Project 3

In this project, we used topology optimization algorithms to minimums the material of wall support. In figure.1, we designed wall support using SOLIDWORKS as the initial design before the optimal solution of the structure design. We found out that topology optimization can reduce the cost of the material as we can know the set constrained variables to a minimum. It is possible volume of the structure design.

- The optimization formation is shown below:

We need to find

$$x = [x_1, x_2, x_3, \ldots\ldots x_n]^T$$
$$minimize\ c(\tilde{x}) = F^T U(\tilde{x})$$
$$Subject\ to\ \ v(\tilde{x}) = \tilde{x}^T v - \tilde{v} \leq 0$$
$$x \in X$$
$$X = 0 \leq x \leq 1$$

Where n is the number of elements in the design. And v is the vector of element volume. Also, F is the nodal force in the design and U is the nodal displacements that solution for F =K(x)*U(x)
We used SQ approximation for minimum compliance as
We have to find d

$$minimize \frac{1}{2}\ d^T \nabla^2 c^{(T)} d + \nabla_c^{(k)^T} d$$

$$Subject\ to\ \ v(\tilde{x}) = Ad = 0$$

- The boundary condition and loading in this problem are

1- Force applied on y-direction, which is -500 N
2- Fixed support on the wall

- Mechanical properties of the Materials:

We tested two different materials as we use topology optimization in our design.
1- Aluminum material
2- copper
We used Aluminum as the material selected. We have some properties as shown below:

- Aluminum material used the Poisson ratio of 0.35 and young models of 68 psi.
- Copper material used the Poisson ratio of 0.34 and young models of 121 psi.

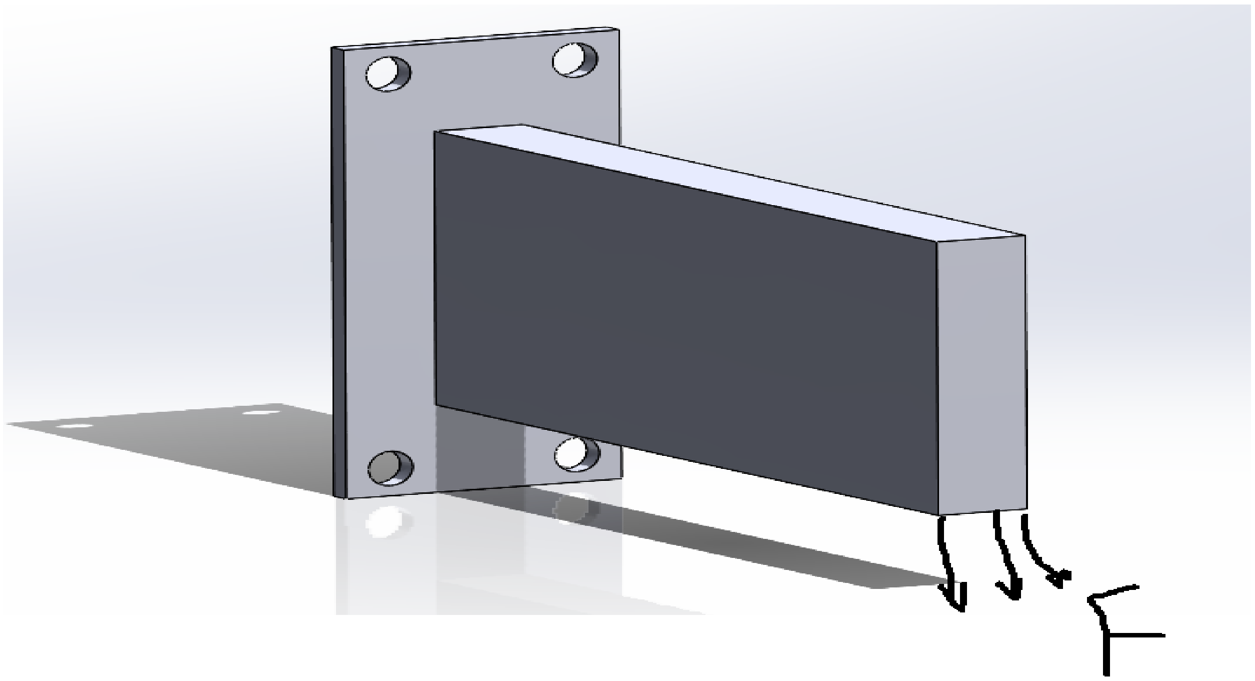

Figure 1: design wall support in SOLIDWORK before optimization

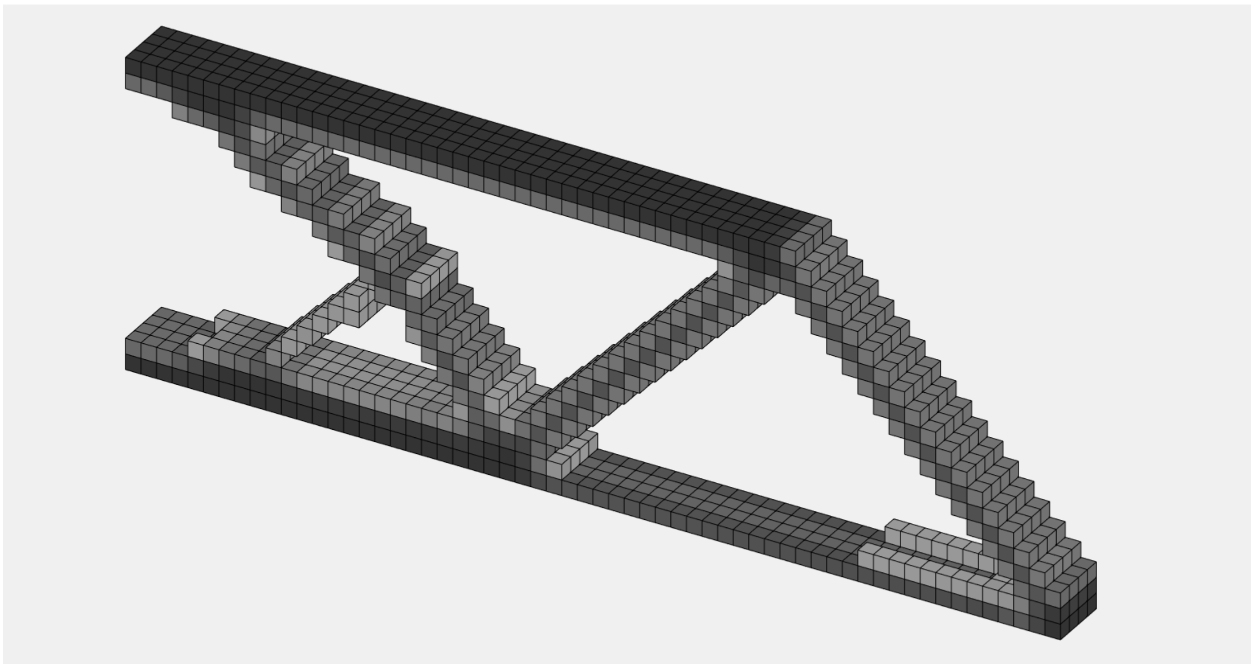
Figure 2: wall support 3-D design for Aluminum after topology optimization in MATLAB

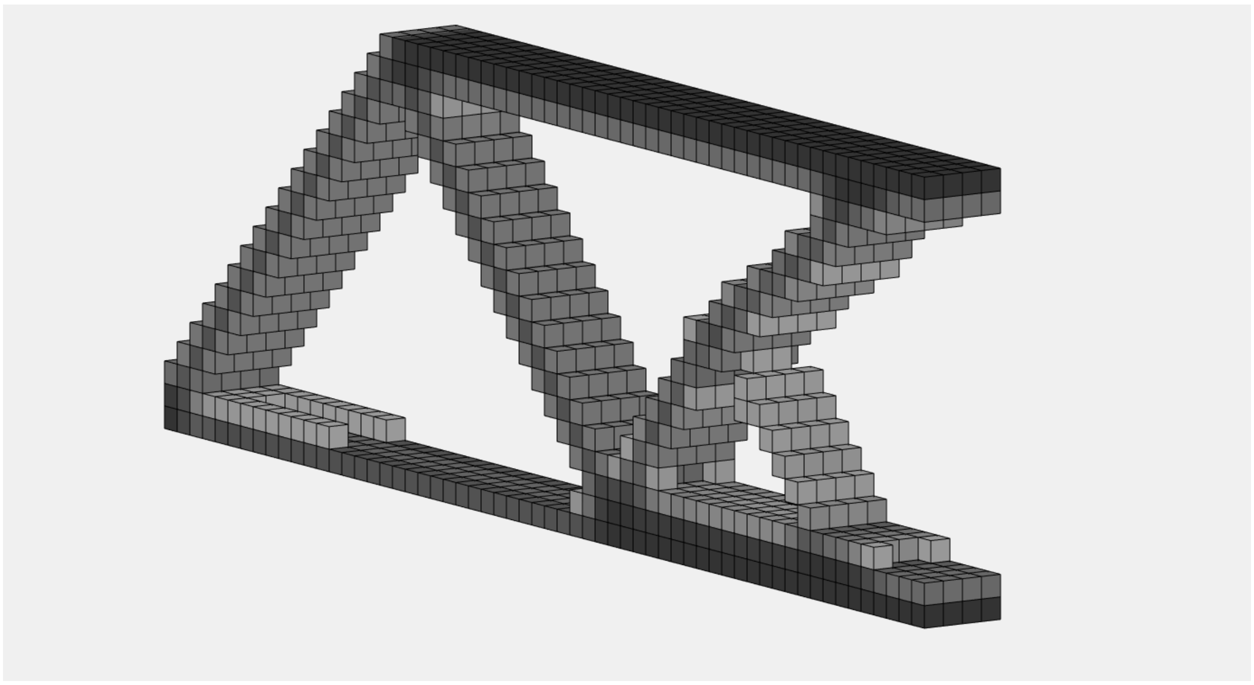

Figure 2: here is the design for Aluminum with different point view

Results

We used MATLAB to find the topology optimization of wall support. These methods can be used to find the minimum structure design possible. We design wall support as shown in figure 1 as the initial design and figure 2 shows optimal solution using topology optimization.

Reference

Liu, Kai, and Andrés Tovar. “An Efficient 3D Topology Optimization Code Written in MATLAB.” *Structural and Multidisciplinary Optimization*, vol. 50, no. 6, 2014, pp. 1175–1196., https://doi.org/10.1007/s00158-014-1107-x.

Contents

```
%%%% AN 88 LINE TOPOLOGY OPTIMIZATION CODE Nov, 2010 %%%%

nelx = 60;
nely = 20;
nelz = 4;
volfrac = 0.3;
penal = 3;
rmin = 1.5;


%function Project_3_3D(nelx,nely,nelz,volfrac,penal,rmin)
%%USER-DEFINED LOOP PARAMETERS
maxloop= 200;
tolx = 0.01;
displayfag = 1;
```

MATERIAL PROPERTIES

```
E0 = 68;
Emin = 1e-9;
nu = 0.35;
```

USER-DEFINED LOAD DOFs

```
i1 = nelx; j1 =0; k1 = 0:nelz;
loadnid = k1*(nelx+1)*(nely+1)+i1*(nely+1)+(nely+1-j1);
loaddof = 3*loadnid(:)-1;
```

USER-DEFINED SUPPORT FIXED DOFs

```
[jf, kf] = meshgrid(1: nely+1,1:nelz+1);
fixednid = (kf-1)*(nely+1)*(nelx+1)+jf;
fixeddof = [3*fixednid(:); 3*fixednid(:)-1;3*fixednid(:)-2];
```

PREPARE FINITE ELEMENT ANALYSIS

```
nele = nelx*nely*nelz;
ndof = 3*(nelx+1)*(nely+1)*(nelz+1);
% 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(loaddof,1,-1, ndof,1);
U = zeros(ndof,1);
% fixeddofs = union([1:2:2*(nely+1)],[2*(nelx+1)*(nely+1)]);
% alldofs = [1:2*(nely+1)*(nelx+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))';
```

PREPARE FILTER

```
iH = ones(nele*(2*(ceil(rmin)-1)+1)^2,1);
jH = ones(size(iH));
sH = zeros(size(iH));
k = 0;
for k1 = 1:nelz
  for i1 = 1:nelx
      for j1 = 1:nely
          e1 = (k1-1)*nelx*nely + (i1-1)*nely+j1;
          for k2 = max(k1-(ceil(rmin)-1),1):min(k1+(ceil(rmin)-1),nelz)
              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 = (k2-1)*nelx*nely+ (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+(k1-k2)^2);
                   end
               end
           end
      end
  end
end
H = sparse(iH,jH,sH);
Hs = sum(H,2);
```

INITIALIZE ITERATION

```
x = repmat(volfrac,[nely,nelx,nelz]);
% for ely = 1: nely
%     for elx = 1:nelx
%         if sqrt(ely-nely/2.)^2 + (elx-nelx/3.)^2) < nely/3.
%            passive(ely,elx) = 1;
%         else
%            passive(ely,elx) = 0;
%         end
%     end
% end
% passive  = repmat(passive,[1,1,nelz]);
% x(passive) = 0;
xPhys = x;
loop = 0;
change = 1;
```

START ITERATION

```
while change > tolx && loop < maxloop
```

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

FE-ANALYSIS

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

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/MODIFICATION OF SENSITIVITIES

```
  dc(:) = H*(dc(:)./Hs);
  dv(:) = H*(dv(:)./Hs);
```

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)))));
    %xnew(find(passive)) = 0;
    xPhys(:) = (H*xnew(:))./Hs;
    if sum(xPhys(:)) > volfrac*nele , l1 = lmid; else l2 = lmid; end
  end
  change = max(abs(xnew(:)-x(:)));
  x = 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);
```

```
 It.:  200 Obj.:    20.1632 Vol.:  0.300 ch.:  0.000
```

PLOT DENSITIES

```
  if displayfag, clf; display_3D(xPhys); end
clf; display_3D(xPhys);
%end

function [KE] = lk_H8(nu)
A = [32 6 -8 6 -6 4 3 -6 -10 3 -3 -3 -4 -8;-48 0 0 -24 24 0 0 0 12 -12 0 12 12 12];
k =  1/72*A'*[1; nu];

K1 = [k(1) k(2) k(2) k(3) k(5) k(5);
    k(2) k(1) k(2) k(4) k(6) k(7);
    k(2) k(2) k(1) k(4) k(7) k(6);
    k(3) k(4) k(4) k(1) k(8) k(8);
    k(5) k(6) k(7) k(8) k(1) k(2);
    k(5) k(7) k(6) k(8) k(2) k(1)];

K2 = [k(9) k(8) k(12) k(6) k(4) k(7);
    k(8) k(9) k(12) k(5) k(3) k(5);
    k(10) k(10) k(13) k(7) k(4) k(6);
    k(6) k(5) k(11) k(9) k(2) k(10);
    k(4) k(3) k(5) k(2) k(9) k(12);
    k(11) k(4) k(6) k(12) k(10) k(13)];

K3 = [k(6) k(7) k(4) k(9) k(12) k(8);
    k(7) k(6) k(4) k(10) k(13) k(10);
    k(5) k(5) k(3) k(8) k(12) k(9);
    k(9) k(10) k(2) k(6) k(11) k(5);
    k(12) k(13) k(10) k(11) k(6) k(4);
    k(2) k(12) k(9) k(4) k(5) k(3)];

K4 = [k(14) k(11) k(11) k(13) k(10) k(10);
    k(11) k(14) k(11) k(12) k(9) k(8);
    k(11) k(11) k(14) k(12) k(8) k(9);
    k(13) k(12) k(12) k(14) k(7) k(7);
    k(10) k(9) k(8) k(7) k(14) k(11);
    k(10) k(8) k(9) k(7) k(11) k(14)];

K5 = [k(1) k(2) k(8) k(3) k(5) k(4);
    k(2) k(1) k(8) k(4) k(6) k(11);
    k(8) k(8) k(1) k(5) k(11) k(6);
    k(3) k(4) k(5) k(1) k(8) k(2);
    k(5) k(6) k(11) k(8) k(1) k(8);
    k(4) k(11) k(6) k(2) k(8) k(1)];

K6 = [k(14) k(11) k(7) k(13) k(10) k(12);
    k(11) k(14) k(7) k(12) k(9) k(2);
    k(7) k(7) k(14) k(10) k(2) k(9);
    k(13) k(12) k(10) k(14) k(7) k(11);
    k(10) k(9) k(2) k(7) k(14) k(7);
    k(12) k(2) k(9) k(11) k(7) k(14)];
KE = 1/((nu+1)*(1-2*nu))*...
    [ K1 K2 K3 K4;
    K2' K5 K6 K3';
    K3' K6 K5' K2';
    K4 K3 K2 K1'];
end

function display_3D(rho)
[nely, nelx, nelz] = size(rho);
hx = 1; hy = 1; hz = 1;
face = [1 2 3 4; 2 6 7 3; 4 3 7 8; 1 5 8 4; 1 2 6 5; 5 6 7 8];
set (gcf, 'Name', 'ISO display','NumberTitle','off');
for k = 1:nelz
    z = (k-1)*hz;
    for i = 1:nelx
        x = (i-1)*hx;
        for j = 1:nely
            y = nely*hy - (j-1)*hy;
            if (rho(j,i,k)> 0.5)
                vert = [x y z;x y-hx z; x+hx y-hx z;x+hx y z; x y z+hx; x y-hx z+hx; x+hx y-hx z+hx;x+hx y z+hx];
                vert(:,[2 3]) = vert(:,[3 2]); vert(:,2,:) = -vert(:,2,:);
                patch('faces', face,'Vertices', vert,'FaceColor',[0.2+0.8*(1-rho(j,i,k)),0.2+0.8*(1-rho(j,i,k)),0.2+0.8*(1-rho(j,i,k))]);
                hold on;
            end
        end
    end
end
axis equal; axis tight; axis off; box on; view([30,30]); pause(1e-6);
end
```