The Chinese University of Hong Kong
Dept of Mechanical & Automation Eng

MAEG5160 Design for Additive Manufacturing

Assignment #1

by

Liuchao JIN (Student ID: 1155184008)

Liuchao Jin

2022-23 Term 2

This code is a MATLAB implementation of topology optimization for a chair. To use the code, you will need to have a basic understanding of MATLAB and finite element analysis. Before running the code, you will need to modify the input parameters to match your specific problem requirements. These parameters include the number of elements in the x and y directions, the penalization parameter, the filter radius, the initial material density, the element size, and the material properties of the structure. You will also need to specify the load vector and the optimization parameters, such as the maximum number of iterations and the convergence tolerance. The basic inputs of this code are shown below:

```
top_chair(nelx,nely,volfrac,penal,rmin)
```

Here are the definitions of inputs.

- nelx and nely: These variables define the number of elements in the x and y direction, respectively. You can change these values to adjust the mesh size.
- volfrac: a variable that represents the desired volume fraction of the optimized structure.
- penal: This variable is the penalization parameter used in the SIMP (Solid Isotropic Material with Penalization) method. It determines how stiff the material is. A larger value of penal means a stiffer material.
- rmin: This variable defines the filter radius for the sensitivity filter. Increasing this value results in a smoother density distribution.

The above is the instruction about how to use this code to topology optimize a chair. Next, the real application of this code will be demonstrated. If we run the code as shown below:

```
top_chair(60,60,0.15,3.0,1.5)
```

The results are shown below:

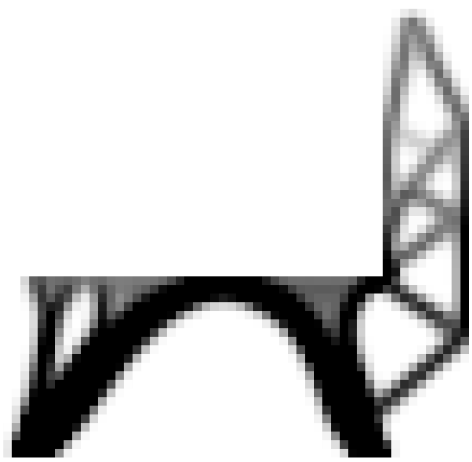

Figure 1: Optimized chair.

Appendix

The MATLAB code in function top_chair :

```
%%%% A 99 LINE TOPOLOGY OPTIMIZATION CODE BY OLE SIGMUND, JANUARY 2000 %%%
%%%% CODE MODIFIED FOR INCREASED SPEED, September 2002, BY OLE SIGMUND %%%
function top_chair(nelx,nely,volfrac,penal,rmin)
    % DEFINE THE PASSIVE VALUE
    for ely = 1:nely
        for elx = 1:nelx
            if elx >=1 && elx<=round(nelx/6*5) && ely>=1 && ely<=round(nely/3*2)-1
                pv(ely,elx) = 1;
                x(ely,elx) = 0.001;
            else
                pv(ely,elx) = 0;
            end
        end
    end
    % INITIALIZE
    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,pv);
        % PRINT RESULTS
        change = max(max(abs(x-xold)));
```

```
        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,passive)
    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)))));
        xnew(find(passive)) = 0.001;
        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-floor(rmin),1):min(i+floor(rmin),nelx)
                for l = max(j-floor(rmin),1):min(j+floor(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 = zeros(2*(nely+1)*(nelx+1),1);
    for elx = 1:nelx
```

```
        for ely = 1:nely
            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 LOADS
    F(2*[(round(nelx/6)-1)*(nely+1)+round(nely/3*2):(nely+1): ...
        (round(nelx/6*5)-1)*(nely+1)+round(nely/3*2)],1) = -1;

    node1 = (round(nelx/6*5)+1)*(nely+1) + round(nely/3*2);
    node2 = (round(nelx/6*5)+1)*(nely+1) + round(nely/2);
    node3 = (round(nelx/6*5)+1)*(nely+1) + round(nely/6);
    F(2*node1 - 1, 1) = 1;
    F(2*node2 - 1, 1) = 1;
    F(2*node3 - 1, 1) = 1;

    % FIXED NODE FOR THE SUPPORT

    posNodes1 = round(nelx/6)*(nely+1);
    posNodes2 = round(nelx/6*5)*(nely+1);

    fixeddofs = [2*posNodes1-1, 2*posNodes1, 2*posNodes2 - 1, 2*posNodes2];
    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
```