %f',DV.Exx(1,1),DV.Exx(5,1),DV.Exx(5,5));
%             text4=    sprintf('combinedTermsExx\t\t[1,1   5,1
 5,5],  %f ,%f
 %f',combinedTermsExx(1,1),combinedTermsExx(5,1),combinedTermsExx(5,5));
%             text5 =    sprintf('DiffXX \t\t\t\t\t[1,1   5,1
    5,5],  %f ,%f %f',DV.Exx(1,1) - DV.ExxSub(1,1),DV.Exx(5,1) -
 DV.ExxSub(5,1),DV.Exx(5,5) - DV.ExxSub(5,5));
%             text6 =    sprintf('lambdaExx
\t\t\t\t[1,1   5,1      5,5],  %f ,%f
 %f',DV.lambdaExx(1,1),DV.lambdaExx(5,1),DV.lambdaExx(5,5));
%
%             text7 =    sprintf('Average E [Target, Current,
 Diff] \t\t\t%f\t%f\t%f', E_target,averageElasticLocal,E_target-
averageElasticLocal);
%
%             disp(text1)
%             disp(text2)
%             disp(text3)
%             disp(text4)
%             disp(text5)
%             disp(text6)
%             disp(text7)
%
%             %             if(config.testingVerGradMaterail ==1)
%             %                 averageNewE = 0.5*(ExxNew+EyyNew);
%             %                 ExxNew=averageNewE;
%             %                 EyyNew=averageNewE;
%             %             end
%
%
%             % -----------------------
%             % Set the valeus.
%             % -----------------------
%             DV.Exx =DV.Exx.*sqrt( ExxNew./  DV.Exx);
%
%             DV.Eyy = DV.Eyy.*sqrt( EyyNew./  DV.Eyy );
%
%
%         end
%
    end
end
```

```
ans = 

  Optimizer with no properties.
```

Published with MATLAB® R2017a