CEE 307 STRUCTURAL DESIGN OPTIMIZATION FALL 2016

Instructor: Dr. Glaucio Paulino

STANFORD UNIVERSITY

RP PROJECT REPORT

Authors:

Lanxi Liu

Wenjin Situ

Date:

Thursday, Dec 8, 2017

Contents:

Listings

1 Problem Statement

The goal of this project is to use optimization tool such as TOP 3D Matlab Code to optimize an inverted 2-story building structure. The physical constraint of the manufactured model is limited to a 2.5'' x 2.5'' x 4'' build envelope.

2 Proposed Model

2.1 Motivation

The inspiration of this project originated from two famous built structures: Geisel Library at UCSD and the China pavilion at Expo 2010 as shown below.

Figure 1. UCSD Geisel Library

Figure 2. China Pabilion

Both inverted structures shown above have extended cantilevers along floor height, and the floor area gradually decreases from top to bottom.

The geometry of the external walls of the inverted building imitates the China pavilion, which contains a large square on top and a small square on the base. The effect of the cantilevers to the shape generated of the façade is explored.

Since the main concern of the structure comes from the cantilevers, we consider only vertical loads acting on the roof and floors for the optimization purpose. The support constraint is simplified as all fixed ends on the base level.

2.2 Boundary value problem

The generation of the finished structure is based on the Matlab code *Top3Dmodified*. The symmetry of the structure is utilized to save computational efforts. Therefore, only a quarter of the structure is modeled and optimized with the code. The boundary condition is modeled as fixed support at the base, and pined connection at symmetric planes. The loads applied are two discretized area loads acting on the roof and the second floor of the

building. To restrict the design domain to external façade, two passive zones including an interior region and an exterior region. The thickness of the façade is restrained within 15% of the edge length of the top square. To model the windows of the building, additional passive zones are placed as two cylinders across each level in two directions. As shown in Figure 3, the applied loads are shown in the blue region, while the red areas represented the pinned condition in the symmetry planes and the fixed condition in the base. As shown in Figure 4, the blue area is the design domain, while the white plain area is the passive zone. In addition, the black area is predefined as solid and will not be optimized. Figure 5 indicates the passive zone for windows in the building; Figure 6 labels the overall dimensions for the whole structure.

1. Domain discretization: nelx = 50, nely = 50, nelz = 80
2. Total number of elements: 200,000
3. Volume fraction: 15%
4. Maximum number of iterations: 200
5. Penalization p: 1
6. Filter exponent q: 2
7. Cutoff ρ: 0.5
8. Numerical solver: Iterative – Jacobi PCG (Preconditioned Conjugate Gradient)

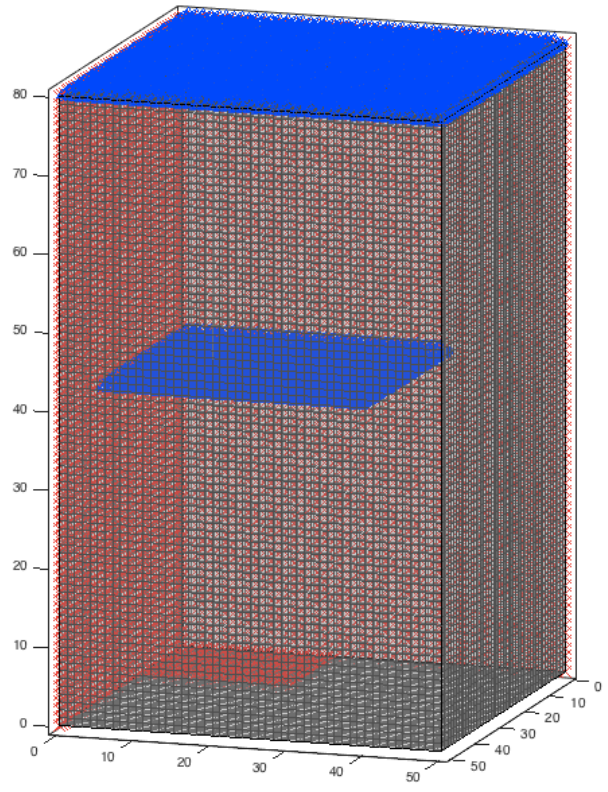

Figure 3. Boundary and Load Condition

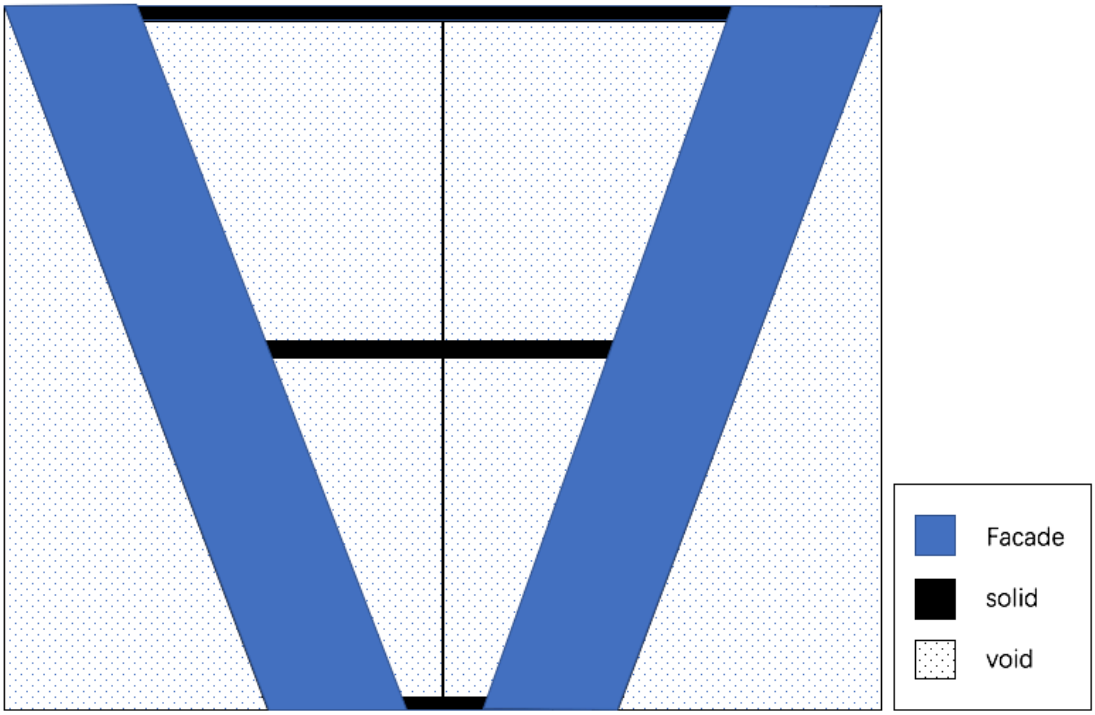

Figure 4. Passive Zone

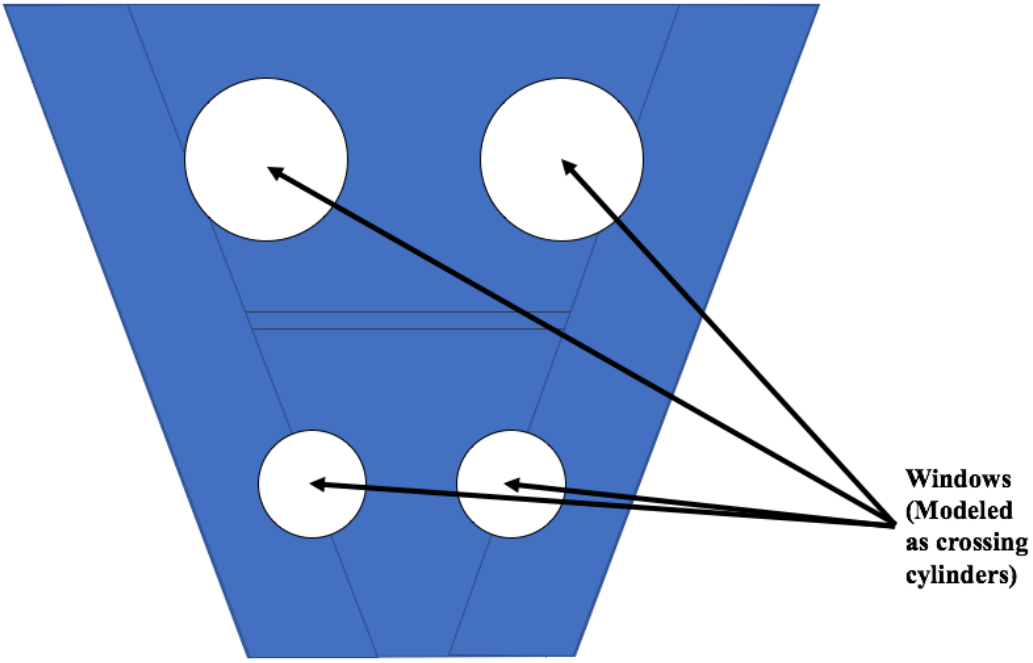

Figure 5. Passive Zone for Windows

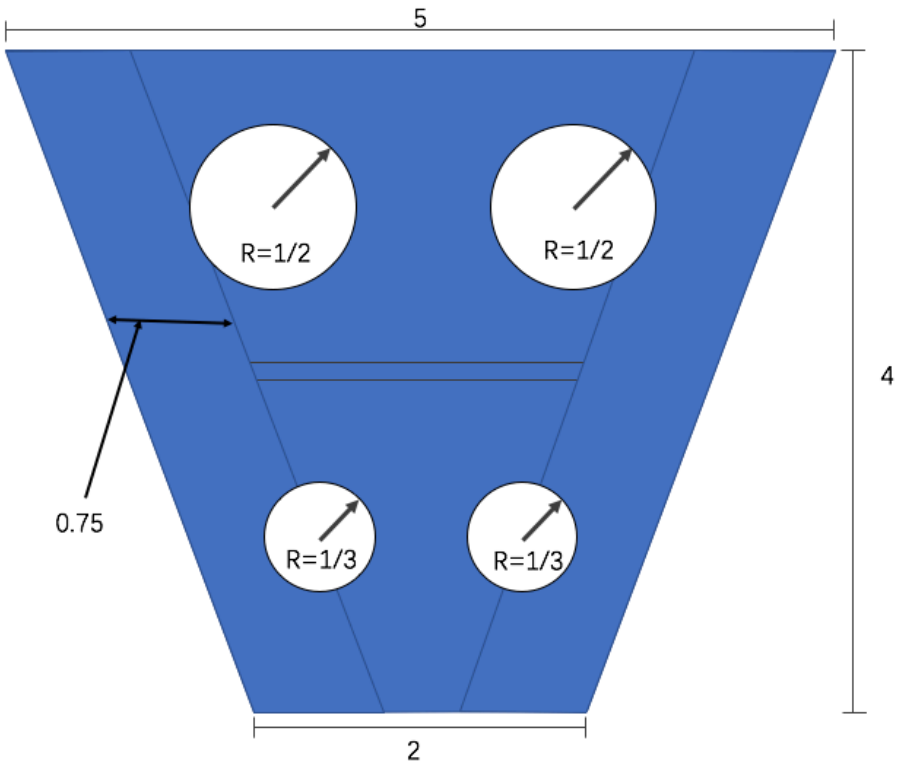

Figure 6. Structure Dimensions

2.3 Results and Comments

The final model is shown in Figure 10 to Figure 9. It confirms with the expected results that it will generate branches connected from the roof to the base as load paths. The tree type structure is referred as the optimal existing shape in transferring loads from large top to narrow bottom.

In addition, the branches in the bottom level is thicker than the second floor. This is also expected as the loads accumulated from the top to the base. As shown in Figure 7, the most direct way to transfer vertical compression loads to ground through walls. However, to remain the functionality of a real building, windows are added as passive zone. After applying passive regions for windows, the structure generates more branches and looks more like a tree, creating aesthetic value. The branches on top of each level collect the area loads acting on the floor and converging the loads into one single large column.

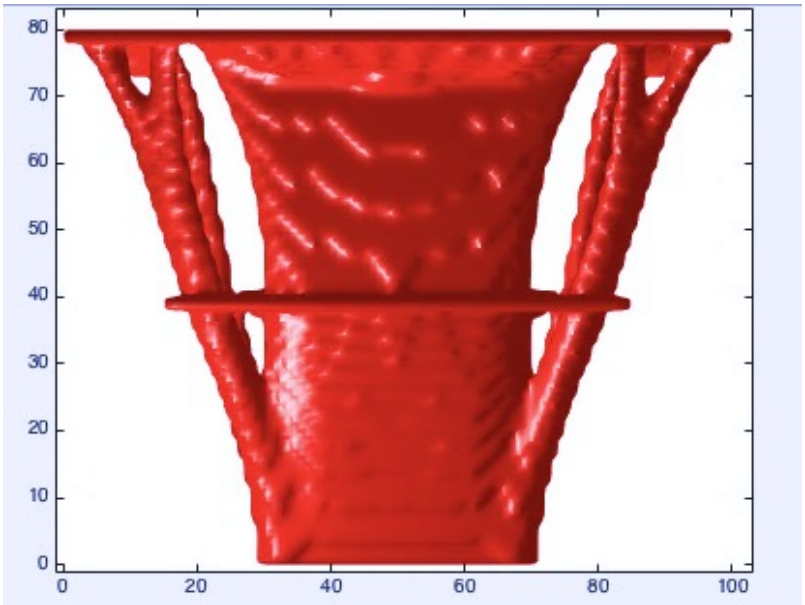

Figure 7. Intermediate Result (Without passive zone for windows)

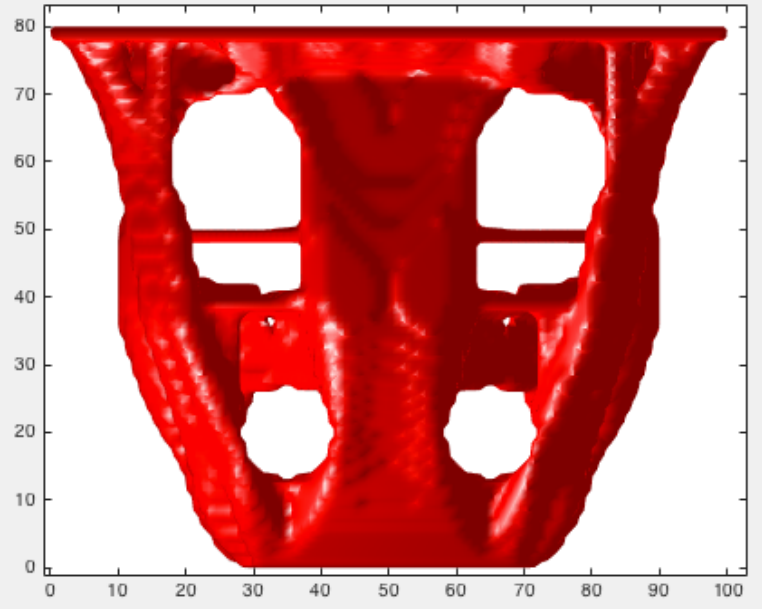

Figure 8.Optimization Result (Side View)

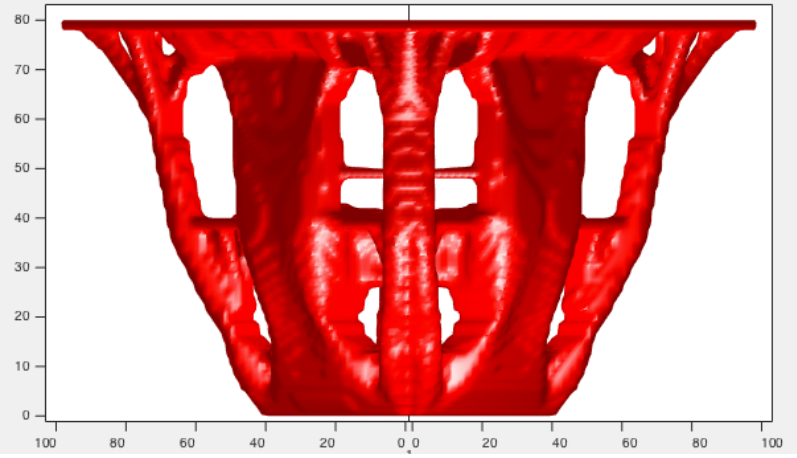

Figure 9.Optimization Result (Continue)

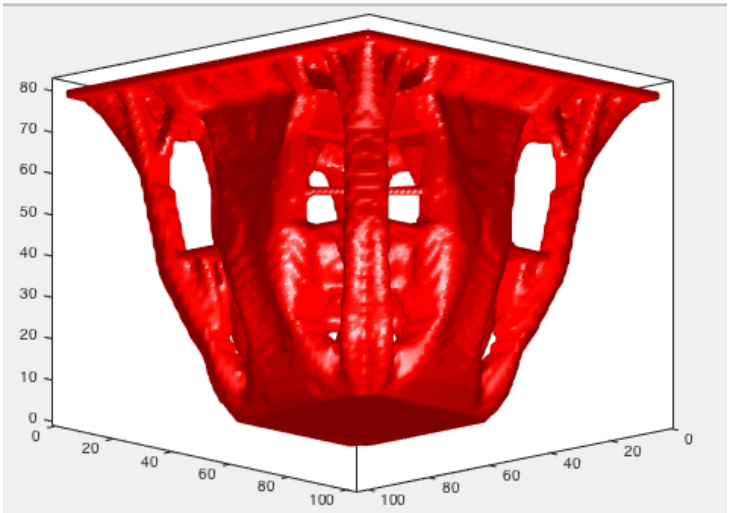

Figure 10. Optimization Result Views

Figure 11. Inspiration from Nature

Matlab Code:

```
%% CEE 307 SDO 2016 FALL Final Project
% Modified by: Lanxi Liu & Wenjin Situ
% Optimization goal: inverted two story structure
% Professor: Glaucio Paulino
% A 169 LINE 3D TOPOLOGY OPITMIZATION CODE BY LIU AND TOVAR (JUL 2013)
% --- MODIFIED BY TOMAS ZEGARD (JAN 2014)
close all;clear;clc
nelx = 50;
nely = 50;
nelz = 80;
volfrac = 0.15;
penal = 1;              % Initial penalization (see lines 127-135 for
continuation)
rmin = 2;               % Filter radius
Q = 2;                   % Filter exponent

% USER-DEFINED LOOP PARAMETERS
maxloop = 200;            % Maximum number of iterations
tolx = 0.00;              % Termination criterion (disabled for
continuation - default: 0.01)
displayflag = true;        % Display structure flag
plotcutoff = 0.50;         % Display density cutoff
storefileprefix = 'output'; % Filename prefix for storage
storeiters = false;        % Store data for each iteration
storefinal = true;         % Store final result

% USER-DEFINED MATERIAL PROPERTIES
E0 = 1;           % Young's modulus of solid material
Emin = 1e-9;       % Young's modulus of void-like material
nu = 0.3;         % Poisson's ratio
% USER-DEFINED LOAD DOFs
[Il,Jl] = meshgrid(1:nelx+1,1:nely+1);                % Coordinates
Fl1 = -ones(size(Il));
Fl1([1 end],:)=Fl1([1 end],:)/2;
Fl1(:,[1 end])=Fl1(:,[1 end])/2;
loadnid = nelz*(nelx+1)*(nely+1)+(Il-1)*(nely+1)+Jl;    % Node IDs, at
the top
loaddof1 = 3*loadnid(:);                              % DOFs
loadval1 = Fl1(:);
```

```
[Il,Jl] =
meshgrid(1:round(nelx*0.7)+1,1:round(nely*0.7)+1);                % Coordinates
Fl2 = -ones(size(Il));
Fl2([1 end],:)=Fl2([1 end],:)/2;
Fl2(:,[1 end])=Fl2(:,[1 end])/2;
loadnid = round(nelz/2)*(nelx+1)*(nely+1)+(Il-1)*(nely+1)+Jl;    % Node IDs, at the top
loaddof = [loaddof1;3*loadnid(:)];                                % DOFs
loadval = [loadval1;Fl2(:)];

% USER-DEFINED SUPPORT FIXED DOFs
[If,Jf] = meshgrid(1:round(0.4*nelx),1:round(0.4*nely)+1);         % Coordinates
fixednid1 = (If-1)*(nely+1)+Jf;                       % Node IDs %1/12 xy surface
[If,Kf] = meshgrid(1:nelx+1,1:nelz+1);                % Coordinates
fixednid2 = (If-1)*(nely+1)+(Kf-1)*(nelx+1)*(nely+1)+1; % Node IDs  %xz surface
[Jf,Kf] = meshgrid(1:nely+1,1:nelz+1);                % Coordinates
fixednid3 = (Kf-1)*(nelx+1)*(nely+1)+Jf;  % Node IDs %yz surface of end of x
fixeddof = [3*fixednid1(:); 3*fixednid1(:)-1; 3*fixednid1(:)-2; ...
    3*fixednid2(:)-1; 3*fixednid3(:)-2];       % DOFs -1 is y, -2 is x, nothing is z
% PREPARE FINITE ELEMENT ANALYSIS
nele = nelx*nely*nelz;
ndof = 3*(nelx+1)*(nely+1)*(nelz+1);
F = sparse(loaddof,1,loadval,ndof,1);
U = zeros(ndof,1);
freedofs = setdiff(1:ndof,fixeddof);
KE = lk_H8(nu);
nodegrd = reshape(1:(nely+1)*(nelx+1),nely+1,nelx+1);
nodeids = reshape(nodegrd(1:end-1,1:end-1),nely*nelx,1);
nodeidz = 0:(nely+1)*(nelx+1):(nelz-1)*(nely+1)*(nelx+1);
nodeids = repmat(nodeids,size(nodeidz))+repmat(nodeidz,size(nodeids));
edofVec = 3*nodeids(:)+1;
edofMat = repmat(edofVec,1,24)+ ...
    repmat([0 1 2 3*nely + [3 4 5 0 1 2] -3 -2 -1 ...
    3*(nely+1)*(nelx+1)+[0 1 2 3*nely + [3 4 5 0 1 2] -3 -2 -
```

```
1]],nele,1);
iK = kron(edofMat,ones(24,1))';
jK = kron(edofMat,ones(1,24))';
% HOUSEKEEPING
clear If Jf Kf Fl Il Jl fixednid1 fixednid2 fixednid3 nodegrd nodeidz
nodeids
% PREPARE FILTER
step = ceil(rmin)-1;
iH = zeros(nele*(2*step+1)^3,1);
jH = zeros(size(iH)); vH = zeros(size(iH));
n = 0;
for el=1:nele
    [i,j,k] = ind2sub([nely,nelx,nelz],el);
    [ispan,jspan,kspan] = meshgrid(max(1,i-step):min(nely,i+step),...
        max(1,j-step):min(nelx,j+step),max(1,k-step):min(nelz,k+step));
    dist = max(0,rmin-sqrt((ispan-i).^2 + (jspan-j).^2 + (kspan-
k).^2)).^Q;
    vH(n+(1:numel(dist))) = dist(:);
    iH(n+(1:numel(dist))) = el;
    jH(n+(1:numel(dist))) = sub2ind([nely nelx nelz],ispan,jspan,kspan);
    n = n + numel(dist);
end
iH(n+1:end)=[]; jH(n+1:end)=[]; vH(n+1:end)=[];
H = sparse(iH,jH,vH);
Hs = sum(H,2);
% HOUSEKEEPING
clear iH jH vH ispan jspan kspan dist

% DEFINE PASSIVE-SOLID ZONES
pass_solid = false(nelx,nely,nelz);
pass_solid(:,:,nelz-1:nelz)=1;
pass_solid(1:round(0.4*nelx),1:round(0.4*nely),1:2)=1;
pass_solid(1:round(0.7*nelx),1:round(0.7*nely),round(nelz/2)-
1:round(nelz/2))=1;

passive = false(nelx,nely,nelz);
for elz=1:nelz
    for ely = 1:nely
        for elx=1:nelx
            ratio_out = 0.4 + 0.6*(elz-1)/nelz; %ratio along z axis
            ratio_in = 0.1 + 0.6*(elz-1)/nelz;
```

```
            if elx > round(ratio_out *nelx) && ely >
round(ratio_out*nely)
                passive(elx,ely,elz)=1;
            end
            if elx < round(ratio_in*nelx) && ely<round(ratio_in*nely)
                if (elz~=1 && elz~=2 && elz~=round(nelz/2)-1 ...
                    && elz~=round(nelz/2) && elz~=nelz && elz~=nelz-1)
                passive(elx,ely,elz)=1;
                end
            end
        end
    end
end
for elx=1:nelx
    for elz = 1:nelz
        for ely=1:nely
            if sqrt(((elz + elz-1)/2-nelz/4.)^2+((ely + ely-1)/2-
nely*0.3)^2)<nelz/12
                passive(elx,ely,elz)=1;
            end
            if sqrt(((elz + elz-1)/2-3*nelz/4.)^2+((ely + ely-1)/2-
nely*0.45)^2)<nelz/8
                passive(elx,ely,elz)=1;
            end
        end
    end
end
for ely=1:nely
    for elz = 1:nelz
        for elx=1:nelx
            if sqrt(((elz + elz-1)/2-nelz/4.)^2+((elx + elx-1)/2-
nelx*0.3)^2)<nelz/12
                passive(elx,ely,elz)=1;
            end
            if sqrt(((elz + elz-1)/2-3*nelz/4.)^2+((elx + elx-1)/2-
nelx*0.45)^2)<nelz/8
                passive(elx,ely,elz)=1;
            end
        end
    end
end
```

```
df_solid = sum(sum(sum(pass_solid)))/(nelx*nely*nelz);  %passive ratio
 volfrac = volfrac + df_solid; % Adjust the volume fraction to consider
passive
% APPLY PASSIVE ZONES
x = repmat(volfrac,[nely,nelx,nelz]);
x((passive))=0; %0 is passive and void
x((pass_solid))=1; %1 is passive and solid


% HOUSEKEEPING
clear Ip Jp Kp


% PLOT DOMAIN AND BCS
plotDomainBCs(nelx,nely,nelz,loaddof,fixeddof,loadval) % Plot the
domain and BCs

% INITIALIZE ITERATION
xPhys = x;
loop = 0;
change = 1;
if displayflag, figure('Color','w'), end
fprintf('=== ITERATIONS BEGIN... ===\n')
% START ITERATION
while change > tolx && loop < maxloop
    if storeiters
        filename = sprintf('%s%03.0f.mat',storefileprefix,loop);
        save(filename,'xPhys','change','c','penal');
    end
    loop = loop+1;
    % FE-ANALYSIS
    sK = KE(:)*(Emin+xPhys(:)'.^penal*(E0-Emin));
    K = sparse(iK(:),jK(:),sK(:)); K = (K+K')/2;

    % OPTION 1: Direct solver (original)
    % U(freedofs,:) = K(freedofs,freedofs)\F(freedofs,:);
    % OPTION 2: Jacobi PCG (suggested by Liu & Tovar for large problems)
    M = diag(diag(K(freedofs,freedofs)));
    U(freedofs,:) = pcg(K(freedofs,freedofs),F(freedofs,:),1e-8,1000,M);
    % OPTION 3: Incomplete Cholesky PCG [fast but might fail]
```

```
%    L = ichol(K(freedofs,freedofs));
%    U(freedofs,:) = pcg(K(freedofs,freedofs),F(freedofs,:),1e-
8,2000,L,L');

    % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
    ce = reshape(sum((U(edofMat)*KE).*U(edofMat),2),[nely,nelx,nelz]);
    c = sum(sum(sum((Emin+xPhys.^penal*(E0-Emin)).*ce)));
    dc = -penal*(E0-Emin)*xPhys.^(penal-1).*ce;
    dv = ones(nely,nelx,nelz);
    % FILTERING AND MODIFICATION OF SENSITIVITIES
    dc(:) = H*(dc(:)./Hs);
    dv(:) = H*(dv(:)./Hs);
    % OPTIMALITY CRITERIA UPDATE
    if loop<round(maxloop/6),        l1 = 0.0;     l2 = 1e9;     move =
0.15;
    elseif loop<round(maxloop/3),    l1 = 0.0;     l2 = 1e9;     move =
0.15; penal = 1.5;
    elseif loop<round(maxloop/2),    l1 = 0.0;     l2 = 1e9;     move =
0.15; penal = 2.0;
    elseif loop<round(maxloop*2/3),  l1 = lmid/1.1; l2 = lmid*1.1; move
= 0.15; penal = 2.5;
    elseif loop<round(maxloop*3/4),  l1 = lmid/1.1; l2 = lmid*1.1; move
= 0.12; penal = 3.0;
    elseif loop<round(maxloop*5/6),  l1 = lmid/1.1; l2 = lmid*1.1; move
= 0.10; penal = 3.5;
    elseif loop<round(maxloop*11/12), l1 = lmid/1.1; l2 = lmid*1.1; move
= 0.08; penal = 4.0;
    else                          l1 = lmid/1.1; l2 = lmid*1.1; move =
0.04; penal = 4.25;
    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)))));
        xnew(pass_solid) = 1;
        xPhys(:) = (H*xnew(:))./Hs;
        if sum(xPhys(:)) > volfrac*nele, l1 = lmid; else l2 = lmid; end
    end
    change = max(abs(xnew(:)-x(:)));
    x = xnew;
    % PRINT RESULTS
```

```
    fprintf(' It.:%5i Obj.:%11.4f Vol.:%7.3f
ch.:%7.3f\n',loop,c,mean(xPhys(:)),change);
    % PLOT DENSITIES
    if displayflag
        plotTOP3D(xPhys,plotcutoff);
        s = sprintf('Iteration = %03.0f     Penal = %.2f',loop,penal);
        title(s), drawnow
    end
end
if storefinal
    filename = sprintf('%s%03.0f.mat',storefileprefix,loop);
    save(filename,'xPhys','change','c','penal');
end
```