TOPOLOGY OPTIMIZATION USING Dr. SIGMUND'S 88 LINES MATLAB CODE

By

Divya Sree Naidu

Under the guidance of

Dr. Yi (Max) Ren

Assistant professor of Aerospace and Mechanical engineering

School for Engineering of Matter, Transport and Energy

Arizona State University, AZ, US

Abstract

In this study, Topology optimization (TO) is introduced, and the algorithm is implemented based on Dr. Sigmund's 88 lines MATLAB code for minimizing the compliance of a statistically loaded structure at its equilibrium state with respect to its topology. Also, a passive element is added to the structure to make it suitable for adding another elements or pipes to the already existing structure.

Keywords

Topology Optimization, Sigmund's 88 lines MATLAB code

Acknowledgements

I thank my Professor, Dr. Yi Ren for his continuous guidance and support to work on this project and to finish this project successfully.

I thank Dr. Ole Sigmund for developing this code for us to learn how to implement topology optimization in various engineering applications.

Contents

1. Introduction

What is Topology Optimization:

Topology Optimization (TO) is a shape optimization method that uses algorithmic models to optimize material layout within a user-defined space for a given set of loads, conditions, and constraints. TO maximizes the performance and efficiency of the design by removing redundant material from areas that do not need to carry significant loads to reduce weight or solve design challenges like reducing resonance or thermal stress.

Designs produced with topology optimization often include free forms and intricate shapes that are complex or impossible to manufacture with traditional production methods. However, TO designs are a perfect match for additive manufacturing processes that have more forgiving design rules and can easily reproduce complex shapes without additional costs.

Advantages of TO:

- ✓ Less expensive because of lower packaging & transportation costs and also heavy machinery isn’t necessary for assembly lines.
- ✓ Solves design challenges such as changes due to resonance and thermal stress.
- ✓ Saves time and reduces environmental impact.
- ✓ Eliminates the errors.

Applications:

Aerospace: To improve the layout design for airframe structures, such as stiffener ribs or brackets for aircraft.

Automotive:

In the automotive industry, topology optimization balances the desirability of lightweight parts for fuel efficiency and power with the stability and strength of a body that can withstand torque and impact.

Besides mass savings, topology optimization can also improve passenger safety by defining the way a structure collapses during an accident.

Medical:

TO tools can also optimize the designs of biodegradable scaffolds for tissue engineering, porous implants, and lightweight orthopedics. Nanotechnology applications—such as cell manipulation, surgery, micro fluids, and optical systems—also use topology optimization.

2. Problem formulation and Boundary Conditions

The design domain, the boundary conditions, and the external load for the beam are shown in below figure.

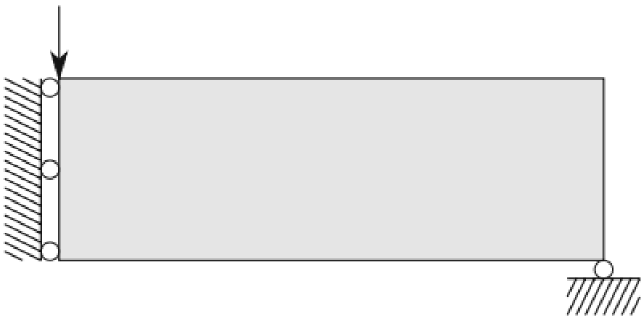


Fig. 1 The design domain, boundary conditions, and external load for the optimization of a symmetric MBB beam

The objective of the optimization problem is to determine the optimal material distribution on the beam with respect to minimum compliance and constant total amount of material.

The mathematical formulation of the optimization problem is as follows:

$$\begin{aligned} \min_{\mathbf{x}} : \quad & c(\mathbf{x}) = \mathbf{U}^{\mathrm{T}}\mathbf{K}\mathbf{U} = \sum_{e=1}^{N} E_e(x_e)\mathbf{u}_e^{\mathrm{T}}\mathbf{k}_0\mathbf{u}_e \\ \text{subject to :} \quad & V(\mathbf{x})/V_0 = f \\ & \mathbf{K}\mathbf{U} = \mathbf{F} \\ & \mathbf{0} \leq \mathbf{x} \leq \mathbf{1} \end{aligned}$$

Where, c is the compliance

U is global displacement vector

F is global Force vector

K is global Stiffness matrix

k_0 is the element stiffness matrix of an element with unit young’s modulus

x is the vector of design variables (i.e., The element densities)

N is the number of elements used to discretize the design domain

V(x) and V0 are the material volume and design domain volume, respectively, and

f is the prescribed volume fraction

2.1. Optimality criteria method

The optimization problem is solved by using a standard optimality criteria method. A heuristic updating scheme is as follows:

$$x_e^{\text{new}} = \begin{cases} \max(0, x_e - m) & \text{if } x_e B_e^{\eta} \leq \max(0, x_e - m) \\ \min(1, x_e + m) & \text{if } x_e B_e^{\eta} \geq \min(1, x_e - m) \\ x_e B_e^{\eta} & \text{otherwise} \end{cases} \quad (3)$$

where m is a positive move limit

η (= 1/2) is a numerical damping coefficient, and

B_e is obtained from the optimality condition

$$B_e = \frac{-\frac{\partial c}{\partial x_e}}{\lambda \frac{\partial V}{\partial x_e}}$$

where λ is the Lagrangian multiplier and it must be chosen in such a way that the volume constraint is satisfied.

The appropriate value for λ can be found by means of a bisection algorithm.

The sensitivities of the objective function c and the material volume V with respect to the element densities x_e is given by:

$$\frac{\partial c}{\partial x_e} = -p x_e^{p-1} (E_0 - E_{\min}) \mathbf{u}_e^{\mathrm{T}} \mathbf{k}_0 \mathbf{u} \qquad (5)$$

$$\frac{\partial V}{\partial x_e} = 1 \qquad (6)$$

Equation (6) assumes that each element has unit volume.

2.2. Filtering

To ensure existence of solutions to the topology optimization problem and to avoid the formation of checker-board patterns (Díaz and Sigmund 1995; Jog and Haber 1996; Sigmund and Petersson 1998), some restriction on the design must be imposed. A common approach is the application of a filter to either the sensitivities or the densities.

In addition to the sensitivity filter (Sigmund 1994, 1997), which is already implemented in the 99-line code, the new 88-line code also includes density filtering (Bruns and Tortorelli 2001; Bourdin 2001).

The sensitivity filter modifies the sensitivities $\partial c/\partial x_e$ as follows:

$$\widehat{\frac{\partial c}{\partial x_e}} = \frac{1}{\max(\gamma, x_e)\sum_{i\in N_e} H_{ei}} \sum_{i\in N_e} H_{ei} x_i \frac{\partial c}{\partial x_i} \qquad (7)$$

Where, N_e is the set of elements i for which the center-to-center distance Δ (e, i) to element e is smaller than the filter radius r_{min} and H_{ei} is a weight factor which is defined as:

$$H_{ei} = \max\left(0, r_{\min} - \Delta(e, i)\right)$$

γ (=10^{-3}) is a small positive number introduced newly in 88-line code in order to avoid division by zero.

The density filter transforms the original densities x_e as follows:

$$\tilde{x}_e = \frac{1}{\sum_{i\in N_e} H_{ei}} \sum_{i\in N_e} H_{ei} x_i$$

So, the sensitivities with respect to design variables x_j are obtained by:

$$\frac{\partial \psi}{\partial x_j} = \sum_{e\in N_j} \frac{\partial \psi}{\partial \tilde{x}_e}\frac{\partial \tilde{x}_e}{\partial x_j} = \sum_{e\in N_j} \frac{1}{\sum_{i\in N_e} H_{ei}} H_{je} \frac{\partial \psi}{\partial \tilde{x}_e} \qquad (10)$$

Where, the function ψ represents either the objective function or the material volume V.

3. MATLAB implementation

```
%%%% Modified by Max Yi Ren (ASU) %%%%%%%%%%%%%%%%%%%%%%%
%%%% AN 88 LINE TOPOLOGY OPTIMIZATION CODE Nov, 2010 %%%%
function top88(nelx,nely,volfrac,penal,rmin,ft)
```

```
nelx=150;
nely=100;
volfrac=0.5;
penal=3;
rmin=5;
ft = 1;
```

MATERIAL PROPERTIES

```
E0 = 1;
Emin = 1e-9;
nu = 0.3;
```

3.1. PREPARE FINITE ELEMENT ANALYSIS

```
A11 = [12 3 -6 -3; 3 12 3 0; -6 3 12 -3; -3 0 -3 12];
A12 = [-6 -3 0 3; -3 -6 -3 -6; 0 -3 -6 3; 3 -6 3 -6];
B11 = [-4 3 -2 9; 3 -4 -9 4; -2 -9 -4 -3; 9 4 -3 -4];
B12 = [ 2 -3 4 -9; -3 2 9 -2; 4 9 2 3; -9 -2 3 2];
KE = 1/(1-nu^2)/24*([A11 A12;A12' A11]+nu*[B11 B12;B12' B11]);
nodenrs = reshape(1:(1+nelx)*(1+nely),1+nely,1+nelx);
edofVec = reshape(2*nodenrs(1:end-1,1:end-1)+1,nelx*nely,1);
edofMat = repmat(edofVec,1,8)+repmat([0 1 2*nely+[2 3 0 1] -2 -1],nelx*nely,1);
iK = reshape(kron(edofMat,ones(8,1))',64*nelx*nely,1);
jK = reshape(kron(edofMat,ones(1,8))',64*nelx*nely,1);
% DEFINE LOADS AND SUPPORTS (HALF MBB-BEAM)
F = sparse(2,1,-1,2*(nely+1)*(nelx+1),1);
U = zeros(2*(nely+1)*(nelx+1),1);
fixeddofs = union((1:2:2*(nely+1)),(2*(nelx+1)*(nely+1)));
alldofs = (1:2*(nely+1)*(nelx+1));
freedofs = setdiff(alldofs,fixeddofs);
```

3.2. PREPARE FILTER

```
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);
%INITIALIZE ITERATION
x = repmat(volfrac,nely,nelx);
xPhys = x;
loop = 0;
change = 1;
% START ITERATION
while change > 0.01
```

```
loop = loop + 1;
```

3.1.1. FE-ANALYSIS

```
sK= reshape(KE(:)*(Emin+xPhys(:)'.^penal*(E0-Emin)),64*nelx*nely,1);
 K = sparse(iK,jK,sK); K = (K+K')/2;
 U(freedofs) = K(freedofs,freedofs)\F(freedofs);
```

3.3. OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

```
ce = reshape(sum((U(edofMat)*KE).*U(edofMat),2),nely,nelx); % element-wise strain energy
 c = sum(sum((Emin+xPhys.^penal*(E0-Emin)).*ce)); % total strain energy
 dc = -penal*(E0-Emin)*xPhys.^(penal-1).*ce; % design sensitivity
 dv = ones(nely,nelx);
```

3.2.1 FILTERING/MODIFICATION OF SENSITIVITIES

3.2.1.1. Heaviside projection filter

```
ft=heaviside(x);
 if ft == 2
 dc(:) = H*(x(:).*dc(:))./Hs./max(1e-3,x(:));
 elseif ft == 3
 dc(:) = H*(dc(:)./Hs);
 dv(:) = H*(dv(:)./Hs);
 end
```

3.3.1. OPTIMALITY CRITERIA UPDATE OF DESIGN VARIABLES AND PHYSICAL DENSITIES

```
l1 = 0; l2 = 1e9; move=0.2;
```

3.3.2. Passive Element

```
passive = zeros(nely,nelx);
for i = 1:nelx
    for j = 1:nely
        if sqrt((j-nely/2)^2+(i-nelx/3)^2) < nely/3
        passive(j,i) = 1;
    end
end
end

 while (l2-l1)/(l1+l2) > 1e-3
 lmid = 0.5*(l2+l1);
 xnew = max(0,max(x-move,min(1,min(x+move,x.*sqrt(-dc./dv/lmid)))));
 if ft == 1
 xPhys = xnew;
 elseif ft==2
 xPhys(:) = (H*xnew(:))./Hs;
 end
 xPhys(passive==1) = 0;
 xPhys(passive==2) = 1;
 if sum(xPhys(:)) > volfrac*nelx*nely, l1 = lmid;
 else
 l2 = lmid;
 end
 end
 change = max(abs(xnew(:)-x(:)));
 x = xnew;
```

4. PRINT RESULTS

```
fprintf(' It.:%5i Obj.:%11.4f Vol.:%7.3f ch.:%7.3f\n',loop,c, ...
 mean(xPhys(:)),change);
```

```
 It.:    1 Obj.:   267.2958 Vol.:  0.500 ch.:  0.200

 It.:    2 Obj.:   163.7187 Vol.:  0.500 ch.:  0.200

 It.:    3 Obj.:    91.4011 Vol.:  0.500 ch.:  0.200

 It.:    4 Obj.:    69.7513 Vol.:  0.500 ch.:  0.200

 It.:    5 Obj.:    64.1504 Vol.:  0.500 ch.:  0.200

 It.:    6 Obj.:    61.8320 Vol.:  0.500 ch.:  0.200

 It.:    7 Obj.:    61.8320 Vol.:  0.500 ch.:  0.200

 It.:    8 Obj.:    61.8320 Vol.:  0.500 ch.:  0.200

 It.:    9 Obj.:    61.8320 Vol.:  0.500 ch.:  0.200

 It.:   10 Obj.:    61.8320 Vol.:  0.500 ch.:  0.200
```

```
It.:   11 Obj.:    61.8320 Vol.:  0.500 ch.:  0.000
```

PLOT DENSITIES

```
image1 = colormap(gray); imagesc(1-xPhys); caxis([0 1]); axis equal; axis off; drawnow;
% Save as png with a resolution of 150 pixels per inch
vwObj = VideoWriter('myfile');
open(vwObj);
f.cdata = image1;
f.colormap = jet(256);
% The colormap will be applied before writing the data to the MPEG4 file
writeVide;o(vwObj, f);
close(vwObj);

end

end
```

4. Results

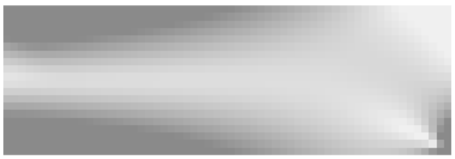
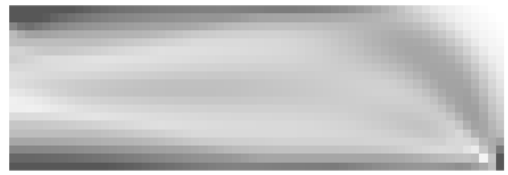
4.1. Without Passive Element:

The below mentioned values are changed in the code:

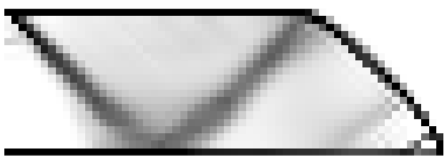
```
nelx=60;
nely=20;
volfrac=0.26;
penal=3;
rmin=1.5;
ft = 1;
```

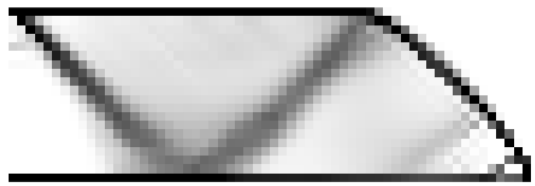
```
It.:    1 Obj.:  7161.9115 Vol.:  0.260 ch.:  0.200
It.:    2 Obj.:  3927.4154 Vol.:  0.260 ch.:  0.200
It.:    3 Obj.:  2511.7836 Vol.:  0.260 ch.:  0.200
It.:    4 Obj.:  1876.5459 Vol.:  0.260 ch.:  0.200
It.:    5 Obj.:  1488.7916 Vol.:  0.260 ch.:  0.200
It.:    6 Obj.:  1248.8078 Vol.:  0.260 ch.:  0.200
It.:    7 Obj.:  1097.5088 Vol.:  0.260 ch.:  0.200
It.:    8 Obj.:   961.3632 Vol.:  0.260 ch.:  0.200
It.:    9 Obj.:   826.1684 Vol.:  0.260 ch.:  0.200
It.:   10 Obj.:   697.1953 Vol.:  0.260 ch.:  0.200
It.:   11 Obj.:   697.1953 Vol.:  0.260 ch.:  0.200
It.:   12 Obj.:   697.1953 Vol.:  0.260 ch.:  0.200
It.:   13 Obj.:   697.1953 Vol.:  0.260 ch.:  0.200
It.:   14 Obj.:   697.1953 Vol.:  0.260 ch.:  0.200
It.:   15 Obj.:   697.1953 Vol.:  0.260 ch.:  0.000
```

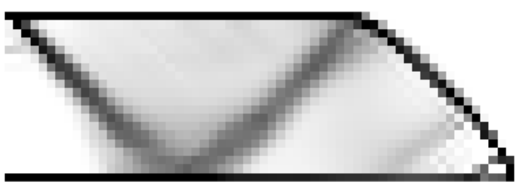
The below list of figures shows the flow of how optimized design of the beam without passive element is obtained.

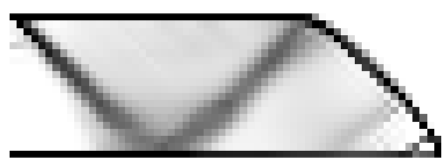





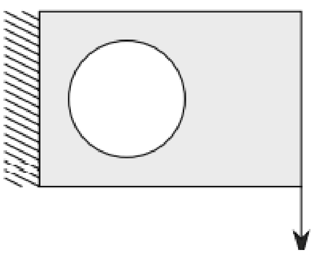




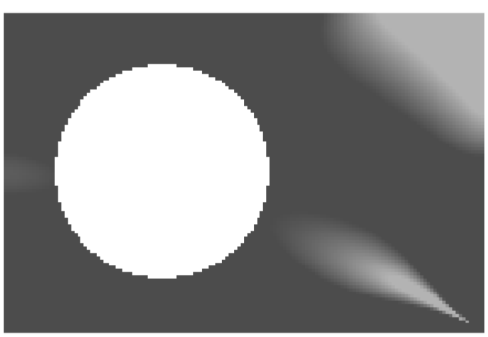


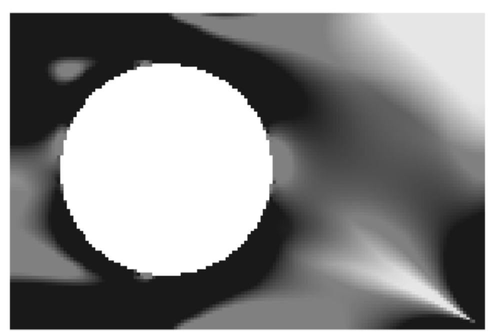


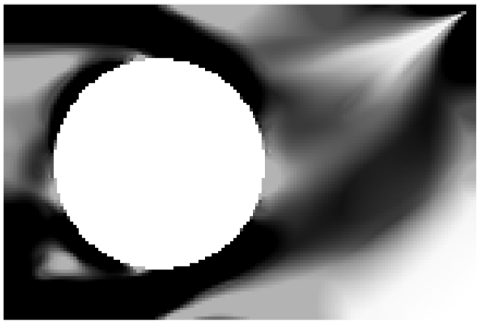
4.2. With Passive Element:

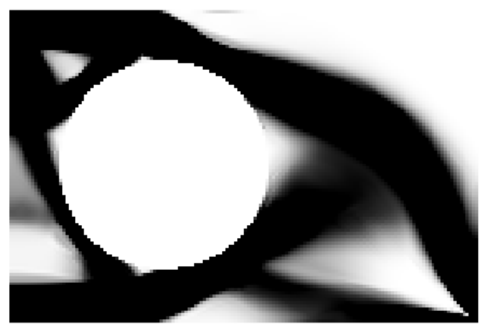


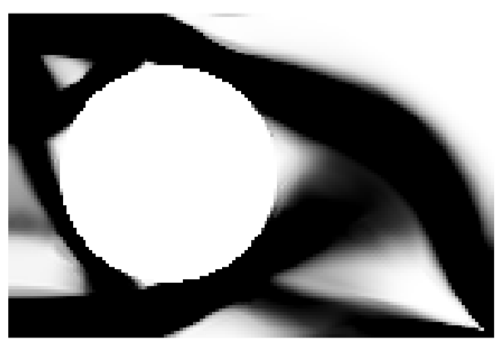
The below list of figures shows the flow of how optimized design of the beam with passive element added to it is obtained.

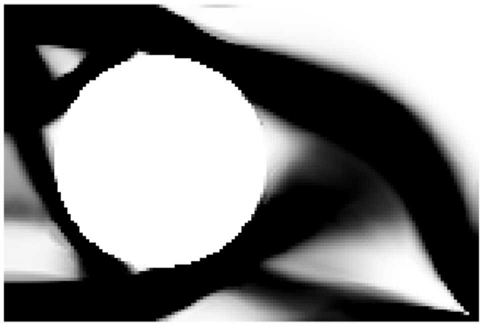












5. Conclusion

This study shows how topology optimization can be implemented using Dr. Sigmund's 88-line MATLAB code on a beam with and without passive element. Through this study, it is observed that 88-line code has more computational efficiency than that of 99-line code (see appendix). An improvement in speed of implementing the TO algorithm is observed between 88-line code and 99-line code.

6. References

1. Topology Optimization tutorial by Yi (Max) Ren
2. https://github.com/DesignInformaticsLab/DesignOptimization2021Fall/blob/292523767a03eb6974ab9e0058731ed5dae5f0f5/Project/Project%203%20topology%20optimization.ipynb
3. Efficient topology optimization in MATLAB using 88 lines of code by Erik Andreassen, Anders Clausen, Mattias Schevenels, Boyan S. Lazarov and Ole Sigmund
4. A 99 line topology optimization code written in MATLAB by Ole Sigmund.

7. Appendix – Dr. Sigmund's 99-line code

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, OCTOBER 1999
 %%%
function P3_99lines(nelx,nely,volfrac,penal,rmin)
% INITIALIZE
nelx=60;
nely=20;
% nelz = 4;
volfrac=0.3;
penal=3;
rmin=1.5;
x(1:nely,1:nelx) = volfrac;
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;
        Ue = U([2*n1-1;2*n1; 2*n2-1;2*n2; 2*n2+1;2*n2+2;
 2*n1+1;2*n1+2],1);
        c = c + x(ely,elx)^penal*Ue'*KE*Ue;
        dc(ely,elx) = -penal*x(ely,elx)^(penal-1)*Ue'*KE*Ue;
    end
end
% FILTERING OF SENSITIVITIES
[dc] = check(nelx,nely,rmin,x,dc);
% DESIGN UPDATE BY THE OPTIMALITY CRITERIA METHOD
[x] = OC(nelx,nely,x,volfrac,dc);
% PRINT RESULTS
change = max(max(abs(x-xold)));
fprintf(' It.:%5i Obj.:%11.4f Vol.:%7.3f ch.:%7.3f\n',loop,c, ...
 sum(sum(x))/(nelx*nely),change);
% disp(['It.: 'sprintf('%4i',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
end
%%%%%%%%%% OPTIMALITY CRITERIA UPDATE %%%%%%%%%
function [xnew]=OC(nelx,nely,x,volfrac,dc)
```

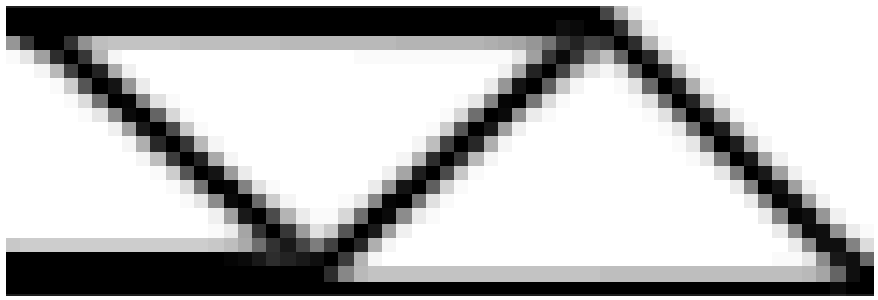
```
l1 = 0; l2 = 100000; move = 0.2;
while (l2-l1 > 1e-4)
lmid = 0.5*(l2+l1);
xnew = max(0.001,max(x-move,min(1.,min(x+move,x.*sqrt(-dc./lmid)))));
if sum(sum(xnew)) - volfrac*nelx*nely > 0
    l1 = lmid;
else
    l2 = lmid;
end
end
end
%%%%%%%%%% MESH-INDEPENDENCY FILTER %%%%%%%%%%%
function [dcn]=check(nelx,nely,rmin,x,dc)
dcn=zeros(nely,nelx);
for i = 1:nelx
    for j = 1:nely
        sum=0.0;
        for k = max(i-round(rmin),1):min(i+round(rmin),nelx)
            for l = max(j-round(rmin),1):min(j+round(rmin), nely)
                fac = rmin-sqrt((i-k)^2+(j-l)^2);
                sum = sum+max(0,fac);
                dcn(j,i) = dcn(j,i) + max(0,fac)*x(l,k)*dc(l,k);
            end
        end
        dcn(j,i) = dcn(j,i)/(x(j,i)*sum);
    end
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),1); U = sparse(2*(nely+1)*(nelx
+1),1);
        for ely = 1:nely
            for elx = 1:nelx
                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 LOADSAND SUPPORTS(HALF MBB-BEAM)
F(2,1) = -1;
fixeddofs = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
alldofs = [1:2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);
% SOLVING
U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);
U(fixeddofs,:)= 0;
end
%%%%%%%%%% 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)];
end
```

```
 It.:    1 Obj.:  4662.1394 Vol.:  0.300 ch.:  0.200
 It.:    2 Obj.:  2567.8233 Vol.:  0.300 ch.:  0.200
 It.:    3 Obj.:  1681.8699 Vol.:  0.300 ch.:  0.200
 It.:    4 Obj.:  1307.7292 Vol.:  0.300 ch.:  0.200
 It.:    5 Obj.:  1104.5787 Vol.:  0.300 ch.:  0.200
 It.:    6 Obj.:   990.0412 Vol.:  0.300 ch.:  0.200
 It.:    7 Obj.:   910.4283 Vol.:  0.300 ch.:  0.161
 It.:    8 Obj.:   845.9893 Vol.:  0.300 ch.:  0.200
 It.:    9 Obj.:   789.8929 Vol.:  0.300 ch.:  0.164
 It.:   10 Obj.:   731.9954 Vol.:  0.300 ch.:  0.200
 It.:   11 Obj.:   664.8502 Vol.:  0.300 ch.:  0.200
 It.:   12 Obj.:   596.5217 Vol.:  0.300 ch.:  0.200
 It.:   13 Obj.:   525.3106 Vol.:  0.300 ch.:  0.200
 It.:   14 Obj.:   467.9681 Vol.:  0.300 ch.:  0.200
 It.:   15 Obj.:   424.6173 Vol.:  0.300 ch.:  0.194
 It.:   16 Obj.:   396.5655 Vol.:  0.300 ch.:  0.173
 It.:   17 Obj.:   378.5753 Vol.:  0.300 ch.:  0.139
 It.:   18 Obj.:   370.0419 Vol.:  0.300 ch.:  0.135
 It.:   19 Obj.:   364.4705 Vol.:  0.300 ch.:  0.123
 It.:   20 Obj.:   361.9186 Vol.:  0.300 ch.:  0.109
 It.:   21 Obj.:   359.7093 Vol.:  0.300 ch.:  0.102
 It.:   22 Obj.:   358.8274 Vol.:  0.300 ch.:  0.089
 It.:   23 Obj.:   357.5683 Vol.:  0.300 ch.:  0.086
 It.:   24 Obj.:   357.0850 Vol.:  0.300 ch.:  0.082
 It.:   25 Obj.:   356.2878 Vol.:  0.300 ch.:  0.079
 It.:   26 Obj.:   355.9764 Vol.:  0.300 ch.:  0.077
 It.:   27 Obj.:   355.4192 Vol.:  0.300 ch.:  0.074
 It.:   28 Obj.:   355.1944 Vol.:  0.300 ch.:  0.072
 It.:   29 Obj.:   354.8280 Vol.:  0.300 ch.:  0.071
 It.:   30 Obj.:   354.6682 Vol.:  0.300 ch.:  0.070
 It.:   31 Obj.:   354.3960 Vol.:  0.300 ch.:  0.071
 It.:   32 Obj.:   354.2509 Vol.:  0.300 ch.:  0.070
 It.:   33 Obj.:   354.0411 Vol.:  0.300 ch.:  0.071
 It.:   34 Obj.:   353.8948 Vol.:  0.300 ch.:  0.069
 It.:   35 Obj.:   353.7146 Vol.:  0.300 ch.:  0.070
 It.:   36 Obj.:   353.5622 Vol.:  0.300 ch.:  0.068
 It.:   37 Obj.:   353.4185 Vol.:  0.300 ch.:  0.069
 It.:   38 Obj.:   353.2650 Vol.:  0.300 ch.:  0.067
 It.:   39 Obj.:   353.1240 Vol.:  0.300 ch.:  0.068
```

```
It.:   40 Obj.:   352.9963 Vol.:  0.300 ch.:  0.065
It.:   41 Obj.:   352.8761 Vol.:  0.300 ch.:  0.066
It.:   42 Obj.:   352.7386 Vol.:  0.300 ch.:  0.064
It.:   43 Obj.:   352.6278 Vol.:  0.300 ch.:  0.065
It.:   44 Obj.:   352.5122 Vol.:  0.300 ch.:  0.063
It.:   45 Obj.:   352.4029 Vol.:  0.300 ch.:  0.063
It.:   46 Obj.:   352.2982 Vol.:  0.300 ch.:  0.061
It.:   47 Obj.:   352.2018 Vol.:  0.300 ch.:  0.062
It.:   48 Obj.:   352.0891 Vol.:  0.300 ch.:  0.060
It.:   49 Obj.:   351.9791 Vol.:  0.300 ch.:  0.060
It.:   50 Obj.:   351.8878 Vol.:  0.300 ch.:  0.058
It.:   51 Obj.:   351.7768 Vol.:  0.300 ch.:  0.059
It.:   52 Obj.:   351.6805 Vol.:  0.300 ch.:  0.056
It.:   53 Obj.:   351.5747 Vol.:  0.300 ch.:  0.057
It.:   54 Obj.:   351.4570 Vol.:  0.300 ch.:  0.055
It.:   55 Obj.:   351.3630 Vol.:  0.300 ch.:  0.056
It.:   56 Obj.:   351.2588 Vol.:  0.300 ch.:  0.053
It.:   57 Obj.:   351.1409 Vol.:  0.300 ch.:  0.054
It.:   58 Obj.:   351.0416 Vol.:  0.300 ch.:  0.052
It.:   59 Obj.:   350.9421 Vol.:  0.300 ch.:  0.053
It.:   60 Obj.:   350.8447 Vol.:  0.300 ch.:  0.050
It.:   61 Obj.:   350.7427 Vol.:  0.300 ch.:  0.051
It.:   62 Obj.:   350.6325 Vol.:  0.300 ch.:  0.049
It.:   63 Obj.:   350.5142 Vol.:  0.300 ch.:  0.050
It.:   64 Obj.:   350.4030 Vol.:  0.300 ch.:  0.048
It.:   65 Obj.:   350.2975 Vol.:  0.300 ch.:  0.048
It.:   66 Obj.:   350.1876 Vol.:  0.300 ch.:  0.046
It.:   67 Obj.:   350.0701 Vol.:  0.300 ch.:  0.047
It.:   68 Obj.:   349.9723 Vol.:  0.300 ch.:  0.045
It.:   69 Obj.:   349.8516 Vol.:  0.300 ch.:  0.046
It.:   70 Obj.:   349.7457 Vol.:  0.300 ch.:  0.044
It.:   71 Obj.:   349.6442 Vol.:  0.300 ch.:  0.044
It.:   72 Obj.:   349.5480 Vol.:  0.300 ch.:  0.042
It.:   73 Obj.:   349.4415 Vol.:  0.300 ch.:  0.043
It.:   74 Obj.:   349.3376 Vol.:  0.300 ch.:  0.041
It.:   75 Obj.:   349.2390 Vol.:  0.300 ch.:  0.042
It.:   76 Obj.:   349.1593 Vol.:  0.300 ch.:  0.040
It.:   77 Obj.:   349.0439 Vol.:  0.300 ch.:  0.041
It.:   78 Obj.:   348.9551 Vol.:  0.300 ch.:  0.039
It.:   79 Obj.:   348.8621 Vol.:  0.300 ch.:  0.040
It.:   80 Obj.:   348.7822 Vol.:  0.300 ch.:  0.038
It.:   81 Obj.:   348.6925 Vol.:  0.300 ch.:  0.039
It.:   82 Obj.:   348.6187 Vol.:  0.300 ch.:  0.037
It.:   83 Obj.:   348.5132 Vol.:  0.300 ch.:  0.038
It.:   84 Obj.:   348.4416 Vol.:  0.300 ch.:  0.036
It.:   85 Obj.:   348.3400 Vol.:  0.300 ch.:  0.037
It.:   86 Obj.:   348.2563 Vol.:  0.300 ch.:  0.035
It.:   87 Obj.:   348.1650 Vol.:  0.300 ch.:  0.036
It.:   88 Obj.:   348.0711 Vol.:  0.300 ch.:  0.034
It.:   89 Obj.:   347.9532 Vol.:  0.300 ch.:  0.035
It.:   90 Obj.:   347.8465 Vol.:  0.300 ch.:  0.033
It.:   91 Obj.:   347.7242 Vol.:  0.300 ch.:  0.034
It.:   92 Obj.:   347.6164 Vol.:  0.300 ch.:  0.032
It.:   93 Obj.:   347.5022 Vol.:  0.300 ch.:  0.033
```

```
It.:   94 Obj.:   347.3885 Vol.:  0.300 ch.:  0.032
It.:   95 Obj.:   347.2654 Vol.:  0.300 ch.:  0.032
It.:   96 Obj.:   347.1450 Vol.:  0.300 ch.:  0.031
It.:   97 Obj.:   347.0125 Vol.:  0.300 ch.:  0.030
It.:   98 Obj.:   346.8833 Vol.:  0.300 ch.:  0.030
It.:   99 Obj.:   346.7355 Vol.:  0.300 ch.:  0.028
It.:  100 Obj.:   346.6204 Vol.:  0.300 ch.:  0.028
It.:  101 Obj.:   346.4783 Vol.:  0.300 ch.:  0.025
It.:  102 Obj.:   346.3646 Vol.:  0.300 ch.:  0.025
It.:  103 Obj.:   346.2289 Vol.:  0.300 ch.:  0.023
It.:  104 Obj.:   346.1211 Vol.:  0.300 ch.:  0.023
It.:  105 Obj.:   346.0047 Vol.:  0.300 ch.:  0.021
It.:  106 Obj.:   345.9250 Vol.:  0.300 ch.:  0.021
It.:  107 Obj.:   345.8228 Vol.:  0.300 ch.:  0.018
It.:  108 Obj.:   345.7359 Vol.:  0.300 ch.:  0.019
It.:  109 Obj.:   345.6407 Vol.:  0.300 ch.:  0.017
It.:  110 Obj.:   345.5534 Vol.:  0.300 ch.:  0.017
It.:  111 Obj.:   345.4508 Vol.:  0.300 ch.:  0.015
It.:  112 Obj.:   345.3898 Vol.:  0.300 ch.:  0.015
It.:  113 Obj.:   345.2807 Vol.:  0.300 ch.:  0.013
It.:  114 Obj.:   345.2142 Vol.:  0.300 ch.:  0.013
It.:  115 Obj.:   345.1331 Vol.:  0.300 ch.:  0.011
It.:  116 Obj.:   345.0632 Vol.:  0.300 ch.:  0.012
It.:  117 Obj.:   344.9707 Vol.:  0.300 ch.:  0.011
It.:  118 Obj.:   344.8948 Vol.:  0.300 ch.:  0.011
It.:  119 Obj.:   344.7865 Vol.:  0.300 ch.:  0.011
It.:  120 Obj.:   344.7220 Vol.:  0.300 ch.:  0.011
It.:  121 Obj.:   344.6258 Vol.:  0.300 ch.:  0.011
It.:  122 Obj.:   344.5381 Vol.:  0.300 ch.:  0.011
It.:  123 Obj.:   344.4788 Vol.:  0.300 ch.:  0.011
It.:  124 Obj.:   344.4169 Vol.:  0.300 ch.:  0.010
It.:  125 Obj.:   344.3757 Vol.:  0.300 ch.:  0.011
It.:  126 Obj.:   344.3241 Vol.:  0.300 ch.:  0.011
It.:  127 Obj.:   344.2844 Vol.:  0.300 ch.:  0.011
It.:  128 Obj.:   344.2343 Vol.:  0.300 ch.:  0.011
It.:  129 Obj.:   344.1866 Vol.:  0.300 ch.:  0.011
It.:  130 Obj.:   344.1387 Vol.:  0.300 ch.:  0.011
It.:  131 Obj.:   344.1048 Vol.:  0.300 ch.:  0.012
It.:  132 Obj.:   344.0837 Vol.:  0.300 ch.:  0.012
It.:  133 Obj.:   344.0905 Vol.:  0.300 ch.:  0.011
It.:  134 Obj.:   344.1153 Vol.:  0.300 ch.:  0.010
It.:  135 Obj.:   344.1320 Vol.:  0.300 ch.:  0.009
```



Published with MATLAB® R2021a