IIT Hyderabad

Assignment 3

Submitted by:
ME21BTECH11001 Abhishek Ghosh

ME5053: Soft Robotics
Mechanical Engineering
27.04.2025

Submitted to:
Dr. Prabhat Kumar

1 Question 1

MATLAB Code for HoneyTop90 with Projection Filter

```matlab
function honey_top90_projection_q1(HNex, HNey, volfrac, penal, rfill, ft)
%% ----------- Material Properties -----------
E0 = 1;
Emin = E0 * 1e-9;

%% ----------- Mesh Generation and DOF Assignment -----------
NstartVs = reshape(1:(1+2*HNex)*(1+HNey), 1+2*HNex, 1+HNey);
DOFstartVs = reshape(2*NstartVs(1:end-1,1:end-1)-1, 2*HNex*HNey, 1);
NodeDOFs = repmat(DOFstartVs, 1, 8) + repmat([2*(2*HNex+1)+[2 3 0 1] 0 1 2 3], 2*HNex*HNey, 1);

% Remove DOFs for edges based on honeycomb pattern
skip = (2*HNex:2*HNex:2*HNex*HNey)' + mod(1:HNey,2)';
ActualDOFs = NodeDOFs(setdiff(1:2*HNex*HNey, skip), :);
HoneyDOFs = [ActualDOFs(2:2:end,1:2), ActualDOFs(1:2:end,:), ActualDOFs(2:2:end,7:8)];

% Nodal coordinates
Ncyi = repmat(reshape(repmat([-0.25 0.25]', HNey+1, 1), 2, HNey+1) + ...
              reshape(1.5*sort(repmat((0:HNey)',2,1)), 2, HNey+1), HNex+1, 1);
Ncyi(:,1:2:end) = flip(Ncyi(:,1:2:end));
Ncyf = Ncyi(1:end-1,:);

HoneyNCO = [repmat((0:cos(pi/6):2*HNex*cos(pi/6)),1,HNey+1)', Ncyf(:)];

if mod(HNey,2) == 0
    HoneyDOFs(end:-1:end-HNex+2,1:6) = HoneyDOFs(end:-1:end-HNex+2,1:6) - 2;
    idx = (2*HNex+1)*(HNey+1);
    HoneyNCO([idx-1, idx], :) = [];
end

[Nelem, Nnode] = deal(size(HoneyDOFs,1), size(HoneyNCO,1));

%% ----------- Force and Boundary Conditions -----------
L = max(HoneyNCO(:,1));
F = sparse(2*Nnode,1);

target_x = [L/4, L/2, 3*L/4];
Fvalue = 1;
force_nodes = [];

for i = 1:length(target_x)
    [~, node_idx] = min(abs(HoneyNCO(:,1) - target_x(i)));
    force_nodes = [force_nodes; node_idx];
end

F(2*force_nodes) = -Fvalue;

% Vertical supports at x=0 and x=L
```

```
fixeddofs = 2*find(abs(HoneyNCO(:,1)) < 1e-6 | abs(HoneyNCO(:,1) - L) < 1e-6);
alldofs = 1:2*Nnode;
freedofs = setdiff(alldofs, fixeddofs);

%% ----------- Assembly Preparation -----------
iK = reshape(kron(HoneyDOFs, ones(12,1))', 144*Nelem, 1);
jK = reshape(kron(HoneyDOFs, ones(1,12))', 144*Nelem, 1);

KE = (E0/1000) * [
    616.43012  92.77147  -168.07333  65.54377  -232.28511 -0.00032 -120.65312 -83.28564 -71.60020 -92.77115 -23.81836  17.74187;
    92.77147  509.30685  101.02751  -71.90335  0.00032  -18.03857 -83.28564 -24.48314 -92.77179 -178.72347 -17.74187 -216.15832;
   -168.07333 101.02751  455.74522   0.00000 -168.07333 -101.02751 -71.60020 -92.77179  23.60185  -0.00000 -71.60020  92.77179;
    65.54377  -71.90335   0.00000   669.99176 -65.54377  -71.90335 -92.77115 -178.72347 -0.00000 -168.73811  92.77115 -178.72347;
   -232.28511  0.00032  -168.07333 -65.54377  616.43012 -92.77147 -23.81836 -17.74187 -71.60020  92.77115 -120.65312  83.28564;
   -0.00032  -18.03857 -101.02751 -71.90335 -92.77147  509.30685 17.74187 -216.15832  92.77179 -178.72347  83.28564 -24.48314;
   -120.65312 -83.28564 -71.60020 -92.77115 -23.81836  17.74187 616.43012  92.77147 -168.07333  65.54377 -232.28511 -0.00032;
   -83.28564 -24.48314 -92.77179 -178.72347 -17.74187 -216.15832 92.77147  509.30685 101.02751 -71.90335  0.00032  -18.03857;
   -71.60020 -92.77179  23.60185  -0.00000 -71.60020  92.77179 -168.07333  101.02751  455.74522  0.00000 -168.07333 -101.02751;
   -92.77115 -178.72347 -0.00000 -168.73811  92.77115 -178.72347 65.54377 -71.90335  0.00000  669.99176 -65.54377 -71.90335;
   -23.81836 -17.74187 -71.60020  92.77115 -120.65312  83.28564 -232.28511  0.00032 -168.07333 -65.54377  616.43012 -92.77147;
    17.74187 -216.15832  92.77179 -178.72347  83.28564 -24.48314 -0.00032 -18.03857 -101.02751 -71.90335 -92.77147  509.30685
];

%% ----------- Filter Preparation -----------
Cxx = repmat([sqrt(3)/2*(1:2:2*HNex-1), sqrt(3)*(1:HNex-1)], 1, ceil(HNey/2));
Cyy = repmat(3/4 + 3/2*(0:HNey-1), HNex, 1);
Cyy(HNex+1:2*HNex:end) = [];
ct = [Cxx(1:length(Cyy))', Cyy(:)];

DD = cell(Nelem,1);
for j = 1:Nelem
    dist = sqrt((ct(j,1) - ct(:,1)).^2 + (ct(j,2) - ct(:,2)).^2);
    idx = find(dist <= rfill);
    DD{j} = [idx, j*ones(size(idx)), dist(idx)];
end
DD = vertcat(DD{:});
HHs = sparse(DD(:,2), DD(:,1), 1 - DD(:,3)/rfill);
HHs = spdiags(1./sum(HHs,2), 0, Nelem, Nelem) * HHs;

%% ----------- Heaviside Projection Filter Parameters -----------
% Adding Heaviside projection filter parameters
beta = 1;         % Initial beta value
betamax = 128;    % Maximum beta value
eta = 0.5;        % Threshold parameter
```

```matlab

%% ----------- Initialization -----------
x = volfrac * ones(Nelem,1);
[loop, change, maxiter, dv, move] = deal(0, 1, 200, ones(Nelem,1), 0.2);

%% ----------- Initialize variables based on filter type -----------
if ft == 0 || ft == 1  % No filter or sensitivity filter
    xPhys = x;
elseif ft == 2         % Density filter
    xPhys = HHs * x;
elseif ft == 3         % Heaviside projection filter
    xTilde = HHs * x;  % First apply density filter
    % Then apply Heaviside projection
    xPhys = (tanh(beta*eta) + tanh(beta*(xTilde-eta))) ./ (tanh(beta*eta) + tanh(beta*(1-eta)));
end

%% ----------- Optimization Loop -----------
while (change > 0.01 && loop < maxiter)
    loop = loop + 1;

    % Finite Element Analysis
    sK = reshape(KE(:) * (Emin + xPhys'.^penal * (E0-Emin)), 144*Nelem, 1);
    K = sparse(iK, jK, sK);

    U = zeros(2*Nnode,1);
    U(freedofs) = decomposition(K(freedofs,freedofs),'chol','lower') \ F(freedofs);

    % Objective and Sensitivities
    ce = sum((U(HoneyDOFs) * KE) .* U(HoneyDOFs), 2);
    c = sum((Emin + xPhys.^penal * (E0-Emin)) .* ce);
    dc = -penal * (E0-Emin) * xPhys.^(penal-1) .* ce;

    % Apply appropriate filtering to sensitivities
    if ft == 0       % No filtering
        % Do nothing
    elseif ft == 1  % Sensitivity filtering
        dc = HHs' * (x.*dc) ./ max(1e-3, x);
    elseif ft == 2  % Density filtering
        dc = HHs' * dc;
        dv = HHs' * dv;
    elseif ft == 3  % Heaviside projection filter
        % Chain rule for sensitivities
        dH = beta * (1 - tanh(beta*(xTilde-eta)).^2) ./ (tanh(beta*eta) + tanh(beta*(1-eta)));
        dc = HHs' * (dc .* dH);
        dv = HHs' * (dv .* dH);
    end

    % Optimality Criteria Update
    xold = x;
    [xUpp, xLow] = deal(xold + move, xold - move);
    OcC = xold .* sqrt(-dc ./ dv);

    % Set limits for the Lagrange multiplier
```

```
    if ft == 3
        l1 = 0; l2 = 1e9;  % For projection filter, use wide range to avoid numerical issues
    else
        l1 = 0; l2 = max(OcC) / volfrac;
    end

    while (l2-l1)/(l2+l1) > 1e-3
        lmid = 0.5*(l2+l1);
        x = max(0, max(xLow, min(1, min(xUpp, OcC / lmid))));
        if mean(x) > volfrac
            l1 = lmid;
        else
            l2 = lmid;
        end
    end

    % Update physical variables based on filter type
    if ft == 0 || ft == 1
        xPhys = x;
    elseif ft == 2
        xPhys = HHs * x;
    elseif ft == 3
        xTilde = HHs * x;
        xPhys = (tanh(beta*eta) + tanh(beta*(xTilde-eta))) ./ (tanh(beta*eta) + tanh(beta*(1-eta)));
    end

    change = max(abs(x - xold));

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

    % Plot intermediate designs
    colormap('gray');
    scatter(ct(:,1), ct(:,2), [], 1-xPhys, 'filled');
    axis equal off;
    drawnow;

    % Update beta parameter (continuation approach)
    if ft == 3 && mod(loop, 60) == 0 && beta < betamax
        beta = 2 * beta;
        fprintf(' Beta updated to: %g\n', beta);
    end
end
end
```

Figure 1:

MATLAB Code for top88 with Helmhotz Filter

```
%%%% TOPOLOGY OPTIMIZATION WITH HELMHOLTZ FILTER FOR BEAM PROBLEM %%%%
function top88_helmholtz_q1(nelx,nely,volfrac,penalMax,rmin)
% input example: >> top88_beam(80,20,0.5,3,2)
% The beam has length L, with a = L/4

% Ensuring nelx is divisible by 4 for exact placement of supports
if mod(nelx, 4) ~= 0
    fprintf('Warning: nelx should be divisible by 4 for exact placement of supports\n');
    nelx = 4 * round(nelx/4);
    fprintf('Adjusted nelx to %d\n', nelx);
end

%% MATERIAL PROPERTIES
E0 = 1;
Emin = 1e-9;
nu = 0.3;
penal = 0.96;
%% 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 FOR BEAM PROBLEM
% Initialize force vector
F = sparse(2*(nely+1)*(nelx+1),1);

```

```
% Apply point loads at a=L/4, 2a=L/2, and 3a=3L/4
loadPositions = [nelx/4, nelx/2, 3*nelx/4];
for i = 1:length(loadPositions)
    F(2*nodenrs(1, loadPositions(i)+1)) = -1; % Negative y-direction
        with force = 1
end

U = zeros(2*(nely+1)*(nelx+1),1);

% Define supports:
% Left support at x=0 (pin)
leftSupportNode = nodenrs(nely+1, 1);
leftFixedDofs = [2*leftSupportNode-1, 2*leftSupportNode]; % Constrain
    both x and y

% Right support at x=L (roller)
rightSupportNode = nodenrs(nely+1, nelx+1);
rightFixedDofs = [2*rightSupportNode]; % Only vertical constraint for
    roller

% Combine fixed DOFs
fixeddofs = union(leftFixedDofs, rightFixedDofs);
alldofs = 1:2*(nely+1)*(nelx+1);
freedofs = setdiff(alldofs, fixeddofs);

%% PREPARE HELMHOLTZ FILTER
Rmin = rmin/2/sqrt(3); % Conversion between classical and PDE filter
    radius
% Define filter stiffness matrix
KEF = [
    2/3 -1/6 -1/3 -1/6
    -1/6 2/3 -1/6 -1/3
    -1/3 -1/6 2/3 -1/6
    -1/6 -1/3 -1/6 2/3];
KEF = Rmin^2*KEF + [
    4/9 1/9 1/9 1/9
    1/9 4/9 1/9 1/9
    1/9 1/9 4/9 1/9
    1/9 1/9 1/9 4/9]/4;
% Setup filter FE matrices
edofVecF = reshape(nodenrs(1:end-1,1:end-1),nelx*nely,1);
edofMatF = repmat(edofVecF,1,4) + repmat([0 nely+[1 2] 1],nelx*nely,1);
iKF = reshape(kron(edofMatF,ones(4,1))',16*nelx*nely,1);
jKF = reshape(kron(edofMatF,ones(1,4))',16*nelx*nely,1);
sKF = reshape(KEF(:)*ones(1,nelx*nely),16*nelx*nely,1);
KF = sparse(iKF,jKF,sKF);
LF = chol(KF,'lower'); % Cholesky factorization for efficient solving
iTF = reshape(edofMatF,4*nelx*nely,1);
jTF = reshape(repmat([1:nelx*nely],4,1)',4*nelx*nely,1);
sTF = repmat(1/4,4*nelx*nely,1);
TF = sparse(iTF,jTF,sTF);

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

```
while change > 0.01
  loop = loop + 1;
  penal = min(penalMax, penal + 0.04);
  %% FE-ANALYSIS
  sK = reshape(KE(:)*(Emin+xPhys(:)'.^penal*(E0-Emin)),64*nelx*nely,1);
  K = sparse(iK,jK,sK); K = (K+K')/2;
  tic; U(freedofs) = K(freedofs,freedofs)\F(freedofs); toc;
  %% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
  ce = reshape(sum((U(edofMat)*KE).*U(edofMat),2),nely,nelx);
  c = sum(sum((Emin+xPhys.^penal*(E0-Emin)).*ce));
  dc = -penal*(E0-Emin)*xPhys.^(penal-1).*ce;
  dv = ones(nely,nelx);

  %% HELMHOLTZ FILTERING OF SENSITIVITIES
  dc(:) = TF'*(LF'\(LF\(TF*dc(:))));
  dv(:) = TF'*(LF'\(LF\(TF*dv(:))));

  %% OPTIMALITY CRITERIA UPDATE OF DESIGN VARIABLES AND PHYSICAL
     DENSITIES
  l1 = 0; l2 = 1e9; move = 0.2;
  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)))));

    %% HELMHOLTZ FILTERING OF DENSITIES
    xPhys(:) = TF'*(LF'\(LF\(TF*xnew(:))));

    if sum(xPhys(:)) > volfrac*nelx*nely, 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
  colormap(gray); imagesc(1-xPhys); caxis([0 1]); axis equal; axis off;
      drawnow;
end
```

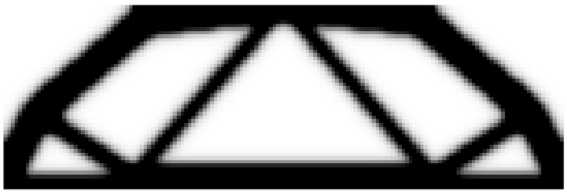

Figure 2:

2 Question 2

MATLAB Code for HoneyTop90 with Projection Filter

```
function honey_top90_projection_Q2(HNex, HNey, volfrac, penal, rfill,
    ft)
%% ----------- Material Properties -----------
E0 = 1;
Emin = E0 * 1e-9;

%% ----------- Mesh Generation and DOF Assignment -----------
NstartVs = reshape(1:(1+2*HNex)*(1+HNey), 1+2*HNex, 1+HNey);
DOFstartVs = reshape(2*NstartVs(1:end-1,1:end-1)-1, 2*HNex*HNey, 1);
NodeDOFs = repmat(DOFstartVs, 1, 8) + repmat([2*(2*HNex+1)+[2 3 0 1] 0
    1 2 3], 2*HNex*HNey, 1);

% Remove DOFs for edges based on honeycomb pattern
skip = (2*HNex:2*HNex:2*HNex*HNey)' + mod(1:HNey,2)';
ActualDOFs = NodeDOFs(setdiff(1:2*HNex*HNey, skip), :);
HoneyDOFs = [ActualDOFs(2:2:end,1:2), ActualDOFs(1:2:end,:), ActualDOFs
    (2:2:end,7:8)];

% Nodal coordinates
Ncyi = repmat(reshape(repmat([-0.25 0.25]', HNey+1, 1), 2, HNey+1) +
    ...
            reshape(1.5*sort(repmat((0:HNey)',2,1)), 2, HNey+1), HNex
    +1, 1);
Ncyi(:,1:2:end) = flip(Ncyi(:,1:2:end));
Ncyf = Ncyi(1:end-1,:);

HoneyNCO = [repmat((0:cos(pi/6):2*HNex*cos(pi/6)),1,HNey+1)', Ncyf(:)];

if mod(HNey,2) == 0
    HoneyDOFs(end:-1:end-HNex+2,1:6) = HoneyDOFs(end:-1:end-HNex+2,1:6)
        - 2;
    idx = (2*HNex+1)*(HNey+1);
```

```
    HoneyNCO([idx-1, idx], :) = [];
end

[Nelem, Nnode] = deal(size(HoneyDOFs,1), size(HoneyNCO,1));

%% ----------- Force and Boundary Conditions -----------
L = max(HoneyNCO(:,1));
F = sparse(2*Nnode,1);

% Apply force F =  2  at 45  angle at the top-right corner
F_magnitude = sqrt(2);
F_x = F_magnitude * cos(pi/4);  % F*cos(45  )
F_y = F_magnitude * sin(pi/4);  % F*sin(45  )

% Find top-right node (maximum x, minimum y)
maxX = max(HoneyNCO(:,1));
minY = min(HoneyNCO(:,2));
[~, topRightNodeIdx] = min(sum([(HoneyNCO(:,1) - maxX).^2, (HoneyNCO(:,2) - minY).^2], 2));

F(2*topRightNodeIdx-1) = F_x;     % Horizontal component (positive x)
F(2*topRightNodeIdx) = -F_y;      % Vertical component (negative y)

% Fix all DOFs at the left end (cantilever beam)
leftNodes = find(abs(HoneyNCO(:,1)) < 1e-6);
fixeddofs = [2*leftNodes-1; 2*leftNodes];
fixeddofs = reshape(fixeddofs, [], 1);

alldofs = 1:2*Nnode;
freedofs = setdiff(alldofs, fixeddofs);

%% ----------- Assembly Preparation -----------
iK = reshape(kron(HoneyDOFs, ones(12,1))', 144*Nelem, 1);
jK = reshape(kron(HoneyDOFs, ones(1,12))', 144*Nelem, 1);

KE = (E0/1000) * [
    616.43012  92.77147  -168.07333  65.54377  -232.28511 -0.00032 -120.65312 -83.28564 -71.60020 -92.77115 -23.81836  17.74187;
    92.77147  509.30685  101.02751  -71.90335  0.00032  -18.03857 -83.28564 -24.48314 -92.77179 -178.72347 -17.74187 -216.15832;
   -168.07333 101.02751  455.74522   0.00000 -168.07333 -101.02751 -71.60020 -92.77179  23.60185  -0.00000 -71.60020  92.77179;
    65.54377  -71.90335   0.00000   669.99176 -65.54377  -71.90335 -92.77115 -178.72347 -0.00000 -168.73811  92.77115 -178.72347;
   -232.28511  0.00032  -168.07333 -65.54377  616.43012 -92.77147 -23.81836 -17.74187 -71.60020  92.77115 -120.65312  83.28564;
   -0.00032  -18.03857 -101.02751 -71.90335 -92.77147  509.30685 17.74187 -216.15832  92.77179 -178.72347  83.28564 -24.48314;
   -120.65312 -83.28564 -71.60020 -92.77115 -23.81836  17.74187 616.43012  92.77147 -168.07333  65.54377 -232.28511 -0.00032;
   -83.28564 -24.48314 -92.77179 -178.72347 -17.74187 -216.15832 92.77147  509.30685 101.02751 -71.90335  0.00032  -18.03857;
   -71.60020 -92.77179  23.60185  -0.00000 -71.60020  92.77179 -168.07333  101.02751  455.74522  0.00000 -168.07333 -101.02751;
   -92.77115 -178.72347 -0.00000 -168.73811  92.77115 -178.72347 65.54377 -71.90335  0.00000  669.99176 -65.54377 -71.90335;
   -23.81836 -17.74187 -71.60020  92.77115 -120.65312  83.28564 -232.28511  0.00032 -168.07333 -65.54377  616.43012 -92.77147;
```

```
    17.74187 -216.15832  92.77179 -178.72347  83.28564 -24.48314
        -0.00032 -18.03857 -101.02751 -71.90335 -92.77147  509.30685
];

%% ----------- Filter Preparation -----------
Cxx = repmat([sqrt(3)/2*(1:2:2*HNex-1), sqrt(3)*(1:HNex-1)], 1, ceil(
    HNey/2));
Cyy = repmat(3/4 + 3/2*(0:HNey-1), HNex, 1);
Cyy(HNex+1:2*HNex:end) = [];
ct = [Cxx(1:length(Cyy))', Cyy(:)];

DD = cell(Nelem,1);
for j = 1:Nelem
    dist = sqrt((ct(j,1) - ct(:,1)).^2 + (ct(j,2) - ct(:,2)).^2);
    idx = find(dist <= rfill);
    DD{j} = [idx, j*ones(size(idx)), dist(idx)];
end
DD = vertcat(DD{:});
HHs = sparse(DD(:,2), DD(:,1), 1 - DD(:,3)/rfill);
HHs = spdiags(1./sum(HHs,2), 0, Nelem, Nelem) * HHs;

%% ----------- Heaviside Projection Filter Parameters -----------
% Adding Heaviside projection filter parameters
beta = 1;         % Initial beta value
betamax = 128;    % Maximum beta value
eta = 0.5;        % Threshold parameter

%% ----------- Initialization -----------
x = volfrac * ones(Nelem,1);
[loop, change, maxiter, dv, move] = deal(0, 1, 200, ones(Nelem,1), 0.2)
    ;

%% ----------- Initialize variables based on filter type -----------
if ft == 0 || ft == 1  % No filter or sensitivity filter
    xPhys = x;
elseif ft == 2          % Density filter
    xPhys = HHs * x;
elseif ft == 3          % Heaviside projection filter
    xTilde = HHs * x;  % First apply density filter
    % Then apply Heaviside projection
    xPhys = (tanh(beta*eta) + tanh(beta*(xTilde-eta))) ./ (tanh(beta*
        eta) + tanh(beta*(1-eta)));
end

%% ----------- Optimization Loop -----------
while (change > 0.01 && loop < maxiter)
    loop = loop + 1;

    % Finite Element Analysis
    sK = reshape(KE(:) * (Emin + xPhys'.^penal * (E0-Emin)), 144*Nelem,
        1);
    K = sparse(iK, jK, sK);

    U = zeros(2*Nnode,1);
    U(freedofs) = decomposition(K(freedofs,freedofs),'chol','lower') \
        F(freedofs);

    % Objective and Sensitivities
```

```
    ce = sum((U(HoneyDOFs) * KE) .* U(HoneyDOFs), 2);
    c = sum((Emin + xPhys.^penal * (E0-Emin)) .* ce);
    dc = -penal * (E0-Emin) * xPhys.^(penal-1) .* ce;

    % Apply appropriate filtering to sensitivities
    if ft == 0      % No filtering
        % Do nothing
    elseif ft == 1  % Sensitivity filtering
        dc = HHs' * (x.*dc) ./ max(1e-3, x);
    elseif ft == 2  % Density filtering
        dc = HHs' * dc;
        dv = HHs' * dv;
    elseif ft == 3  % Heaviside projection filter
        % Chain rule for sensitivities
        dH = beta * (1 - tanh(beta*(xTilde-eta)).^2) ./ (tanh(beta*eta)
            + tanh(beta*(1-eta)));
        dc = HHs' * (dc .* dH);
        dv = HHs' * (dv .* dH);
    end

    % Optimality Criteria Update
    xold = x;
    [xUpp, xLow] = deal(xold + move, xold - move);
    OcC = xold .* sqrt(-dc ./ dv);

    % Set limits for the Lagrange multiplier
    if ft == 3
        l1 = 0; l2 = 1e9;  % For projection filter, use wide range to
           avoid numerical issues
    else
        l1 = 0; l2 = max(OcC) / volfrac;
    end

    while (l2-l1)/(l2+l1) > 1e-3
        lmid = 0.5*(l2+l1);
        x = max(0, max(xLow, min(1, min(xUpp, OcC / lmid))));
        if mean(x) > volfrac
            l1 = lmid;
        else
            l2 = lmid;
        end
    end

    % Update physical variables based on filter type
    if ft == 0 || ft == 1
        xPhys = x;
    elseif ft == 2
        xPhys = HHs * x;
    elseif ft == 3
        xTilde = HHs * x;
        xPhys = (tanh(beta*eta) + tanh(beta*(xTilde-eta))) ./ (tanh(
           beta*eta) + tanh(beta*(1-eta)));
    end

    change = max(abs(x - xold));

    % Print Results
```

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

    % Plot intermediate designs
    colormap('gray');
    scatter(ct(:,1), ct(:,2), [], 1-xPhys, 'filled');
    axis equal off;
    drawnow;

    % Update beta parameter (continuation approach)
    if ft == 3 && mod(loop, 60) == 0 && beta < betamax
        beta = 2 * beta;
        fprintf(' Beta updated to: %g\n', beta);
    end
end
end
```

Figure 3:

MATLAB Code for top88 with Helmholtz Filter

```
%%%% TOPOLOGY OPTIMIZATION WITH HELMHOLTZ FILTER FOR CANTILEVER BEAM
    %%%%
function top88_helmholtz_Q2(nelx,nely,volfrac,penalMax,rmin)
% input example: >> top88_cantilever(80,20,0.5,3,2)
% Cantilever beam with angled force at the top-right corner

%% MATERIAL PROPERTIES
E0 = 1;
Emin = 1e-9;
nu = 0.3;
penal = 0.96;
%% 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 FOR CANTILEVER BEAM PROBLEM
% Initialize force vector
F = sparse(2*(nely+1)*(nelx+1),1);

% Calculate the force components (F =   12   at 45   )
F_magnitude = sqrt(12);
F_x = F_magnitude * cos(pi/4);   % F*cos(45   )
F_y = F_magnitude * sin(pi/4);   % F*sin(45   )

% Apply force at the top-right corner
topRightNode = nodenrs(1, nelx+1);
F(2*topRightNode-1) = F_x;       % Horizontal component (positive x)
F(2*topRightNode) = -F_y;        % Vertical component (negative y)

U = zeros(2*(nely+1)*(nelx+1),1);

% Define supports: Fixed at left end (cantilever)
fixeddofs = [];
for i = 1:nely+1
    leftNode = nodenrs(i, 1);
    fixeddofs = [fixeddofs, 2*leftNode-1, 2*leftNode]; % Fix both x and
        y DOFs
end

alldofs = 1:2*(nely+1)*(nelx+1);
freedofs = setdiff(alldofs, fixeddofs);

%% PREPARE HELMHOLTZ FILTER
Rmin = rmin/2/sqrt(3); % Conversion between classical and PDE filter
    radius
% Define filter stiffness matrix
KEF = [
    2/3 -1/6 -1/3 -1/6
    -1/6 2/3 -1/6 -1/3
    -1/3 -1/6 2/3 -1/6
    -1/6 -1/3 -1/6 2/3];
KEF = Rmin^2*KEF + [
    4/9 1/9 1/9 1/9
    1/9 4/9 1/9 1/9
    1/9 1/9 4/9 1/9
    1/9 1/9 1/9 4/9]/4;
% Setup filter FE matrices
edofVecF = reshape(nodenrs(1:end-1,1:end-1),nelx*nely,1);
edofMatF = repmat(edofVecF,1,4) + repmat([0 nely+[1 2] 1],nelx*nely,1);
iKF = reshape(kron(edofMatF,ones(4,1))',16*nelx*nely,1);
jKF = reshape(kron(edofMatF,ones(1,4))',16*nelx*nely,1);
sKF = reshape(KEF(:)*ones(1,nelx*nely),16*nelx*nely,1);
KF = sparse(iKF,jKF,sKF);
LF = chol(KF,'lower'); % Cholesky factorization for efficient solving
iTF = reshape(edofMatF,4*nelx*nely,1);
jTF = reshape(repmat([1:nelx*nely],4,1)',4*nelx*nely,1);
```

```
sTF = repmat(1/4,4*nelx*nely,1);
TF = sparse(iTF,jTF,sTF);

%% INITIALIZE ITERATION
x = repmat(volfrac,nely,nelx);
xPhys = x;
loop = 0;
change = 1;
%% START ITERATION
while change > 0.01
  loop = loop + 1;
  penal = min(penalMax, penal + 0.04);
  %% FE-ANALYSIS
  sK = reshape(KE(:)*(Emin+xPhys(:)'.^penal*(E0-Emin)),64*nelx*nely,1);
  K = sparse(iK,jK,sK); K = (K+K')/2;
  tic; U(freedofs) = K(freedofs,freedofs)\F(freedofs); toc;
  %% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
  ce = reshape(sum((U(edofMat)*KE).*U(edofMat),2),nely,nelx);
  c = sum(sum((Emin+xPhys.^penal*(E0-Emin)).*ce));
  dc = -penal*(E0-Emin)*xPhys.^(penal-1).*ce;
  dv = ones(nely,nelx);

  %% HELMHOLTZ FILTERING OF SENSITIVITIES
  dc(:) = TF'*(LF'\(LF\(TF*dc(:))));
  dv(:) = TF'*(LF'\(LF\(TF*dv(:))));

  %% OPTIMALITY CRITERIA UPDATE OF DESIGN VARIABLES AND PHYSICAL DENSITIES
  l1 = 0; l2 = 1e9; move = 0.2;
  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)))));

    %% HELMHOLTZ FILTERING OF DENSITIES
    xPhys(:) = TF'*(LF'\(LF\(TF*xnew(:))));

    if sum(xPhys(:)) > volfrac*nelx*nely, 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
  colormap(gray); imagesc(1-xPhys); caxis([0 1]); axis equal; axis off; drawnow;
end
```

Figure 4:

3 Question 3

MATLAB Code for HoneyTop90 with Projection Filter

```matlab
function honey_top90_projection_Q3(HNex, HNey, volfrac, penal, rfill, ft)
%% ----------- Material Properties -----------
E0 = 1;
Emin = E0 * 1e-9;

%% ----------- Mesh Generation and DOF Assignment -----------
NstartVs = reshape(1:(1+2*HNex)*(1+HNey), 1+2*HNex, 1+HNey);
DOFstartVs = reshape(2*NstartVs(1:end-1,1:end-1)-1, 2*HNex*HNey, 1);
NodeDOFs = repmat(DOFstartVs, 1, 8) + repmat([2*(2*HNex+1)+[2 3 0 1] 0 1 2 3], 2*HNex*HNey, 1);

skip = (2*HNex:2*HNex:2*HNex*HNey)' + mod(1:HNey,2)';
ActualDOFs = NodeDOFs(setdiff(1:2*HNex*HNey, skip), :);
HoneyDOFs = [ActualDOFs(2:2:end,1:2), ActualDOFs(1:2:end,:), ActualDOFs(2:2:end,7:8)];

Ncyi = repmat(reshape(repmat([-0.25 0.25]', HNey+1, 1), 2, HNey+1) + ...
            reshape(1.5*sort(repmat((0:HNey)',2,1)), 2, HNey+1), HNex+1, 1);
Ncyi(:,1:2:end) = flip(Ncyi(:,1:2:end));
Ncyf = Ncyi(1:end-1,:);

HoneyNCO = [repmat((0:cos(pi/6):2*HNex*cos(pi/6)),1,HNey+1)', Ncyf(:)];

if mod(HNey,2) == 0
    HoneyDOFs(end:-1:end-HNex+2,1:6) = HoneyDOFs(end:-1:end-HNex+2,1:6) - 2;
    idx = (2*HNex+1)*(HNey+1);
    HoneyNCO([idx-1, idx], :) = [];
end
```

```
[Nelem, Nnode] = deal(size(HoneyDOFs,1), size(HoneyNCO,1));

%% ----------- Force and Boundary Conditions (Uniformly Distributed Simply Supported Beam) -----------
L = max(HoneyNCO(:,1));
F = sparse(2*Nnode,1);

% Apply uniform downward load
totalLoad = -1;  % Total load (negative = downward)
topNodes = find(abs(HoneyNCO(:,2) - max(HoneyNCO(:,2))) < 1e-6);
loadPerNode = totalLoad / length(topNodes);

F(2*topNodes) = F(2*topNodes) + loadPerNode;

% Find left and right nodes
leftNodes = find(abs(HoneyNCO(:,1)) < 1e-6);
rightNodes = find(abs(HoneyNCO(:,1) - max(HoneyNCO(:,1))) < 1e-6);

% Fix vertical DOFs at left and right nodes
fixeddofs = [2*leftNodes; 2*rightNodes];

% Additionally, fix horizontal movement at one left node
[~, idxMin] = min(HoneyNCO(:,1));
fixeddofs = [fixeddofs; 2*idxMin-1];

alldofs = 1:2*Nnode;
freedofs = setdiff(alldofs, fixeddofs);

%% ----------- Assembly Preparation -----------
iK = reshape(kron(HoneyDOFs, ones(12,1))', 144*Nelem, 1);
jK = reshape(kron(HoneyDOFs, ones(1,12))', 144*Nelem, 1);

KE = (E0/1000) * [
    616.43012  92.77147  -168.07333  65.54377  -232.28511 -0.00032  -120.65312 -83.28564 -71.60020 -92.77115 -23.81836  17.74187;
    92.77147  509.30685  101.02751  -71.90335  0.00032  -18.03857  -83.28564 -24.48314 -92.77179 -178.72347 -17.74187 -216.15832;
   -168.07333 101.02751  455.74522   0.00000 -168.07333 -101.02751  -71.60020 -92.77179  23.60185  -0.00000 -71.60020  92.77179;
    65.54377  -71.90335   0.00000   669.99176 -65.54377  -71.90335  -92.77115 -178.72347 -0.00000 -168.73811  92.77115 -178.72347;
   -232.28511  0.00032  -168.07333 -65.54377  616.43012 -92.77147  -23.81836 -17.74187 -71.60020  92.77115 -120.65312  83.28564;
   -0.00032  -18.03857 -101.02751 -71.90335 -92.77147  509.30685  17.74187 -216.15832  92.77179 -178.72347  83.28564 -24.48314;
   -120.65312 -83.28564 -71.60020 -92.77115 -23.81836  17.74187  616.43012  92.77147 -168.07333  65.54377 -232.28511 -0.00032;
   -83.28564 -24.48314 -92.77179 -178.72347 -17.74187 -216.15832  92.77147  509.30685 101.02751 -71.90335  0.00032  -18.03857;
   -71.60020 -92.77179  23.60185  -0.00000 -71.60020  92.77179  -168.07333  101.02751  455.74522  0.00000 -168.07333 -101.02751;
   -92.77115 -178.72347 -0.00000 -168.73811  92.77115 -178.72347  65.54377 -71.90335  0.00000  669.99176 -65.54377 -71.90335;
   -23.81836 -17.74187 -71.60020  92.77115 -120.65312  83.28564  -232.28511  0.00032 -168.07333 -65.54377  616.43012 -92.77147;
    17.74187 -216.15832  92.77179 -178.72347  83.28564 -24.48314  -0.00032 -18.03857 -101.02751 -71.90335 -92.77147  509.30685
```

```
];

%% ----------- Filter Preparation -----------
Cxx = repmat([sqrt(3)/2*(1:2:2*HNex-1), sqrt(3)*(1:HNex-1)], 1, ceil(
    HNey/2));
Cyy = repmat(3/4 + 3/2*(0:HNey-1), HNex, 1);
Cyy(HNex+1:2*HNex:end)*
```

Figure 5:

MATLAB Code for top88 with Helmhotz Filter

```
%%%% TOPOLOGY OPTIMIZATION WITH HELMHOLTZ FILTER FOR CANTILEVER BEAM
    %%%%
function top88_helmholtz_Q3(nelx, nely, volfrac, penalMax, rmin)
% Example usage: >> top88_helmholtz_Q2(80, 20, 0.5, 3, 2)
% Cantilever beam with angled force at top-right corner.

%% 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
penal = 0.96;       % Starting penalization factor

%% PREPARE FINITE ELEMENT ANALYSIS
KE = element_stiffness_matrix(nu);
[nodenrs, edofMat, iK, jK] = prepare_fea(nelx, nely);

%% DEFINE LOADS AND SUPPORTS
[F, freedofs] = define_loads_supports(nelx, nely, nodenrs);

%% PREPARE HELMHOLTZ FILTER
[LF, TF] = prepare_helmholtz_filter(nelx, nely, nodenrs, rmin);

%% INITIALIZE ITERATION
x = repmat(volfrac, nely, nelx);
xPhys = x;
loop = 0;
change = 1;

%% ITERATIVE OPTIMIZATION LOOP
while change > 0.01
    loop = loop + 1;
    penal = min(penalMax, penal + 0.04);

    % FE-Analysis
    U = finite_element_analysis(xPhys, KE, iK, jK, freedofs, Emin, E0,
       penal);
```

```
    % Objective function and sensitivity
    [c, dc] = objective_and_sensitivity(xPhys, U, KE, nelx, nely, Emin, E0, penal);

    % Filtering sensitivities
    dc(:) = TF' * (LF' \ (LF \ (TF * dc(:))));

    % Optimality criteria update
    [xnew, xPhys] = optimality_criteria(x, dc, TF, LF, volfrac, nelx, nely);

    % Compute change
    change = max(abs(xnew(:) - x(:)));
    x = xnew;

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

    % Plot densities
    plot_densities(xPhys);
end
end

%% SUBFUNCTIONS

function KE = element_stiffness_matrix(nu)
    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]);
end

function [nodenrs, edofMat, iK, jK] = prepare_fea(nelx, nely)
    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);
end

function [F, freedofs] = define_loads_supports(nelx, nely, nodenrs)
    ndof = 2*(nelx+1)*(nely+1);
    F = sparse(ndof, 1);

    F_magnitude = sqrt(12);
    F_x = F_magnitude * cos(pi/4);
    F_y = F_magnitude * sin(pi/4);

    topRightNode = nodenrs(1, nelx+1);
    F(2*topRightNode-1) = F_x;
    F(2*topRightNode) = -F_y;

    fixeddofs = [];
```

```
    for i = 1:nely+1
        leftNode = nodenrs(i, 1);
        fixeddofs = [fixeddofs, 2*leftNode-1, 2*leftNode];
    end

    alldofs = 1:ndof;
    freedofs = setdiff(alldofs, fixeddofs);
end

function [LF, TF] = prepare_helmholtz_filter(nelx, nely, nodenrs, rmin)
    Rmin = rmin / (2*sqrt(3));
    KEF = [2/3 -1/6 -1/3 -1/6;
           -1/6 2/3 -1/6 -1/3;
           -1/3 -1/6 2/3 -1/6;
           -1/6 -1/3 -1/6 2/3];
    KEF = Rmin^2 * KEF + [4/9 1/9 1/9 1/9;
                          1/9 4/9 1/9 1/9;
                          1/9 1/9 4/9 1/9;
                          1/9 1/9 1/9 4/9]/4;
    edofVecF = reshape(nodenrs(1:end-1,1:end-1), nelx*nely, 1);
    edofMatF = repmat(edofVecF, 1, 4) + repmat([0 nely+[1 2] 1], nelx*nely, 1);
    iKF = reshape(kron(edofMatF, ones(4,1))', 16*nelx*nely, 1);
    jKF = reshape(kron(edofMatF, ones(1,4))', 16*nelx*nely, 1);
    sKF = reshape(KEF(:) * ones(1, nelx*nely), 16*nelx*nely, 1);
    KF = sparse(iKF, jKF, sKF);
    LF = chol(KF, 'lower');

    iTF = reshape(edofMatF, 4*nelx*nely, 1);
    jTF = reshape(repmat(1:nelx*nely, 4, 1)', 4*nelx*nely, 1);
    sTF = repmat(1/4, 4*nelx*nely, 1);
    TF = sparse(iTF, jTF, sTF);
end

function U = finite_element_analysis(xPhys, KE, iK, jK, freedofs, Emin, E0, penal)
    sK = reshape(KE(:) * (Emin + xPhys(:)'.^penal * (E0 - Emin)), 64*numel(xPhys), 1);
    K = sparse(iK, jK, sK);
    K = (K + K')/2;
    F = evalin('caller', 'F');
    U = zeros(length(F), 1);
    U(freedofs) = K(freedofs, freedofs) \ F(freedofs);
end

function [c, dc] = objective_and_sensitivity(xPhys, U, KE, nelx, nely, Emin, E0, penal)
    edofMat = evalin('caller', 'edofMat');
    ce = reshape(sum((U(edofMat) * KE) .* U(edofMat), 2), nely, nelx);
    c = sum(sum((Emin + xPhys.^penal * (E0 - Emin)) .* ce));
    dc = -penal * (E0 - Emin) * xPhys.^(penal-1) .* ce;
end

function [xnew, xPhys] = optimality_criteria(x, dc, TF, LF, volfrac, nelx, nely)
    l1 = 0; l2 = 1e9; move = 0.2;
    dv = ones(size(x));
    dv(:) = TF' * (LF' \ (LF \ (TF * dv(:))));
```

```
    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)))));
        xPhys(:) = TF' * (LF' \ (LF \ (TF * xnew(:))));
        if sum(xPhys(:)) > volfrac * nelx * nely
            l1 = lmid;
        else
            l2 = lmid;
        end
    end
end

function plot_densities(xPhys)
    colormap(gray);
    imagesc(1 - xPhys);
    caxis([0 1]);
    axis equal;
    axis off;
    drawnow;
end
```

Figure 6:

4 Question 4

MATLAB Code for top88 with Helmhotz Filter

```
% code
function top88(nelx, nely, volfrac, penal, rmin, ft)
    % MATERIAL PROPERTIES
    E0 = 1;
    Emin = 1e-9;
    nu = 0.3;
    % 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*(nely+1)*nelx+2,2*(nely+1)*(nelx+1)], ...
[1 2],[1-1],2*(nely+1)*(nelx+1),2);
U = zeros(2*(nely+1)*(nelx+1),2);
fixeddofs = [1:2*nely+1];
alldofs = [1:2*(nely+1)*(nelx+1)];
freedofs = setdiff(alldofs,fixeddofs);
% PREPARE FILTER
Rmin = rmin/2/sqrt(3);
KEF = Rmin^2*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4]/6 + ...
[4 2 1 2; 2 4 2 1; ...
1 2 4 2; 2 1 2 4]/36;
edofVecF = reshape(nodenrs(1:end-1,1:end-1),nelx*nely,1);
edofMatF = repmat(edofVecF,1,4) + ...
repmat([0 nely+[1:2] 1],    nelx*nely,1);
iKF = reshape(kron(edofMatF,ones(4,1))',    16*nelx*nely,1);
jKF = reshape(kron(edofMatF,ones(1,4))',    16*nelx*nely,1);
sKF = reshape(KEF(:)*ones(1,nelx*nely),    16*nelx*nely,1);
KF = sparse(iKF,jKF,sKF);
LF = chol(KF,'lower');
iTF = reshape(edofMatF,4*nelx*nely,1);
jTF = reshape(repmat([1:nelx*nely],4,1)',4*nelx*nely,1);
sTF = repmat(1/4,4*nelx*nely,1);
TF = sparse(iTF,jTF,sTF);
% INITIALIZE ITERATION
x = repmat(volfrac,nely,nelx);
xPhys = x;
loop = 0;
change = 1;
% START ITERATION
while (change > 0.01)
    loop = loop + 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,:);
    % OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS
    c=0;
    dc=0;
    for i = 1:size(F,2)
     Ui = U(:,i);
     ce = reshape(sum((Ui(edofMat)*KE).*Ui(edofMat),2), ...
     nely,nelx);
     c = c + sum(sum((Emin+xPhys.^penal*(E0-Emin)).*ce));
     dc = dc- penal*(E0-Emin)*xPhys.^(penal-1).*ce;
    end
    dv = ones(nely,nelx);
    % FILTERING/MODIFICATION OF SENSITIVITIES
    if ft == 1
     dc(:) = (TF'*(LF'\(LF\(TF*(dc(:).*xPhys(:)))))) ...
     ./max(1e-3,xPhys(:));
    elseif ft == 2
     dc(:) = TF'*(LF'\(LF\(TF*dc(:))));
     dv(:) = TF'*(LF'\(LF\(TF*dv(:))));
    end
    % OPTIMALITY CRITERIA UPDATE
    l1 = 0; l2 = 1e9; move = 0.2;
```

```
        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(:) = (TF'*(LF'\(LF\(TF*xnew(:)))));
            end
            if sum(xPhys(:)) > volfrac*nelx*nely, 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
        colormap(gray); imagesc(1-xPhys); caxis([0 1]); axis equal;
           axis off; drawnow;
    end
end
```

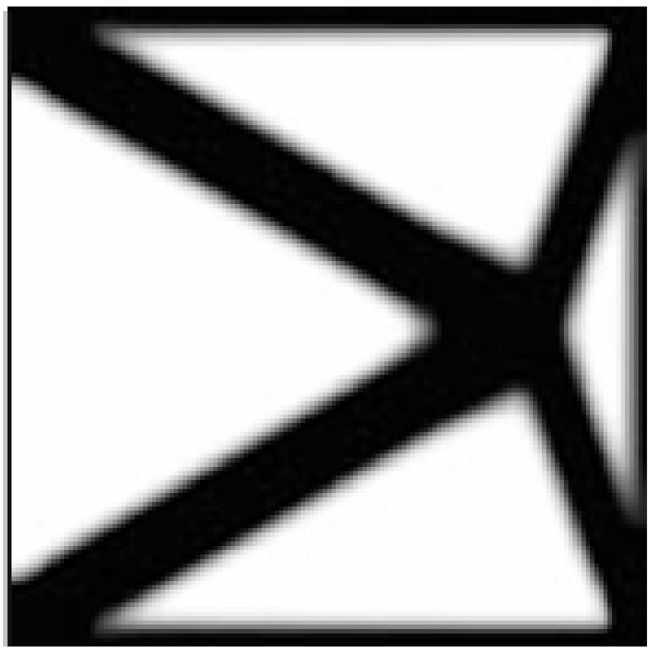

Figure 7:

MATLAB Code for HoneyTop90 with Projection Filter

```
function HoneyTop90(HNex,HNey,volfrac,penal,rfill,ft)
%% ---------- Material properties ----------
E0 = 1;
Emin = E0*1e-9;
```

```matlab
%% ---Element connectivity, nodal coordinates, Finite element analysis preparation--
NstartVs = reshape(1:(1+2*HNex)*(1+HNey),1+2*HNex,1+HNey);
DOFstartVs = reshape(2*NstartVs(1:end-1,1:end-1)-1,2*HNex*HNey,1);
NodeDOFs = repmat(DOFstartVs,1,8) + repmat([2*(2*HNex+1) + [2 3 0 1] 0 1 2 3 ],2*HNex*HNey,1);
ActualDOFs = NodeDOFs(setdiff(1:2*HNex*HNey,(2*HNex:2*HNex:2*HNex*HNey)' + mod(1:HNey,2)'),:);
HoneyDOFs = [ActualDOFs(2:2:end,1:2), ActualDOFs(1:2:end,:), ActualDOFs(2:2:end,7:8)];
Ncyi = repmat(reshape(repmat([-0.25 0.25]',HNey+1,1),2,HNey+1)+reshape(1.5*sort(repmat((0:HNey)',2,1)),2,(HNey+1)),HNex+1,1);
Ncyi(:,1:2:end) = flip(Ncyi(:,1:2:end));
Ncyf = Ncyi(1:end-1,:); % final arrays containing y-coordinates
HoneyNCO=(1)*[repmat((0:cos(pi/6):2*HNex*cos(pi/6)),1,HNey+1)' Ncyf(:)];%node co
if(mod(HNey,2)==0)
 HoneyDOFs(end:-1:end-(HNex)+2,1:6) = HoneyDOFs(end:-1:end-(HNex)+2,1:6)-2; % Updating
 HoneyNCO([(2*HNex+1)*HNey+1;(2*HNex+1)*(HNey+1)],:) = []; % Removing hangining nodes
end
[Nelem,Nnode] = deal(size(HoneyDOFs,1),size(HoneyNCO,1)); % elem #, node #
 F = sparse(2*((2*HNex+1)*HNey+1),1,-1,2*Nnode,1);          % Applied load
 U = zeros(2*Nnode,1);                                       % Initializing U
 fixeddofs = [2*(1:2*HNex+1:(2*HNex+1)*HNey+1)-1,(2*(2*HNex+1))];  % Fixed DOFs
 alldofs = 1:2*Nnode;                                              % Total DOFs
 freedofs = setdiff(alldofs,fixeddofs);                            % Free DOFs
 iK = reshape(kron(HoneyDOFs,ones(12,1))',144*Nelem,1);
 jK = reshape(kron(HoneyDOFs,ones(1,12))',144*Nelem,1);
 KE = E0*[616.43012 92.77147 -168.07333 65.54377 -232.28511 -0.00032 -120.65312 -83.28564 -71.60020 -92.77115 -23.81836 17.74187;
 92.77147 509.30685 101.02751 -71.90335 0.00032 -18.03857 -83.28564 -24.48314 -92.77179 -178.72347 -17.74187 -216.15832;
-168.07333 101.02751 455.74522 0.00000 -168.07333 -101.02751 -71.60020 -92.77179 23.60185 -0.00000 -71.60020 92.77179;
65.54377 -71.90335 0.00000 669.99176 -65.54377 -71.90335 -92.77115 -178.72347 -0.00000 -168.73811 92.77115 -178.72347;
-232.28511 0.00032 -168.07333 -65.54377 616.43012 -92.77147 -23.81836 -17.74187 -71.60020 92.77115 -120.65312 83.28564;
-0.00032 -18.03857 -101.02751 -71.90335 -92.77147 509.30685 17.74187 -216.15832 92.77179 -178.72347 83.28564 -24.48314;
-120.65312 -83.28564 -71.60020 -92.77115 -23.81836 17.74187 616.43012 92.77147 -168.07333 65.54377 -232.28511 -0.00032;
-83.28564 -24.48314 -92.77179 -178.72347 -17.74187 -216.15832 92.77147 509.30685 101.02751 -71.90335 0.00032 -18.03857;
-71.60020 -92.77179 23.60185 -0.00000 -71.60020 92.77179 -168.07333 101.02751 455.74522 0.00000 -168.07333 -101.02751;
-92.77115 -178.72347 -0.00000 -168.73811 92.77115 -178.72347 65.54377 -71.90335 0.00000 669.99176 -65.54377 -71.90335;
-23.81836 -17.74187 -71.60020 92.77115 -120.65312 83.28564 -232.28511 0.00032 -168.07333 -65.54377 616.43012 -92.77147;
```

```
17.74187 -216.15832 92.77179 -178.72347 83.28564 -24.48314 -0.00032 -18.03857 -101.02751 -71.90335 -92.77147 509.30685;]/1000;% elem stiffness
%% ---------- Filter preperation ---------
Cxx = repmat([sqrt(3)/2*(1:2:2*HNex-1) sqrt(3)*(1:1:HNex-1)],1,ceil(HNey/2));
Cyy= (repmat(3/4,HNex,HNey) + repmat(3/2*(0:HNey-1),HNex,1));
Cyy(HNex+1:2*HNex:length(Cyy(:))) = [];
ct = [Cxx(1:length(Cyy))' Cyy']*(1);             % Centre coordinates
DD = cell(Nelem,1);                                    % Initializing
for j = 1:Nelem
    Cent_dist = sqrt((ct(j,1)-ct(:,1)).^2+((ct(j,2)-ct(:,2)).^2));
    [I,J] = find(Cent_dist<=rfill);
    DD{j} = [I,J+(j-1),Cent_dist(I)];
end
 DD = cell2mat(DD);
 HHs = sparse(DD(:,2),DD(:,1),1-DD(:,3)/rfill);
 HHs = spdiags(1./sum(HHs,2),0,Nelem,Nelem)*HHs;
%% ---------- Initialization -------------
x = volfrac*ones(Nelem,1);                                          % Initial guess
[xPhys,loop,change,maxiter,dv,move] = deal(x,0,1,200,ones(Nelem,1),0.2); % Parameters
%% ---------- Start optimization ----------
while (change > 0.01 && loop < maxiter)
  loop = loop + 1;
  %% ---------- Finite element analysis ----------
  sK = reshape(KE(:)*(Emin + xPhys'.^penal*(E0-Emin)),144*Nelem,1);
  K = sparse(iK,jK,sK);                                          % Global stiffness
  U(freedofs) = decomposition(K(freedofs,freedofs),'chol','lower')\F(freedofs);
  %% ---------- Objective and sensitivities evaluation ----------
  ce = sum((U(HoneyDOFs)*KE).*U(HoneyDOFs),2);
  c =  sum(sum((Emin+xPhys.^penal*(E0-Emin)).*ce));              % Finding objective
  dc = -penal*(E0-Emin)*xPhys.^(penal-1).*ce;                   % Obj. sensitivities
   %% ---------- Using Filters ---------------------------
   if ft == 1
    dc = HHs'*(x.*dc)./max(1e-3,x);
  elseif ft == 2
    dc = HHs'*(dc);
    dv = HHs'*(dv);
   end
  %% ---------- Optimality criteria update ----------------
   xOpt = x;
  [xUpp, xLow] = deal (xOpt + move, xOpt - move);                % Upp. & low. limits
  OcC = xOpt.*(sqrt(-dc./dv));                                   % Opt. parameter
  inL = [0, mean(OcC)/volfrac];                                  % Lag. Mul. range
  while (inL(2)-inL(1))/(inL(2)+inL(1))> 1e-3
    lmid = 0.5*(inL(2)+ inL(1));
    x = max(0,max(xLow,min(1,min(xUpp,OcC/lmid))));
    if mean(x)>volfrac, inL(1) = lmid; else, inL(2) = lmid; end
  end
```

```
  if ft == 1 || ft ==0, xPhys = x; elseif ft == 2, xPhys = HHs'*x; end
  change = max(abs(xOpt-x));
  %% ---------- Results printing ----------------------------
  fprintf(' It.:%5i Obj.:%11.4f Vol.:%7.3f ch.:%7.3f\n',loop,c, mean(
      xPhys),change);
  %% ---------- Plotting intermediate designs ---------------
  colormap('gray'); scatter(ct(:,1),ct(:,2),[],1-xPhys,'filled'); axis
      off equal; pause(1e-6);
end
```

References

1 The code used to generate plots is in the zip file.