**INDIAN INSTITUTE OF TECHNOLOGY MADRAS**

**DEPARTMENT OF MECHANICAL ENGINEERING**

**INTERNSHIP REPORT**

**ON**

**MATLAB CODE IMPLEMENTATION OF VARIOUS TOPOLOGY OPTIMISATION METHODS**

SUBMITTED BY

**MUNI KUMAR P V**

B.TECH (MECHANICAL ENGINEERING),

NATIONAL INSTITUTE OF TECHNOLOGY TIRUCHIRAPALLI.

UNDER THE GUIDANCE OF

**DR. SOURAV RAKSHIT**

ASSISTANT PROFESSOR,

DEPARTMENT OF MECHANICAL ENGINEERING,

INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

**CONTENTS**

1. Introduction
   1. Scope
   2. Objectives
2. Background

2.1 Related Work

1. Various Topology Optimisation Methods
   1. Evolutionary Structural Optimization(ESO) Method
   2. Bi-directional Evolutionary Structural Optimization(BESO) Method
      1. Hard kill BESO Method
      2. Soft kill BESO Method
         1. Concentrated Load Optimization Problems
         2. Gravity Load Optimization Problems
         3. Gravity and Concentrated Load Optimization Problems
      3. SIMP Method
2. Matlab Codes for the Optimization Methods
   1. ESO Method
   2. BESO Method
      1. Concentrated Load Optimization Problems
      2. Gravity Load Optimization Problems
      3. Gravity and Concentrated Load Optimization Problems
3. Conclusion
4. Appendix
5. Bibliography

**INTRODUCTION**

Structural optimization is the process of finding best design parameters for the structures in order to achieve the best measurable performance under given constraints.

Classification of Structural optimization

1. Shape optimization
2. Size optimization
3. Topology optimization

Shape optimization is an optimization technique which is used to find the optimal shape without modifying the size of the body for a given set of constraints. Shape optimization is mainly performed on continuum structures by modifying the predetermined boundaries to achieve the optimal designs

Size optimization is to find the optimal design by changing the size variables such as the cross-sectional dimensions of trusses and frames, or the thicknesses of plates. Here only the reduction or addition of size takes with consistent shape.

Topology optimization of continuum structures is to find the optimal designs by determining the best locations and geometries of cavities in the design domains.

Finite element analysis (FEA) is the way of discretizing the design domain into a fine mesh of elements. The optimization procedure is to find the topology of a structure by determining for every point in the design domain if there should be material (solid element) or not (void element).

**BACKGROUND**

Topology optimization is a relatively new and rapidly expanding field of structural mechanics, which can result in much greater savings than mere cross section or shape optimization. Topology optimization (TO) is a mathematical method that optimizes material layout within a given design space, for a given set of loads, boundary conditions and constraints with the goal of maximizing the performance of the system. Here the function which is going to be minimised is called objective function or compliance. The mean compliance is the total strain energy of the structure or work done by the applied external loads. The restrictions which are imposed for the increase in efficiency of the body is called constraints. Generally the constraint is given on the volume of the body. Other constraints include displacement, buckling load, frequency, temperature, etc.

Without topology optimization, excess materials are used and loads on moving parts are higher than required. So the energy efficiency is compromised. Topology optimization is often aimed at searching for the stiffest structure with a given volume of material. The latest methods and theories to improve mechanical performances and save structural weight under static, dynamic and thermal loads are summarized and explained in detail here, in addition to potential applications of topology optimization techniques such as shape preserving design, smart structure design and additive manufacturing.

**VARIOUS TOPOLOGY OPTIMIZATION MATERIALS**

**EVOLUTIONARY TOPOLOGY OPTIMIZATION (ESO) METHOD:**

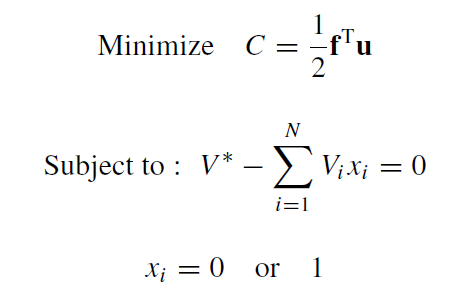
ESO’ stands for Evolutionary Structural Optimization, which is a design method based on the simple concept of gradually removing inefficient material from a structure. When a solid element is removed from a structure, the change of the mean compliance or total strain energy is equal to the elemental strain energy. This change is define as the elemental sensitivity number. The increase in elemental strain energy due to removal of the ith element is equal to elemental strain energy. The number of elements to be removed is determined by the element removal ratio (ERR) which is defined as the ratio of the number of elements removed at each iteration to the total number of elements in the initial or the current FEA model. An inefficient use of material is the places where there is low values of stress (or strain) in some parts of the structure. The low-stressed material is assumed to be under-utilized and is therefore removed.

The ESO method seems to follow a logical procedure to gradually remove material from the structure to reduce its weight until the constraint is no longer satisfied. The ESO algorithm is unable to recover the material once it has been prematurely or wrongly deleted from the structure. Hence, while the ESO method is capable of producing a much improved solution over an initial guess design in most cases, the result may not necessarily be the absolute optimum. If once the element is changed into void then again it will not be converted into a solid. The ESO method does not require regenerating new finite element meshes even when the final structure has departed substantially from the initial design. Element removal can be done by simply assigning the material property number of the rejected elements to zero and ignoring those elements when the global stiffness matrix is assembled in the subsequent finite element analysis. Early ESO method neglected the numerical topology optimisation problem which includes solution to existence of solution, checker-board, mesh-dependency and local optimum, etc.

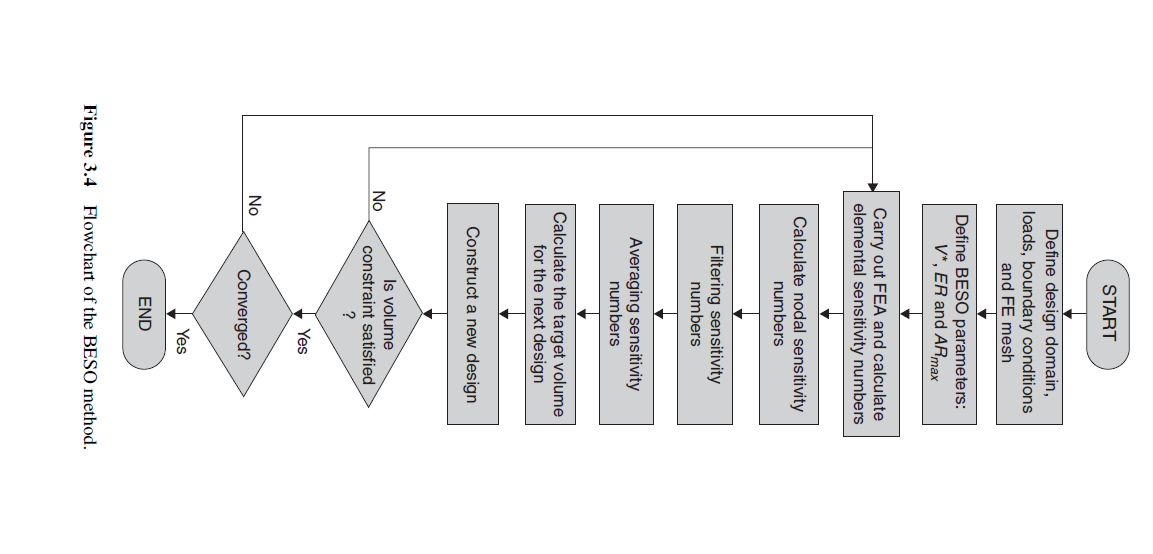
**BI-DIRECTIONAL EVOLUTIONARY STRUCTURAL OPTIMIZATION (BESO) METHOD:**

The bi-directional evolutionary structural optimization (BESO) method allows material to be removed and added simultaneously. The sensitivity numbers of the void elements are estimated through a linear extrapolation of the displacement field after the finite element analysis. Then, the solid elements with the lowest sensitivity numbers are removed from the structure, and the void elements with the highest sensitivity numbers are changed into solid elements. The numbers of removed and added elements in each iteration are determined by two unrelated parameters: the rejection ratio (RR) and the inclusion ratio (IR).

In this method, elements with the lowest von Mises stresses are removed and void elements near the highest von Mises stress regions are switched on as solid elements. The user must carefully select the rejection and inclusion ratio to obtain an optimum solution.



Here, f – applied load, u – displacement vector, C – compliance, V\* - optimum volume, xi –0 for void and 1 for solid, N – total number of elements in the structure. BESO method is to make the algorithm to be stably convergent towards a solution which exactly addresses the above optimization problem statement.



**HARD KILL BESO METHOD:**

In hard kill BESO method, the density of the voids present in the optimal design is initialised as zero. This means that the material will not be present in that regions.The hard kill BESO method lacks existence of solution in general continuum setting. The reason is that, with different mesh sizes, more number of holes are introduced without changing the structural volume leads to increase in efficiency of the design. This leads to numerical instability and is known as mesh dependency. The sensitivity numbers of solid elements and soft elements are equal to the elemental strain energy and zero.

**SOFT KILL BESO METHOD:**

In soft kill BESO method, the density of the void elements is very less (xmin) but not zero. These elements are known as soft elements. The design variables in the soft-kill BESO method can either be 1 or xmin. Since xmin is usually very small (e.g. 0.001), the soft elements are structurally negligible and therefore the topologies obtained are virtually solid-void designs with no intermediate material densities between 1 and xmin. The averaging scheme will suppress unwarranted changes of design variables for solid elements with high historical sensitivity numbers and void elements with low historical sensitivity numbers. Hard kill BESO method is one of the cases in the soft kill BESO method where the penalisation tends to infinity.

**RECTIFICATION OF VARIOUS DRAWBACKS IN PREVIOUS METHODS:**

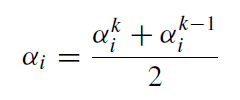
**Checkerboard and Mesh-dependency problems:**

When the continuum structure is discretised into lower order finite elements, the sensitivity numbers become discontinuous across the boundary elements. This leads to checkerboard patterns in resulting topologies. To avoid that checkerboard patterns in ESO method, smoothing scheme of averaging the sensitivity numbers of neighbouring elements should be done.

Mesh-dependency refers to the problem of obtaining different topologies from using different finite element meshes. Mesh-refinement should result in a better finite element modelling of the same optimal structure and a better description of boundaries – not in a more detailed or qualitatively different structure. A filter scheme is used to carry out this process. The filter has a length scale rmin that does not change with mesh refinement. The primary role of the scale parameter rmin in the filter scheme is to identify the nodes that will influence the sensitivity of the ith element. This can be visualized by drawing a circle of radius rmin centred at the centroid of ith element, thus generating the circular sub-domain. Usually i value of rmin should be big enough so that covers more than one element. The size of the sub-domain does not change with mesh size. Nodes located inside Ωi contribute to the computation of the improved sensitivity number of the ith element. The filter scheme smoothens the sensitivity numbers in the whole design domain. Thus, the sensitivity numbers for void elements are automatically obtained. They may have high values due to high sensitivity numbers of solid elements within the sub-domain Ωi. Therefore, some of the void elements may be changed to solid elements in the next iteration.

**STABILIZING EVOLUTIONARY PROCESSES:**

In ESO/BESO methods,the objective function and the corresponding topology may not be convergent. The reason for such chaotic behaviour is that the sensitivity numbers of the solid (1) and void (0) elements are based on discrete design variables of element presence (1) and absence (0). This makes the objective function and the topology difficult to converge. The simple averaging scheme is given as



Where, k is the current iteration number. Then let αik = αi which will be used for the next iteration. Thus, the updated sensitivity number includes the whole history of the sensitivity information in the previous iterations. It affects the search path of the BESO algorithm it has very little effect on the final solution when it becomes convergent.

**ELEMENT REMOVAL AND ADDITION AND THE CONVERGENT CRITERION:**

The volume constraint can be greater or smaller than the volume of the initial guess design, the target volume in each iteration may decrease or increase step by step until the constraint volume is achieved. The evolution of the volume can be expressed by

Vk+1 = Vk (1 +or- ER), k=1, 2, 3,…

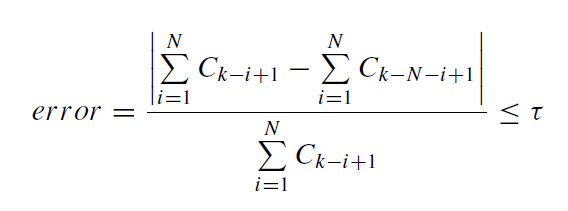
Where ER is the evolutionary volume ratio. The elements are sorted according to the values of their sensitivity numbers (from the highest to the lowest). For solid element (1), it will be removed (switched to 0) if



For void elements (0), it will be added (switched to 1) if



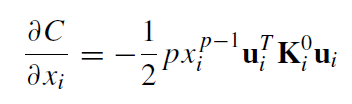
Here, αdelth and αaddth are the threshold sensitivity numbers for removing and adding elements, αdelth is always less than or equal to αaddth. The volume addition ratio (AR), which is defined as the number of added elements divided by the total number of elements in the design domain. It is introduced to ensure that not too many elements is added in a single iteration. New element removal and addition scheme ranks all elements (void and solid) together, while in the previous BESO method elements for removal and those for addition are treated differently. This ensures that the objective volume is reached and the following convergence criterion is satisfied.



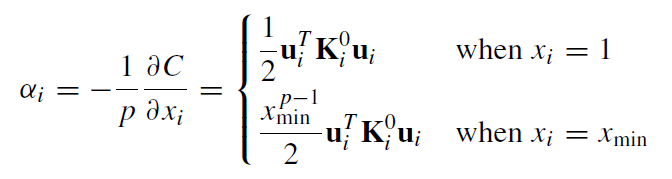
Where k is the current iteration number, τ is an allowable convergence tolerance and N is an t integer number. Normally, N is selected to be 5 which implies that the change in the mean compliance over the last 10 iterations is acceptably small.

**BESO UTILIZING MATERIAL INTERPOLATION SCHEME WITH PENALIZATION (SIMP METHOD):**

In SIMP method, continuous values between 1 and xmin are considered for the design variables. This leads to the appearance of grey elements in the final optimized result.The complete removal of a solid element from the design domain could result in theoretical difficulties in topology optimization. It appears to be rather irrational when the design variable (an element) is directly eliminated from the topology optimization problem. An alternative way of effectively ‘removing’ an element is to reduce the elastic modulus of the element or one of the characteristic dimensions (such as the thickness) of the element to a very small value. The void element is replaced by a soft element with a very low density. The sensitivity number in the original ESO/BESO methods for stiffness optimization is defined by the approximate variation of the objective function due to removing an element. The sensitivity is given by



And the sensitivity number is given by



Here the constant is multiplied with the solid sensitivity to give a very small value. The elements with higher strain energy density should have xi increased and the elements with lower strain energy density should have xi decreased. For the soft-kill BESO method, as the design variables are restricted to be either xmin or 1, the optimality criterion can be described as that strain energy densities of solid elements are always higher than those of soft elements. Therefore, we devise an update scheme for the design variables xi by changing from 1 to xmin for elements with lowest sensitivity numbers and from xmin to 1 for elements with highest sensitivity numbers. The averaging scheme will suppress unwarranted changes of design variables for solid elements with high historical sensitivity numbers and void elements with low historical sensitivity numbers. The optimal designs from the generalized BESO method are independent of the selection of the penalty exponent (normally p >= 1.5), even when p tends to infinity.

**MATLAB CODES FOR THESE OPTIMIZATION METHODS**

The original code for all the below mentioned codes was “**BESO method for Concentrated load on Cantilever Beam**” which is the third code from the top. This code was taken from the book named “**Evolutionary Topology Optimization of Continuum Structures: Methods and Applications by X. Haung and Y. M. Xie (Pg. 48 - 50)**”. The loading and the boundary conditions for the remaining codes are modified according to its specifications.

**ESO METHOD FOR CONCENTRATED LOAD ON CANTILEVER BEAM:**

In this code, the changes are made in the “**Optimality Criteria Update**” section of the program from the original one.

%%%% ESO METHOD IN TOPOLOGY OPTIMIZATION %%%%

function sbeso3(nelx,nely,volfrac,er,rmin)

% INITIALIZE

x(1:nely,1:nelx) = 1.; vol=1.; i = 0; change = 1.; penal = 3.;

% START iTH ITERATION

while i < 100

i = i + 1; vol = max(vol\*(1-er),volfrac);

if i >1; olddc = dc; end

% FE-ANALYSIS

[U]=FE(nelx,nely,x,penal);

% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

[KE] = lk;

c(i) = 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(i) = c(i) + 0.5\*x(ely,elx)^penal\*Ue'\*KE\*Ue;

dc(ely,elx) = 0.5\*x(ely,elx)^(penal-1)\*Ue'\*KE\*Ue;

end

end

% FILTERING OF SENSITIVITIES

[dc] = check(nelx,nely,rmin,x,dc);

% STABLIZATION OF EVOLUTIONARY PROCESS

if i > 1; dc = (dc+olddc)/2.; end

% BESO DESIGN UPDATE

[x] = ADDDEL(nelx,nely,vol,er,dc,x);

% PRINT RESULTS

if i>10;

change=abs(sum(c(i-9:i-5))-sum(c(i-4:i)))/sum(c(i-4:i));

end

disp([' It.: ' sprintf('%4i',i) ' Obj.:' sprintf('%10.4f',c(i))...

' 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

%%%%%%%%%% OPTIMALITY CRITERIA UPDATE%%%%%%%%%%%%%%%%%%%%

function [x]=ADDDEL(nelx,nely,volfrac,er,dc,x)

b = [];count = 1;

dc = reshape(dc,size(dc,1)\*size(dc,2),1);

for i = 0.95:-er:volfrac

b = sort(dc);

d = round(count\*er\*size(dc,1));

c = b(d,1);

e = find(dc<=c);

x(e) = 0.001;

dc = reshape(dc,nelx,nely);

count = count+1;

end

%%%%%%%%%% MESH-INDEPENDENCY FILTER%%%%%%%%%%%%%%%%%%%%%%

function [dcf]=check(nelx,nely,rmin,x,dc)

dcf=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);

dcf(j,i) = dcf(j,i) + max(0,fac)\*dc(l,k);

end

end

dcf(j,i) = dcf(j,i)/sum;

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 AND SUPPORTS (Cantilever)

F(2\*(nelx+1)\*(nely+1)-nely,1)=-1.0;

fixeddofs=[1:2\*(nely+1)];

alldofs = [1:2\*(nely+1)\*(nelx+1)];

freedofs = setdiff(alldofs,fixeddofs);

% SOLVING

U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);

U(fixeddofs,:)= 0;

%%%%%%%%%% 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)];

**ESO METHOD FOR CONCENTRATED LOAD ON SIMPLY SUPPORTED BEAM:**

In this code, the changes are made in the “**Optimality Criteria Update**” and “**Define Loads and Supports**” sections of the program from the original one.

%%%% ESO METHOD IN TOPOLOGY OPTIMIZATION %%%%

function eso\_ssb(nelx,nely,volfrac,er,rmin)

% INITIALIZE

x(1:nely,1:nelx) = 1.; vol=1.; i = 0; change = 1.; penal = 3.;

% START iTH ITERATION

while i < 100

i = i + 1; vol = max(vol\*(1-er),volfrac);

if i >1; olddc = dc; end

% FE-ANALYSIS

[U]=FE(nelx,nely,x,penal);

% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

[KE] = lk;

c(i) = 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(i) = c(i) + 0.5\*x(ely,elx)^penal\*Ue'\*KE\*Ue;

dc(ely,elx) = 0.5\*x(ely,elx)^(penal-1)\*Ue'\*KE\*Ue;

end

end

% FILTERING OF SENSITIVITIES

[dc] = check(nelx,nely,rmin,x,dc);

% STABLIZATION OF EVOLUTIONARY PROCESS

if i > 1; dc = (dc+olddc)/2.; end

% BESO DESIGN UPDATE

[x] = ADDDEL(nelx,nely,vol,er,dc,x);

% PRINT RESULTS

if i>10;

change=abs(sum(c(i-9:i-5))-sum(c(i-4:i)))/sum(c(i-4:i));

end

disp([' It.: ' sprintf('%4i',i) ' Obj.:' sprintf('%10.4f',c(i))...

' 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

%%%%%%%%%% OPTIMALITY CRITERIA UPDATE%%%%%%%%%%%%%%%%%%%%

function [x]=ADDDEL(nelx,nely,volfrac,er,dc,x)

b = [];count = 1;

dc = reshape(dc,size(dc,1)\*size(dc,2),1);

for i = 0.95:-er:volfrac

b = sort(dc);

d = round(count\*er\*size(dc,1));

c = b(d,1);

e = find(dc<=c);

x(e) = 0.001;

dc = reshape(dc,nelx,nely);

count = count+1;

end

%%%%%%%%%% MESH-INDEPENDENCY FILTER%%%%%%%%%%%%%%%%%%%%%%

function [dcf]=check(nelx,nely,rmin,x,dc)

dcf=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);

dcf(j,i) = dcf(j,i) + max(0,fac)\*dc(l,k);

end

end

dcf(j,i) = dcf(j,i)/sum;

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 AND SUPPORTS (Simply supported beam)

F((nelx\*(nely+1)+1),1)=-1.0;

fixeddofs=[(2\*(nely+1)-1),2\*(nely+1),(2\*(nelx+1)\*(nely+1)-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;

%%%%%%%%%% 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)];

**BESO METHOD FOR CONCENTRATED LOAD ON CANTILEVER BEAM:**

[Refer [1] in Bibliography, Pg.48]

This the original code that have been taken directly from the book.

%%%% A SOFT-KILL BESO METHOD %%%%

function sbeso(nelx,nely,volfrac,er,rmin)

% INITIALIZE

x(1:nely,1:nelx) = 1.; vol=1.; i = 0; change = 1.; penal = 3.;

% START iTH ITERATION

while change > 0.001

i = i + 1; vol = max(vol\*(1-er),volfrac);

if i >1; olddc = dc; end

% FE-ANALYSIS

[U]=FE(nelx,nely,x,penal);

% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

[KE] = lk;

c(i) = 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(i) = c(i) + 0.5\*x(ely,elx)^penal\*Ue'\*KE\*Ue;

dc(ely,elx) = 0.5\*x(ely,elx)^(penal-1)\*Ue'\*KE\*Ue;

end

end

% FILTERING OF SENSITIVITIES

[dc] = check(nelx,nely,rmin,x,dc);

% STABLIZATION OF EVOLUTIONARY PROCESS

if i > 1; dc = (dc+olddc)/2.; end

% BESO DESIGN UPDATE

[x] = ADDDEL(nelx,nely,vol,dc,x);

% PRINT RESULTS

if i>10;

change=abs(sum(c(i-9:i-5))-sum(c(i-4:i)))/sum(c(i-4:i));

end

disp([' It.: ' sprintf('%4i',i) ' Obj.:' sprintf('%10.4f',c(i))...

' 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

%%%%%%%%%% OPTIMALITY CRITERIA UPDATE%%%%%%%%%%%%%%%%%%%%

function [x]=ADDDEL(nelx,nely,volfra,dc,x)

l1 = min(min(dc)); l2 = max(max(dc));

while ((l2-l1)/l2 > 1.0e-5)

th = (l1+l2)/2.0;

x = max(0.001,sign(dc-th));

if sum(sum(x))-volfra\*(nelx\*nely) > 0;

l1 = th;

else

l2 = th;

end

end

%%%%%%%%%% MESH-INDEPENDENCY FILTER%%%%%%%%%%%%%%%%%%%%%%

function [dcf]=check(nelx,nely,rmin,x,dc)

dcf=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);

dcf(j,i) = dcf(j,i) + max(0,fac)\*dc(l,k);

end

end

dcf(j,i) = dcf(j,i)/sum;

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 AND SUPPORTS (Cantilever)

F(2\*(nelx+1)\*(nely+1)-nely,1)=-1.0;

fixeddofs=[1:2\*(nely+1)];

alldofs = [1:2\*(nely+1)\*(nelx+1)];

freedofs = setdiff(alldofs,fixeddofs);

% SOLVING

U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);

U(fixeddofs,:)= 0;

%%%%%%%%%% 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)];

**BESO METHOD FOR CONCENTRATED LOAD ON CLAMPED BEAM:**

In this code, the changes are made in the “**Define Loads and Supports**” section of the program from the original one.

%%%% A SOFT-KILL BESO METHOD %%%%

function sbesossb(nelx,nely,volfrac,er,rmin)

% INITIALIZE

x(1:nely,1:nelx) = 1.; vol=1.; i = 0; change = 1.; penal = 3.;

% START iTH ITERATION

while change > 0.001

i = i + 1; vol = max(vol\*(1-er),volfrac);

if i >1; olddc = dc; end

% FE-ANALYSIS

[U]=FE(nelx,nely,x,penal);

% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

[KE] = lk;

c(i) = 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(i) = c(i) + 0.5\*x(ely,elx)^penal\*Ue'\*KE\*Ue;

dc(ely,elx) = 0.5\*x(ely,elx)^(penal-1)\*Ue'\*KE\*Ue;

end

end

% FILTERING OF SENSITIVITIES

[dc] = check(nelx,nely,rmin,x,dc);

% STABLIZATION OF EVOLUTIONARY PROCESS

if i > 1; dc = (dc+olddc)/2.; end

% BESO DESIGN UPDATE

[x] = ADDDEL(nelx,nely,vol,dc,x);

% PRINT RESULTS

if i>10

change=abs(sum(c(i-9:i-5))-sum(c(i-4:i)))/sum(c(i-4:i));

end

disp([' It.: ' sprintf('%4i',i) ' Obj.:' sprintf('%10.4f',c(i))...

' 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

%%%%%%%%%% OPTIMALITY CRITERIA UPDATE%%%%%%%%%%%%%%%%%%%%

function [x]=ADDDEL(nelx,nely,volfra,dc,x)

l1 = min(min(dc)); l2 = max(max(dc));

while ((l2-l1)/l2 > 1.0e-5)

th = (l1+l2)/2.0;

x = max(0.001,sign(dc-th));

if sum(sum(x))-volfra\*(nelx\*nely) > 0

l1 = th;

else

l2 = th;

end

end

%%%%%%%%%% MESH-INDEPENDENCY FILTER%%%%%%%%%%%%%%%%%%%%%%

function [dcf]=check(nelx,nely,rmin,x,dc)

dcf=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);

dcf(j,i) = dcf(j,i) + max(0,fac)\*dc(l,k);

end

end

dcf(j,i) = dcf(j,i)/sum;

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 AND SUPPORTS (Simply supported clamped beam)

F((nelx\*(nely+1)+1),1)=-1.0;

fixeddofs=[(2\*(nely+1)-1),2\*(nely+1),(2\*(nelx+1)\*(nely+1)-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;

%%%%%%%%%% 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)];

**BESO METHOD FOR CONCENTRATED LOAD ON SIMPLY SUPPORTED BEAM:**

In this code, the changes are made in the “**Define Loads and Supports**” section of the program from the original one.

%%%% A SOFT-KILL BESO METHOD %%%%

function sbesossb(nelx,nely,volfrac,er,rmin)

% INITIALIZE

x(1:nely,1:nelx) = 1.; vol=1.; i = 0; change = 1.; penal = 3.;

% START iTH ITERATION

while change > 0.001

i = i + 1; vol = max(vol\*(1-er),volfrac);

if i >1; olddc = dc; end

% FE-ANALYSIS

[U]=FE(nelx,nely,x,penal);

% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

[KE] = lk;

c(i) = 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(i) = c(i) + 0.5\*x(ely,elx)^penal\*Ue'\*KE\*Ue;

dc(ely,elx) = 0.5\*x(ely,elx)^(penal-1)\*Ue'\*KE\*Ue;

end

end

% FILTERING OF SENSITIVITIES

[dc] = check(nelx,nely,rmin,x,dc);

% STABLIZATION OF EVOLUTIONARY PROCESS

if i > 1; dc = (dc+olddc)/2.; end

% BESO DESIGN UPDATE

[x] = ADDDEL(nelx,nely,vol,dc,x);

% PRINT RESULTS

if i>10

change=abs(sum(c(i-9:i-5))-sum(c(i-4:i)))/sum(c(i-4:i));

end

disp([' It.: ' sprintf('%4i',i) ' Obj.:' sprintf('%10.4f',c(i))...

' 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

%%%%%%%%%% OPTIMALITY CRITERIA UPDATE%%%%%%%%%%%%%%%%%%%%

function [x]=ADDDEL(nelx,nely,volfra,dc,x)

l1 = min(min(dc)); l2 = max(max(dc));

while ((l2-l1)/l2 > 1.0e-5)

th = (l1+l2)/2.0;

x = max(0.001,sign(dc-th));

if sum(sum(x))-volfra\*(nelx\*nely) > 0

l1 = th;

else

l2 = th;

end

end

%%%%%%%%%% MESH-INDEPENDENCY FILTER%%%%%%%%%%%%%%%%%%%%%%

function [dcf]=check(nelx,nely,rmin,x,dc)

dcf=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);

dcf(j,i) = dcf(j,i) + max(0,fac)\*dc(l,k);

end

end

dcf(j,i) = dcf(j,i)/sum;

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 AND SUPPORTS (Simply supported beam at edges)

F((nelx\*(nely+1)+1),1)=-1.0;

fixeddofs=horzcat(1:2\*(nely+1),(2\*nelx\*(nely+1))+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;

%%%%%%%%%% 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)];

**BESO METHOD FOR CONCENTRATED LOAD ON SIMPLY SUPPORTED BEAM WITH ROLLER SUPPORT ON ONE END:**

In this code, the changes are made in the “**Define Loads and Supports**” section of the program from the original one.

%%%% A SOFT-KILL BESO CODE FOR ROLLER SUPPORT BEAM %%%%

function sbesossb\_with\_roller(nelx,nely,volfrac,er,rmin)

% INITIALIZE

x(1:nely,1:nelx) = 1.; vol=1.; i = 0; change = 1.; penal = 3.;

% START iTH ITERATION

while i < 100

i = i + 1; vol = max(vol\*(1-er),volfrac);

if i >1; olddc = dc; end

% FE-ANALYSIS

[U]=FE(nelx,nely,x,penal);

% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

[KE] = lk;

c(i) = 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(i) = c(i) + 0.5\*x(ely,elx)^penal\*Ue'\*KE\*Ue;

dc(ely,elx) = 0.5\*x(ely,elx)^(penal-1)\*Ue'\*KE\*Ue;

end

end

% FILTERING OF SENSITIVITIES

[dc] = check(nelx,nely,rmin,x,dc);

% STABLIZATION OF EVOLUTIONARY PROCESS

if i > 1; dc = (dc+olddc)/2.; end

% BESO DESIGN UPDATE

[x] = ADDDEL(nelx,nely,vol,dc,x);

% PRINT RESULTS

if i>10

change=abs(sum(c(i-9:i-5))-sum(c(i-4:i)))/sum(c(i-4:i));

end

disp([' It.: ' sprintf('%4i',i) ' Obj.:' sprintf('%10.4f',c(i))...

' 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

%%%%%%%%%% OPTIMALITY CRITERIA UPDATE%%%%%%%%%%%%%%%%%%%%

function [x]=ADDDEL(nelx,nely,volfra,dc,x)

l1 = min(min(dc)); l2 = max(max(dc));

while ((l2-l1)/l2 > 1.0e-5)

th = (l1+l2)/2.0;

x = max(0.001,sign(dc-th));

if sum(sum(x))-volfra\*(nelx\*nely) > 0

l1 = th;

else

l2 = th;

end

end

%%%%%%%%%% MESH-INDEPENDENCY FILTER%%%%%%%%%%%%%%%%%%%%%%

function [dcf]=check(nelx,nely,rmin,x,dc)

dcf=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);

dcf(j,i) = dcf(j,i) + max(0,fac)\*dc(l,k);

end

end

dcf(j,i) = dcf(j,i)/sum;

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 AND SUPPORTS (Simply supported beam with roller support)

F((nelx\*(nely+1)+1),1)=-1.0;

fixeddofs=[(2\*(nely+1)-1),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;

%%%%%%%%%% 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)];

**BESO METHOD FOR GRAVITY LOAD ON SIMPLY SUPPORTED BEAM:**

In this code, the changes are made in the “**FE Analysis**” and “**Define Loads and Supports**” sections of the program from the original one.

%%%% A SOFT-KILL BESO CODE FOR GRAVITY LOAD %%%%

function sbesog(nelx,nely,volfrac,er,rmin)

% INITIALIZE

x(1:nely,1:nelx) = 1.; vol=1.; i = 0; change = 1.; penal = 5;

% START iTH ITERATION

while sum(sum(x)) >= (volfrac\*nelx\*nely)

i = i + 1; vol = max(vol\*(1-er),volfrac);

if i >1; olddc = dc; end

% FE-ANALYSIS

[U]=FE(nelx,nely,x,penal);

% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

[KE] = lk;

c(i) = 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(i) = c(i) + 0.5\*x(ely,elx)^penal\*Ue'\*KE\*Ue;

fe = -[0 0.25 0 0.25 0 0.25 0 0.25];

% dc(ely,elx) = fe\*Ue + 0.5\*penal\*x(ely,elx)^(penal-1)\*Ue'\*KE\*Ue;

dc(ely,elx) = 0.5\*penal\*x(ely,elx)^(penal-1)\*Ue'\*KE\*Ue;

end

end

% FILTERING OF SENSITIVITIES

[dc] = check(nelx,nely,rmin,x,dc);

% STABLIZATION OF EVOLUTIONARY PROCESS

if i > 1; dc = (dc+olddc)/2.; end

% BESO DESIGN UPDATE

[x] = ADDDEL(nelx,nely,vol,dc,x);

% PRINT RESULTS

if i>10

change=abs(sum(c(i-9:i-5))-sum(c(i-4:i)))/sum(c(i-4:i));

end

disp([' It.: ' sprintf('%4i',i) ' Obj.:' sprintf('%10.4f',c(i))...

' 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

%%%%%%%%%% OPTIMALITY CRITERIA UPDATE%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [x]=ADDDEL(nelx,nely,volfra,dc,x)

l1 = min(min(dc)); l2 = max(max(dc));

while ((l2-l1)/l2 > 1.0e-5)

th = (l1+l2)/2.0;

x = max(0.001,sign(dc-th));

if sum(sum(x))-volfra\*(nelx\*nely) > 0

l1 = th;

else

l2 = th;

end

end

%%%%%%%%%% MESH-INDEPENDENCY FILTER%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [dcf]=check(nelx,nely,rmin,x,dc)

dcf=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);

dcf(j,i) = dcf(j,i) + max(0,fac)\*dc(l,k);

end

end

dcf(j,i) = dcf(j,i)/sum;

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;

FE = -9.81\*[0 0.25 0 0.25 0 0.25 0 0.25]';

F(edof) = F(edof) + x(ely,elx)^penal\*FE;

end

end

% DEFINE LOADS AND SUPPORTS (Simply supported beam)

fixeddofs=[1,2,2\*(nelx)\*(nely+1)+1,2\*nelx\*(nely+1)+2];

alldofs = [1:2\*(nely+1)\*(nelx+1)];

freedofs = setdiff(alldofs,fixeddofs);

% SOLVING

U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);

U(fixeddofs,:)= 0;

%%%%%%%%%% ELEMENT STIFFNESS MATRIX%%%%%%%%%%%%%%%%%%%%%%

function [KE]=lk

E = 1e9;

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)];

**BESO METHOD FOR GRAVITY AND CONCENTRATED LOAD ON SIMPLY SUPPORTED BEAM:**

In this code, the changes are made in the “**FE Analysis**” and “**Define Loads and Supports**” sections of the program from the original one.

%%%% A SOFT-KILL BESO CODE FOR GRAVITY AND CONCENTRATED LOAD %%%%

function sbesog\_with\_concload(nelx,nely,volfrac,er,rmin)

% INITIALIZE

x(1:nely,1:nelx) = 1.; vol=1.; i = 0; change = 1.; penal = 5;

% START iTH ITERATION

while sum(sum(x)) >= (volfrac\*nelx\*nely)

i = i + 1; vol = max(vol\*(1-er),volfrac);

if i >1; olddc = dc; end

% FE-ANALYSIS

[U]=FE(nelx,nely,x,penal);

% OBJECTIVE FUNCTION AND SENSITIVITY ANALYSIS

[KE] = lk;

c(i) = 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(i) = c(i) + 0.5\*x(ely,elx)^penal\*Ue'\*KE\*Ue;

fe = -[0 0.25 0 0.25 0 0.25 0 0.25];

dc(ely,elx) = 0.5\*penal\*x(ely,elx)^(penal-1)\*Ue'\*KE\*Ue;

end

end

% FILTERING OF SENSITIVITIES

[dc] = check(nelx,nely,rmin,x,dc);

% STABLIZATION OF EVOLUTIONARY PROCESS

if i > 1; dc = (dc+olddc)/2.; end

% BESO DESIGN UPDATE

[x] = ADDDEL(nelx,nely,vol,dc,x);

% PRINT RESULTS

if i>10

change=abs(sum(c(i-9:i-5))-sum(c(i-4:i)))/sum(c(i-4:i));

end

disp([' It.: ' sprintf('%4i',i) ' Obj.:' sprintf('%10.4f',c(i))...

' 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

%%%%%%%%%% OPTIMALITY CRITERIA UPDATE%%%%%%%%%%%%%%%%%%%%

function [x]=ADDDEL(nelx,nely,volfra,dc,x)

l1 = min(min(dc)); l2 = max(max(dc));

while ((l2-l1)/l2 > 1.0e-5)

th = (l1+l2)/2.0;

x = max(0.001,sign(dc-th));

if sum(sum(x))-volfra\*(nelx\*nely) > 0

l1 = th;

else

l2 = th;

end

end

%%%%%%%%%% MESH-INDEPENDENCY FILTER%%%%%%%%%%%%%%%%%%%%%%

function [dcf]=check(nelx,nely,rmin,x,dc)

dcf=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);

dcf(j,i) = dcf(j,i) + max(0,fac)\*dc(l,k);

end

end

dcf(j,i) = dcf(j,i)/sum;

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;

FE = -9.81\*[0 0.25 0 0.25 0 0.25 0 0.25]';

F(edof) = F(edof) + x(ely,elx)^penal\*FE;

end

end

% DEFINE LOADS AND SUPPORTS (Simply supported beam with gravity and concentrated load)

F(nelx\*(nely+1)+2,1)=1.0;

fixeddofs=[1,2,2\*(nelx)\*(nely+1)+2];

alldofs = [1:2\*(nely+1)\*(nelx+1)];

freedofs = setdiff(alldofs,fixeddofs);

% SOLVING

U(freedofs,:) = K(freedofs,freedofs) \ F(freedofs,:);

U(fixeddofs,:)= 0;

%%%%%%%%%% ELEMENT STIFFNESS MATRIX%%%%%%%%%%%%%%%%%%%%%%

function [KE]=lk

E = 1e9;

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)];

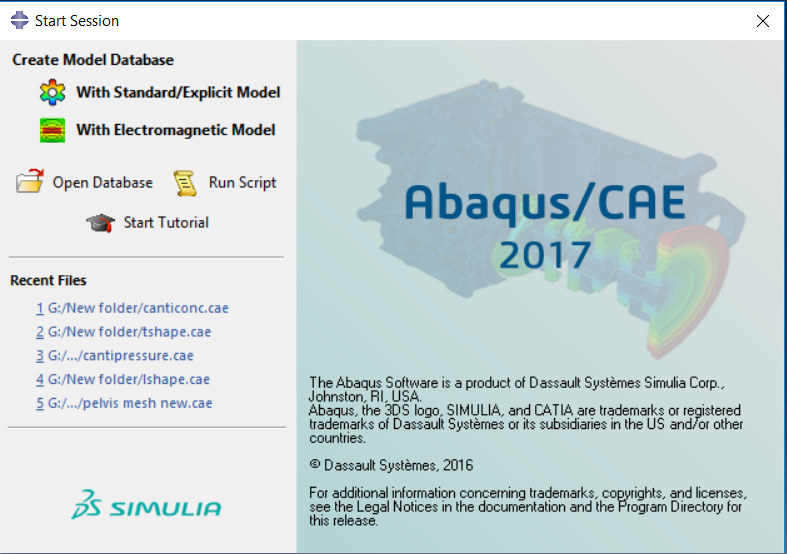
**CONCLUSION**

Topology optimization has many applications in present and future due to reduction of unwanted material usage. With the advent of rapid prototyping techniques, it is now possible to fabricate complicated topologies directly from computer models using 3D printers to reproduce structural details point-by-point and layer-by-layer. It enables architects and engineers to greatly expand the possible structural forms of their projects and to obtain designs that are not only structurally efficient but also exhibit distinctive aesthetical appeal. One could well see the day when more landmark structures are designed using ESO/BESO and constructed in many parts of the world. It will be a one of the most eminent fields in the upcoming years of research and applications.

**APPENDIX**

**TOPOLOGY OPTIMIZATION WITH THE HELP OF ABAQUS SOFTWARE:**

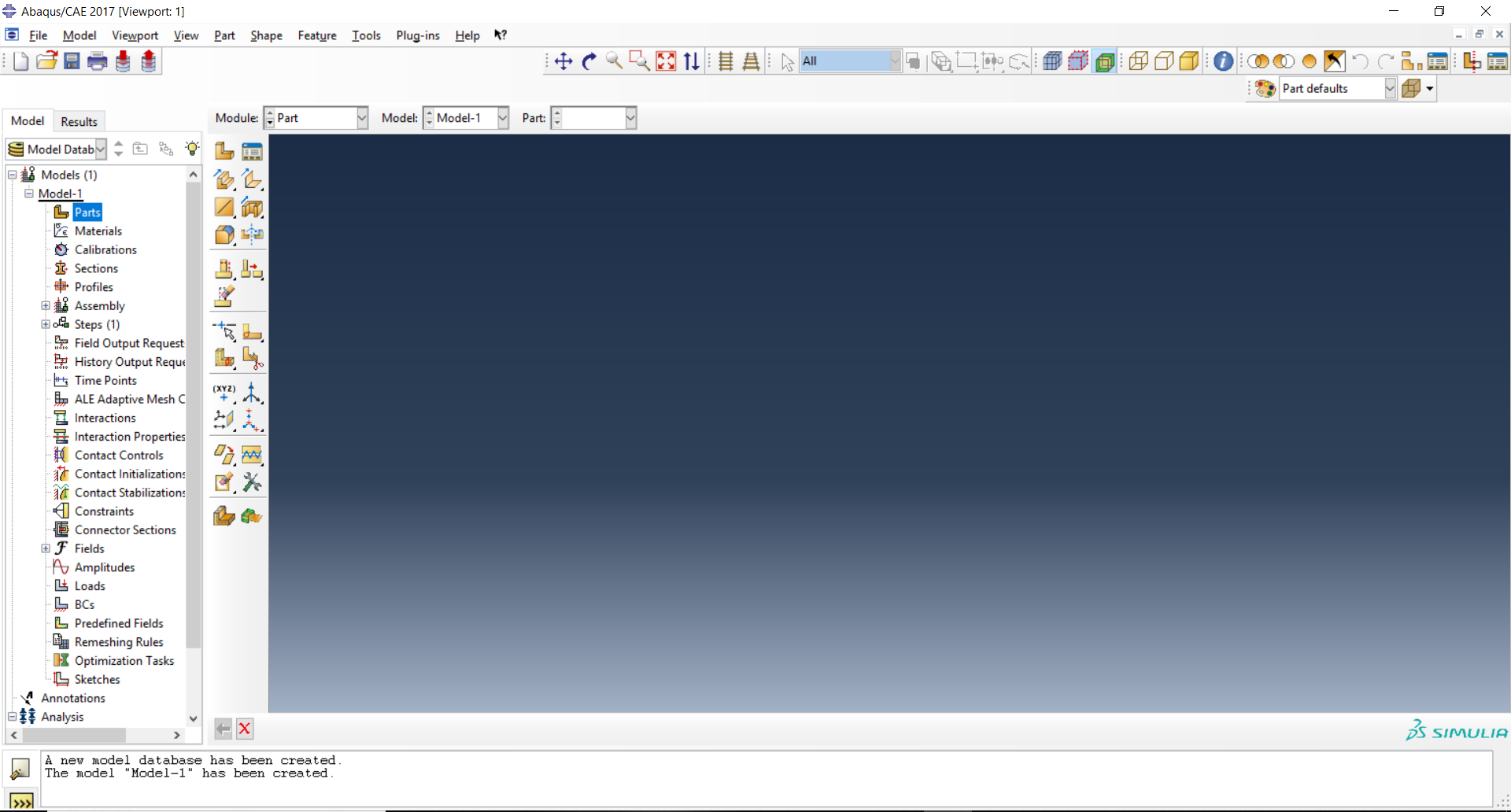
Once the Abaqus CAE software is opened there will be window which opens like the picture given below.



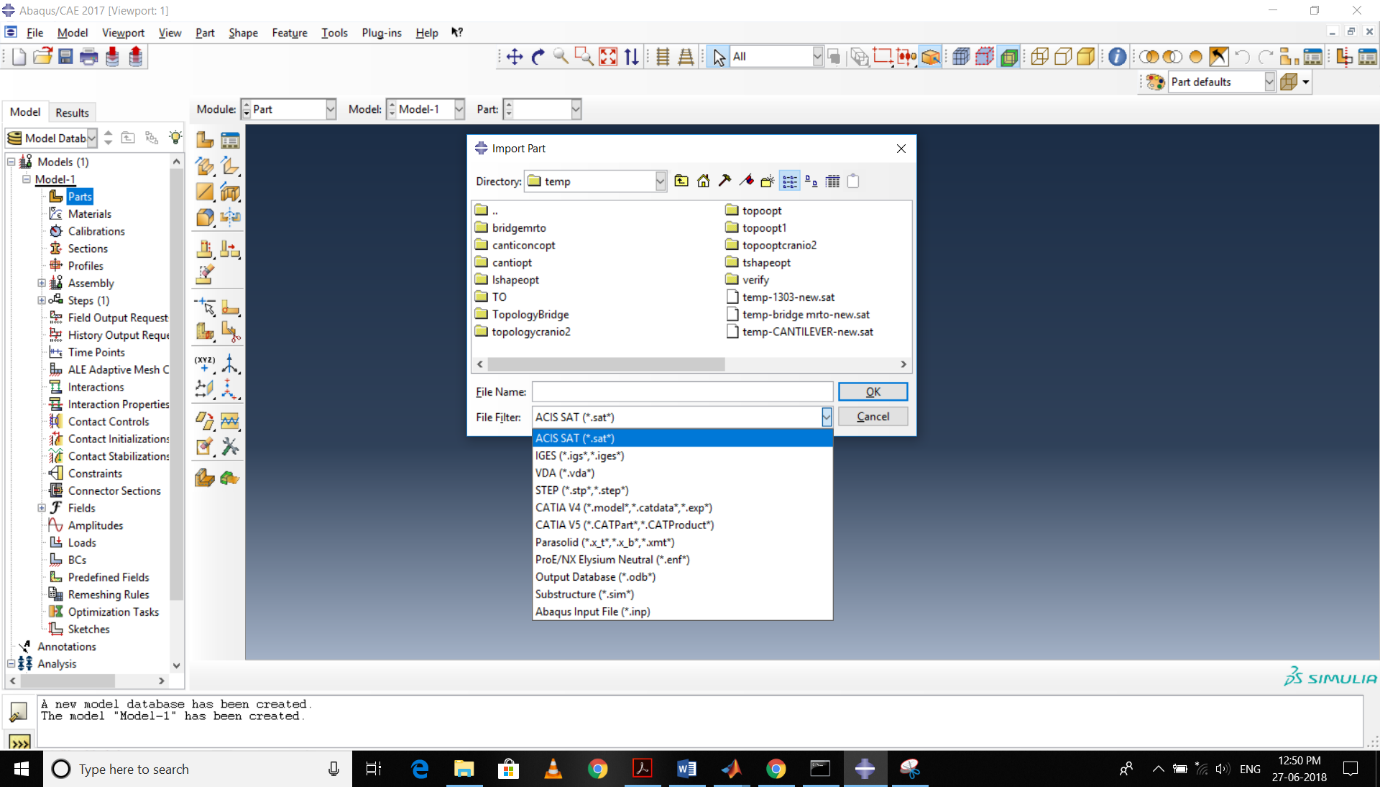
Select -> With Standard/Explicit Model.

Now you can either create a part or import a part file into this software. This can be done as follows:

1. When right click on the Part present in the Feature tree present in the left side of the screen.

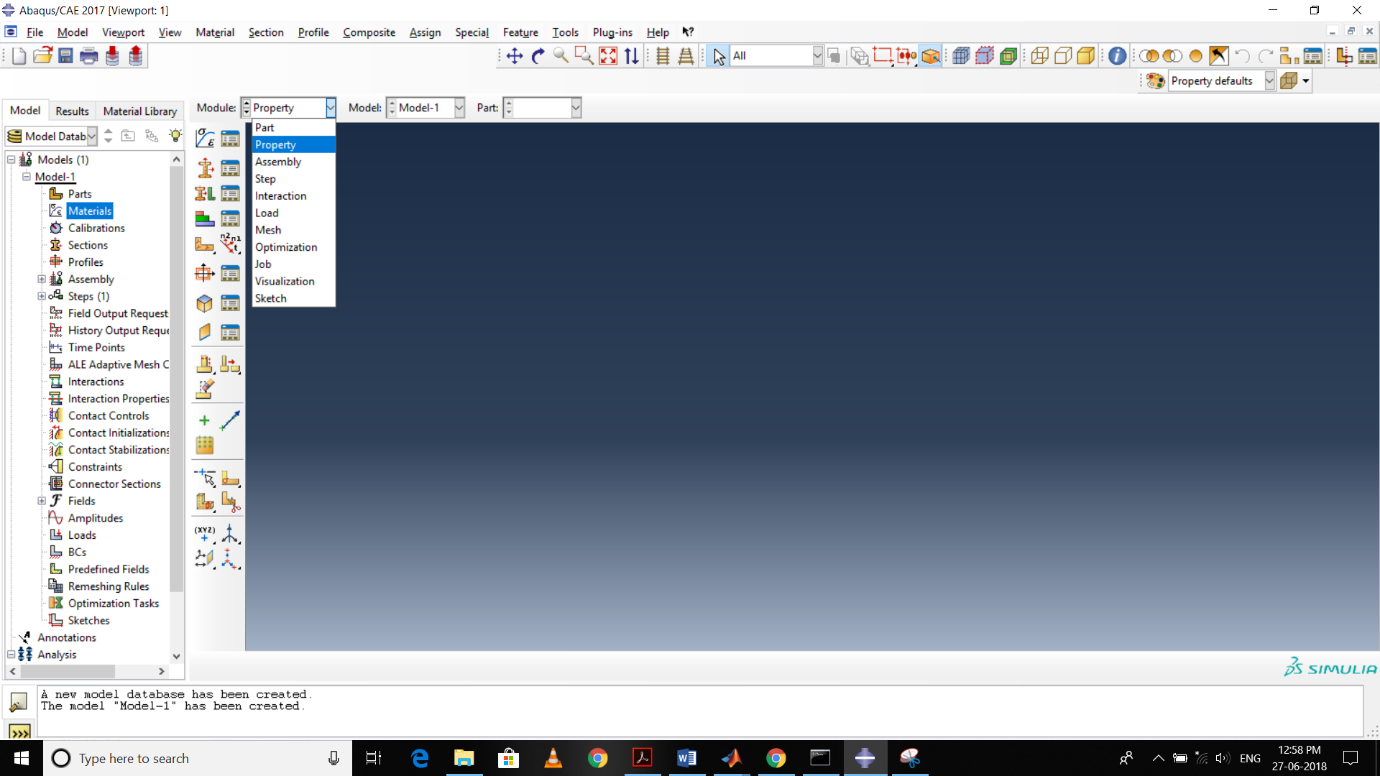


1. Select -> **Create** to create the part file for which we have to find the optimized results.
2. If you have already created a part file, the Select -> **Import**.

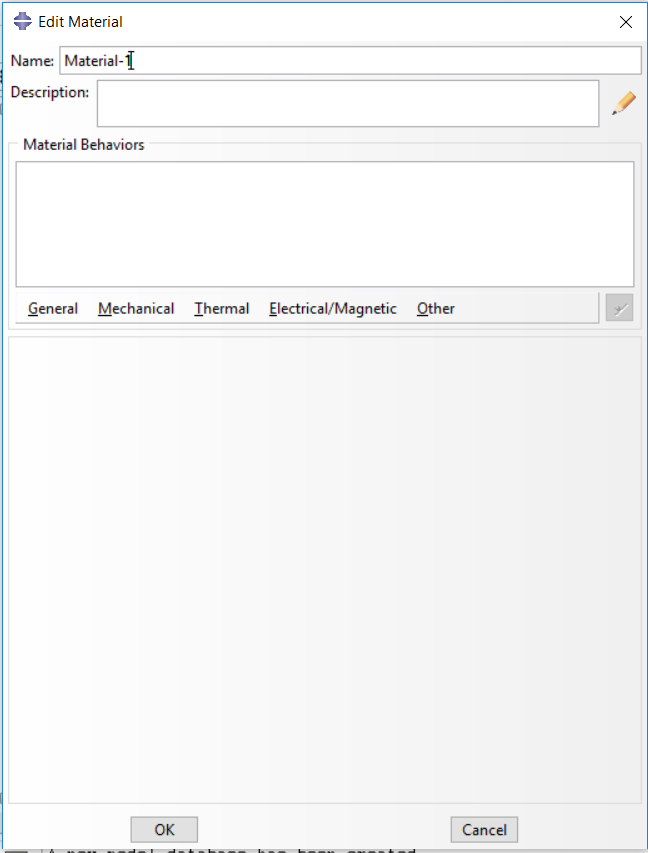


1. The parts with following extensions can only be imported to this software. If the part has extensions other than these then try to convert the part file into any one these extensions and import it.

After the creating or importing the part/assembly files, Click -> **Property** in the Module drop menu present above the workspace.



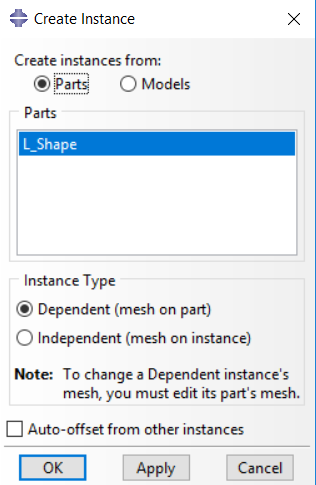
1. Once you have selected the Property, we have to define the properties of the materials which are going to assigned to the part/assembly files.
2. Select -> **Create material** present in the material feature tree. Define all the Mechanical, Thermal and Electric/Magnetic properties of the material here and Click -> OK.



1. Create as many materials as required for the optimization process.
2. Select -> **Create Section** in feature tree. Here you are going to create sections and provide the characteristics of each section.
3. Select -> **Assign Section**. Create -> **Set** and click on the respective part for each corresponding sections.

Click -> **Assembly** in Module drop menu.

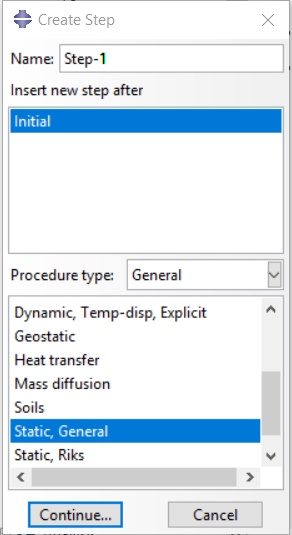
1. Select -> **Create Instance**.



This dialogue box will open and select either dependent or independent depending on your requirement. Most of the structural optimization cases are done with dependent assembly.

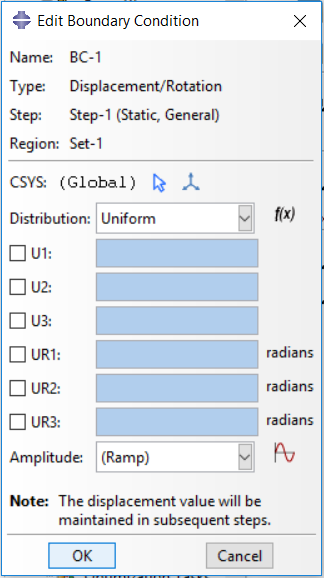
Now Select -> Step in the Module menu. Select -> **Create Step** in the step’s feature tree.

1. Select the required type of analysis which you are going to do for the structure.

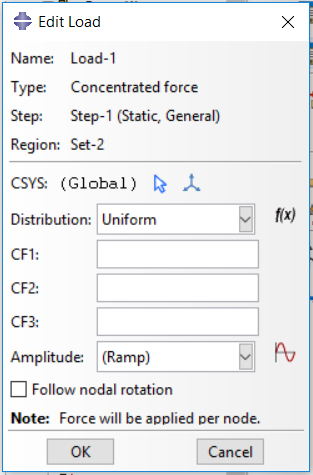


Then Select -> **Interaction**. If you have any interaction between surfaces of the parts, you can define them in this section. Select -> **Create Interaction**. Select -> **Create Interaction Property**. Specify the appropriate constraints by Select -> **Create Constraint**.

Loads for the structure can be prescribed by Select -> **Load** down the Module tree. Select -> **Create Boundary Condition** in left side menu. In category, Select -> **Mechanical** and in Types for Selected Step, Assign -> **Displacement and Rotation**. In select regions for boundary condition, specify the region where you wish to apply the boundary condition by selecting it.



The above mentioned dialogue box opens in the screen. Here, U1, U2, U3 represents the displacement degrees of freedom on the x, y and z directions respectively. UR1, UR2, UR3 represents the rotational degrees of freedom on the x, y and z directions respectively. Determine the degrees of freedom as per your conditions. In Amplitude, leave it as default. If you need to provide an amplitude, Click ->. Go to **Create Load** for giving the forces acting on the structure. Select -> Mechanical and in types for selected steps, specify the type of load which is enforced in the conditions. After that, you have to create a **Set** for indicating the point or area on which your given load is going to act on the structure.

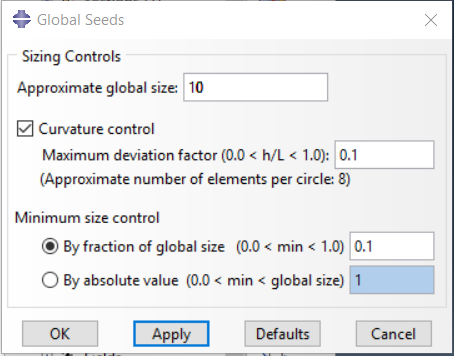


CF1, CF2, CF3 represents the loads which is acting on the x, y and z directions respectively as per the given conditions. . In Amplitude, leave it as default. If you need to provide an amplitude, Click ->.

Specifying all the loads, Click -> **Mesh**. Select -> **Part** as given below.

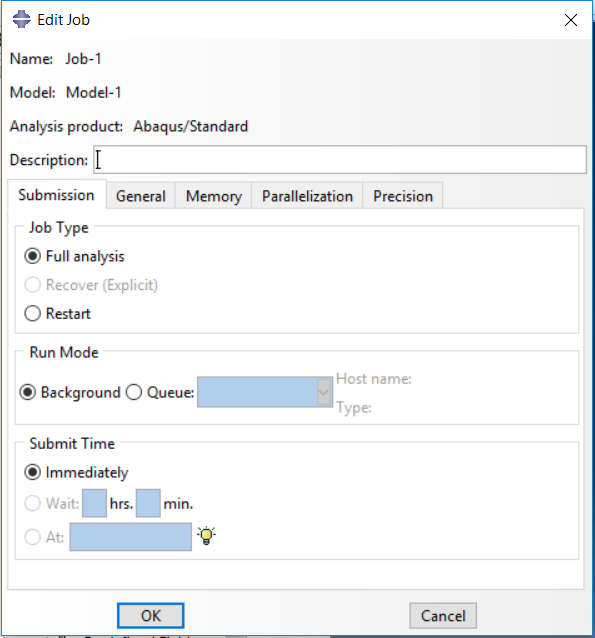


Thereafter, Click -> **Seed Part**. Consign the **Approximate global size** and other details (if you want to or leave as default) as per your concern.

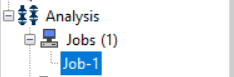


Click -> **Mesh Part**. Select **yes** in the OK to mesh part present below the working space. The part will be meshed and displayed on the screen.

Click -> **Job** in the Module drop menu. Click -> **Create Job**. If you have any preconditions to establish, fill in the respective boxes else proceed by Selecting -> **OK**.



Your job will be created. Now go to **Analysis** in the feature tree and click the **Plus** present to the left of it and again do the same for **Jobs**. Then you can notice your **Job**. **Right Click** on it and provide **Submit**. Your analysis you have determined before starts running. If you want to look at process behind Right Click on your Job and click **Monitor**.

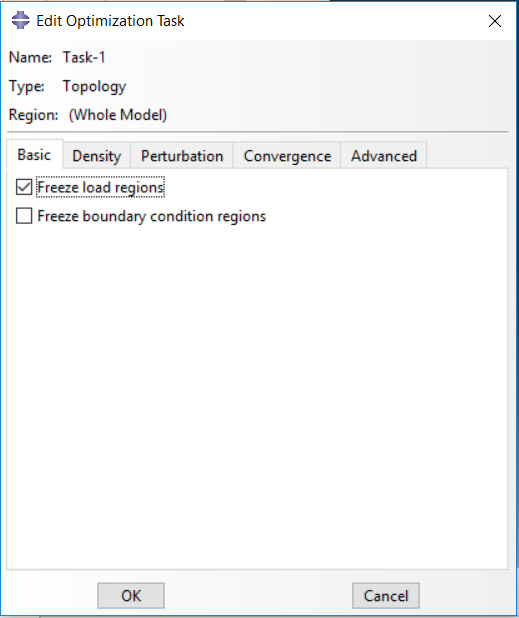


Once your job got completed you can view the results by **Right Click** on your job in the tree and click **Results**.

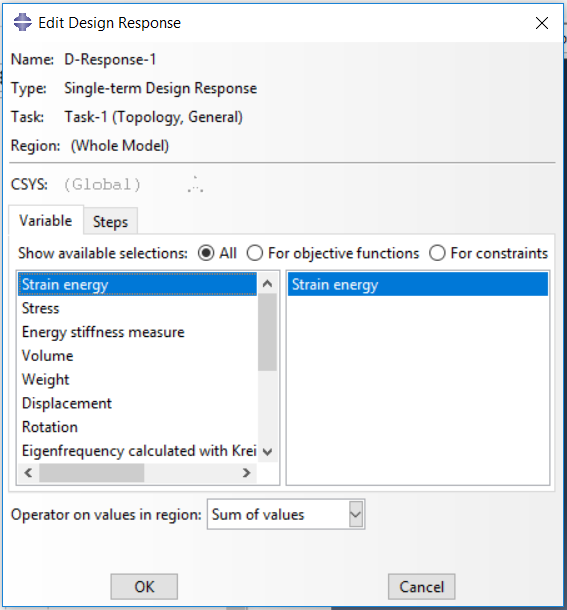
Directly Select -> **Optimization** in the Module drop Menu. Click -> Create optimization task. A dialogue box will open as given below.



Provide the type of the optimization which you have to do with the given part. Click -> **Continue**. Select the optimization region or clear to consider the Whole Model. You have to define a **Set** for the region. Soon this dialogue box will open where you to mention the conditions for optimization.

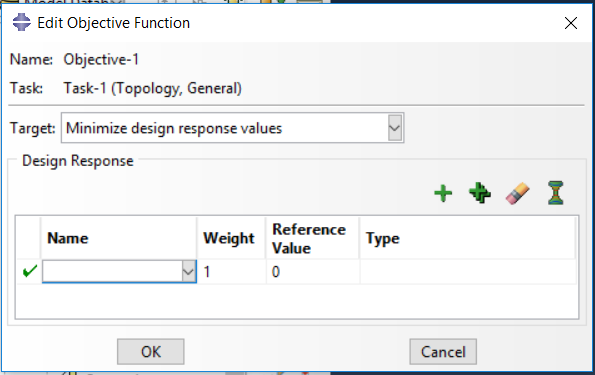


Click -> **OK** to continue for further steps. Your optimization task will be created. Click -> **Create Design Response** -> **Single term** to mention the type of variables for which the task is going to depend upon. In select the design response region type, give the areas of the design responses.

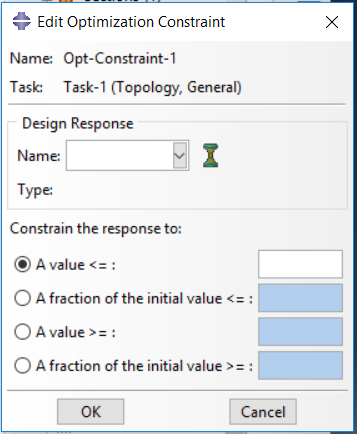


Provide the variable for this design response so that this response will denote the variable specified. **Repeat the above steps for further design variables** and at the last step select the variable accordingly.

Click -> **Create Objective Function**. Give the function a name and continue. Click the box below the **Name** and select the variable which you wish to give it as objective function in the drop down menu.



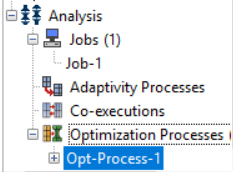
Click -> **Create Constraint**. Create a constraint and continue. Then below dialogue box will open where in **Name** you specify the variable which is the constraint and provide its values in any form in the types below the Name. If you already created a variable select from the drop down menu or create a new one by clicking the icon.



Once all these procedures are done then go to **Analysis**. **Right Click** on the **Optimization processes** and click -> **Create**. The dialogue box as below opens. Click -> **OK** or if you want to specify any details in the optimization process provide here and Click -> OK.



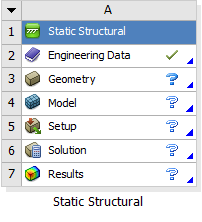
Go to **Analysis** and in **Optimization Processes** if you click the Plus near to it, you can view the optimization process you have defined as given below.



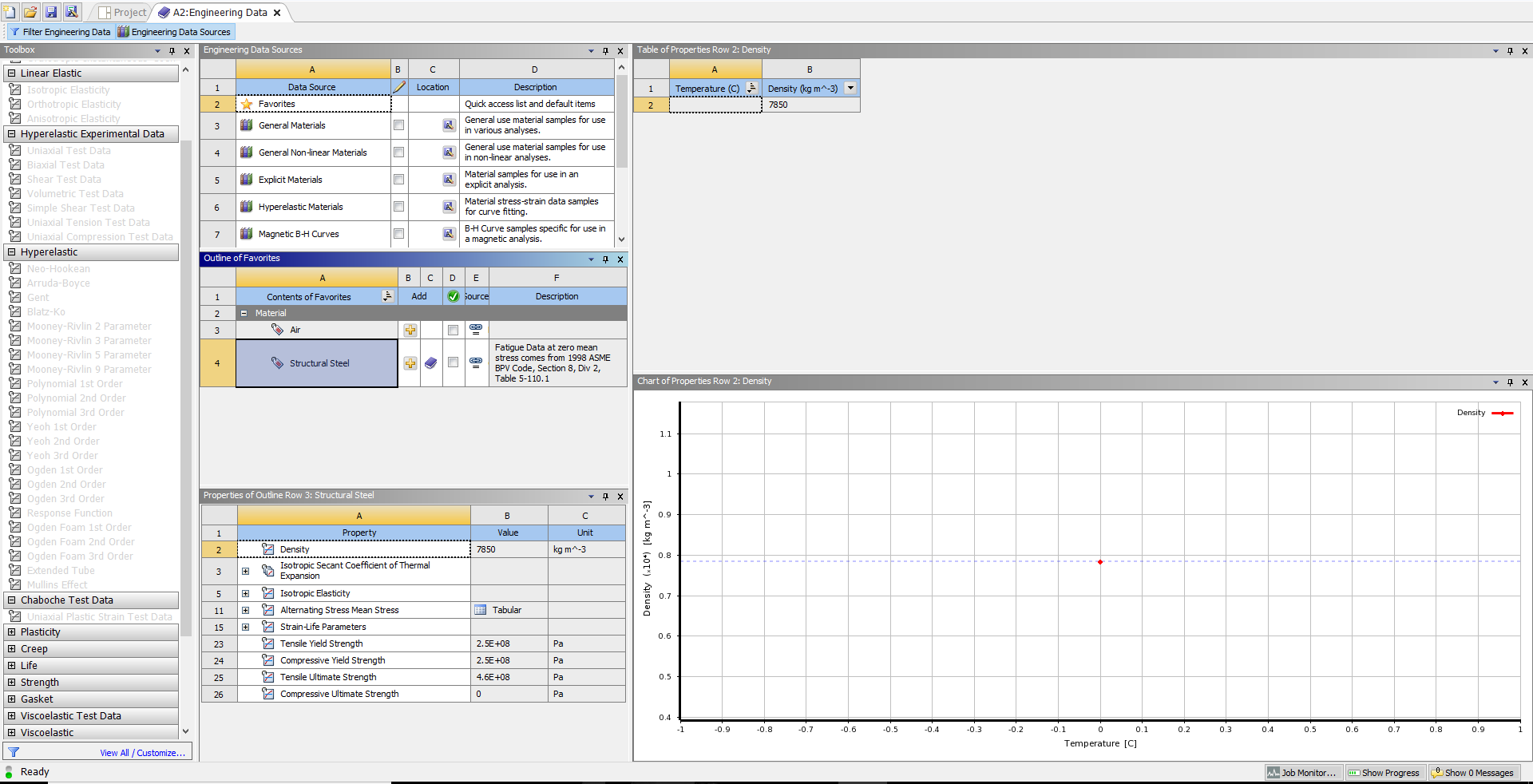
**Right Click** on your process and Click -> **Submit** to run the process. If you have submitted the task, it starts running. If you wish to look at the background process, Right Click on your task and Click -> **Monitor**. Once the process gets completed you can sight the results by Right Clicking on your task and Click -> **Results**. Or you can the Click -> **Visualization** in Module drop down menu. Hence obtained result is the topologically optimized result of the initially granted design.

**TOPOLOGY OPTIMIZATION WITH THE HELP OF ANSYS SOFTWARE:**

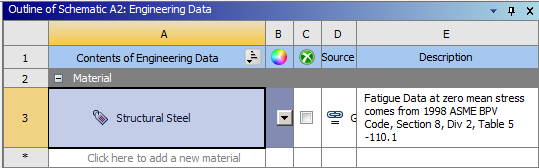
Once you have opened the ANSYS software, drag and drop the **Static Structural** option in the **Analysis Systems** to the **Project Schematic** workspace. After doing this you can see the box as given below in project schematic space.



Double click on the Engineering Data. Here you have to select the materials which you are going to use for your systems or you can create a new material on your own of the analysis.

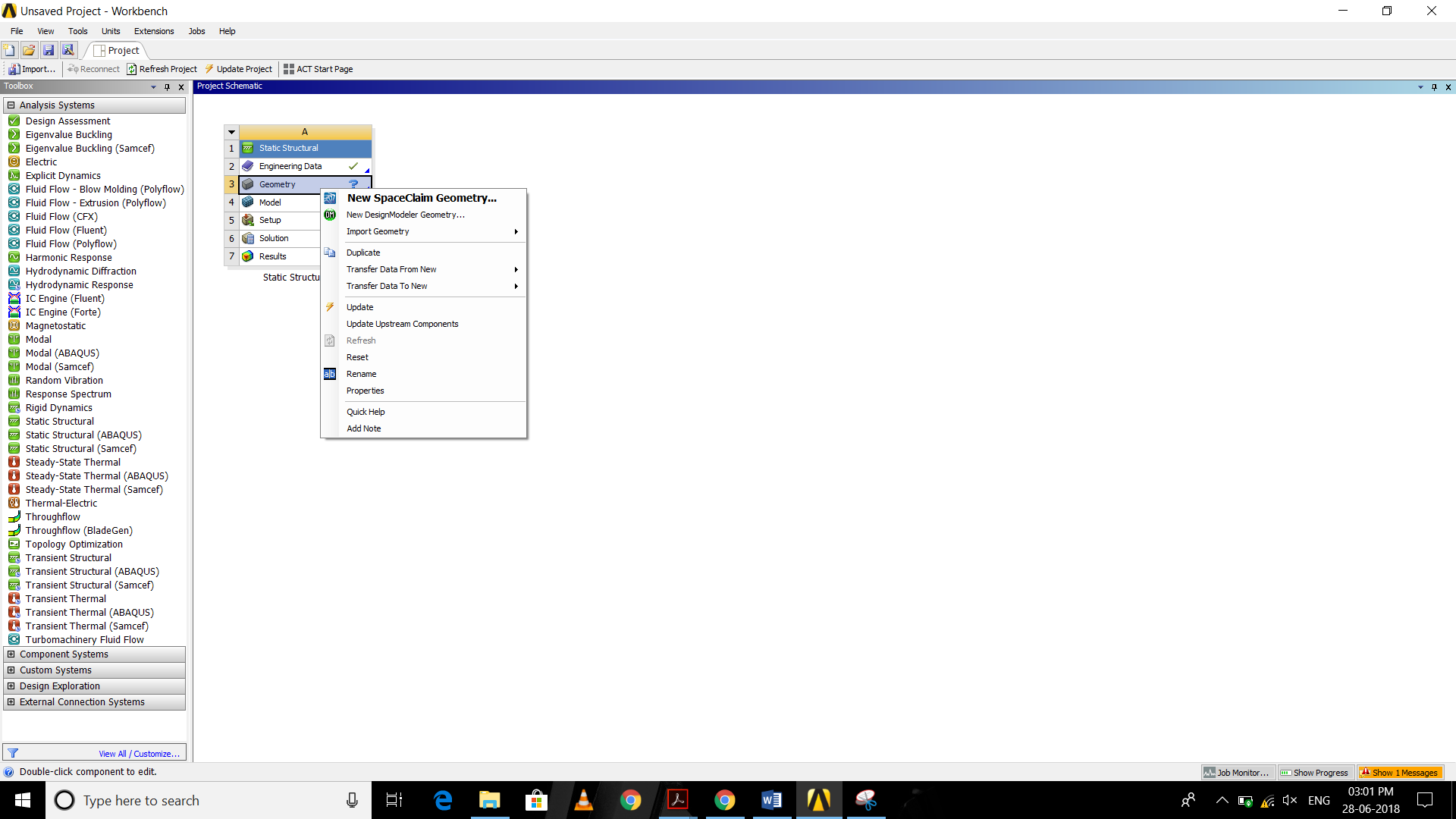


Click **Engineering Data Sources** present in the top left corner of the window for selecting the required material.



Click -> **Click here to add a new material** to provide a new material for analysis. After selecting the materials for the analysis, Close the Engineering Data Sources screen.

**Right Click** on the **Geometry**.

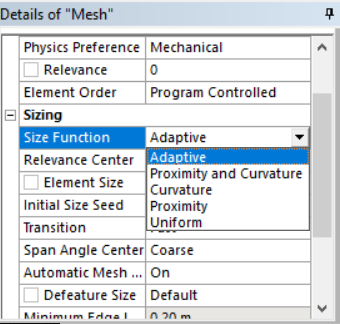


You will see this menu box opened. If you want to create a new part/assembly, then select **New Space Claim Geometry** or **New Design Modeller Geometry**. Otherwise if you want to import part or assembly files select **Import Geometry**. Change the extension as **All Geometry Files** for selecting any type of files. Click **Update Project** present above the Project Schematic space.

Double Click on the **Model**. Ansys Mechanical will be expanded as a new window.

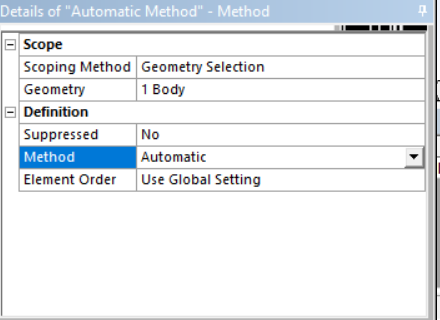


Click -> **Mesh** on the Outline menu. Click the plus icon present near the **Sizing**. In the **Size Function,** select the type of the mesh mandatory for the analysis.



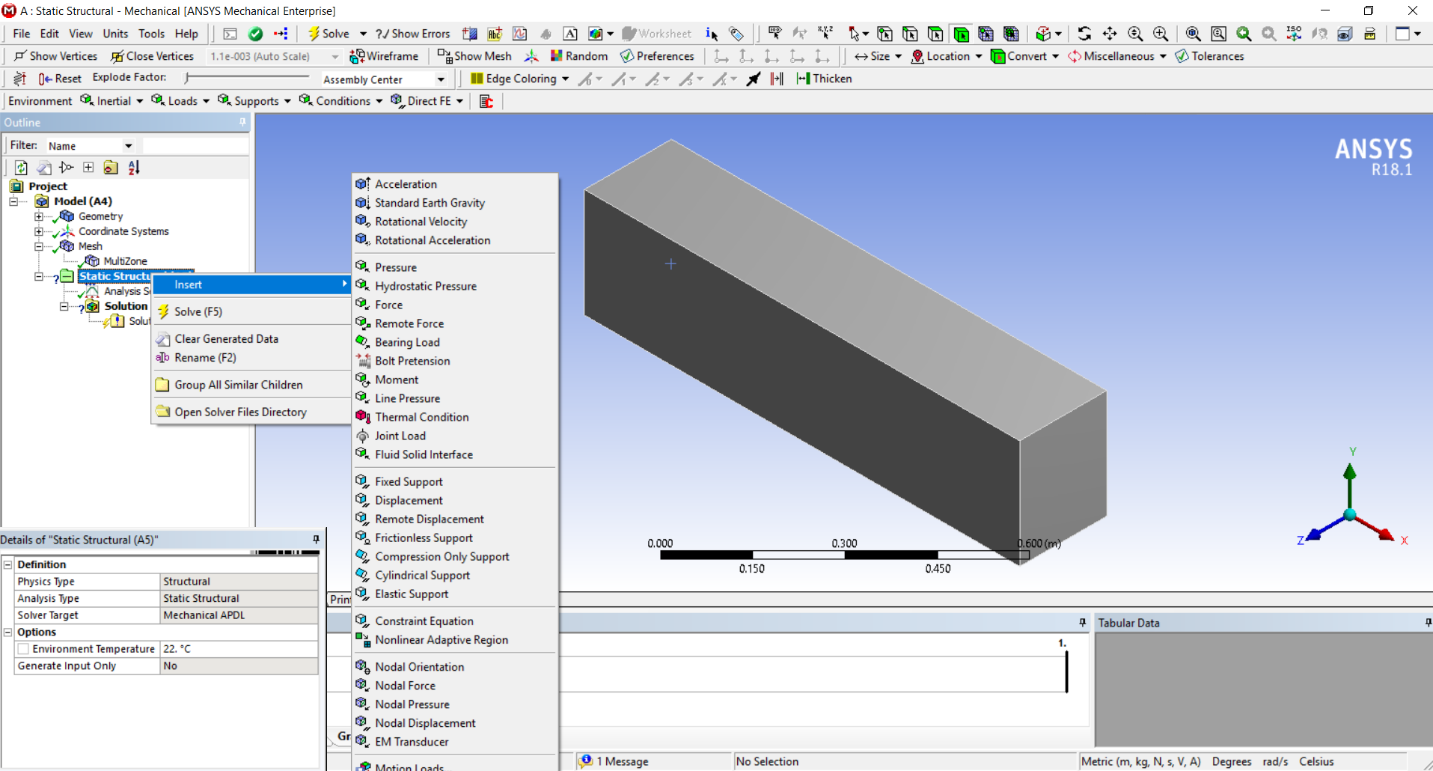
In the **Relevance Centre**, select the whether the mesh is to be a coarse, medium or a fine mesh. Specify the **Element Size** or leave it as default. Fill other details under the **Sizing** menu according to given conditions. Provide rest of the details in the **Details of Mesh** or continue with the default ones.

If you wish to define the type of element for the mesh, Right Click -> **Mesh**. Go to **Insert** where you can find an option named **Method**. In **Geometry**, select the whole body and Click -> Apply. In **Method**, Select the type of elements in the drop down menu. Prefer **Multi Zone** or **Hex Dominant** for better results since they are the ones mostly preferred by experts.



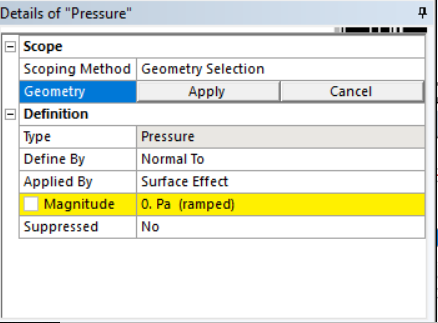
Determine other details or continue with the default settings. Then Right click -> Mesh and Select -> **Generate Mesh**. The part will be meshed for analysis.

Right Click on the **Static Structural (A5)** in the feature tree. Got to **Insert** and Select -> **Fixed Support** for providing the boundary conditions.

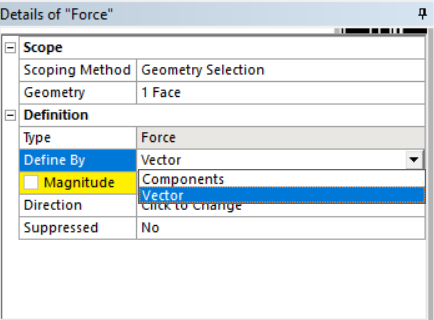


Select ->Surface, Point or Line where you have to apply the fixed support. Then Click -> Apply.

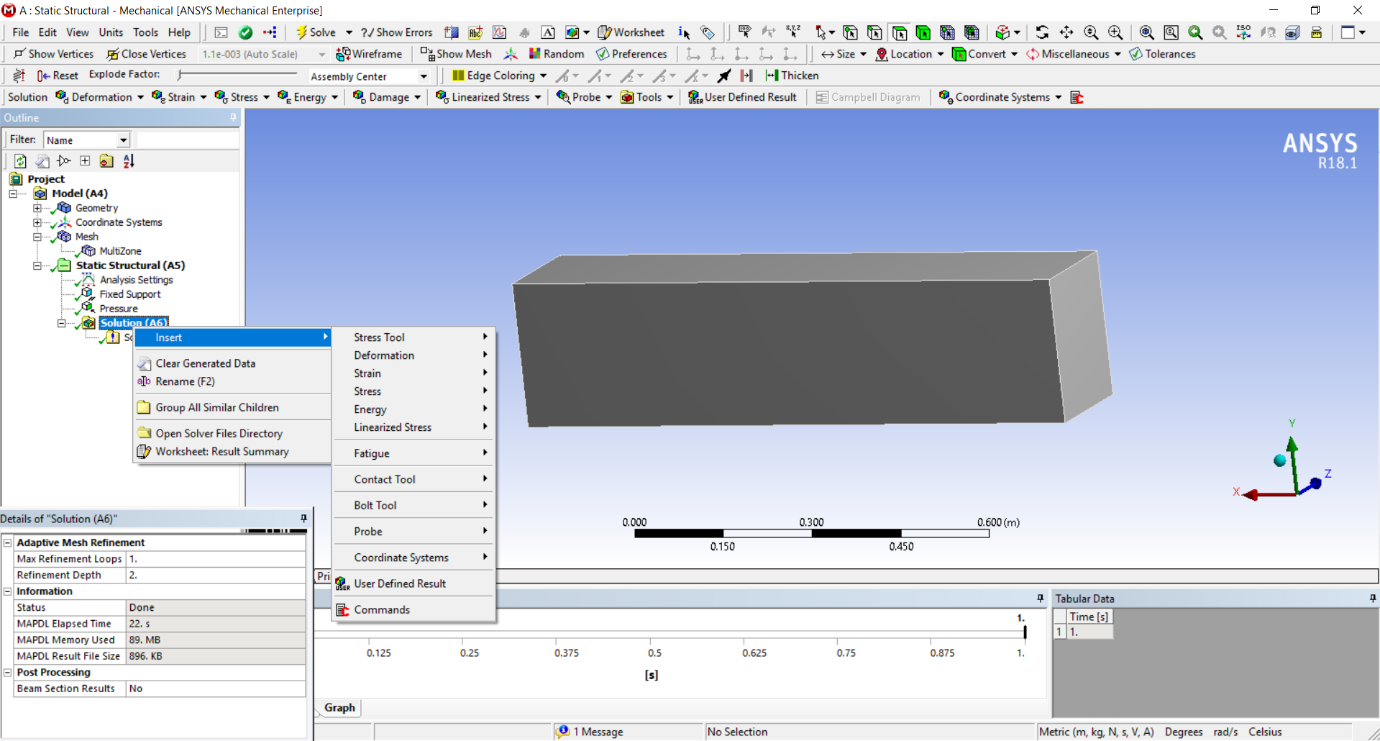
Again Right Click on the Static Structural and Select the kind of load which has to be applied on the part/assembly. Fill the **Details of Load** as per the conditions.



If you wish to provide the forces along each and every axes separately, Click -> **Define By** in the **Details of Force** and Select -> **Components**. Give the forces along each coordinate axes. Specify the direction and magnitude along the axes. Also define the **Applied By** row in the details.

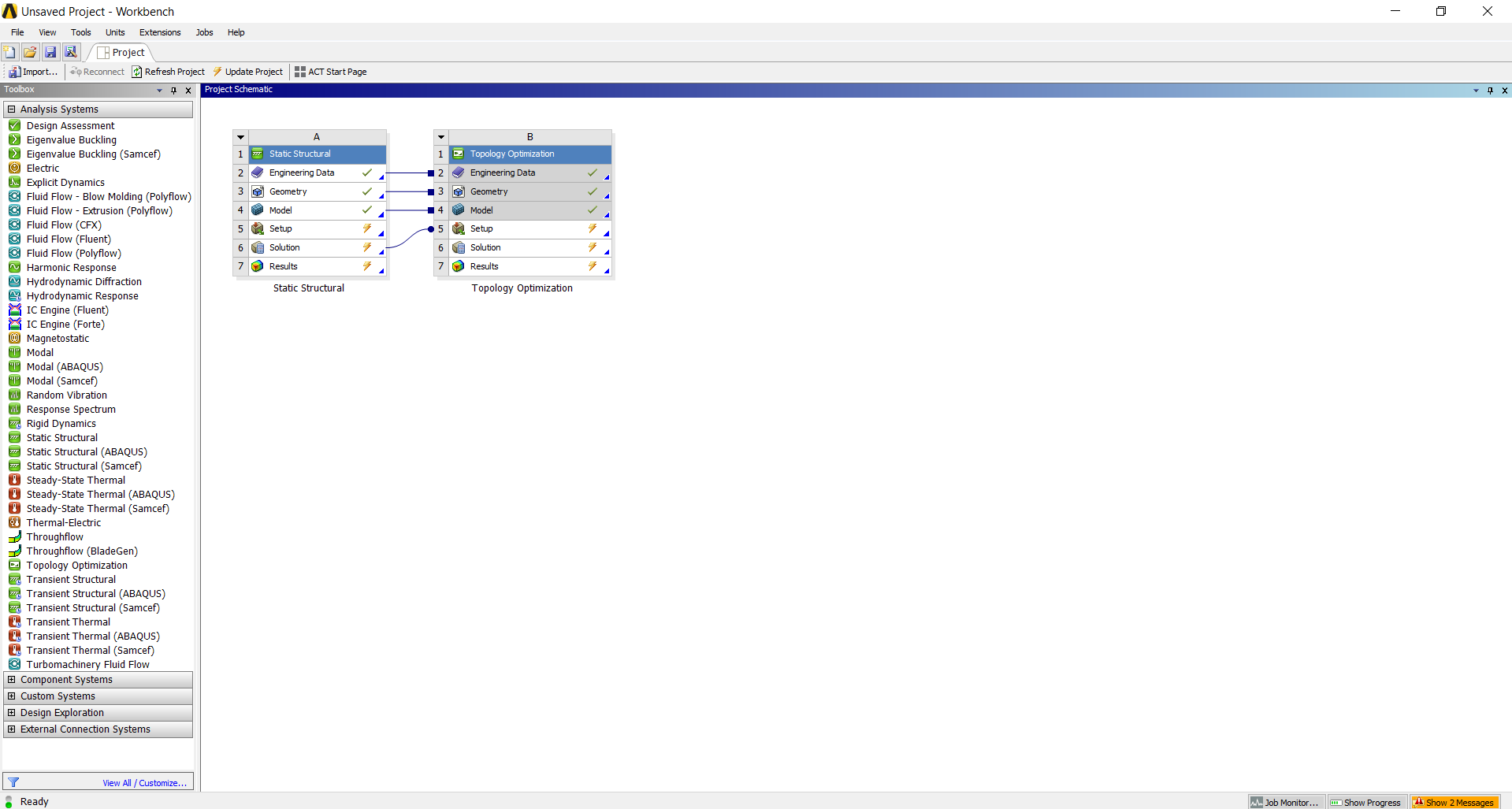


After providing all the boundary conditions and the loads for the structure, Right Click -> **Solution** **(A6)** present below the **Static Structural (A5)**. Select -> **Insert** and mention the category of the results you are looking for.

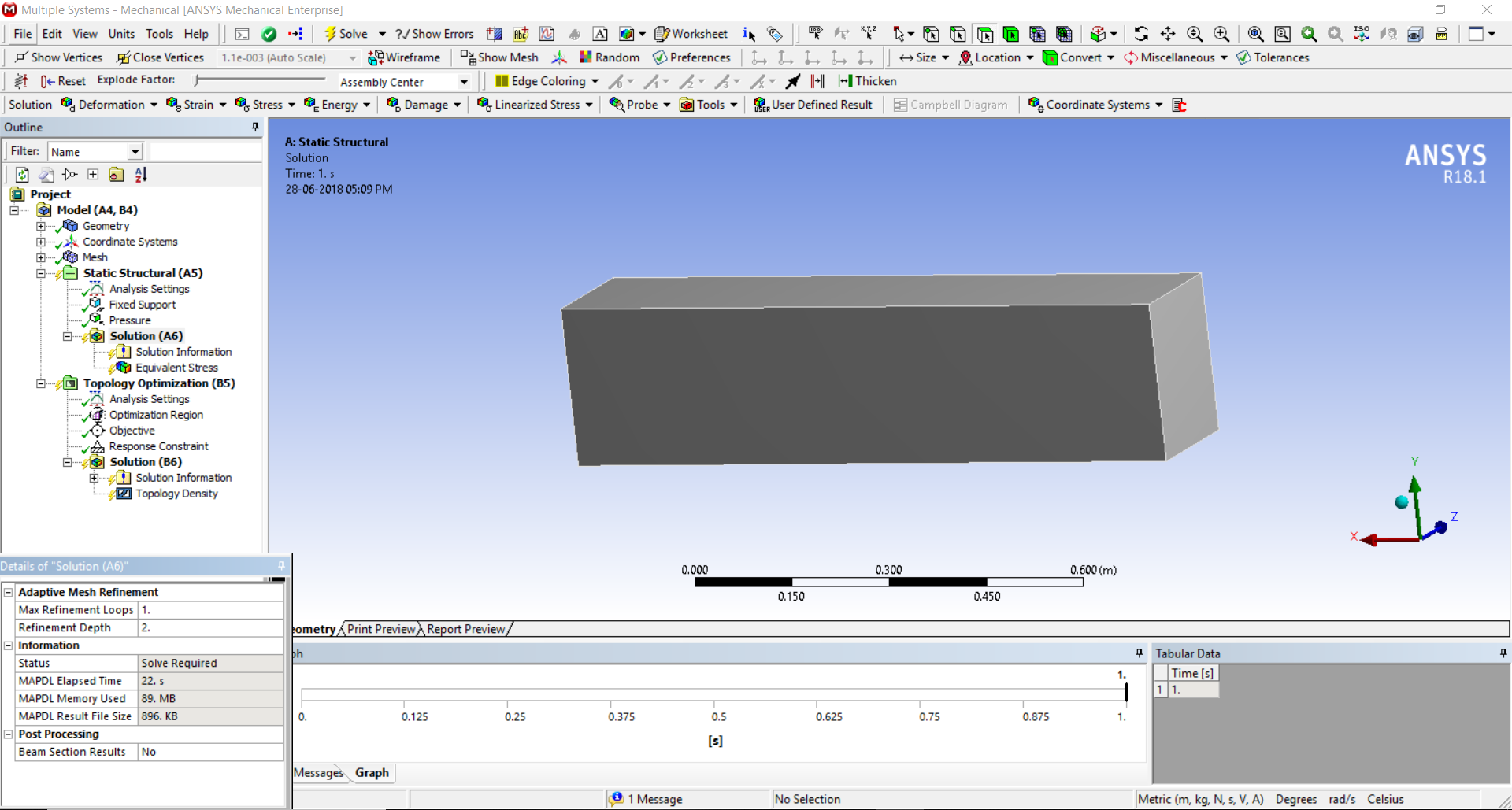


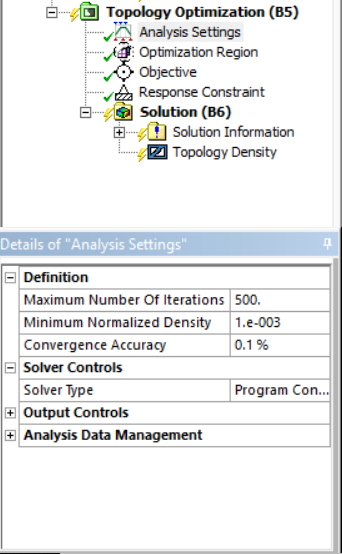
Select -> **Solve** present in the **Title bar** for the Ansys to analyse and provide results.

Go to **Ansys Workbench** Window again. Drag and drop the **Topology Optimization** in the feature tree to the **Solution** in the **Static Structural** present in the Project Schematic. Now the Project Schematic window looks like the picture give below.



Go to **Mechanical** window. You can view the **Topology Optimization tools** that have incorporated into the mechanical.

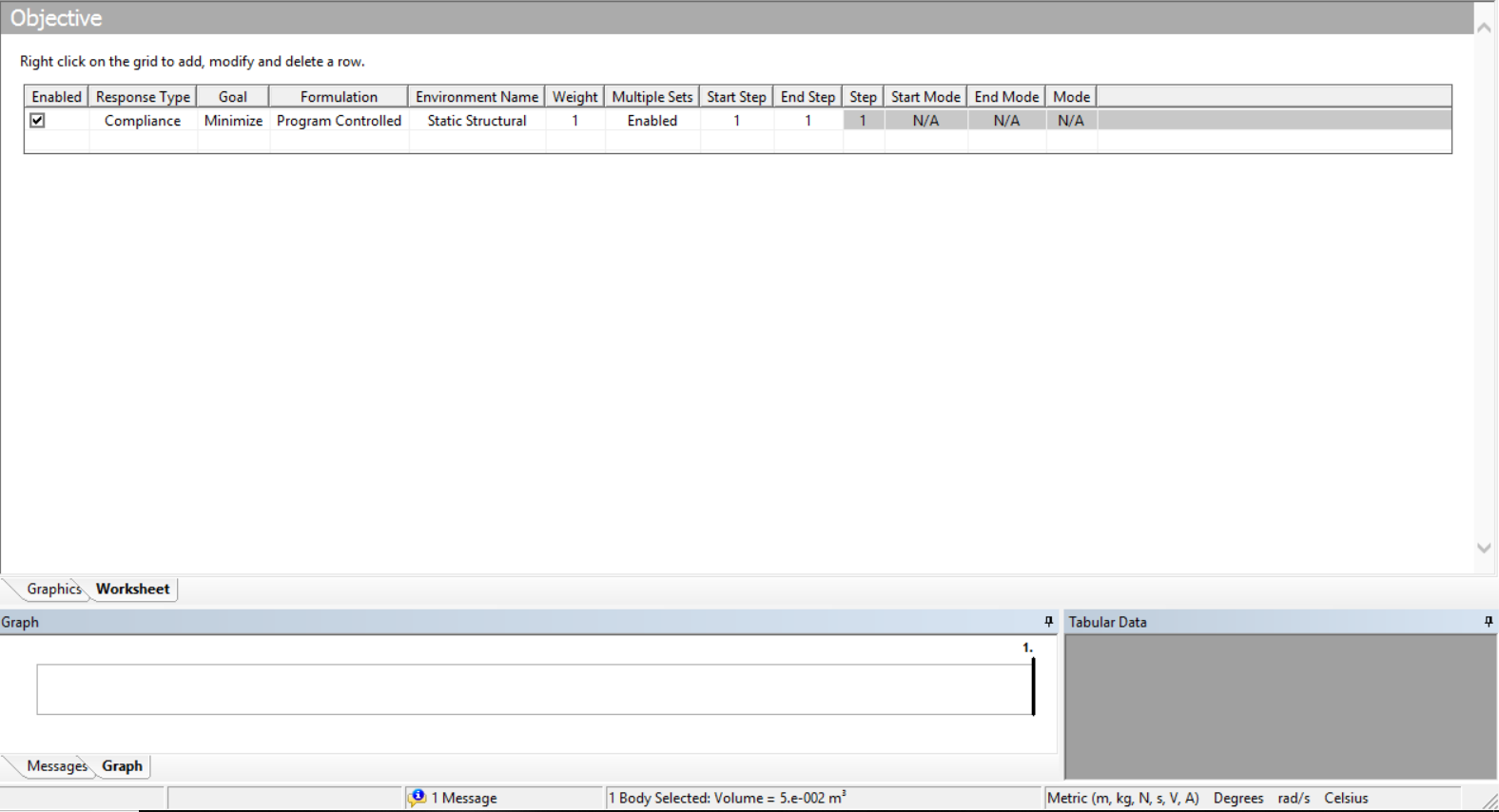




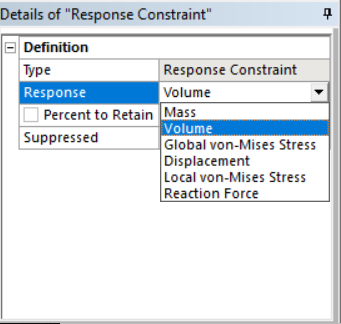
Select -> **Analysis Settings** under the Topology Optimization and give the necessary details.

Select -> **Optimization Region** and provide the regions for which you have to find the optimised results.

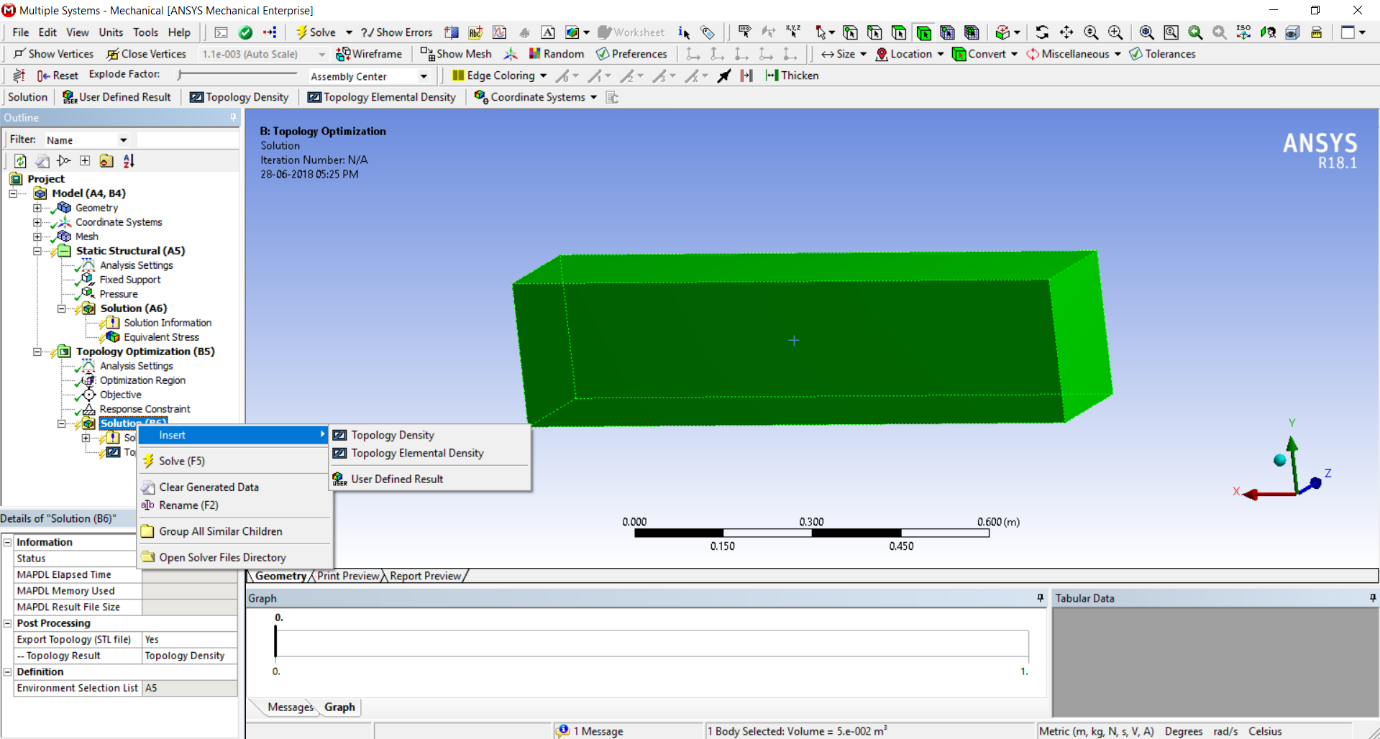
Select -> **Objective** for specifying the objective function for the optimization. Once you select it you see the workspace as given below. Right click on the box below Enabled if you want to Add/Modify/Delete the objective function present as default. Else carryon with the present one.



Select -> **Response Constraint** and determine the field of constraint from the options in the **Response**. Give the **Percent to Retain** in the box below once after optimization is completed.



Right Click on the **Solution** **(B6)** in the Topology menu. Go to **Insert** and select the form of the optimization results which you need.



Click -> **Solve** present in the Title bar of the Ansys. The process starts to optimise the structure and once it is completes you can look at the results by selecting the form which you have provided present below the Solution menu.



Final result is the topologically optimised result for the specified structure. Hence the mission has been accomplished by finding the desired outcome.

**BIBLIOGRAPHY**

1. Evolutionary Topology Optimization of Continuum Structures : Methods and Applications by X. Haung and Y. M. Xie
2. https://en.wikipedia.org/wiki/Topology\_optimization
3. https://caeai.com/blog/what-topology-optimization-and-why-use-it
4. https://www.ethz.ch/content/dam/ethz/special-interest/mavt/department-dam/news/documents/sigmund-presentation-dls-hs-15.pdf
5. http://web.mit.edu/16.810/www/16.810\_L8\_Optimization.pdf
6. https://www.youtube.com/watch?v=htHkr3l7iMQ
7. https://www.youtube.com/watch?v=PMcP7mK1Zec