Universitat Politècnia de Catalunya

ESEIAAT

DEGREE IN AEROSPACE TECHNOLOGY ENGINEERING

Assignment 3

ELASTOSTATIC ANALYSIS OF A CANTILEVER BOX BEAM

Author
Bernardo Márquez López

Professor
Joaquín Hernández Ortega
Department of materials and structural resistance in engineering

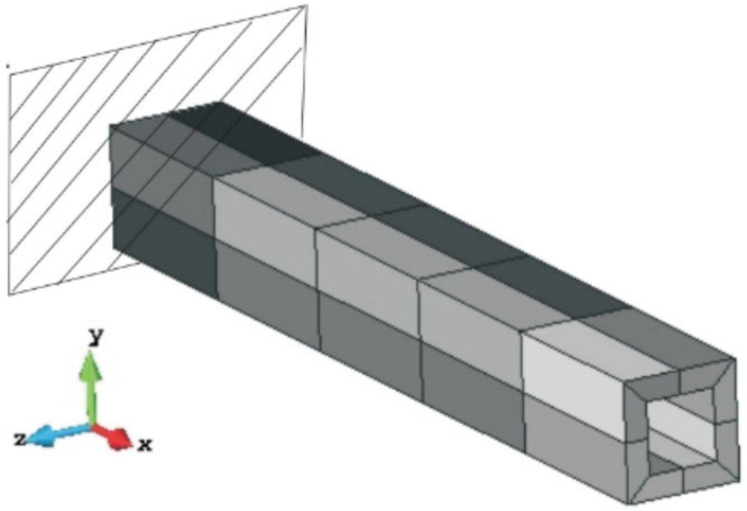
November 2022

Contents

List of Figures

General statement of the assignment

Consider the cantilever box beam depicted in Figure 1.



The length of the beam is $L_X = 2m$, its width and height $h_y = h_z = 0.25m$, and its thickness $e = 0.05m$. The material is isotropic, with $E = 70 \cdot 10^6 kN/m^2$, and $\nu = 0.3$. The beam is subjected to a uniformly distributed load of value $t^{(2)} = 500kN/m^2$ on its top surface ($y = y_{max}$)

Assess the convergence upon mesh refinement, by launching 4 different analyses with increasing number of finite elements (hexahedral). In particular, use semi-structured meshes with:

- *MESH 1: n_{ex} = 2 divisions in the $x-$direction, and typical size in the y-z plane of 0.1 m.*
- *MESH 2: n_{ex} = 10 divisions in the $x-$direction, and typical size in the y-z plane of 0.05 m.*
- *MESH 3: n_{ex} = 15 divisions in the $x-$direction, and typical size in the y-z plane of 0.05 m.*
- *MESH 4: n_{ex} = 20 divisions in the $x-$ direction, and typical size in the y-z plane of 0.025 m.*

Once the performance of the code has been properly assessed, study, for MESH 4, the finite element solution corresponding to the load state in which a torque T = 100 kN m is applied at the free end (see figure below). HINT: replace this point torque by any statically equivalent system of point/distributed forces. Code a matlab routine able to automatically compute the resultant of the reaction forces at the fixed end (Rx, Ry, Rz as well as Mx, My and Mz). Check that such reactions are in equilibrium with the prescribed forces.

1 Part 1 (Basic)

1.1 Generation of the mesh

The first step in the development of this assignment will be the definition of the geometry of the problem and the generation of the mesh in that geometry. In this assignment it is important to notice the rellevance of the axis directions in the definition of this geometry, as all the boundary conditions will be defined by that definition.

First of all, the cross-sectional surface is created, as it is shown in figure 1. Next, this surface is extruded, creating the 3D geometry of the beam, as it is shown in figure 2.

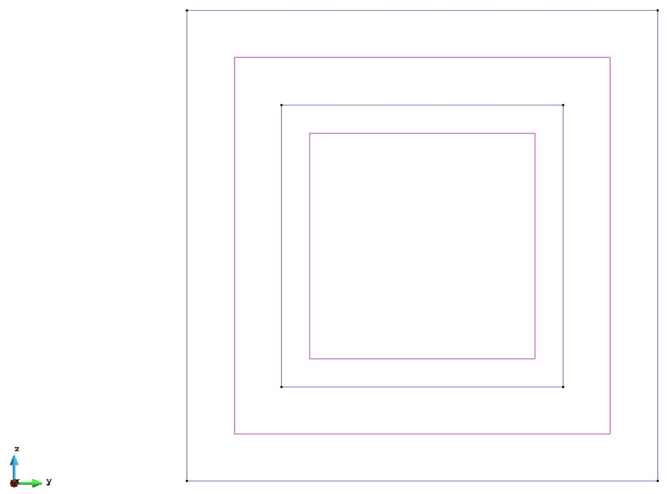


Figure 1: Creation of the cross-sectional surface of the beam.

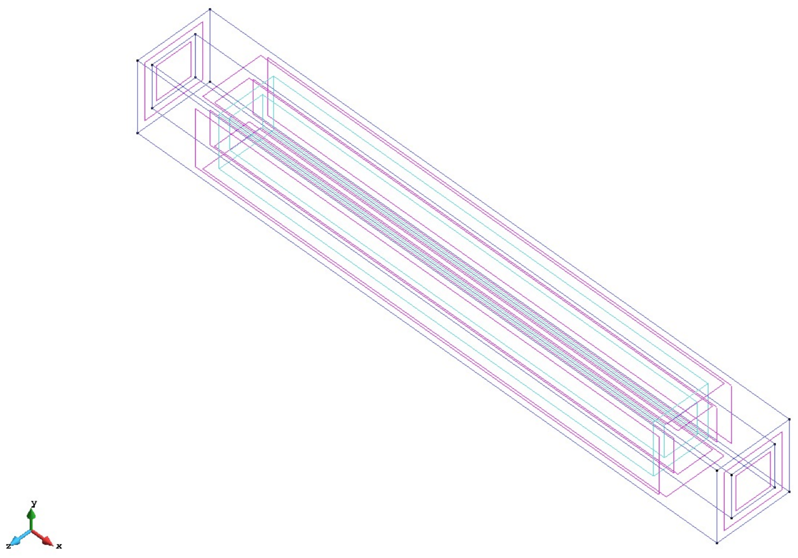


Figure 2: 3D extrusion of the beam.

Finally, once the beam geometry has been created, the material and conditions are assigned, and the different meshes are created, using in this case different semi-structured meshes of hexaedral elements, as defined in the statement.

Figure 3 shows the definition of the first mesh requested in the statement on the beam geometry, corresponding of 2 divisions in the x-direction, and typical size of the plane y-a of 0.10m.

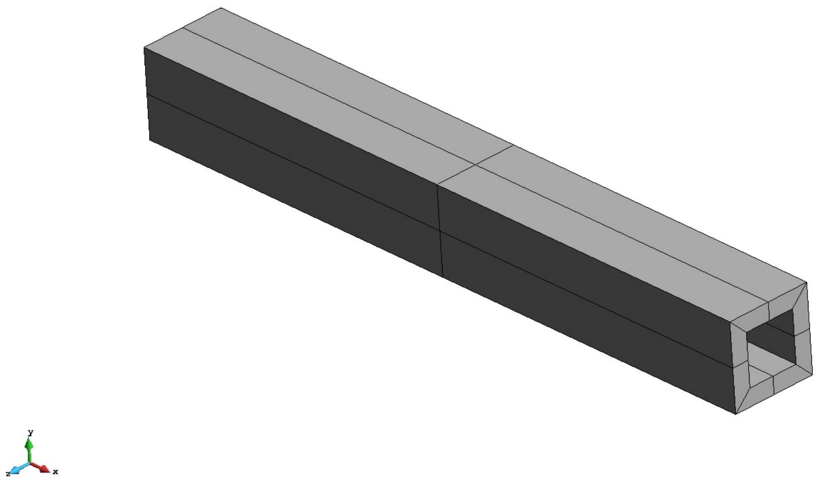


Figure 3: Generation of the first mesh on the beam geometry.

1.2 Development of the finite elements program

As in the assignment 2, in this assignment most of the functions necessary for the finite elements program were already defined, and there where only some parts missing which were needed to implement. In the case of this assignment, these parts have been the assembly of stiffness matrix K, the implementation of shape functions routine for trilinear hexaedral elements and the code for solving the final system of equations. The developed codes are shown in the Appendix of this report, as well as some other rellevant auxiliar functions implemented. In the next subsections, the different Matlab codes developed will be explained in detail.

1.2.1 Assembly of the stiffness matrix K

The complete code for this part is shown in Appendix A.1. In this code, it was necessary to assembly the global system stiffness matrix K by storing the stiffness submatrix corresponding to each element, computed by the function $ComputeKeMatrix$. As shown in the code, the assembly of K matrix was done by a loop over the number of elements. Differently as in assignment 2, as in assignment 3 the problem has a 3D geometry, there are 3 degrees of freedom per node.

1.2.2 Implementation of shape functions routine for trilinear hexaedral elements

The code developed for this function is shown in Appendix A.2. The base for this code is the same as in the case of the function *Quadrilateral4InPoint* developed for assignment 2, but now it has been implemented for trilinear hexaedral elements. This function provides the weights and positions of the points for 2D gaussian quadrature, and then computes the element shape functions N^e and B^e for these Gauss points positions. These functions have the following structures in this case:

$$N^e(1:4) = \frac{1}{8}\left[(1-\xi)\cdot(1-\eta)\cdot(1-\zeta) \quad (1+\xi)\cdot(1-\eta)\cdot(1-\zeta) \quad (1+\xi)\cdot(1+\eta)\cdot(1-\zeta) \quad (1-\xi)\cdot(1+\eta)\cdot(1-\zeta)\right]$$

$$N^e(5:8) = \frac{1}{8}\left[(1-\xi)\cdot(1-\eta)\cdot(1+\zeta) \quad (1+\xi)\cdot(1-\eta)\cdot(1+\zeta) \quad (1+\xi)\cdot(1+\eta)\cdot(1+\zeta) \quad (1-\xi)\cdot(1+\eta)\cdot(1+\zeta)\right]$$

$$B^e(:,1:4) = \begin{bmatrix} -(1-\eta)\cdot(1-\zeta) & (1-\eta)\cdot(1-\zeta) & (1+\eta)\cdot(1-\zeta) & -(1+\eta)\cdot(1-\zeta) \\ -(1-\xi)\cdot(1-\zeta) & -(1+\xi)\cdot(1-\zeta) & (1+\xi)\cdot(1-\zeta) & (1-\xi)\cdot(1-\zeta) \\ -(1-\xi)\cdot(1-\eta) & -(1+\xi)\cdot(1-\eta) & -(1+\eta)\cdot(1+\eta) & -(1-\eta)\cdot(1+\zeta) \end{bmatrix}$$

$$B^e(:,5:8) = \begin{bmatrix} -(1-\eta)\cdot(1+\zeta) & (1-\eta)\cdot(1+\zeta) & -(1-\eta)\cdot(1+\zeta) & (1+\eta)\cdot(1+\zeta) \\ -(1-\xi)\cdot(1+\zeta) & -(1+\xi)\cdot(1+\zeta) & (1+\xi)\cdot(1+\zeta) & (1-\xi)\cdot(1zeta) \\ -(1-\xi)\cdot(1-\eta) & (1+\xi)\cdot(1-\eta) & (1+\xi)\cdot(1+\eta) & (1-\xi)\cdot(1+\eta) \end{bmatrix}$$

These shape functions N and B will be determined using the adequate weights of each point of the gaussian quadrature.

1.2.3 Solution of the final system of equations

The code for solving the final system of equations is shown in Appendix A.3. The function *SolveELAS* operates and gives the solution of the final system of equations, which has the following matricial expression:

$$\begin{bmatrix} K_{RR} & K_{RL} \\ K_{LR} & k_{LL} \end{bmatrix} \cdot \begin{bmatrix} d_R \\ d_L \end{bmatrix} = \begin{bmatrix} f_R \\ f_L \end{bmatrix} \tag{1}$$

Then, the operation made by this function is the following:

$$d_L = K_{LL}^{-1} \cdot (f_L - K_{LR} \cdot d_R) \tag{2}$$

1.3 Simulation results

After the implementation of the necessary functions so that the code is fully functional, in this section the post-processing results given by *GID CIMNE* software taking the Matlab code results will be shown. The analysis will be undertaken for the different meshes defined in the statement in order to study the convergence in the results for an increasing number of discretization elements of the beam geometry.

1.3.1 Simulation for the first Mesh (2 divisions in X direction, typical size of 0.1m in plane YZ)

The first mesh studied is a semi-structured mesh which has a geometry characterised for having 2 divisions in the X direction, and a typical length of 0.10m m in the plane YZ. This mesh is shown in figure 4.

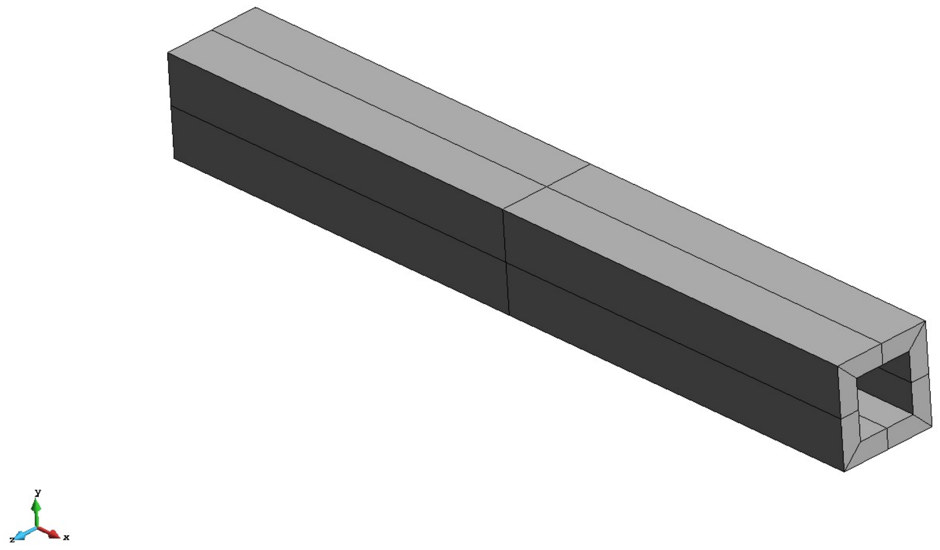


Figure 4: Generated mesh of the problem for the 2 divisions in the x-direction case.

After solving the systems for displacements implemented in the finite elements program, a file *.res* has been generated and it has been processed by *GID* in its post-processing feature. Analysing the vertical diplacements of the beam, the simulation shows the following results:

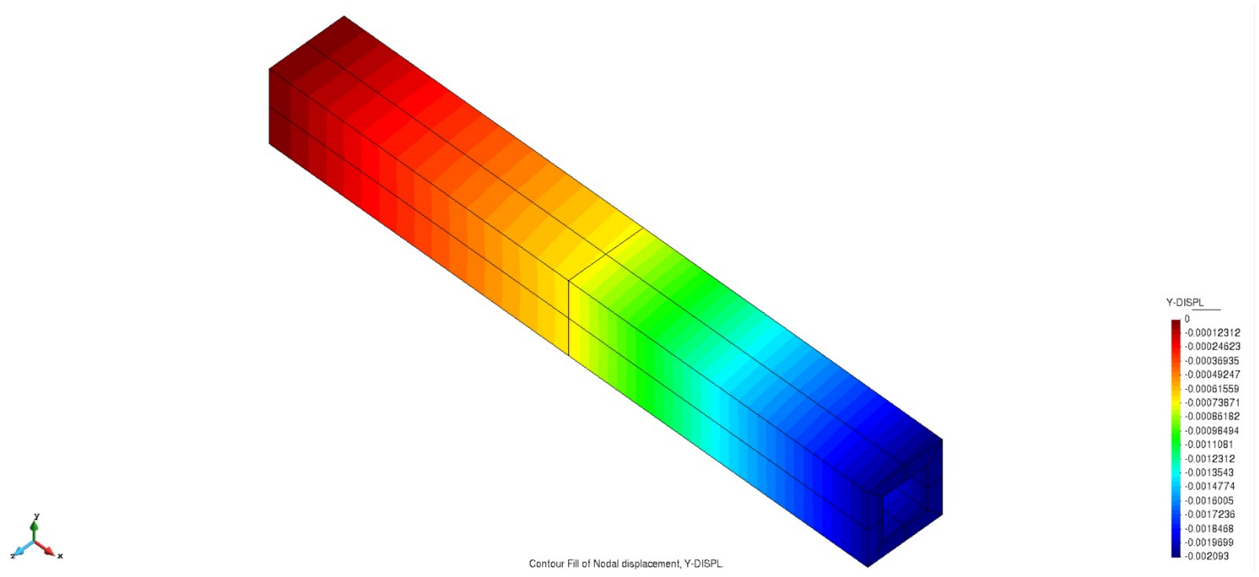


Figure 5: Vertical displacements on the beam for the first mesh($n_x = 2$).

Analysing the vertical displacements on the upper surface of the beam for this discretization, the *GID* simulation has provided the results shown in figure 6.

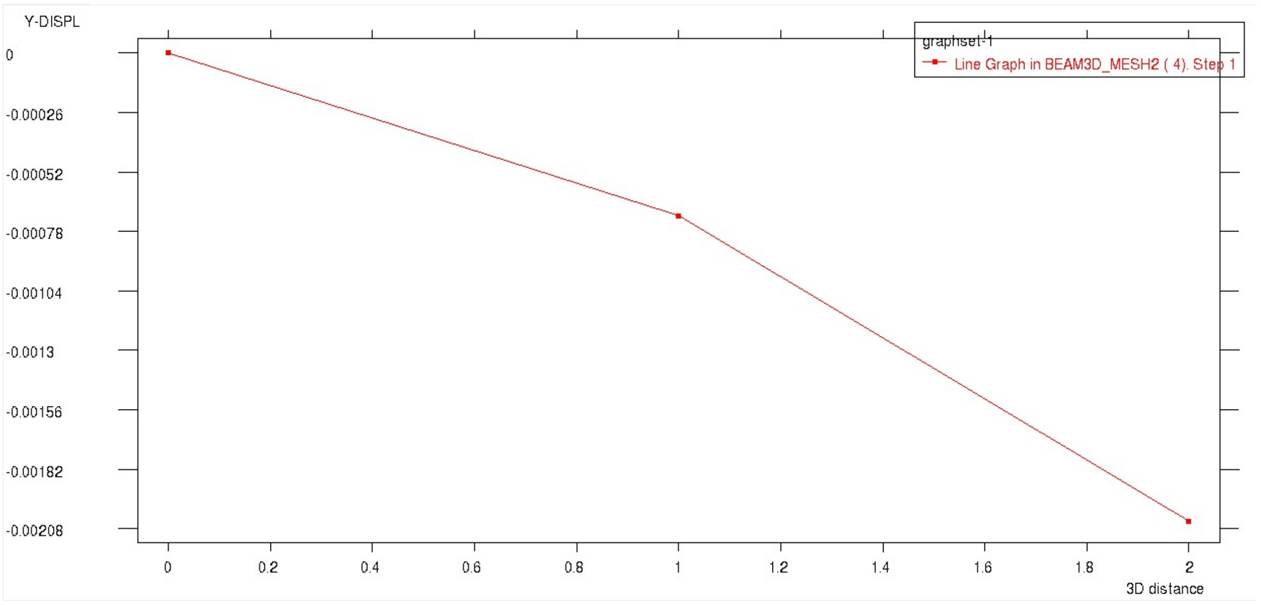


Figure 6: Vertical displacements distribution on the upper surface of the beam for the first mesh($n_x = 2$).

1.3.2 Simulation for the First Mesh (10 divisions in X direction, typical size of 0.05 m in plane YZ)

The second mesh that has been analysed is a semi-structured mesh of hexaedral elements with similar features as the previous one, but in this case this mesh is charactecterised for having 10 divisions in the X direction and a typical size of 0.05m in the YZ plane. The generated mesh in this case is shown in figure 7

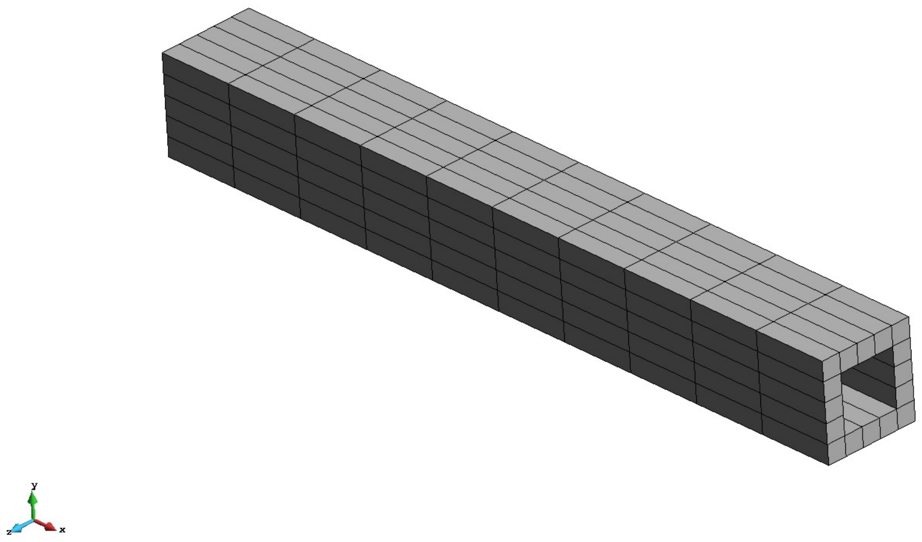


Figure 7: Generated mesh of the problem for the 10 divisions in the x-direction case.

Post-processing the results computed by the Matlab code, the vertical displacements field over the beam in this case is shown in the following *GID* simulation:

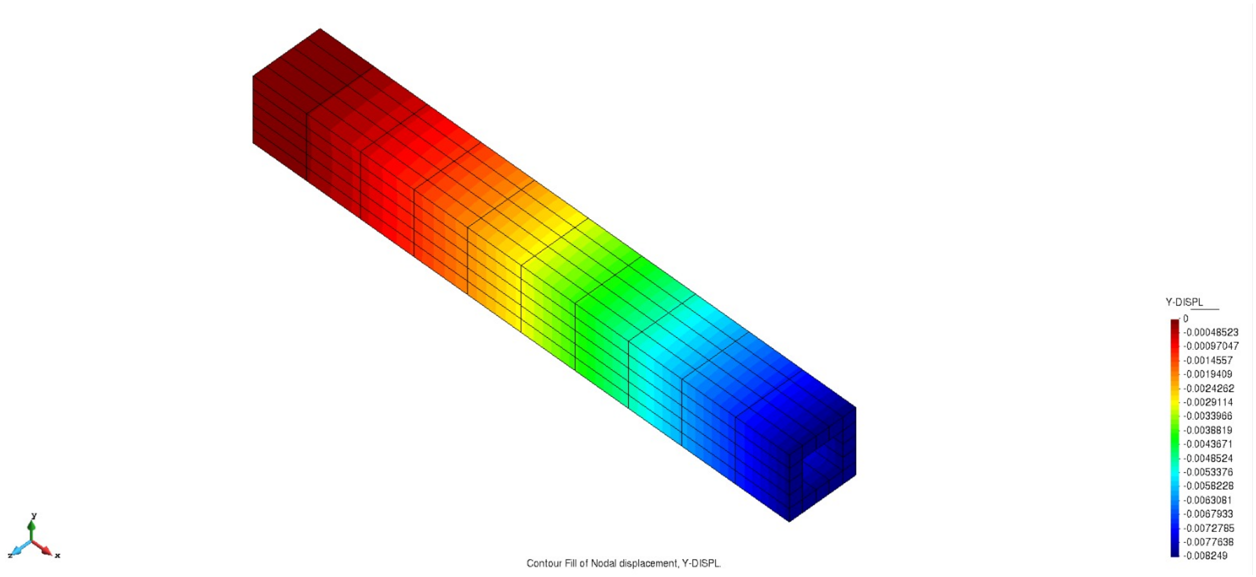


Figure 8: Vertical displacements field distribution on the beam for the second mesh($n_x = 10$).

Studying the vertical displacements distributions in this case, as it was done in the previous one, the *GID* post-processing interface has provided the results presented in the graph of figure 9.

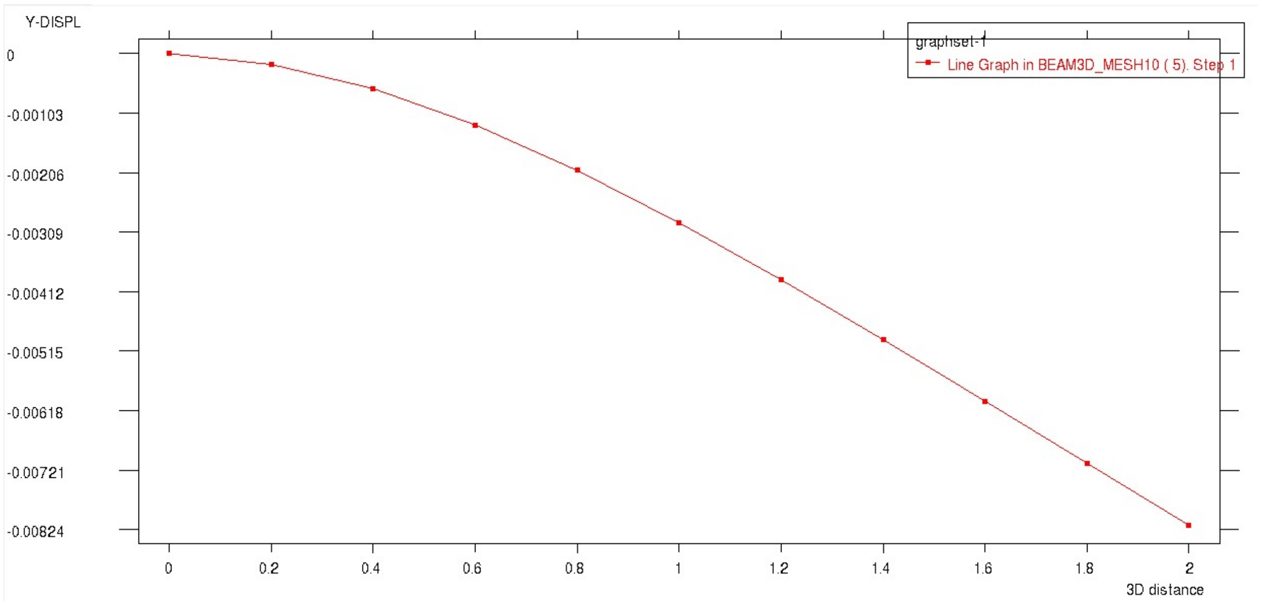


Figure 9: Vertical displacements distribution on the upper surface of the beam for the second mesh($n_x = 10$).

1.3.3 Simulation fo the Second Mesh (15 divisions in X direction, typical size of 0.05 m in plane YZ)

The next mesh to analyse is a semi-structured mesh of hexaedral elemenents which has the same typical size of 0.05m in the plane XZ as in the mesh in the previous subsection, but in this case the number of divisions in the X direction has been increased to 15. The generated mesh in this case is shown in figure 10.

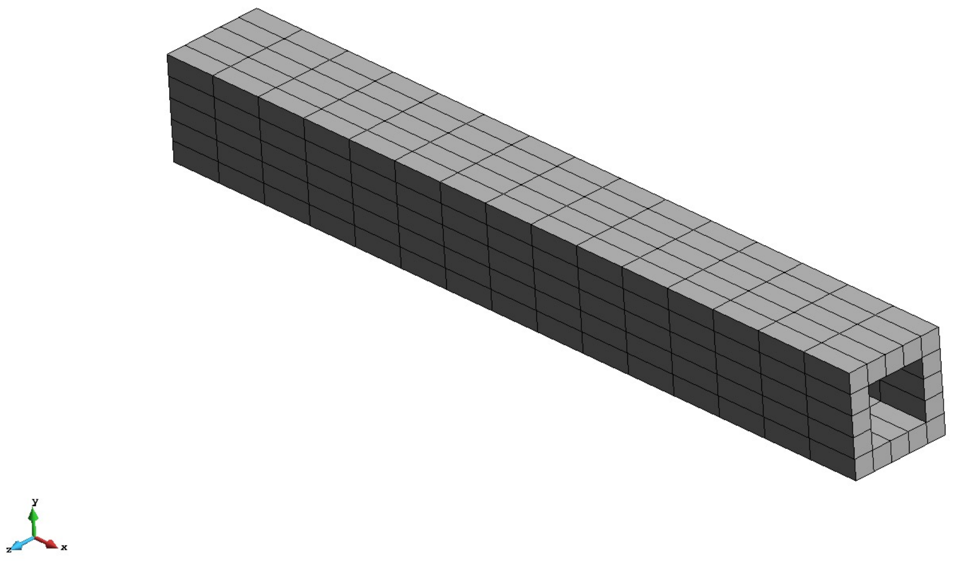


Figure 10: Generated mesh of the problem for the 15 divisions in the x-direction case

After post-processing the results computed by the Matlab finite elements program, the simulation of the vertical displacements field distribution over the beam has been obtained from *GID*, as it is shown in figure 11.

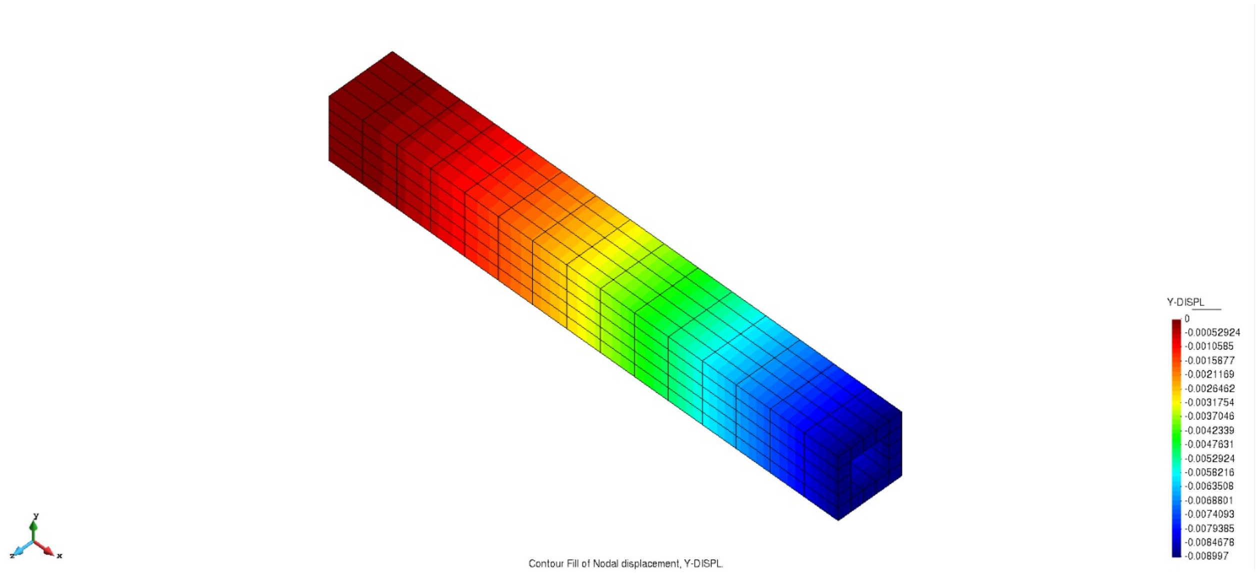


Figure 11: Vertical displacements field distribution on the beam for the third mesh($n_x = 15$).

For this post-processing results, the vertical displacements field distribution over the upper surface of the beam has been studied again for this new discretization, as it is shown in the graph of figure 12.

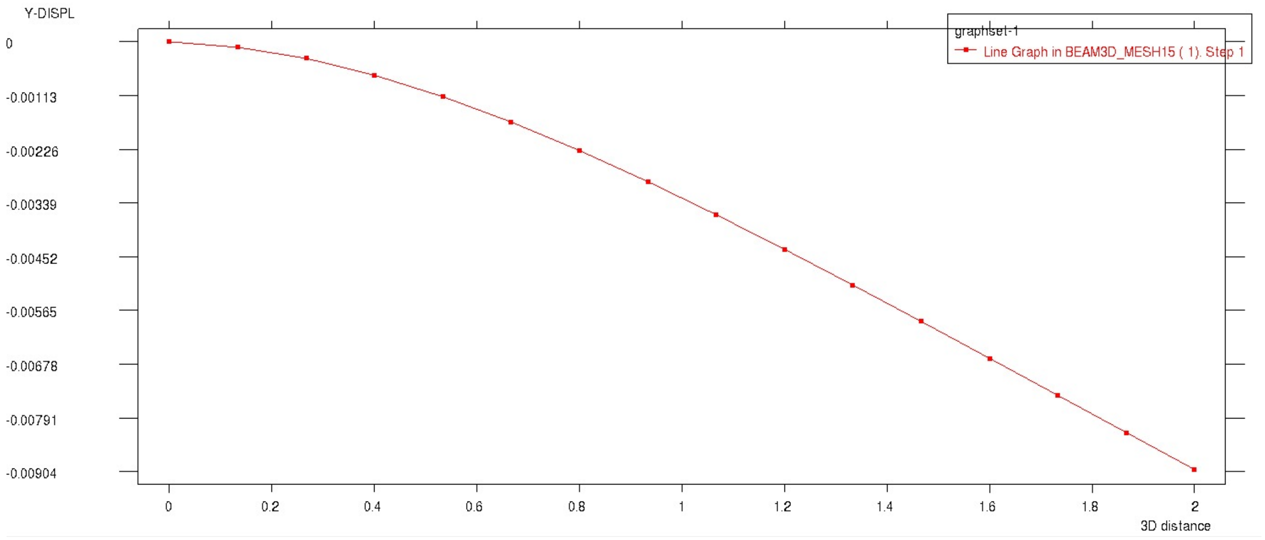


Figure 12: Vertical displacements distribution on the upper surface of the beam for the third mesh($n_x = 15$).

1.3.4 Simulation fo the Third Mesh (20 divisions in X direction, typical size of 0.025 m in plane YZ)

Finally, the last mesh that has been assessed in the finite elements analysis of the problem is a semi-structured hexaedral mesh which has 20 divisions in the X direction and a typical size of 0.025m in the plane YZ. The result of the mesh generation in the *GID* software is shown in figure 13

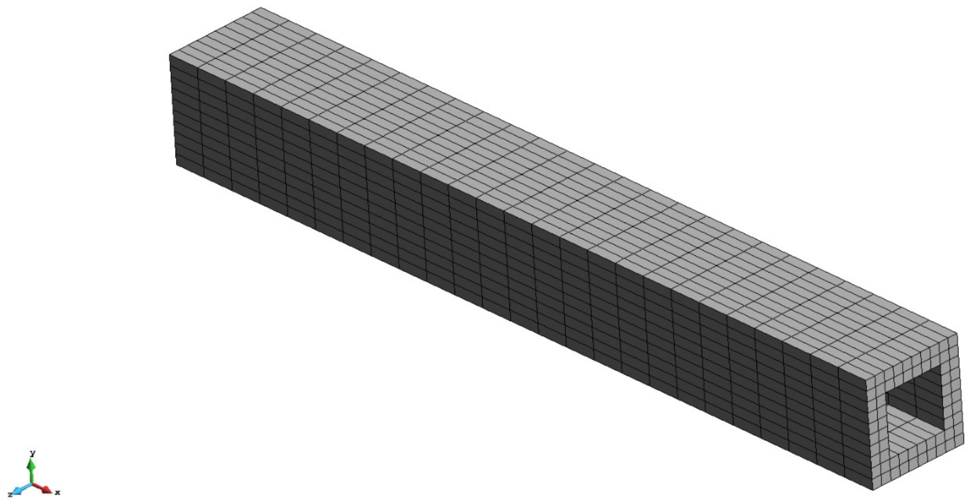


Figure 13: Generated mesh of the problem for the 20 divisions in the x-direction case

In this final case, the simulation of the vertical displacements field obtained from the *GID* post-processing feature for the results computed by the Matlab code for this mesh are shown in figure 14.

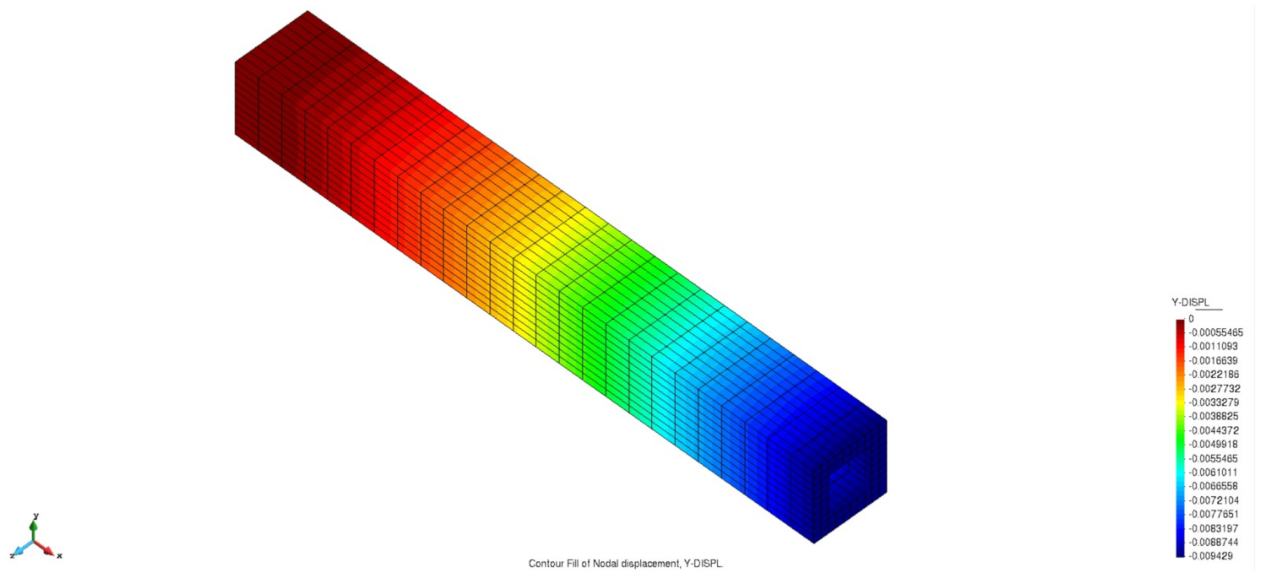


Figure 14: Vertical displacements on the beam for the fourth mesh($n_x = 20$).

Finally, as in the previous cases, the vertical displacements distribution over the upper surface of the beam has been simulated in this post-process stage. The results provided by the *GID* simulation can bee seen in the graph of figure 15.

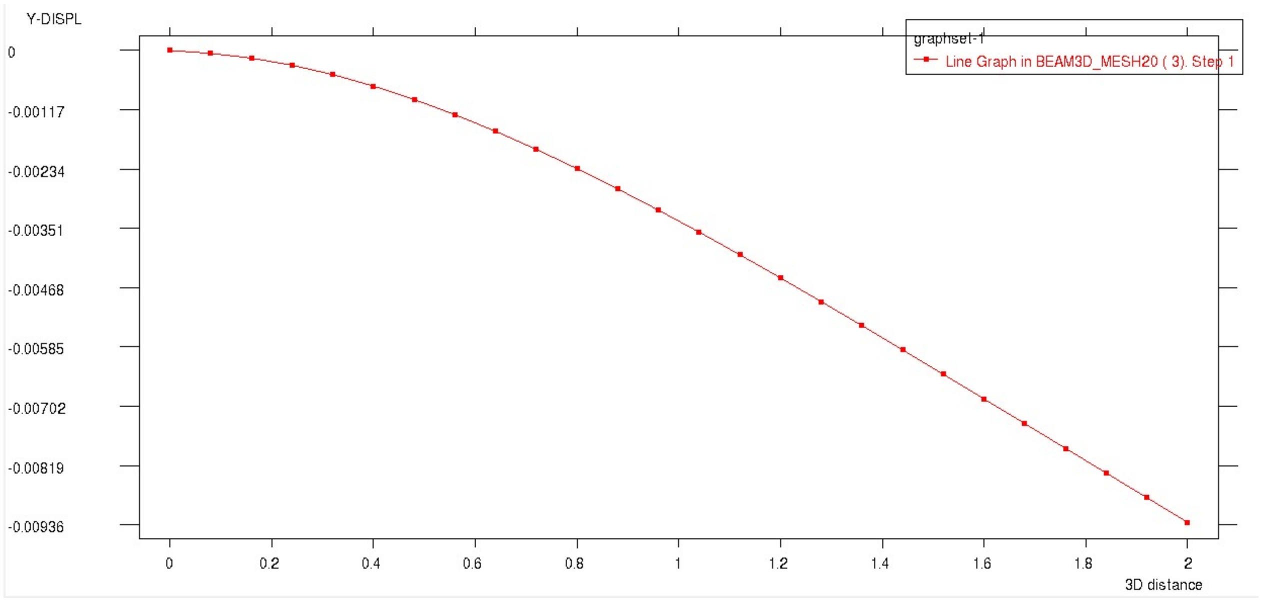


Figure 15: Vertical displacements on the beam for the fourth mesh($n_x = 20$).

1.4 Theoretical tip deflection

In order to analyse more accurately the convergence of the results provided by the simulations presented in the previous section for different meshes, in this section the teorethical value of the deflection at the beam tip will be determined taking into account the geometrical parameters information given in the statement.

First of all, the forces and momentum diagrams of the body are determined,as it is shown in figure 16. In this figure, t represents the uniform load distributed over the beam, and h is the beam length in the Y direction.

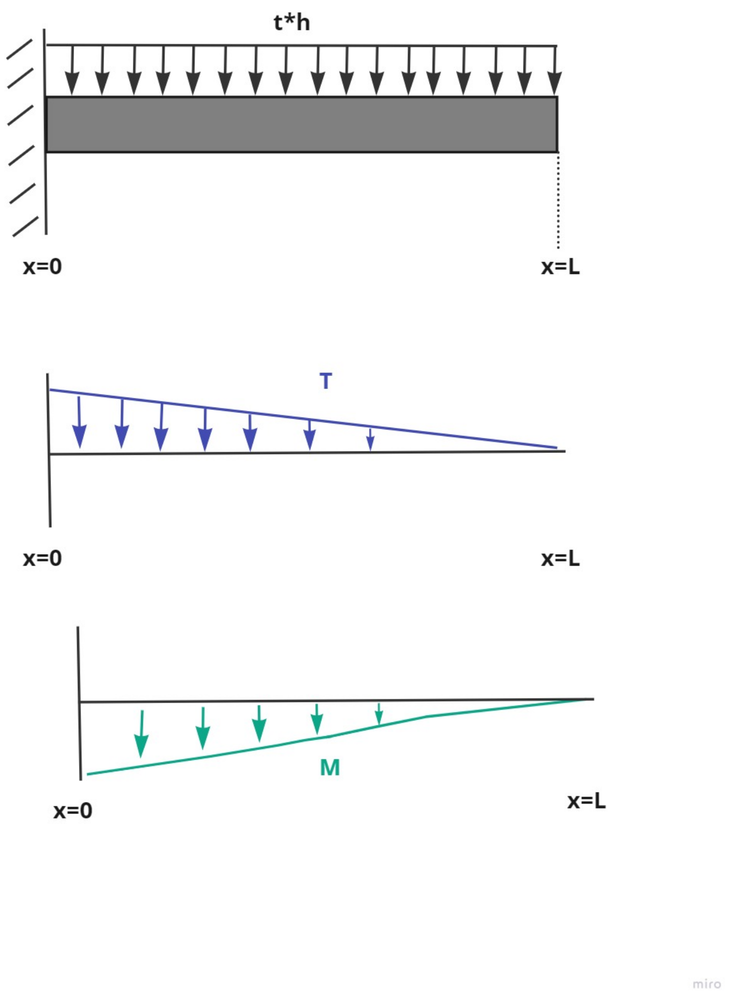


Figure 16: Distribution of forces, shear and momentum over the beam.

From these diagrams, the expressions of the forces and momentums on the beam can be obtained:

$$T = th(L-x) \tag{3}$$

$$M = \int T dx = th(Lx - \frac{x^2}{2}) \tag{4}$$

Then, the vertical displacements of the beam (δ) can be determined applying the following expression:

$$\delta = \iint \frac{qh}{EI}\left(Lx - \frac{x^2}{2}\right) dxdx \tag{5}$$

Applying the boundary conditions of the problem ($\delta = 0m$ at $x = 0m$ and the rotation in the tip is considered 0, $\theta = 0$ at $X = L$), then the displacement is formulated as:

$$\delta(x) = -\frac{h \cdot t \cdot x^2}{24 \cdot E \cdot I} \cdot (6 \cdot L^2 - 4 \cdot L \cdot x + x^2) \tag{6}$$

As it is a square-section beam, the inertia of the section can be determined as:

$$I = \frac{1}{12}[h^4 - (h-2e)^4] = 2.83 \cdot 10^{-4} m \tag{7}$$

Finally, substituting this value for the previous equation, $\delta(L)$ can be obtained from the physical parameters given in the statement:

$$\delta(L) \approx -0.0126m \tag{8}$$

Once the theoretical value of the vertical displacement at the tip of the beam has been obtained, in the next section the convergence of the simulations from finite elements program will be assessed.

1.5 Convergence analysis of the finite elements program results

In this section the convergence of finite elements programs results will be studied by comparing the simulation results with the theoretical prescribed displacements given by equation 6. Figure 17 represents a graph of this convergence.

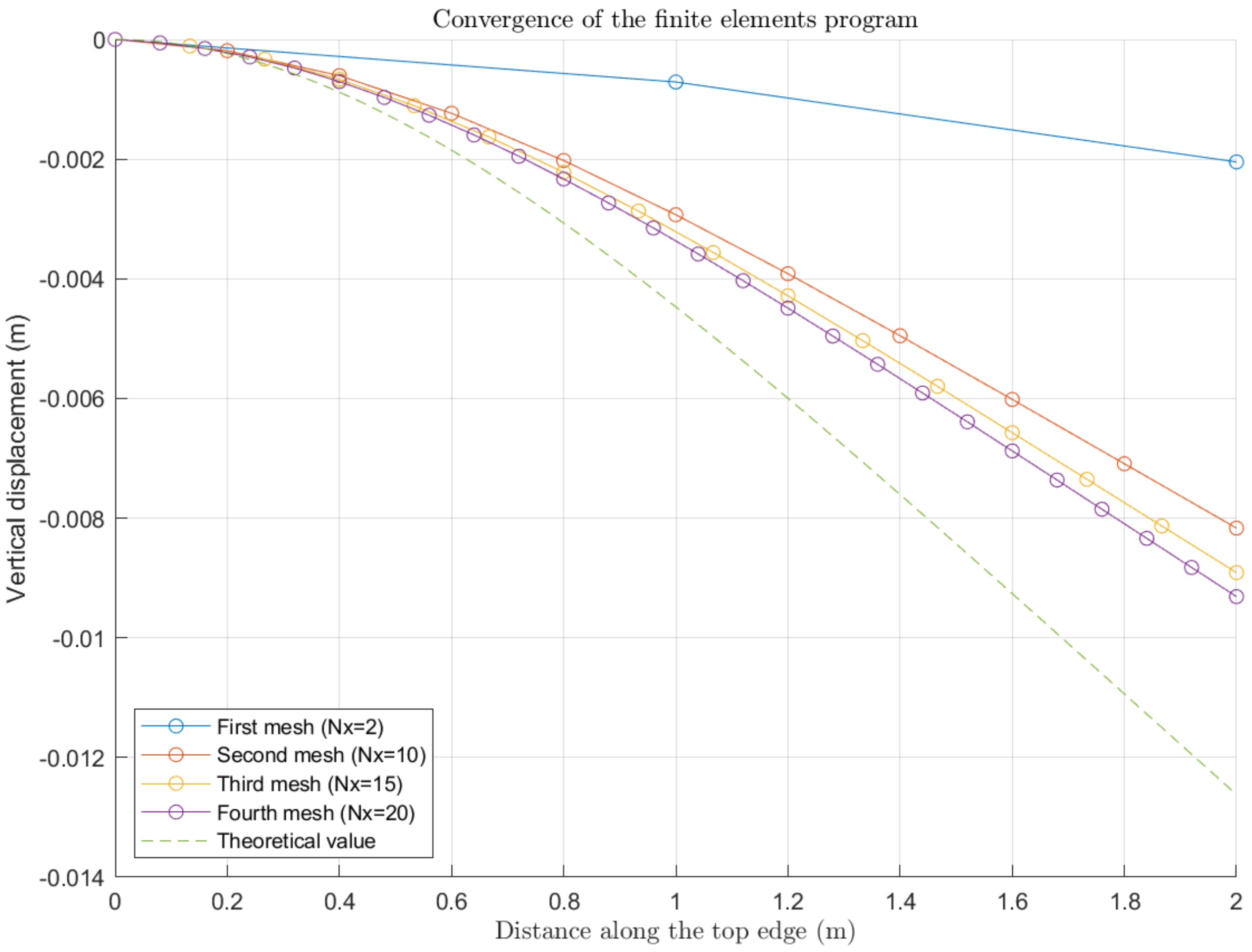


Figure 17: Convergence of the finite elements program with the theoretical values for vertical displacements.

In this graph it can be seen that the relative error between simulation results and theoretical results decreases as the number of discretization elements in the mesh increases. As commented in the previous assignment, mesh refinement can be seen as the relative error between numerical results tends to minimize as the discretization has a higher number of elements, which implies the existence of convergence in the results.

1.6 Finite elements program with a torque applied to the beam tip

The program implemented in the previous sections allows calculations with forces, but cannot operate directly with torques. Therefore, in order to determine the reactions of the beam subjected to a torque at its tip, Saint Venant's principle will be applied. From this principle, it is concluded that a torque applied at the tip of a beam has a similar effect as two forces with opposed directions applied at the nodes located in the half of the left and right side edges of the beam. This equivalent force can be determined the following way:

$$F = \frac{T}{h} = \frac{100}{0.25} = 400kN \tag{9}$$

For this analysis of the torque effects in the beam tip, the fourth mesh is used. Then, it is necessary to determine the nodes where the forces will be applied. These are the nodes 37 and 236 defined in the mesh, as shown in figure 18.

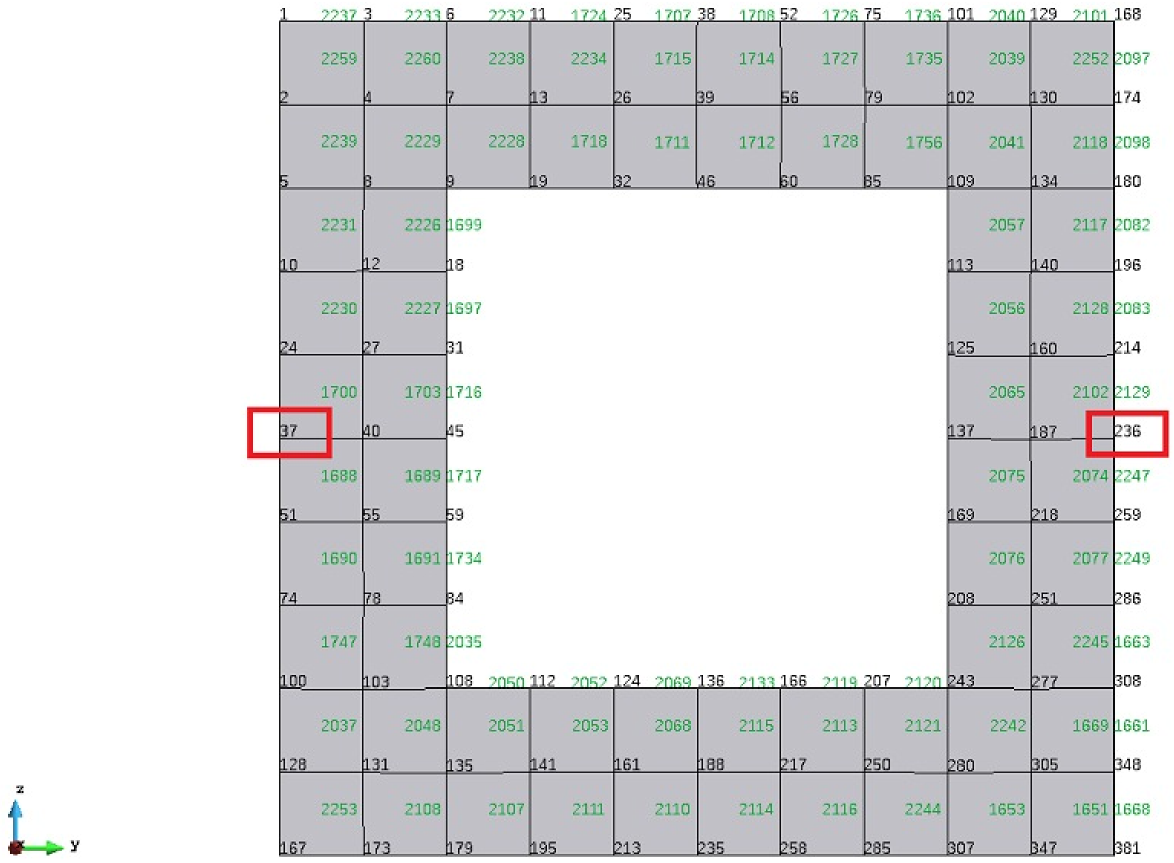


Figure 18: Location of the nodes where the equivalent forces to the torque will be applied.

Then, introducing the designations of these nodes and the forces values, and running the finite elements program implemented in the preprocess section, the following results are obtained for vertical and horizontal displacements of the beam when both the torque and the uniform load are applied:

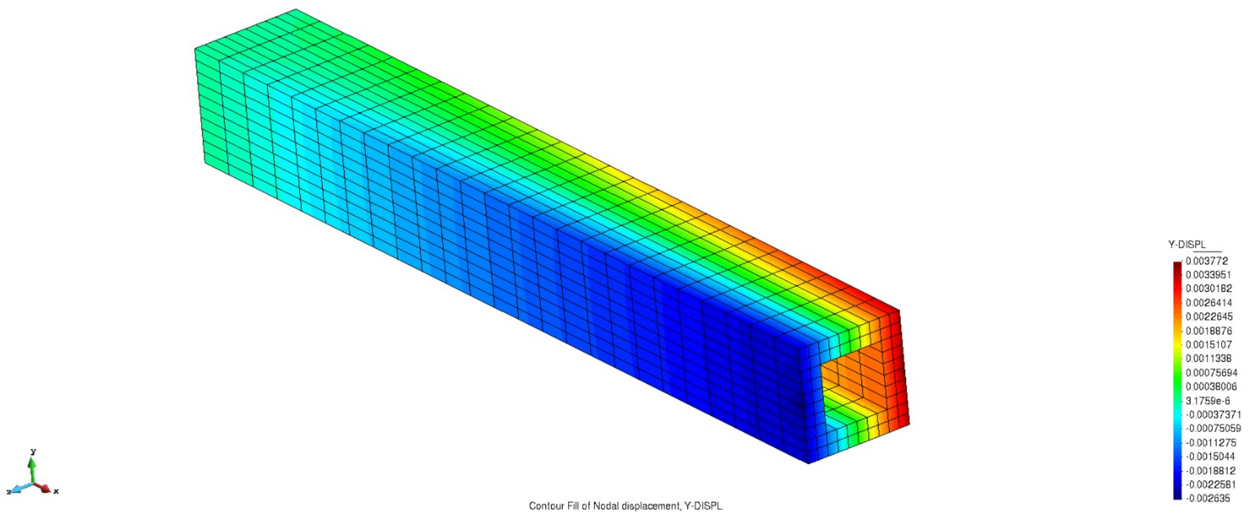


Figure 19: Vertical displacements field distribution on the beam in the conditions of application of a 100kNm torque in the tip and an uniform load over the upper surface of 500kN/m^2.

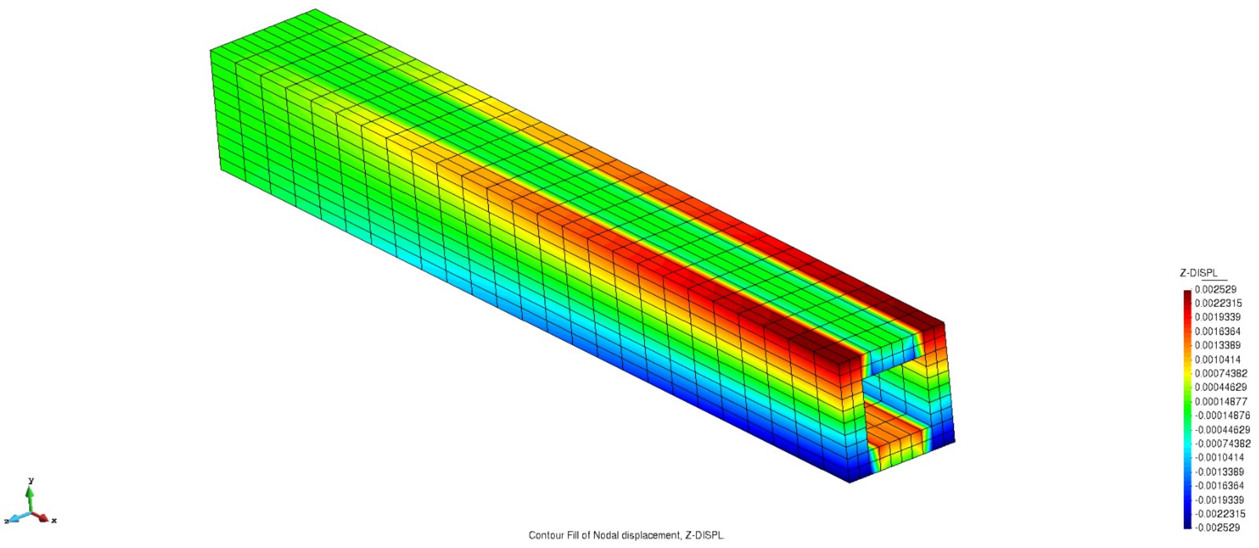


Figure 20: Vertical displacements field distribution on the beam in the conditions of application of a 100kNm torque in the tip, and an uniform load over the upper surface of 500kN/m^2.

The distribution of displacements is assymetrical the uniform load is acting in the Y axis, affecting the horizontal displacement generated from the rotation of the torque at the tip.

1.7 Computation of the reaction forces of the system

In this section, a Matlab function will be developed in order to compute the reaction forces and moments at the fixed end of the beam in different conditions. The code implemented for this program is shown in Appendix A.5. This program gives the following results:

$$R_x = 0MN \quad ; \quad R_y = 0.194MN \quad ; \quad R_z = 0MN$$

$$M_x = 0.001862MN \quad ; \quad M_y = 0.109MN \quad ; \quad M_z = 0.488MN$$

These results are the corresponding to a situation where both the uniform load and the torque are being applied simultaneously to the beam, therefore the system is not in equilibrium. If only the torque was applied, the system would be in static equilibrium.

2 Part 2 (Advanced)

In this second part, the derivation of a discrete system of a discrete system of equations needs to be achieved, taking into account the thermo-mechanical constitutive equation in the case that stress field is dependent on the temperature. This constituve equation is the following:

$$\sigma(T) = C \cdot (\nabla^s u - \alpha \Delta T) = C \cdot \nabla^s u - \beta \Delta T \tag{10}$$

The Variational Formulation of the Boundary Value Problem corresponding is given by the following expression:

$$\int_\Omega \nabla^s v^T \cdot \sigma \cdot d\Omega = \int_\Omega v^T f d\Omega + \int_{\Gamma_\sigma} v^T \bar{t} d\Gamma \tag{11}$$

Then, substituting this definition of the stress field in the previous formulation, the following equation is obtained:

$$\int_\Omega \nabla^s v^T (C\nabla^s u - \beta \nabla T) d\Omega = \int_\Omega v^T f d\Omega + \int_{\Gamma_\sigma} v^T \bar{t} d\Gamma \tag{12}$$

Rearranging the terms algebraically:

$$\int_\Omega \nabla^s v^T \cdot C\nabla^s u d\Omega - \beta \nabla T \cdot d\Omega = \int_\Omega v^T f d\Omega + \int_{\Gamma_\sigma} v^T \bar{t} d\Gamma \tag{13}$$

$$\int_\Omega \nabla^s v^T \cdot C\nabla^s u d\Omega - \int_\Omega \beta \nabla T \cdot d\Omega = \int_\Omega v^T f d\Omega + \int_{\Gamma_\sigma} v^T \bar{t} d\Gamma \tag{14}$$

The following relation is defined: $\Delta T(x) = N^e(x) \cdot \theta^e \quad x \in \Omega$ from theory. Then, expressing $u(x)$ and $v(x)$ as combinations of elemental basis functions: $u(x) = d \cdot N(x)$, $v(x) = c \cdot N(x)$, and knowing that the derivative shape functions matrix can be expressed as: $B = \nabla^s N$, then the previous formulation is equivalent to the following:

$$\int_\Omega \nabla^s (N^T \cdot c^T) \cdot C \cdot B \cdot d \cdot d\Omega = \int_\Omega (N^T \cdot c^T \cdot f + \beta \cdot N \cdot \theta) \cdot d\Omega + \int_{\Gamma_\sigma} N^T \cdot c^T \cdot \bar{t} \cdot d\Gamma \tag{15}$$

$$\int_\Omega B^T \cdot c^T \cdot C \cdot Bd \cdot d\Omega = \int_\Omega (N^T \cdot c^T \cdot f + \beta \cdot N \cdot \theta) \cdot d\Omega + \int_{\Gamma_\sigma} N^T \cdot c^T \cdot \bar{t} \cdot d\Gamma \tag{16}$$

Cancelling the factor c^T, the following formulation is obtained:

$$d \cdot \int_\Omega B^T \cdot C \cdot B \cdot d\Omega = \int_\Omega (N^T \cdot c^T \cdot f + \beta \cdot N \cdot \theta) \cdot d\Omega + \int_{\Gamma_\sigma} N^T \cdot c^T \cdot \bar{t} \cdot d\Gamma \tag{17}$$

Taking terms to the left-hand side:

$$d \cdot \int_\Omega B^T \cdot C \cdot B \cdot d\Omega - \left(\int_\Omega (N^T \cdot c^T \cdot f + \beta \cdot N \cdot \theta) \cdot d\Omega + \int_{\Gamma_\sigma} N^T \cdot c^T \cdot \bar{t} \cdot d\Gamma \right) = 0 \tag{18}$$

It is known that the system equations have the matricial structure $c^T \cdot (K \cdot d - F) = 0 \longrightarrow K \cdot d - F = 0$.
Comparing this structure with the previous formulation obtained, it is deducted that the expression of F^e_{th} is the correspondent to the independent term. Therefore:

$$F^e_{th} = \int_{\Omega^e} N^T \cdot (f + \beta^T \cdot \theta) \cdot d\Omega + \int_{\Gamma^e_\sigma} N^T \cdot \bar{t} \cdot d\Gamma \tag{19}$$

A Appendix

A.1 Code for Assembly of Stiffness Matrix (K) Function

```

  function K = ComputeK(COOR,CN,TypeElement, celasglo) ;
%%%%
% This subroutine   returns the global stiffness matrix K (ndim*nnode x ndim*nnode)
% Inputs:   COOR: Coordinate matrix (nnode x ndim), % CN: Connectivity matrix (nelem x ...
    nnodeE), % TypeElement: Type of finite element (quadrilateral,...),  celasglo (nstrain ...
    x nstrain x nelem)  % Array of elasticity matrices
% Dimensions of the problem
if nargin == 0
    load('tmp1.mat')
end
nnode = size(COOR,1);
ndim = size(COOR,2);
nelem = size(CN,1);
nnodeE = size(CN,2);

% nstrain = size(celasglo,1) ;
% Shape function routines (for calculating shape functions and derivatives)
TypeIntegrand = 'K';
[weig,posgp,shapef,dershapef] = ComputeElementShapeFun(TypeElement,nnodeE,TypeIntegrand) ;
% Assembly of matrix K
% ----------------
K = zeros(nnode*ndim) ;
for e = 1:nelem
    celas = celasglo(:,:,e) ;  % Stiffness matrix of element "e"
    CNloc = CN(e,:) ;          % Coordinates of the nodes of element "e"
    Xe = COOR(CNloc,:)' ;      % Computation of elemental stiffness matrix
    Ke = ComputeKeMatrix(celas,weig,dershapef,Xe);
    submatrix=zeros(nnodeE*3,1);
    for n=1:nnodeE
    submatrix((n-1)*3+1,1)=(CN(e,n)-1)*3+1;
    submatrix((n-1)*3+2,1)=(CN(e,n)-1)*3+2;
    submatrix((n-1)*3+3,1)=(CN(e,n)-1)*3+3;
    end

    K(submatrix,submatrix,1)=K(submatrix,submatrix,1)+Ke;

    if mod(e,10)==0  % To display on the screen the number of element being assembled
        disp(['e=',num2str(e)])
    end
end
end
```

A.2 Code for Hexaedral8InPoints Function

```
   function [weig,posgp,shapef,dershapef] = Hexaedra8NInPoints()

ndimensions=3;
nnodeE=2^ndimensions;
weig=ones(1,nnodeE);
ngauss=length(weig);

signs=[-1 1 1 -1 -1 1 1 -1;  % signs of xi
       -1 -1 1 1 -1 -1 1 1;  % signs of eta
       -1 -1 -1 -1 1 1 1 1];  % signs of zeta

posgp=(1/sqrt(3))*signs;

```

```
shapef=zeros(ngauss,nnodeE);
dershapef=zeros(ndimensions,nnodeE,ngauss);

for g=1:ngauss
    xi=posgp(1,g);
    eta=posgp(2,g);
    zeta=posgp(3,g);

    shapef(g,:)=(1/nnodeE)*[(1-xi)*(1-eta)*(1-zeta) (1+xi)*(1-eta)*(1-zeta) ...
        (1+xi)*(1+eta)*(1-zeta) (1-xi)*(1+eta)*(1-zeta) (1-xi)*(1-eta)*(1+zeta) ...
        (1+xi)*(1-eta)*(1+zeta) (1+xi)*(1+eta)*(1+zeta) (1-xi)*(1+eta)*(1+zeta)];

    dershapef(:,:,g)=(1/nnodeE)*signs.*[(1-eta)*(1-zeta) (1-eta)*(1-zeta) (1+eta)*(1-zeta) ...
        (1+eta)*(1-zeta) (1-eta)*(1+zeta) (1-eta)*(1+zeta) (1+eta)*(1+zeta) (1+eta)*(1+zeta);
        (1-xi)*(1-zeta) (1+xi)*(1-zeta) (1+xi)*(1-zeta) (1-xi)*(1-zeta) (1-xi)*(1+zeta) ...
            (1+xi)*(1+zeta) (1+xi)*(1+zeta) (1-xi)*(1+zeta);
        (1-xi)*(1-eta) (1+xi)*(1-eta) (1+xi)*(1+eta) (1-xi)*(1+eta) (1-xi)*(1-eta) ...
            (1+xi)*(1-eta) (1+xi)*(1+eta) (1-xi)*(1+eta)];
end

end
```

A.3 Code for Solving the System of Equations

```
    function [d,strainGLO,stressGLO,React,posgp] = ...
        SolveELAS(K,Fb,Ftrac,dR,DOFr,COOR,CN,TypeElement,celasglo,typePROBLEM,celasgloINV,DATA) ...
        ;
% This function returns  the (nnode*ndim x 1) vector of nodal displacements (d),
% as well as the arrays of stresses and strains
%%% points  (qheatGLO)
% Input data
% K = Global stiffness matrix   (nnode*ndim x nnode*ndim)
% Fb = External force vector due to  body forces  (nnode*ndim x 1)
% Ftrac = External force vector due to  boundary tractions    (nnode*ndim x 1)
% DOFr = Set of restricted DOFs
% dR = Vector of prescribed displacements
% ----------------------
if nargin == 0
    load('tmp.mat')
end
nnode = size(COOR,1);
ndim = size(COOR,2);
nelem = size(CN,1);
nnodeE = size(CN,2) ;

% Solution of the system of FE equation
% Right-hand side
F = Fb + Ftrac ;
% Set of nodes at which temperature is unknown
DOFl = 1:nnode*ndim ;
DOFl(DOFr) = [] ;

% dL =  K^{-1}*(Fl .Klr*dR)
% ***

d=zeros(nnode*ndim,1);
React=zeros(size(d));

d(DOFl,1) = K(DOFl,DOFl)\(F(DOFl,1)-K(DOFl,DOFr)*dR); % Nodal displacements (initialization)
React(DOFr) = K(DOFr,DOFr)*d(DOFr,1)+K(DOFr,DOFl)*d(DOFl,1)-F(DOFr,1);  %  Reaction forces ...
     (initialization)


%%%% COmputation of strain and stress vector at each gauss point
disp('Computation of stress and strains at each Gauss point')
```

```
[strainGLO,stressGLO,posgp]= ...
    StressStrains(COOR,CN,TypeElement,celasglo,d,typePROBLEM,celasgloINV,DATA) ;
end
```

A.4 Code for Convergence Analysis

```
  clc; clear all; close all;

%% Convergence study of finite elements program

% Mesh 1

M1=[
0 0
1 -0.000711
2 -0.002047];

% Mesh 2

M2=[0 0
0.2 -0.00018800001
0.40000001 -0.00060500001
0.60000002 -0.001234
0.80000001 -0.002023
1 -0.002929
1.2 -0.0039149998
1.4 -0.004952
1.6 -0.0060160002
1.8 -0.0070910002
2 -0.0081660002];

% Mesh 3

M3=[0 0
0.133333 -0.00011
0.26666701 -0.000329
0.40000001 -0.00066700001
0.533333 -0.001106
0.66666698 -0.0016280001
0.80000001 -0.0022199999
0.93333298 -0.002868
1.0666699 -0.0035600001
1.2 -0.004284
1.33333 -0.0050329999
1.46667 -0.0057970001
1.6 -0.0065720002
1.73333 -0.0073509999
1.86667 -0.0081310002
2 -0.0089100003];

% Mesh 4

M4=[0 0
0.079999998 -5.8000001e-05
0.16 -0.000151
0.23999999 -0.00029200001
0.31999999 -0.00047699999
0.40000001 -0.00070400001
0.47999999 -0.00096899999
0.56 -0.001267
0.63999999 -0.001596
0.72000003 -0.001952
0.80000001 -0.0023320001
0.88 -0.0027330001
0.95999998 -0.003151
1.04 -0.0035850001
1.12 -0.004032
1.2 -0.0044900002
1.28 -0.0049569998
1.36 -0.0054299999
1.4400001 -0.0059090001
1.52 -0.006391
1.6 -0.006877
1.6799999 -0.007363
1.76 -0.0078509999
1.84 -0.0083379997
1.92 -0.0088250004
2 -0.0093109999];

% Theoretical displacements

x=0:0.001:2;
Δ=-125000.*x.^2/(24.*70*10^9.*0.000283).*(x.^2+6.*4-8.*x);

hold on;
plot(M1(:,1),M1(:,2),'-o')
plot(M2(:,1),M2(:,2),'-o')
plot(M3(:,1),M3(:,2),'-o')
plot(M4(:,1),M4(:,2),'-o')
plot(x,Δ,'--')
grid on
title('Convergence of the finite elements program',Interpreter='latex');
xlabel('Distance along the top edge (m)',Interpreter='latex');
ylabel('Vertical displacement (m)')
legend({'First mesh (Nx=2)','Second mesh (Nx=10)','Third mesh (Nx=15)','Fourth mesh ...
    (Nx=20)','Theoretical value'},'Location','southwest')
```

A.5 Code for System Reactions Computation

```
    clc; clear all; close all;
load('INFO_FE.mat');

    Reactions = React(DOFr);
    Reactions_mat=reshape(Reactions,[length(DOFr)/3,3]);
    Total_Reactions=sum(Reactions_mat,1);
    Fx=Total_Reactions(1);
    Fy=Total_Reactions(2);
    Fz=Total_Reactions(3);

   cg = [0 0.25/2 0.25/2]; % reference point for the moments calculation
   dist=zeros(length(DOFr)/3,3);


    for i = 1:size(Reactions_mat,1)

        dist(i,:) = [(COOR(i,1)-cg(1,1)) (COOR(i,2)-cg(1,2)) (COOR(i,3)-cg(1,3))];

    end

        M = cross(dist, Reactions_mat);
        M_Total=sum(M,1);
        Mx = M_Total(1);
        My = M_Total(2);
        Mz = M_Total(3);

Reactions=[Fx,Fy,Fz,Mx,My,Mz];

    fprintf("\nReaction forces and moments of the system:\n\n  Rx = %f MN\n  Ry = %f MN\n  ...
        Rz = %f MN\n  Mx = %f MNm\n  My = %f MNm\n  Mz = %f MNm\n",Fx, Fy, Fz, Mx, My, Mz);
```