ASSIGNMENT 1

OPTIMISATION (MT2528)

AKHIL MORA (950728-6954)
YASHPAL SURENDHAR GOUD PULICHERLA (950419-4995)

TASK1:

Problem Statement:

Given an I-section steel cantilever beam is shown in fig below. Weight of the beam is to be minimized while fulfilling following constraints. This is to be solved using MATLAB as server and ABAQUS as client.

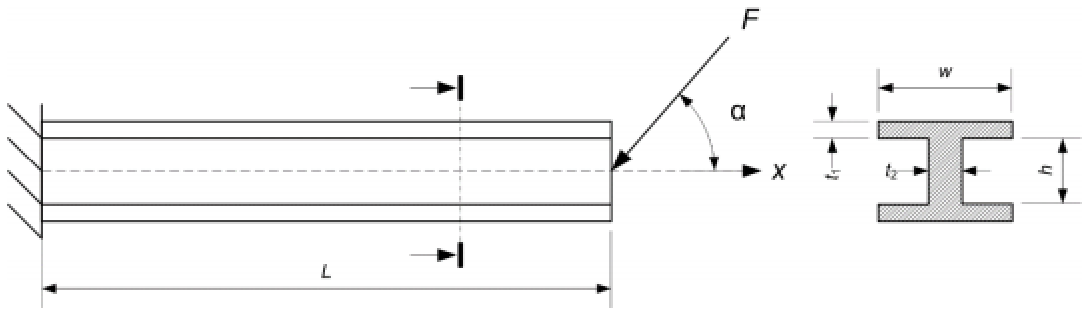


Table 1: Given Parameters

Parameter	Value
Young's Modulus	210GPa
Shear Modulus	70GPa
Yield stress	190MPa
Force	75kN
Length	1.2m
α	45^o

1. Effective stress of the beam shouldn't exceed the yield stress of the material.
2. Maximum deflection of the beam shouldn't exceed 15mm.
3. Maximum available mounting space in y-z plane is bounded with base =150mm & h=175mm.

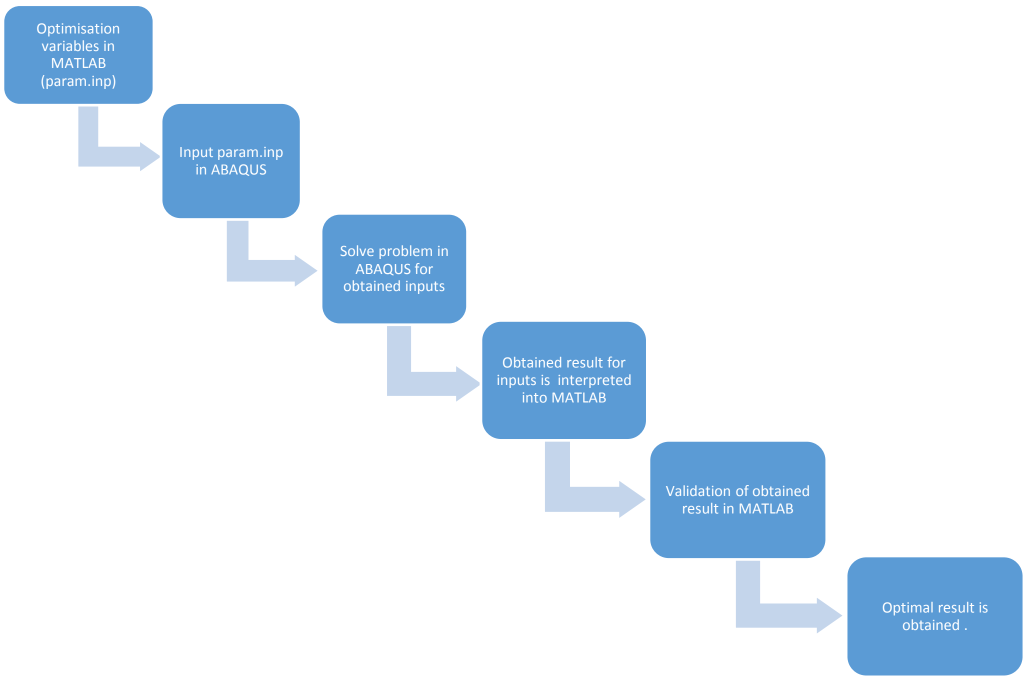
As length is given constant and density is an arbitrary constant the objective function of the beam is cross-sectional area of the beam which is given as

$$A = 2 * w * t1 + (h - 2 * t1) * t2$$

Method:

The given problem is to be solved using MATLAB as server and ABAQUS as client.

The beam model is designed in ABAQUS and solved with arbitrary constants and initial variables. Input variables are given in MATLAB and given input to ABAQUS. The beam is solved for the obtained inputs and resulted output comes in the form of a text file. That text file is interpreted in and read in MATLAB for optimized value.


Optimisation variables in MATLAB (param.inp)
Input param.inp in ABAQUS
Solve problem in ABAQUS for obtained inputs
Obtained result for inputs is interpreted into MATLAB
Validation of obtained result in MATLAB
Optimal result is obtained .

Solution:

The objective function of the problem is to minimize the mass of the system. Calculation of mass is given as a function in MATLAB.

In MATLAB input parameters are generated by *param.inp*. The beam model is designed in ABAQUS and solved with arbitrary constants and initial variables. A python file (*.py*) is generated from the obtained *jnl* file of solved ABAQUS model. The generated file *param.imp* is included in python script and parameters are defined in python script. The python script is studied by ABAQUS and solved for every inputs given from MATLAB. By running MATLAB a new set of input variables is generated and sent to ABQUS and solved. The results obtained (*odb file*) are interpreted again to MATLAB. MATLAB checks the result for constraints and objective functions and gives another set of input variables to ABAQUS for next iteration. The number of iterations are run until an optimal solution is obtained in MATLAB.

Result:

The optimal solution for minimizing the mass of the system with applied constraints are as follows:

w = 75 mm
h = 89 mm
t1 = 10 mm
t2 = 10 mm

TASK2:

Problem Statement:

Given an I-section steel cantilever beam is shown in fig below. Weight of the beam is to be minimized while fulfilling following constraints. This is to be solved using ISIGHT as server and ABAQUS as client.

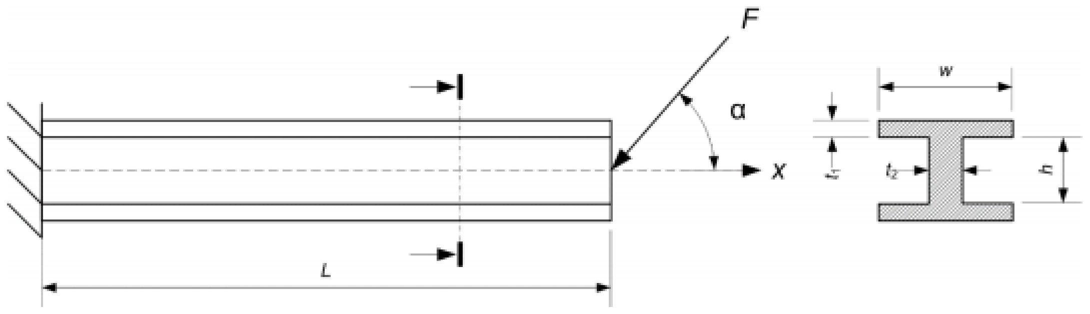


Table 2: Given Parameters

Parameter	Value
Young’s Modulus	210GPa
Shear Modulus	70GPa
Yield stress	190MPa
Force	75kN
Length	1.2m
α	45^{o}

4. Effective stress of the beam shouldn’t exceed the yield stress of the material.
5. Maximum deflection of the beam shouldn’t exceed 15mm.
6. Maximum available mounting space in y-z plane is bounded with base =150mm & h=175mm.

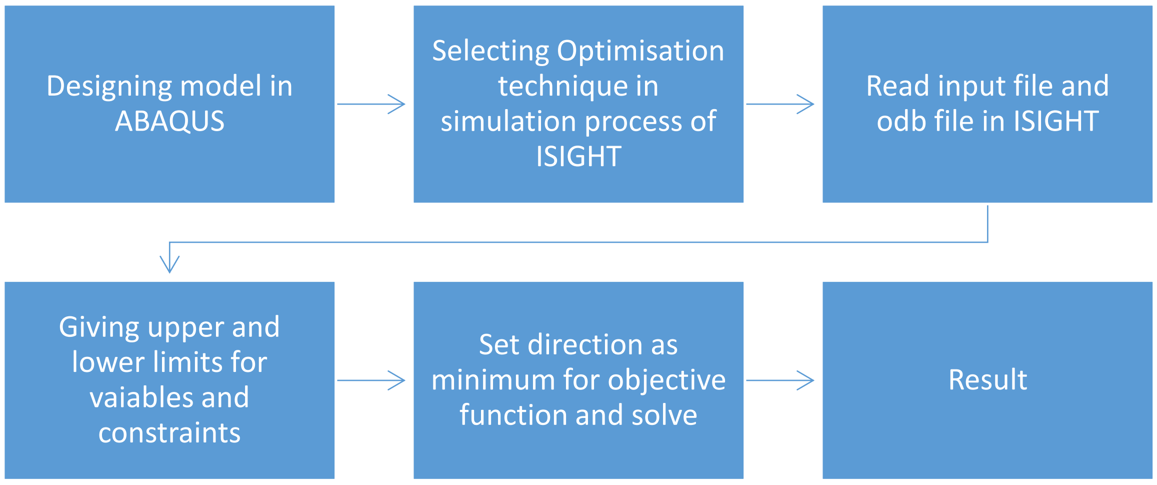
As length is given constant and density is an arbitrary constant the objective function of the beam is cross-sectional area of the beam which is given as

$$A = 2 * w * t1 + (h - 2 * t1) * t2$$

Method:

The given problem is to be solved using ISIGHT as server and ABAQUS as client.

A model of the beam is designed in ABAQUS by giving arbitrary constants and initial guess values for design variables and then solved. The designed model i.e. input file and output database file are then read in ISIGHT by interpreting ABAQUS module in simulation process. Optimisation technique is used in simulation process to solve the problem for obtaining optimal solution. Lower and upper boundary values are assigned for variables and constraints and set the direction as minimum for objective function and then solved.


Designing model in ABAQUS
Selecting Optimisation technique in simulation process of ISIGHT
Read input file and odb file in ISIGHT
Giving upper and lower limits for vaiables and constraints
Set direction as minimum for objective function and solve
Result

Solution:

The objective function of the problem is to minimize the mass of the system.
width w,
height h,
thickness of flange t1 and
thickness of the web t2.

The problem is designed in ABAQUS with initial guess values for width, height, t1 & t2. The designed model is meshed. Material properties, load and boundary conditions are given and solved.

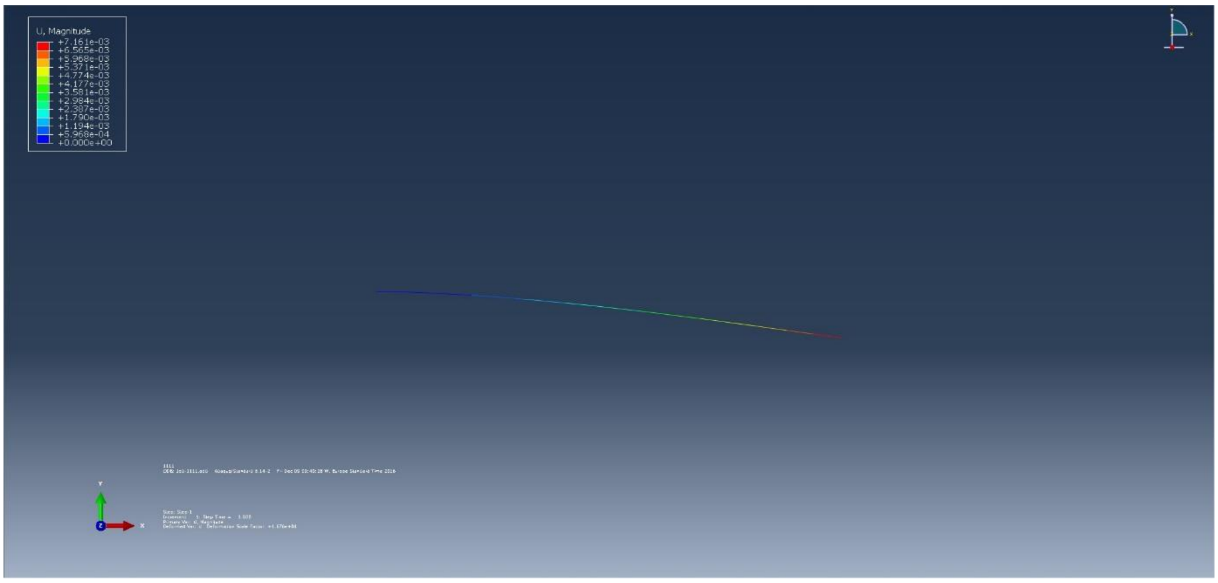


Figure 1. ABAQUS result of displacement of beam with initial values

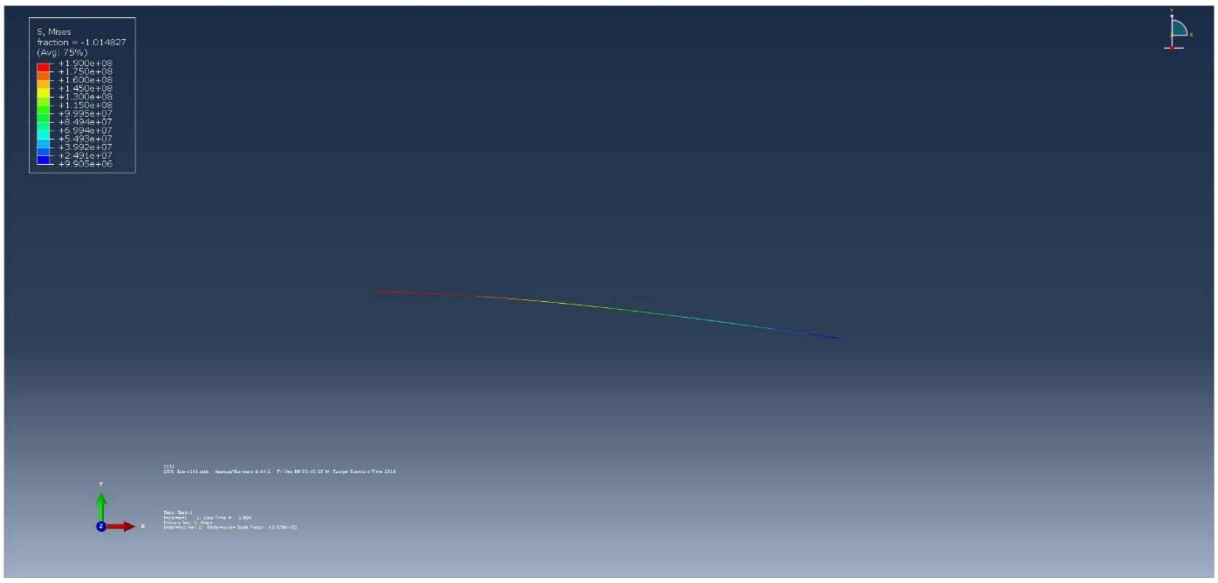


Figure 2. ABAQUS result of Von-Mises stresses with initial values

w = 150 mm
h = 175 mm
t1 = 10 mm
t2 = 10 mm
density ρ = 7800 kg/m^3

Optimisation technique is selected in ISIGHT for simulation process. ABAQUS module is given.

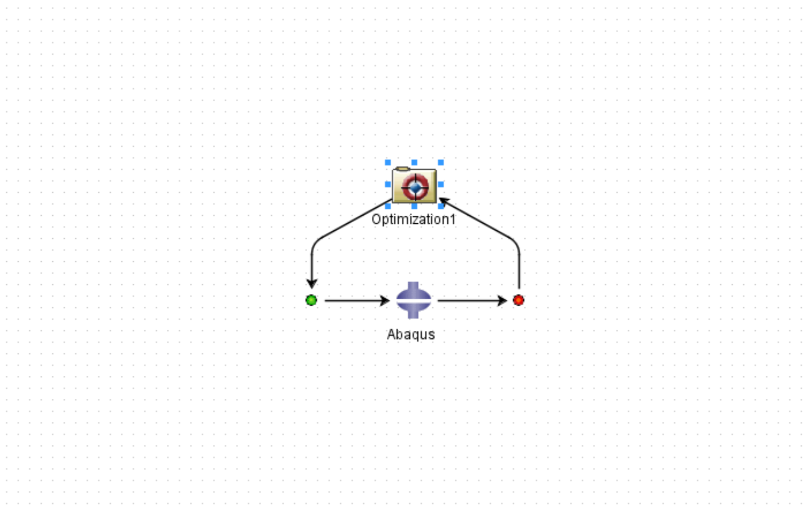
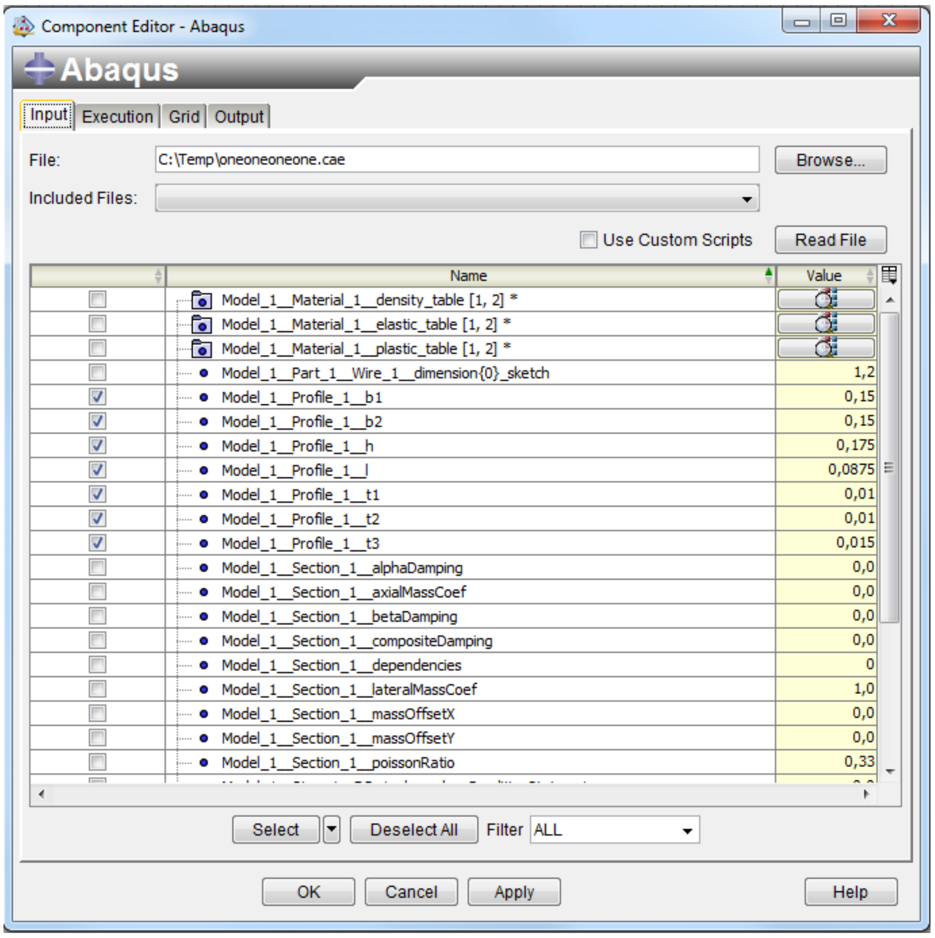


Figure 3. Optimization technique with ABAQUS module

Input and output database files are imported and read. w, h, t1, t2 are assigned as input parameters while von-mises stress and deflection are assigned as output data.



Component Editor - Abaqus

Abaqus

Input | Execution | Grid | Output

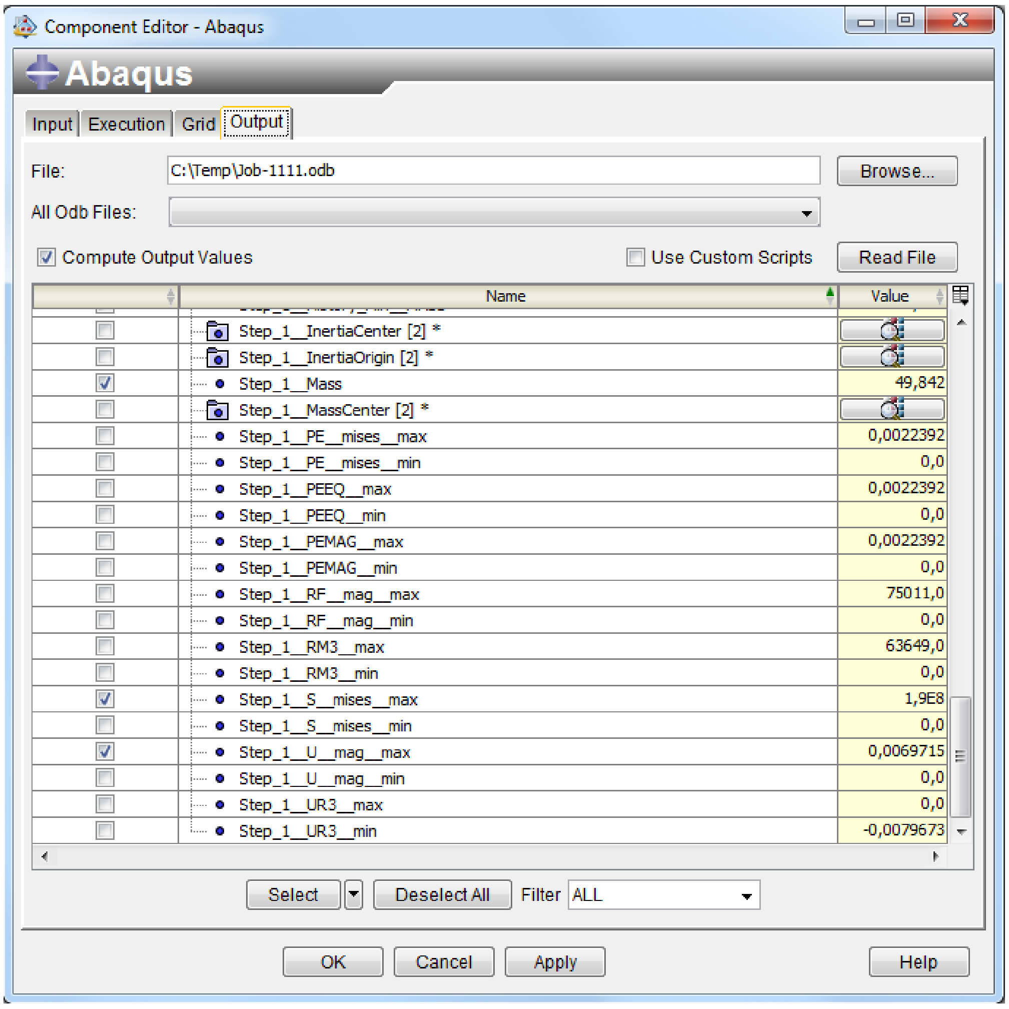
File: C:\Temp\oneoneoneone.cae

Included Files:

☐ Use Custom Scripts

	Name	Value
☐	Model_1__Material_1__density_table [1, 2] *	
☐	Model_1__Material_1__elastic_table [1, 2] *	
☐	Model_1__Material_1__plastic_table [1, 2] *	
☐	Model_1__Part_1__Wire_1__dimension{0}_sketch	1,2
✓	Model_1__Profile_1__b1	0,15
✓	Model_1__Profile_1__b2	0,15
✓	Model_1__Profile_1__h	0,175
✓	Model_1__Profile_1__l	0,0875
✓	Model_1__Profile_1__t1	0,01
✓	Model_1__Profile_1__t2	0,01
✓	Model_1__Profile_1__t3	0,015
☐	Model_1__Section_1__alphaDamping	0,0
☐	Model_1__Section_1__axialMassCoef	0,0
☐	Model_1__Section_1__betaDamping	0,0
☐	Model_1__Section_1__compositeDamping	0,0
☐	Model_1__Section_1__dependencies	0
☐	Model_1__Section_1__lateralMassCoef	1,0
☐	Model_1__Section_1__massOffsetX	0,0
☐	Model_1__Section_1__massOffsetY	0,0
☐	Model_1__Section_1__poissonRatio	0,33

Figure 4. Input parameters read from Input file


Component Editor - Abaqus

Abaqus

Input | Execution | Grid | Output

File: C:\Temp\Job-1111.odb

All Odb Files:

[✓] Compute Output Values

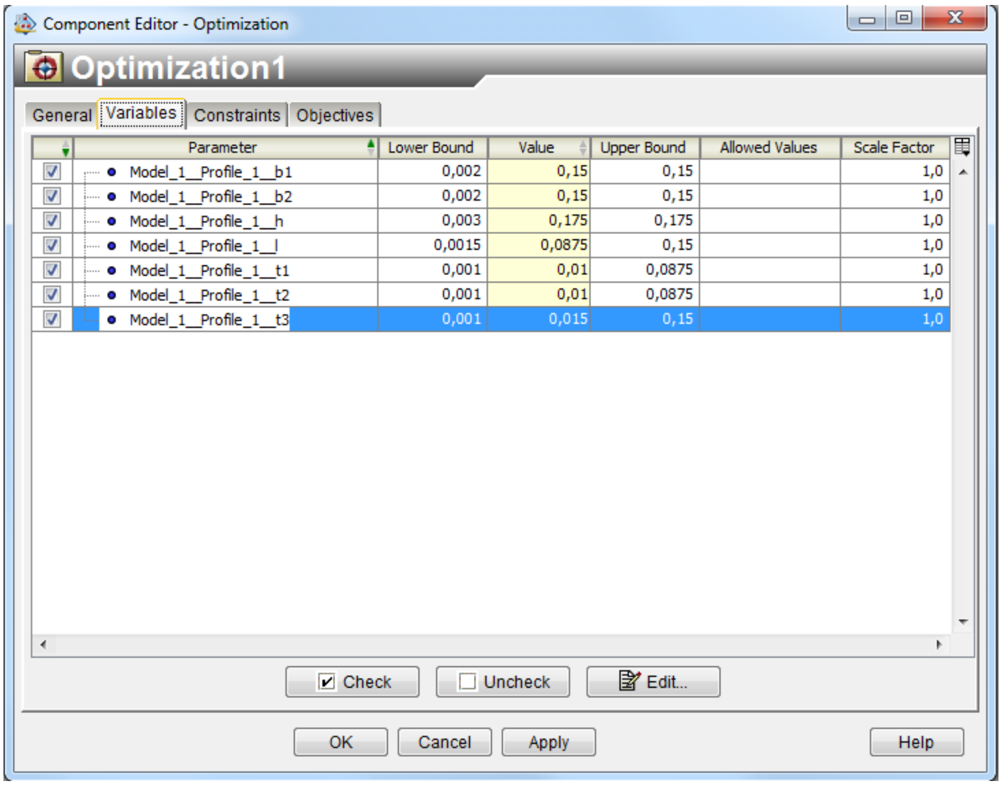
[ ] Use Custom Scripts

	Name	Value
	[illegible]	
	Step_1__InertiaCenter [2] *	
	Step_1__InertiaOrigin [2] *	
✓	Step_1__Mass	49,842
	Step_1__MassCenter [2] *	
	Step_1__PE__mises__max	0,0022392
	Step_1__PE__mises__min	0,0
	Step_1__PEEQ__max	0,0022392
	Step_1__PEEQ__min	0,0
	Step_1__PEMAG__max	0,0022392
	Step_1__PEMAG__min	0,0
	Step_1__RF__mag__max	75011,0
	Step_1__RF__mag__min	0,0
	Step_1__RM3__max	63649,0
	Step_1__RM3__min	0,0
✓	Step_1__S__mises__max	1,9E8
	Step_1__S__mises__min	0,0
✓	Step_1__U__mag__max	0,0069715
	Step_1__U__mag__min	0,0
	Step_1__UR3__max	0,0
	Step_1__UR3__min	-0,0079673

Filter: ALL

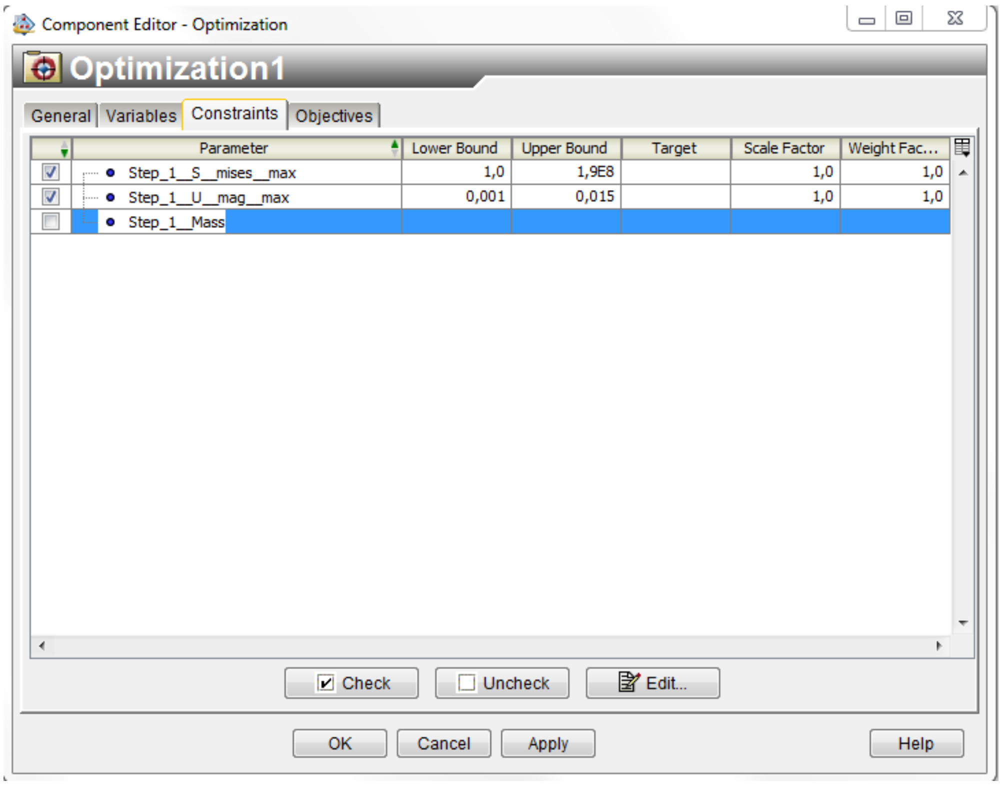
Figure 5. Output functions read from output database file

Setting up lower and upper bounds for variables and constraints and then solved to obtain results for number of iterations and finally giving an optimal solution.



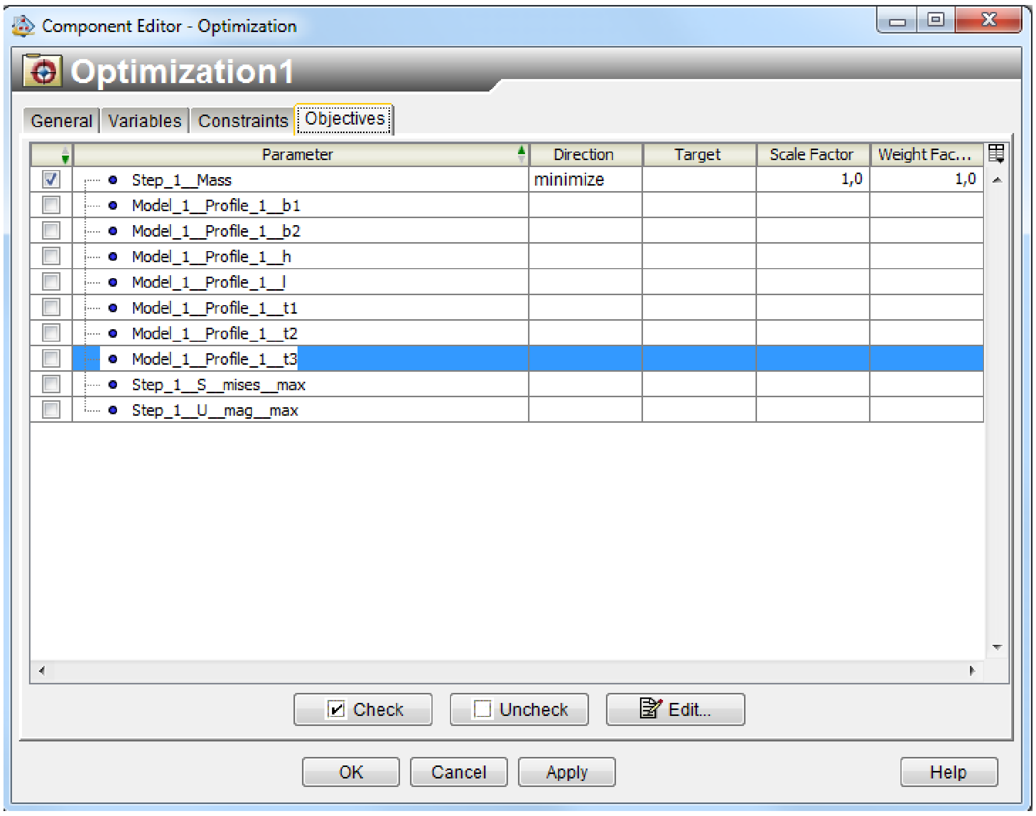
	Parameter	Lower Bound	Value	Upper Bound	Allowed Values	Scale Factor
✓	Model_1__Profile_1__b1	0,002	0,15	0,15		1,0
✓	Model_1__Profile_1__b2	0,002	0,15	0,15		1,0
✓	Model_1__Profile_1__h	0,003	0,175	0,175		1,0
✓	Model_1__Profile_1__l	0,0015	0,0875	0,15		1,0
✓	Model_1__Profile_1__t1	0,001	0,01	0,0875		1,0
✓	Model_1__Profile_1__t2	0,001	0,01	0,0875		1,0
✓	Model_1__Profile_1__t3	0,001	0,015	0,15		1,0

Figure 6. Lower and Upper boundaries for input variables



	Parameter	Lower Bound	Upper Bound	Target	Scale Factor	Weight Fac...
✓	Step_1__S__mises__max	1,0	1,9E8		1,0	1,0
✓	Step_1__U__mag__max	0,001	0,015		1,0	1,0
	Step_1__Mass					

Figure 7. Lower and boundaries for constraints



	Parameter	Direction	Target	Scale Factor	Weight Fac...
✓	Step_1__Mass	minimize		1,0	1,0
	Model_1__Profile_1__b1				
	Model_1__Profile_1__b2				
	Model_1__Profile_1__h				
	Model_1__Profile_1__l				
	Model_1__Profile_1__t1				
	Model_1__Profile_1__t2				
	Model_1__Profile_1__t3				
	Step_1__S__mises__max				
	Step_1__U__mag__max				

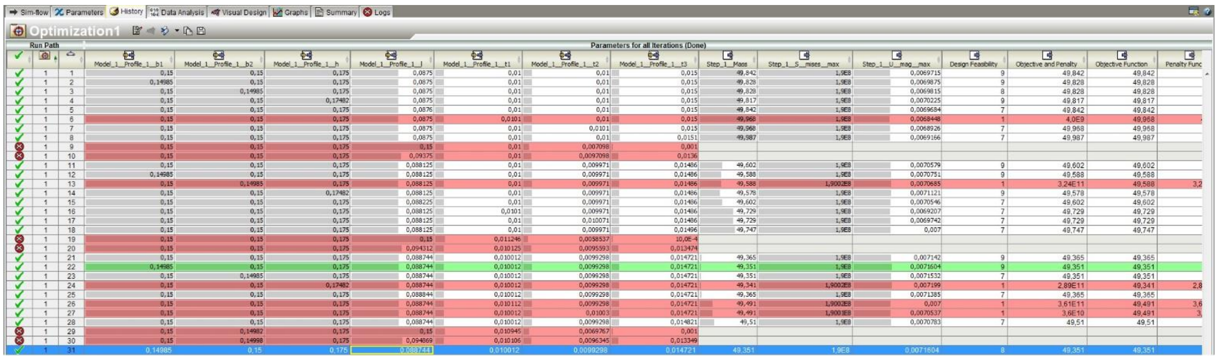
Figure 8. Setting direction for objective function

$0.002 \leq w \leq 0.15$
$0.003 \leq h \leq 0.175$
$0.001 \leq t1 \leq 0.0875$
$0.001 \leq t2 \leq 0.15$
$1 \leq R_{el} \leq 1.9e8$
$0.001 \leq \delta \leq 0.015$

Results:

The optimal solution for minimizing the mass of the system with applied constraints are as follows:

w = 149.85 mm
h = 175 mm
t1 = 9.9298 mm
t2 = 14.721 mm



Parameters for all Iterations (Done)

✓			Model_1__Profile_1__b1	Model_1__Profile_1__b2	Model_1__Profile_1__h	Model_1__Profile_1__l	Model_1__Profile_1__t1	Model_1__Profile_1__t2	Model_1__Profile_1__t3	Step_1__Mass	Step_1__S__mises__max	Step_1__U__mag__max	Design Feasibility	Objective and Penalty	Objective Function	Penalty Func
✓	1	1	0,15	0,15	0,175	0,0875	0,01	0,01	0,015	49,842	1,9E8	0,0069715	9	49,842	49,842	
✓	1	2	0,14985	0,15	0,175	0,0875	0,01	0,01	0,015	49,828	1,9E8	0,0069875	9	49,828	49,828	
✓	1	3	0,15	0,14985	0,175	0,0875	0,01	0,01	0,015	49,828	1,9E8	0,0069815	8	49,828	49,828	
✓	1	4	0,15	0,15	0,17482	0,0875	0,01	0,01	0,015	49,817	1,9E8	0,0070225	9	49,817	49,817	
✓	1	5	0,15	0,15	0,175	0,0876	0,01	0,01	0,015	49,842	1,9E8	0,0069684	7	49,842	49,842	
✓	1	6	0,15	0,15	0,175	0,0875	0,0101	0,01	0,015	49,968	1,9E8	0,0068448	1	4,0E9	49,968	[illegible]
✓	1	7	0,15	0,15	0,175	0,0875	0,01	0,0101	0,015	49,968	1,9E8	0,0068926	7	49,968	49,968	
✓	1	8	0,15	0,15	0,175	0,0875	0,01	0,01	0,0151	49,987	1,9E8	0,0069166	7	49,987	49,987	
✗	1	9	0,15	0,15	0,175	0,15	0,01	0,007098	0,001							
✗	1	10	0,15	0,15	0,175	0,09375	0,01	0,0097098	0,0136							
✓	1	11	0,15	0,15	0,175	0,088125	0,01	0,009971	0,01486	49,602	1,9E8	0,0070579	9	49,602	49,602	
✓	1	12	0,14985	0,15	0,175	0,088125	0,01	0,009971	0,01486	49,588	1,9E8	0,0070751	9	49,588	49,588	
✓	1	13	0,15	0,14985	0,175	0,088125	0,01	0,009971	0,01486	49,588	1,9002E8	0,0070685	1	3,24E11	49,588	3,2
✓	1	14	0,15	0,15	0,17482	0,088125	0,01	0,009971	0,01486	49,578	1,9E8	0,0071121	9	49,578	49,578	
✓	1	15	0,15	0,15	0,175	0,088225	0,01	0,009971	0,01486	49,602	1,9E8	0,0070546	7	49,602	49,602	
✓	1	16	0,15	0,15	0,175	0,088125	0,0101	0,009971	0,01486	49,729	1,9E8	0,0069207	7	49,729	49,729	
✓	1	17	0,15	0,15	0,175	0,088125	0,01	0,010071	0,01486	49,729	1,9E8	0,0069742	7	49,729	49,729	
✓	1	18	0,15	0,15	0,175	0,088125	0,01	0,009971	0,01496	49,747	1,9E8	0,007	7	49,747	49,747	
✗	1	19	0,15	0,15	0,175	0,15	0,011246	0,0058537	10,0E-4							
✗	1	20	0,15	0,15	0,175	0,094312	0,010125	0,0095593	0,013474							
✓	1	21	0,15	0,15	0,175	0,088744	0,010012	0,0099298	0,014721	49,365	1,9E8	0,007142	9	49,365	49,365	
✓	1	22	0,14985	0,15	0,175	0,088744	0,010012	0,0099298	0,014721	49,351	1,9E8	0,0071604	9	49,351	49,351	
✓	1	23	0,15	0,14985	0,175	0,088744	0,010012	0,0099298	0,014721	49,351	1,9E8	0,0071532	7	49,351	49,351	
✓	1	24	0,15	0,15	0,17482	0,088744	0,010012	0,0099298	0,014721	49,341	1,9002E8	0,007199	1	2,89E11	49,341	2,8
✓	1	25	0,15	0,15	0,175	0,088844	0,010012	0,0099298	0,014721	49,365	1,9E8	0,0071385	7	49,365	49,365	
✓	1	26	0,15	0,15	0,175	0,088744	0,010112	0,0099298	0,014721	49,491	1,9002E8	0,007	1	3,61E11	49,491	3,6
✓	1	27	0,15	0,15	0,175	0,088744	0,010012	0,01003	0,014721	49,491	1,9001E8	0,0070537	1	3,6E10	49,491	3,
✓	1	28	0,15	0,15	0,175	0,088744	0,010012	0,0099298	0,014821	49,51	1,9E8	0,0070783	7	49,51	49,51	
✗	1	29	0,15	0,14982	0,175	0,15	0,010945	0,0069767	0,001							
✗	1	30	0,15	0,14998	0,175	0,094869	0,010106	0,0096345	0,013349							
✓	1	31	0,14985	0,15	0,175	0,088744	0,010012	0,0099298	0,014721	49,351	1,9E8	0,0071604	8	49,351	49,351	

Figure 10. Results obtained for various iterations

- The minimized mass of the system obtained is 49.351 kg for density 7800 kg/ m^3.
- As the length of the beam and density are arbitrary constants the minimum value for objective function is obtained as 0.0046 m^2.

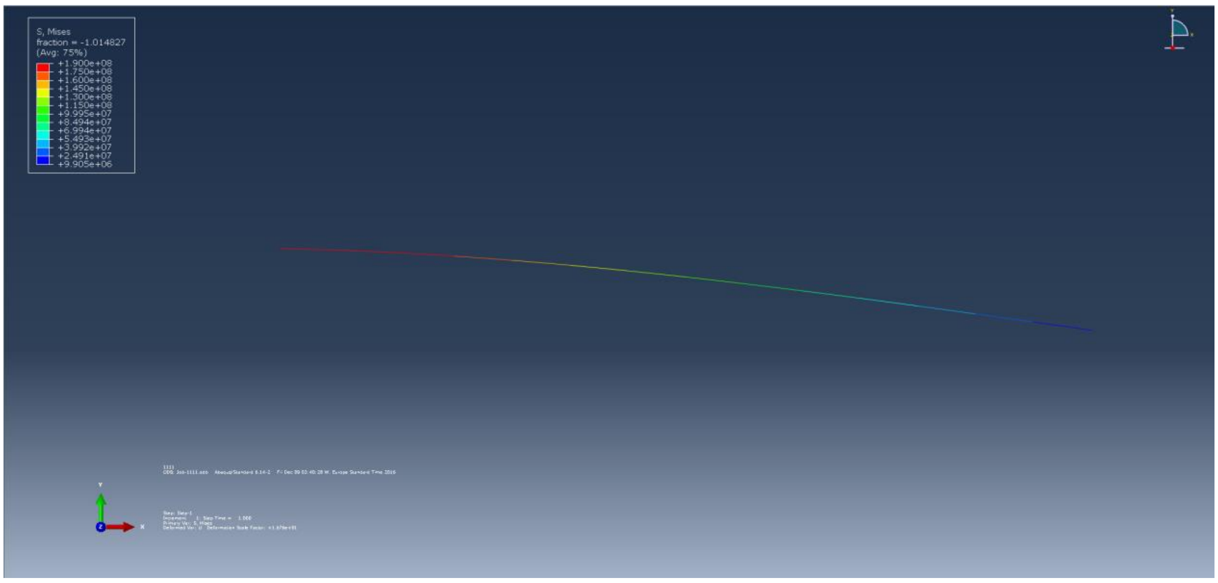
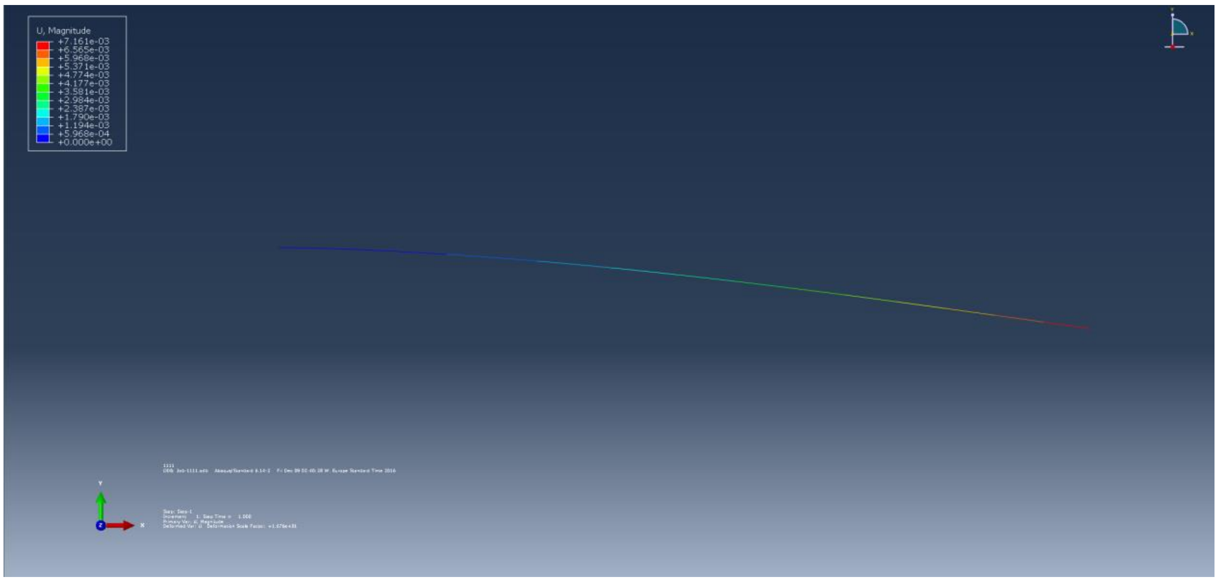


Figure 11.Von-Mises stresses for optimal solution



1111
ODB: Job-1111.odb Abaqus/Standard 6.14-2 Fri Dec 09 01:45:28 W. Europe Standard Time 2016

Step: Step-1
Increment 1: Step Time = 1.000
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +1.076e+01

Figure 12. Displacement of the beam for optimal solution

TASK 3:

Problem Statement:

An anchor system is with three beams and two connecting rods is given as shown in the figure below.

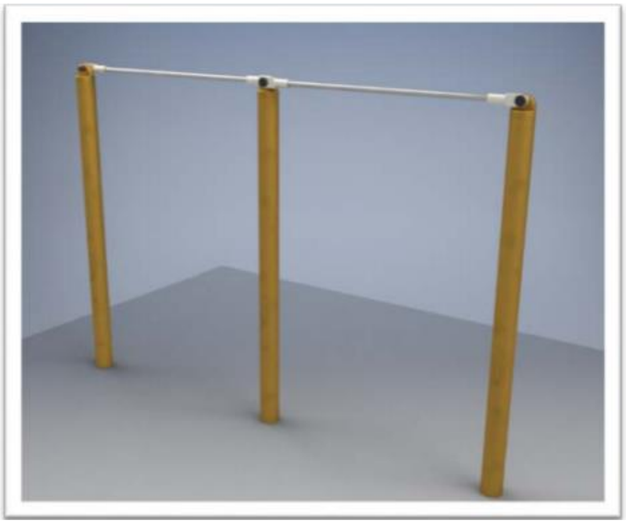


Figure 1: Anchor Design

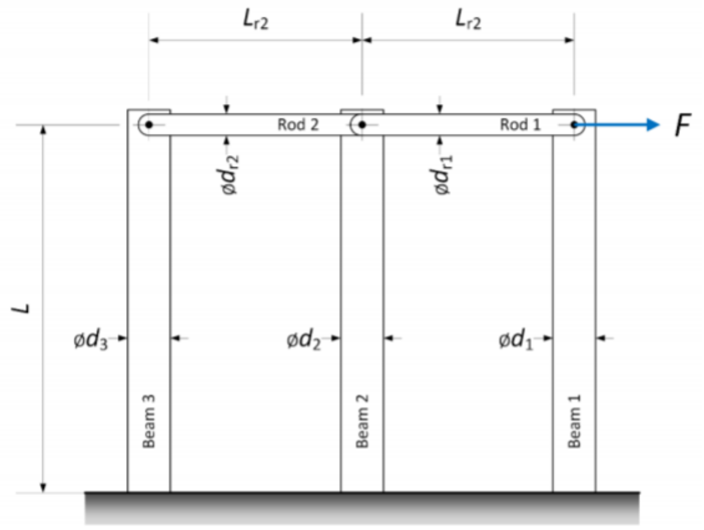


Figure 2: Schematic description

The problem is to be solved using Metamodelling technique.

Table 3: Given parameters

Parameter	Value	Unit
L	1	m
L_{r1}	1	m
L_{r2}	1	m
E	70	GPa
ρ	2700	Kg/m^3

Objective function: (i) Minimization of mass of the system

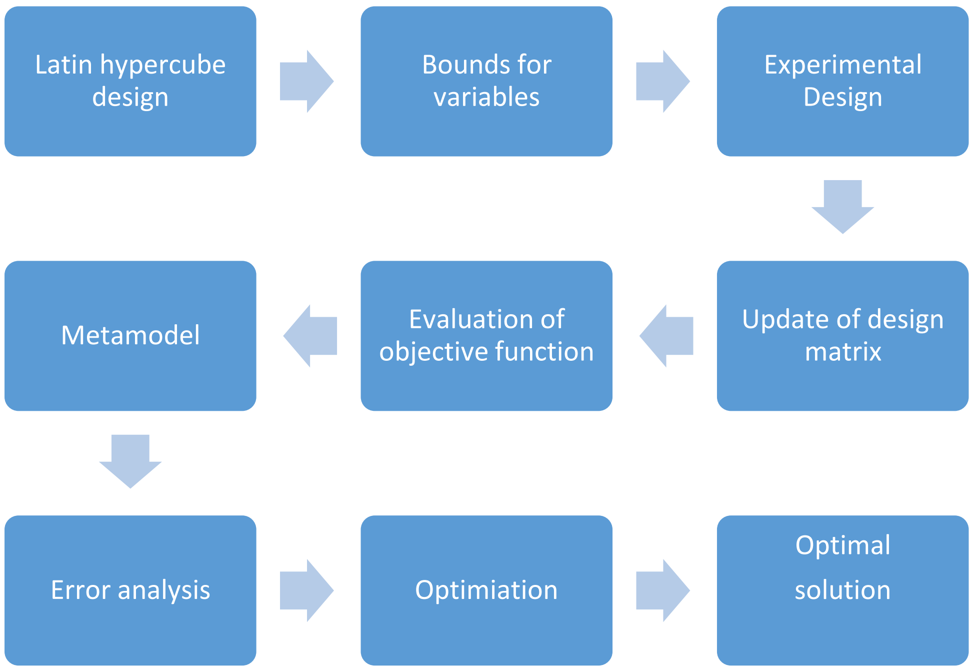
(ii) Minimization of deflection of beam 1

Constraints: (i) Induced stress < 127 MPa

(ii) Transfer of force to the ground < 400 N

Method:

The problem is solved using Metamodeling technique 'Latin hypercube design' with the following steps.



(i) Initially lower and upper bounds are set for diameters of beams 1, 2, 3 and rods 1 and 2.
(ii) An experimental design is generated with the help of MATLAB function *'lhsdesign'* for 10 points.
(iii) Now the matrix of experimental design is replaced with the different values of diameters in the range of bounds.
(iv) Evaluation of objective function and constraints for all the values.\
(v) Then a metamodel is built.
(vi) An error analysis is carried out based on root mean square values.
(vii) Then, the design is optimized.
(viii) Finally, a decision for optimal solution.

Solution:

The following figures represents the variation of stresses and ground forces with the diameters. With various diameters of beams and rods, it was found that the induced stresses are varying.

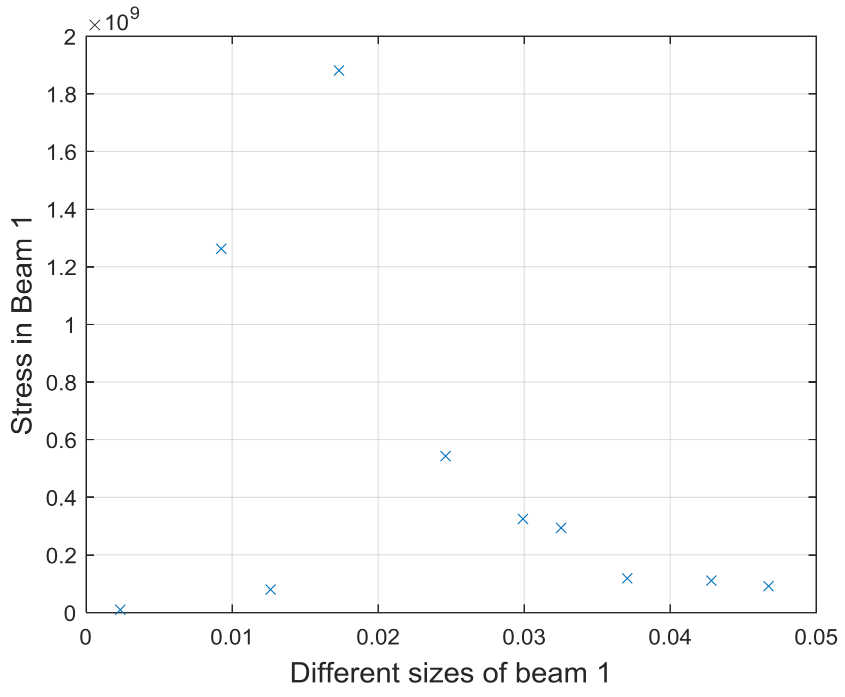


Figure 3: Stress in beam 1

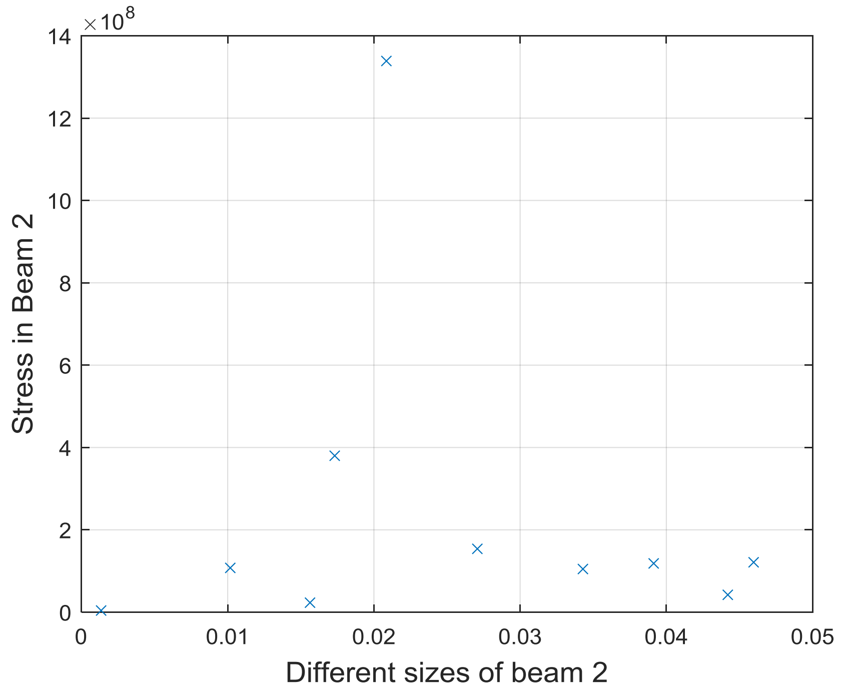


Figure 4: Stress in beam 2

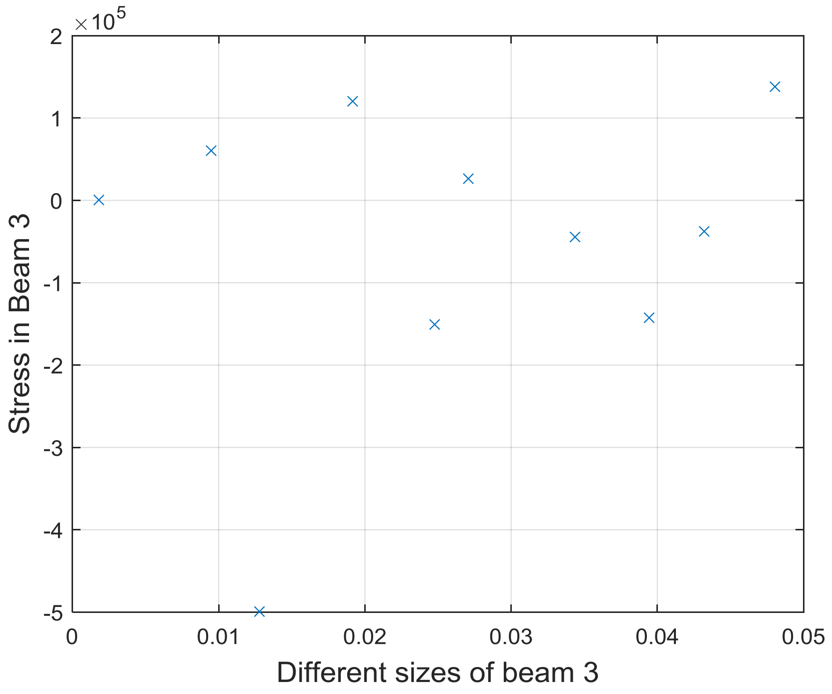


Figure 5: Stress in beam 3

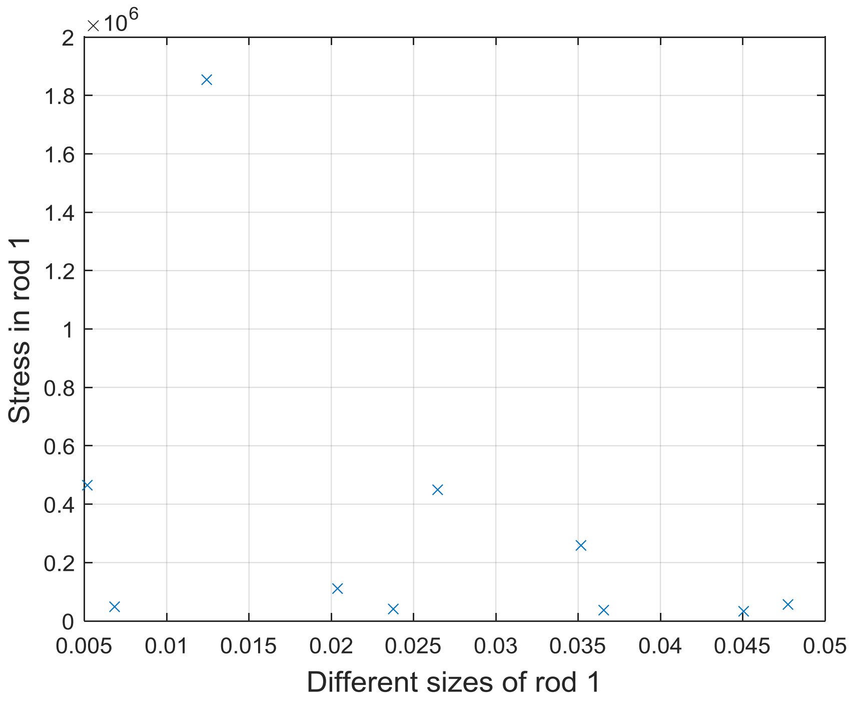


Figure 6: Stress in rod 1

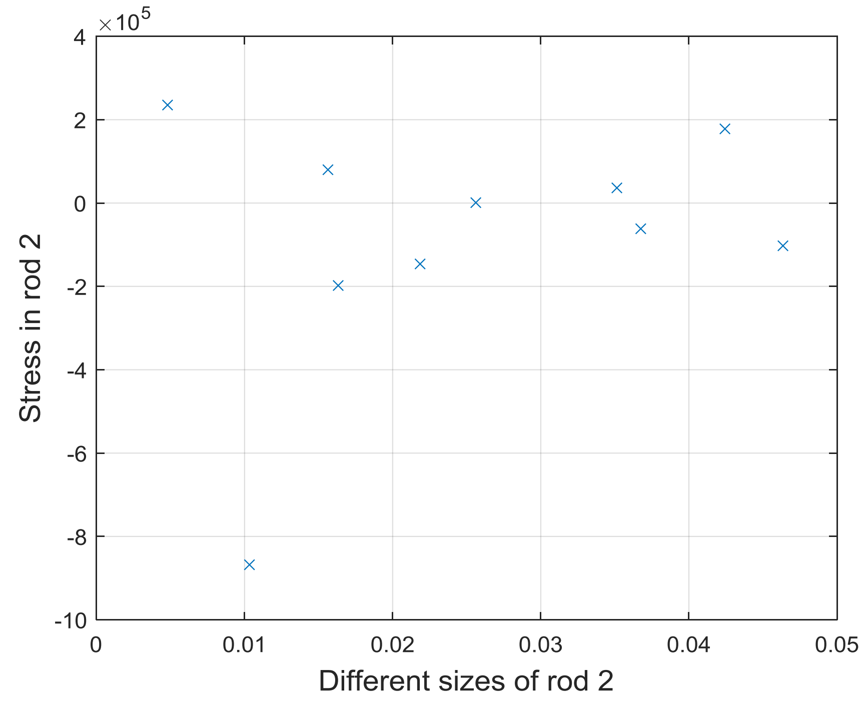


Figure 7: Stress in rod 2

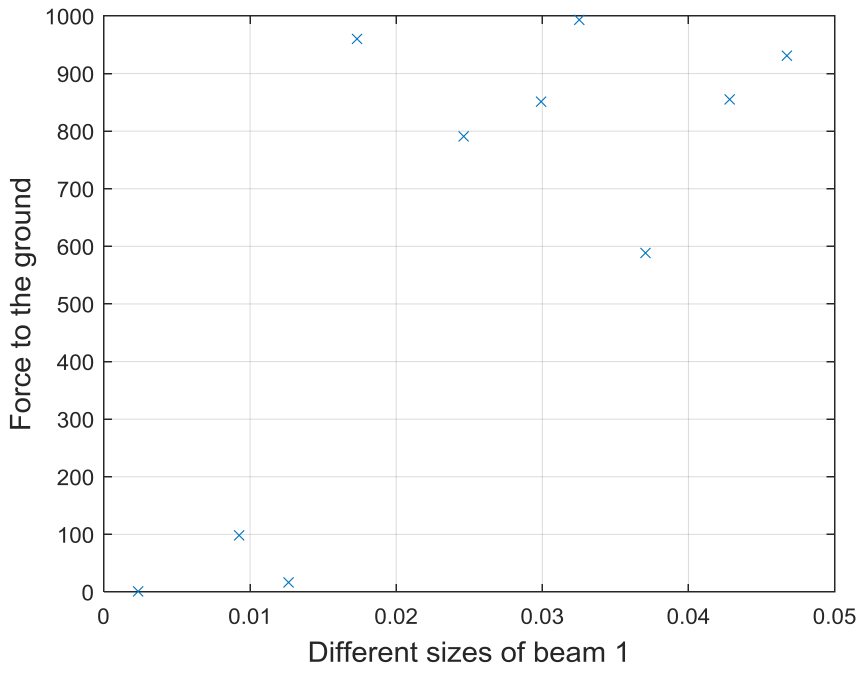


Figure 8: Ground force from beam 1

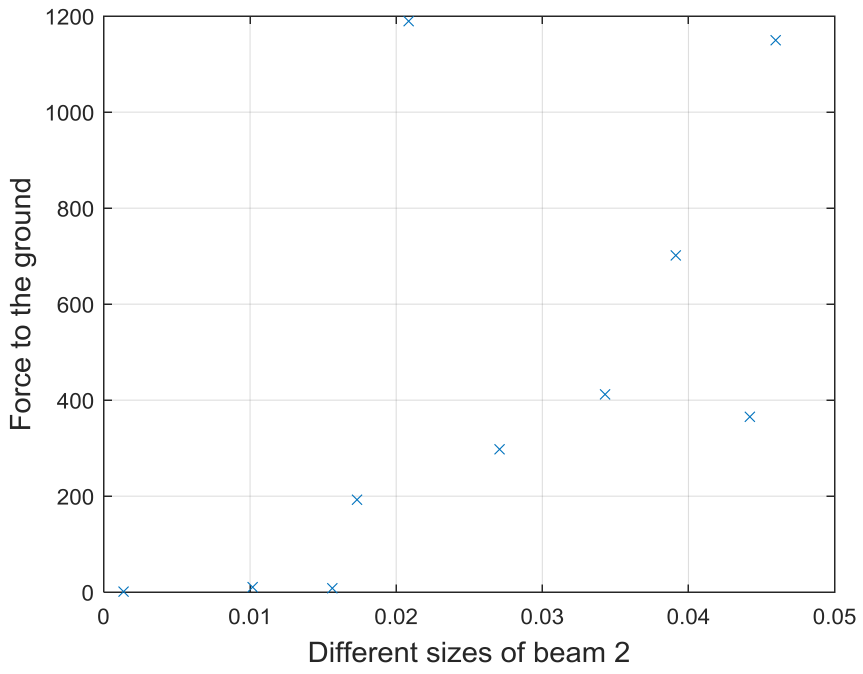


Figure 9: Ground force from beam 2

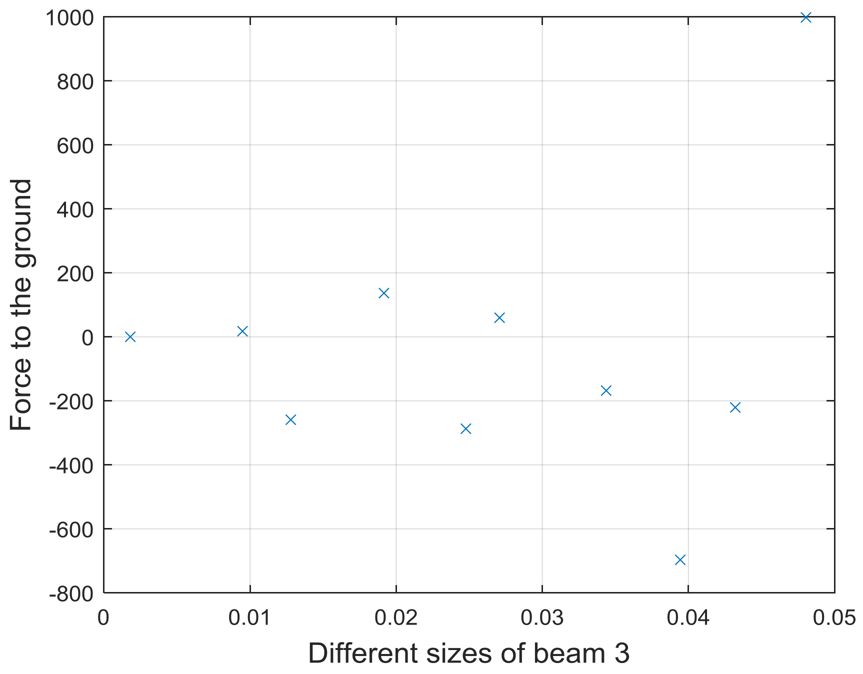


Figure 10: Ground force from beam 3

Result:

From the feasible set of all the variables, that satisfying the constraints, the values for diameters of all the beams and rods are selected in such a way that minimizes the objective functions.

Optimal solution:

Diameter of beam 1 = 5 mm

Diameter of beam 2 = 4 mm

Diameter of beam 3 = 6 mm

Diameter of rod 1 = 8 mm

Diameter of rod 2 = 7 mm

References:

1. Simulation process automation, Wall.J
2. Learn Abaqus script in one hour, J.T.B. Overvelde
3. Metamodelling and experimental design, Wall.J
4. Mathworks
5. Course material ItsLearning

APPENDIX:

TASK1:

MATLAB:

```
clc;
close all;
clear all;
E=210e9;
YStr=190e6;
F=75e3;
l=1.2;
var=[E YStr F l];
wlowbou=0.002; wuppbou=0.15;
hlowbou=0.003; huppbou=0.175;
t1lowbou=0.001; t1uppbou=0.02;
t2lowbou=0.001; t2uppbou=0.02;
x0=[(wlowbou+wuppbou)/2 (hlowbou+huppbou)/2 (t1lowbou+t1uppbou)/2 (t2lowbou+t2uppbou)/2];
lowbou=[wlowbou hlowbou t1lowbou t2lowbou];
uppbou=[wuppbou huppbou t1uppbou t2uppbou];
options=optimoptions(@fmincon,'Algorithm','interior-point','display','iter','MaxFunEvals',3000,'TolX',1E-
9,'Tolfun',1E-9,'DiffMinChange',1E-6);
[x, fval, exitflag, output]=fmincon('area',x0,[],[],[],[],lowbou,uppbou,'myconst',options,var)
```

Functions:

```
function ar=area(x,par)
l=par(4);
ar=l*((2*x(1)*x(3)+((x(2)-(2*x(3)))*x(4))));
```

```
function [a,b]=myconst(k,var)
E=var(1);
Ystr=var(2);
F=var(3);
l=var(4);
b2=(k(2))/2;
fid=fopen('parameters.inp','w');
fprintf(fid,'*PARAMETER\n');
fprintf(fid, '%s=%5.10f\n', 'w',k(1));
fprintf(fid, '%s=%5.10f\n', 'h',k(2));
fprintf(fid, '%s=%5.10f\n', 't1',k(3));
fprintf(fid, '%s=%5.10f\n', 't2',k(4));
fprintf(fid,'%s=%5.10f\n', 'centroid',b2);
fclose(fid);
!C:\SIMULIA\Abaqus\Commands\Abaqus python ODB_read_MaxMises.py akpu.odb
stress=load('femres_stress.txt','-ascii');
a(2)=(stress)-Ystr;
b=0;
```

PYTHON:

```
# -*- coding: mbcs -*-
*INCLUDE,INPUT=parameters.inp
from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *
mdb.models['Model-1'].ConstrainedSketch(name='__profile__', sheetSize=1.0)
mdb.models['Model-1'].sketches['__profile__'].Line(point1=(-0.18, 0.035),
    point2=(0.0949999999860302, 0.035))
mdb.models['Model-1'].sketches['__profile__'].HorizontalConstraint(
    addUndoState=False, entity=
    mdb.models['Model-1'].sketches['__profile__'].geometry[2])
mdb.models['Model-1'].sketches['__profile__'].ObliqueDimension(textPoint=(
    -0.0490557849407196, -0.0054083913564682), value=1.2, vertex1=
    mdb.models['Model-1'].sketches['__profile__'].vertices[0], vertex2=
    mdb.models['Model-1'].sketches['__profile__'].vertices[1])
mdb.models['Model-1'].Part(dimensionality=TWO_D_PLANAR, name='Part-1', type=
    DEFORMABLE_BODY)
mdb.models['Model-1'].parts['Part-1'].BaseWire(sketch=
    mdb.models['Model-1'].sketches['__profile__'])
del mdb.models['Model-1'].sketches['__profile__']
mdb.models['Model-1'].Material(name='Material-1')
mdb.models['Model-1'].materials['Material-1'].Density(table=((7850.0, ), ))
mdb.models['Model-1'].materials['Material-1'].Elastic(table=((210000000000.0,
    0.3), ))
mdb.models['Model-1'].IProfile(b1=w, b2=w, h=h, l=centroid, name=
    'Profile-1', t1=t1, t2=t1, t3=t2)
mdb.models['Model-1'].BeamSection(consistentMassMatrix=False, integration=
    DURING_ANALYSIS, material='Material-1', name='Section-1', poissonRatio=0.0,
    profile='Profile-1', temperatureVar=LINEAR)
mdb.models['Model-1'].sections['Section-1'].setValues(poissonRatio=0.3)
mdb.models['Model-1'].parts['Part-1'].Set(edges=
    mdb.models['Model-1'].parts['Part-1'].edges.getSequenceFromMask(('[#1 ]',
    ), ), name='Set-1')
mdb.models['Model-1'].parts['Part-1'].SectionAssignment(offset=0.0,
    offsetField='', offsetType=MIDDLE_SURFACE, region=
    mdb.models['Model-1'].parts['Part-1'].sets['Set-1'], sectionName=
    'Section-1', thicknessAssignment=FROM_SECTION)
mdb.models['Model-1'].parts['Part-1'].Set(edges=
    mdb.models['Model-1'].parts['Part-1'].edges.getSequenceFromMask(('[#1 ]',
    ), ), name='Set-2')
mdb.models['Model-1'].parts['Part-1'].assignBeamSectionOrientation(method=
    N1_COSINES, n1=(0.0, 0.0, -1.0), region=
    mdb.models['Model-1'].parts['Part-1'].sets['Set-2'])
mdb.models['Model-1'].StaticStep(name='Step-1', previous='Initial')
mdb.models['Model-1'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['Model-1'].rootAssembly.Instance(dependent=OFF, name='Part-1-1',
    part=mdb.models['Model-1'].parts['Part-1'])
mdb.models['Model-1'].rootAssembly.seedPartInstance(deviationFactor=0.1,
    minSizeFactor=0.1, regions=(
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'], ), size=0.12)
mdb.models['Model-1'].rootAssembly.generateMesh(regions=(
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'], ))
mdb.models['Model-1'].rootAssembly.Set(name='Set-1', vertices=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].vertices.getSequenceFromMask(
    ('[#2 ]', ), ))
mdb.models['Model-1'].ConcentratedForce(cf1=-53030.0, cf2=-53030.0,
    createStepName='Step-1', distributionType=UNIFORM, field='', localCsys=None
    , name='Load-1', region=mdb.models['Model-1'].rootAssembly.sets['Set-1'])
mdb.models['Model-1'].rootAssembly.Set(name='Set-2', vertices=
    mdb.models['Model-1'].rootAssembly.instances['Part-1-1'].vertices.getSequenceFromMask(
    ('[#1 ]', ), ))
mdb.models['Model-1'].DisplacementBC(amplitude=UNSET, createStepName='Step-1',
    distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=None, name=
    'BC-1', region=mdb.models['Model-1'].rootAssembly.sets['Set-2'], u1=0.0,
    u2=0.0, ur3=0.0)
mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
    explicitPrecision=SINGLE, getMemoryFromAnalysis=True, historyPrint=OFF,
    memory=90, memoryUnits=PERCENTAGE, model='Model-1', modelPrint=OFF,
    multiprocessingMode=DEFAULT, name='akpu', nodalOutputPrecision=SINGLE,
    numCpus=1, numGPUs=0, queue=None, resultsFormat=ODB, scratch='', type=
    ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0)
mdb.jobs['akpu'].submit(consistencyChecking=OFF)
mdb.jobs['akpu']._Message(STARTED, {'phase': BATCHPRE_PHASE,
    'clientHost': 'G513-0030', 'handle': 0, 'jobName': 'akpu'})
mdb.jobs['akpu']._Message(ODB_FILE, {'phase': BATCHPRE_PHASE,
    'file': 'C:\\Temp\\akpu.odb', 'jobName': 'akpu'})
mdb.jobs['akpu']._Message(COMPLETED, {'phase': BATCHPRE_PHASE,
    'message': 'Analysis phase complete', 'jobName': 'akpu'})
mdb.jobs['akpu']._Message(STARTED, {'phase': STANDARD_PHASE,
    'clientHost': 'G513-0030', 'handle': 7124, 'jobName': 'akpu'})
mdb.jobs['akpu']._Message(STEP, {'phase': STANDARD_PHASE, 'stepId': 1,
    'jobName': 'akpu'})
mdb.jobs['akpu']._Message(ODB_FRAME, {'phase': STANDARD_PHASE, 'step': 0,
    'frame': 0, 'jobName': 'akpu'})
mdb.jobs['akpu']._Message(STATUS, {'totalTime': 0.0, 'attempts': 0,
    'timeIncrement': 1.0, 'increment': 0, 'stepTime': 0.0, 'step': 1,
    'jobName': 'akpu', 'severe': 0, 'iterations': 0, 'phase': STANDARD_PHASE,
    'equilibrium': 0})
mdb.jobs['akpu']._Message(MEMORY_ESTIMATE, {'phase': STANDARD_PHASE,
    'jobName': 'akpu', 'memory': 24.0})
mdb.jobs['akpu']._Message(ODB_FRAME, {'phase': STANDARD_PHASE, 'step': 0,
    'frame': 1, 'jobName': 'akpu'})
mdb.jobs['akpu']._Message(STATUS, {'totalTime': 1.0, 'attempts': 1,
    'timeIncrement': 1.0, 'increment': 1, 'stepTime': 1.0, 'step': 1,
    'jobName': 'akpu', 'severe': 0, 'iterations': 1, 'phase': STANDARD_PHASE,
    'equilibrium': 1})
mdb.jobs['akpu']._Message(END_STEP, {'phase': STANDARD_PHASE, 'stepId': 1,
    'jobName': 'akpu'})
mdb.jobs['akpu']._Message(COMPLETED, {'phase': STANDARD_PHASE,
    'message': 'Analysis phase complete', 'jobName': 'akpu'})
mdb.jobs['akpu']._Message(JOB_COMPLETED, {'time': 'Fri Dec 09 12:59:11 2016',
    'jobName': 'akpu'})
```

```
# Save by akmo16 on 2016_12_09-12.59.26; build 6.14-2 2014_08_22-16.00.46 134497
# Save by akmo16 on 2016_12_09-13.00.11; build 6.14-2 2014_08_22-16.00.46 134497
```

For displacement:

```
# ODB_read_disp.py
# A script to read displacement from the Abaqus Odb-file
# Four arguments (Name of ODB-file, Name of studied step, Name of instance, Name of nodeset)
#
# Johan Wall
# Blekinge Tekniska Hogskola
# 2015-12-09

from odbAccess import *
import sys

# Input arguments
ODBFilneName = sys.argv[1]
StepName = sys.argv[2]
InstanceName = sys.argv[3]
NodesetName = sys.argv[4]

odb = openOdb(path=ODBFilneName)

# Determine number of frames in load step.
myFrames = odb.steps[StepName].frames
numFrames = len(myFrames)

myoutfile = open('femresdisp.txt', 'w+')

# Create variables that refers to the node sets from where
# results are to be extracted.
# The sets is  associated with the part instance InstanceName (user submitted).
dispnode = odb.rootAssembly.instances[InstanceName].\
    nodeSets[NodesetName]

# Extract results from last frame in load step
CurFrame = odb.steps[StepName].frames[-1]

# Create a variable that refers to the displacement 'U'
displacement = CurFrame.fieldOutputs['U']

# Create variables that refers to the displacement
centerDisplacement = displacement.getSubset(region=dispnode)

for v in centerDisplacement.values:
    myoutfile.write(str(v.data[0]))
    myoutfile.write(' ')
    myoutfile.write(str(v.data[1]))
    myoutfile.write('\n')

myoutfile.close()
odb.close()
```

For Von-Mises stress:

```
# ODB_read_disp.py
# A script to read max vonMises from the Abaqus Odb-file, based on example in ABAQUS help files
# One argument (Name of ODB-file)
#
# Johan Wall
# Blekinge Tekniska Hogskola
# 2015-12-09

from odbAccess import *
import sys
import site

# Input arguments
ODBFilneName = sys.argv[1]
odb = openOdb(path=ODBFilneName)

elsetName = ' ALL ELEMENTS'

# Print max mises location and value given odbName and elset(optional)
elset = elemset = None
region = "over the entire model"
assembly = odb.rootAssembly

#Check to see if the element set exists in the assembly
if elsetName:
    try:
        elemset = assembly.elementSets[elsetName]
        region = " in the element set : " + elsetName;
    except KeyError:
        print 'An assembly level elset named %s does' \
              'not exist in the output database %s' \
              %(elsetName, odbName)
        odb.close()

#""" Initialize maximum values """
maxMises = -0.1
maxElem = 0
maxStep = "_None_"
maxFrame = -1
Stress = 'S'
isStressPresent = 0
for step in odb.steps.values():
    print 'Processing Step:', step.name
    for frame in step.frames:
        allFields = frame.fieldOutputs
        if (allFields.has_key(Stress)):
            isStressPresent = 1
            stressSet = allFields[Stress]
            if elemset:
                stressSet = stressSet.getSubset(
                    region=elemset)
            for stressValue in stressSet.values:
                if (stressValue.mises > maxMises):
                    maxMises = stressValue.mises
                    maxElem = stressValue.elementLabel
                    maxStep = step.name
                    maxFrame = frame.incrementNumber
if(isStressPresent):
    myoutfile = open('femres_stress.txt', 'w+')
    myoutfile.write(str(maxMises))
    myoutfile.close()

else:
    print 'Stress output is not available in' \
          'the output database : %s\n' %(odb.name)

#""" Close the output database before exiting the program """
odb.close()
```

TASK3:

```
clc;
clear all;
close all;
%
l=1;
lr1=1;
lr2=1;
E=70e9;
d=2700;
F=1000;
gf=400;
ms=127e6;
syms P1 P2
d1=[1e-3 50e-3];
d2=[1e-3 50e-3];
d3=[1e-3 50e-3];
dr1=[1e-3 50e-3];
dr2=[1e-3 50e-3];
doe=lhsdesign(10, 5, 'criterion','maximin', 'iterations',200);
doe(:,1)=d1(1)+doe(:,1)*(d1(2)-d1(1));
doe(:,2)=d2(1)+doe(:,2)*(d2(2)-d2(1));
doe(:,3)=d3(1)+doe(:,3)*(d3(2)-d3(1));
doe(:,4)=dr1(1)+doe(:,4)*(dr1(2)-dr1(1));
doe(:,5)=dr2(1)+doe(:,5)*(dr2(2)-dr2(1));
for i=1:10
    di1(i)=doe(i,1);
    di2(i)=doe(i,2);
    di3(i)=doe(i,3);
    dir1(i)=doe(i,4);
    dir2(i)=doe(i,5);
    eqn1(i)=((F-P1)*(l.^3)*(64)/(3*E*pi*di1(i).^4))-(4*P1*lr1/(pi*(dir1(i).^4)*E))-((P1-
P2)*(l^3)*64/(3*E*pi*di2(i).^4));
    eqn2(i)=((P1-P2)*(l.^3)*64/(3*E*pi*di2(i).^4))-(4*P1*lr1/(pi*(dir1(i).^4)*E))-
(P2*(l^3)*(64)/(3*E*pi*di3(i).^4));
    [Tensile1(i),Tensile2(i)]=solve(eqn1(i)==0,eqn2(i)==0);
    c1(i)=(F-Tensile1(i))./(pi*di1(i)^2);
    c2(i)=(Tensile1(i)-Tensile2(i))./(pi*di2(i)^2);
    c3(i)=(Tensile2(i))./(pi*di3(i)^2);
    c4(i)=F-Tensile1(i);
    c5(i)=Tensile1(i)-Tensile2(i);
    c6(i)=Tensile2(i);
    A(i)=pi*(di1(i).^2+di2(i).^2+di3(i).^2+dir1(i).^2+dir2(i).^2);
    Sr1=(F-Tensile1(i))*(l^3)/(3*E.*A(i));
    sig1(i)=(F-Tensile1(i)).*l.*(di1(i)./2)/(pi*di1(i).^4/64);
    sig2(i)=(Tensile1(i)-Tensile2(i)).*l.*(di2(i)./2)/(pi*di2(i).^4/64);
    sig3(i)=(Tensile2(i))/(pi*di3(i).^2);
    sig4(i)=Tensile1(i)./(pi*dir1(i).^2);
    sig5(i)=Tensile2(i)./(pi*dir2(i).^2);
    ar(i)=pi.*(di1(i).^2+di2(i).^2+di3(i).^2+dir1(i).^2+dir2(i).^2);
    def(i)=(F-Tensile1(i).^3)./(3*E*pi.*di1(i).^4/64);
    ExpRes(i)=(ar(i)+def(i))/2;
    ResMatEst(i)=[ones(10,1) doe(:,1) doe(:,2) doe(:,3) doe(:,4) doe(:,5) doe(:,1).*doe(:,2) doe(:,2).*doe(:,3)
doe(:,3).*doe(:,4) doe(:,4).*doe(:,5) doe(:,1).*doe(:,3) doe(:,1).*doe(:,4) doe(:,1).*doe(:,5) doe(:,2).*doe(:,4)
doe(:,2).*doe(:,5) doe(:,3).*doe(:,5) doe(:,1).*doe(:,2).*doe(:,3) doe(:,1).*doe(:,2).*doe(:,4)
doe(:,1).*doe(:,2).*doe(:,5) doe(:,2).*doe(:,3).*doe(:,4) doe(:,2).*doe(:,3).*doe(:,5) doe(:,3).*doe(:,4).*doe(:,5)
doe(:,2).*doe(:,4).*doe(:,5) doe(:,1).*doe(:,4).*doe(:,5) doe(:,1).*doe(:,3).*doe(:,5) doe(:,1).*doe(:,3).*doe(:,4)
doe(:,1).*doe(:,2).*doe(:,3).*doe(:,4) doe(:,1).*doe(:,2).*doe(:,3).*doe(:,5) doe(:,1).*doe(:,2).*doe(:,4).*doe(:,5)
doe(:,2).*doe(:,3).*doe(:,4).*doe(:,5) doe(:,1).*doe(:,3).*doe(:,4).*doe(:,5)
doe(:,1).*doe(:,2).*doe(:,3).*doe(:,4).*doe(:,5)];
    ResMatTrans(i)=ResMatEst(i);
    ft=ResMatTrans(i)./doe(i);
end
%%
diabeam1=doe(:,1);
Strbeam1=[sig1];
plot(diabeam1,Strbeam1,'x');
xlabel('Different sizes of beam 1','fontsize',13);
ylabel('Stress in Beam 1','fontsize',13);
grid on
figure;
diabeam2=doe(:,2);
Strbeam2=[sig2];
plot(diabeam2,Strbeam2,'x');
xlabel('Different sizes of beam 2','fontsize',13);
ylabel('Stress in Beam 2','fontsize',13);
grid on
figure;
diabeam3=doe(:,3);
Strbeam3=[sig3];
plot(diabeam3,Strbeam3,'x');
xlabel('Different sizes of beam 3','fontsize',13);
ylabel('Stress in Beam 3','fontsize',13);
grid on
figure;
Gf1=[c4];
Gf2=[c5];
Gf3=[c6];
plot(diabeam1,Gf1,'x');
xlabel('Different sizes of beam 1','fontsize',13);
ylabel('Force to the ground','fontsize',13);
grid on
figure;
plot(diabeam2,Gf2,'x');
xlabel('Different sizes of beam 2','fontsize',13);
ylabel('Force to the ground','fontsize',13);
grid on
figure;
plot(diabeam3,Gf3,'x');
```

```
xlabel('Different sizes of beam 3','fontsize',13);
ylabel('Force to the ground','fontsize',13);
grid on
figure;
diarod1=doe(:,4);
Strrod1=[sig4];
plot(diarod1,Strrod1,'x');
xlabel('Different sizes of rod 1','fontsize',13);
ylabel('Stress in rod 1','fontsize',13);
grid on;
figure;
diarod2=doe(:,5);
Strrod2=[sig5];
plot(diarod2,Strrod2,'x');
xlabel('Different sizes of rod 2','fontsize',13);
ylabel('Stress in rod 2','fontsize',13);
grid on;
```