Aerospace Computational Engineering — GrETA

Assignment 3

Elastostatic

David Beurnio
Biel Riera

Polithecnic University of Catalonia

January 16, 2023

Contents

1 Introduction

The aim of this assignment is to compute the forces and displacements that occur in an specific problem. In this case, consider the cantilever box beam depicted in Figure 1. The length of the beam is $L_X = 2$ m, its width and height $h_y = h_z = 0.25m$, and its thickness $e = 0.05$ m. 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 ($y = y_{max}$).

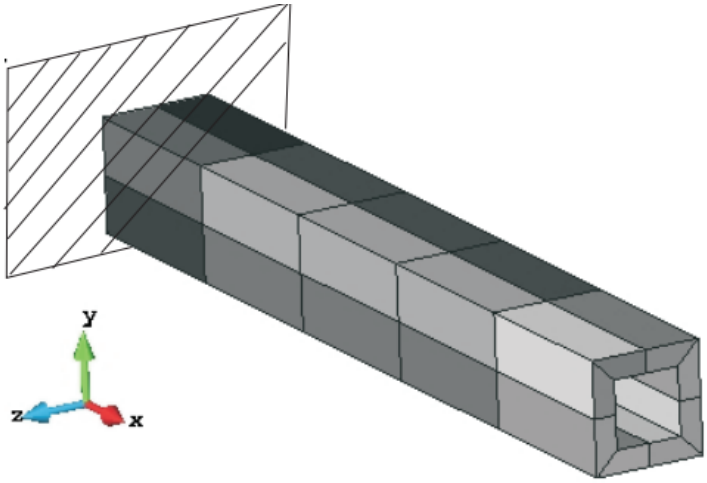

Figure 1: Geometry of the cantilever box beam

The convergence upon mesh refinement will be assessed by launching 4 different analyses with increasing number of finite elements (hexaedral). In particular, semi-structured meshes will be used 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.

2 Part 1

2.1 Creation of the semi-structured mesh

The program used to create the 3D geometry and the mesh will be GiD. Following the guidelines given, the structure is defined in GiD. After defining the structure making sure the axes (x, y and z) are placed correctly to ensure the data generated is correct, the mesh is generated. The process will be repeated 4 times in order to mesh the 4 different semi-structured meshes.

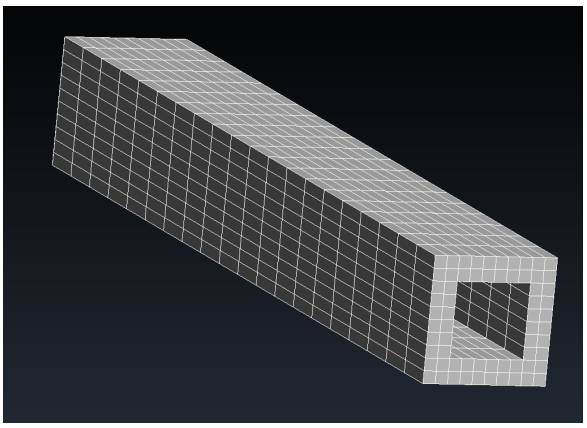

Figure 2: Meshed geometry with 20 divisions and a typical size in the y-z plane of 0.05m.

2.2 Programming the finite elements method

Most part of the code was given in the assignment guidelines, with only a few functions and subprogrammes to be completed in order to generate results. The missing parts of the program are as follows:

- Assembly of the stiffness matrix.
- Assembly of the shape function routines of trilinear hexaedral elements.
- Assembly of the final system of equations.

2.2.1 Assembly of the stiffness matrix

In this part of the program, what is left to do is assembly the K matrix of the problem. The only thing left to program is a loop to compute the K matrix of each element (in this case of each node) and assembly it in a global stiffness matrix. See code in Appendix A1.

2.2.2 Assembly of the shape function routines for trilinear hexaedral elements

The trilinear hexaedral shape function is a generalization of the bilinear quadrilateral. As seen in the lecture notes [1, pg. 91-94], the N^e and B^e matrices must be created (specifically, $N^e(1:4)$, $N^e(5:8)$, $B^e(1:4)$ and $B^e(5:8)$). See code in Appendix A2.

2.2.3 Assembly of the final system of equations

To solve the system of equations, it is needed to solve the following equation:

$$d_l = K_{ll}^{-1}(F_l - K_{lr}u) \tag{1}$$

Code can be found in Appendix A3.

2.3 Results for an uniformly distributed load

2.3.1 Mesh 1

The first mesh consists in a mesh with 2 divisions and a typical size in the y-z plane of 0.1 m. For this specific mesh, the following results are obtained:

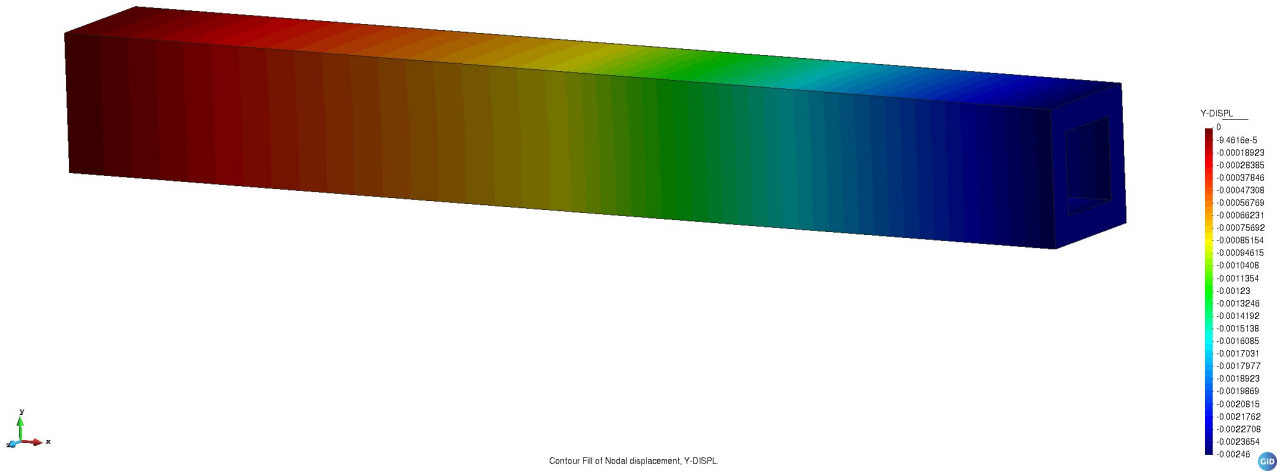

Figure 3: Vertical displacements distribution for the first mesh.

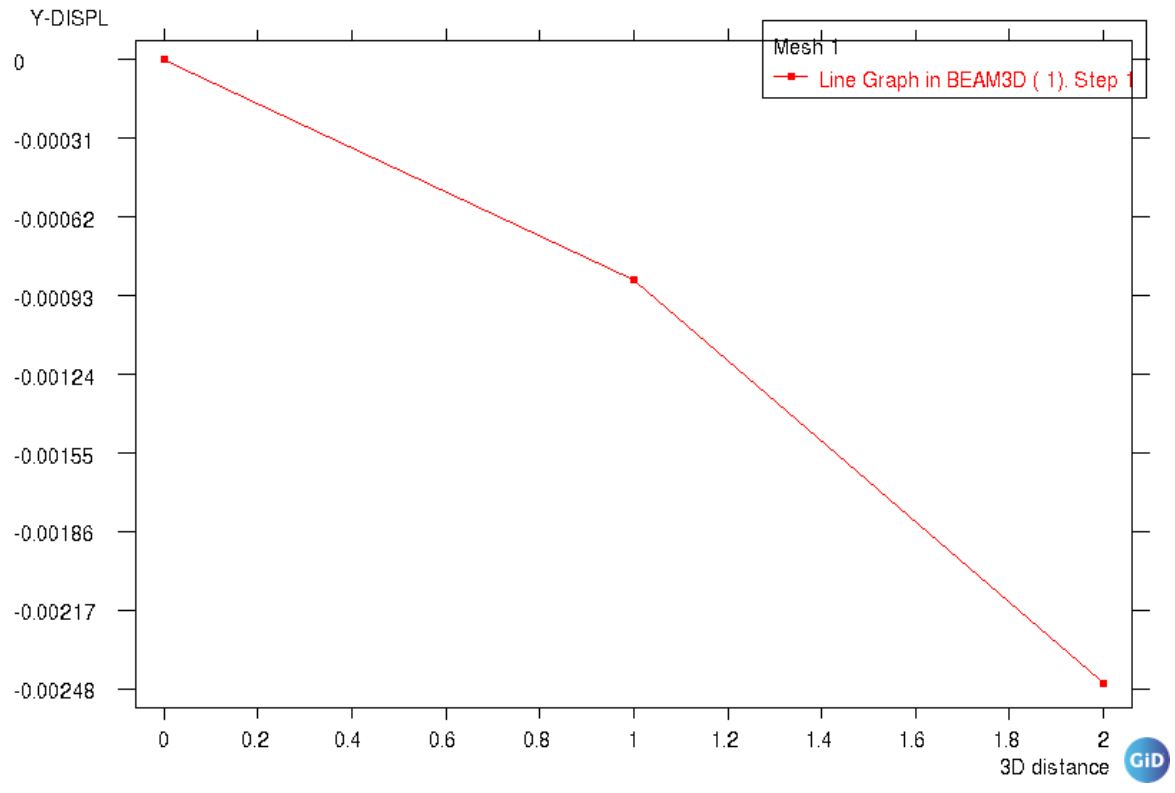

Figure 4: Vertical displacements (in the y-direction) along one of the edges on the top surface (parallel to the x-axis) for the first mesh.

2.3.2 Mesh 2

The second mesh consists in a mesh with 10 divisions and a typical size in the y-z plane of 0.05 m. For this specific mesh, the following results are obtained:

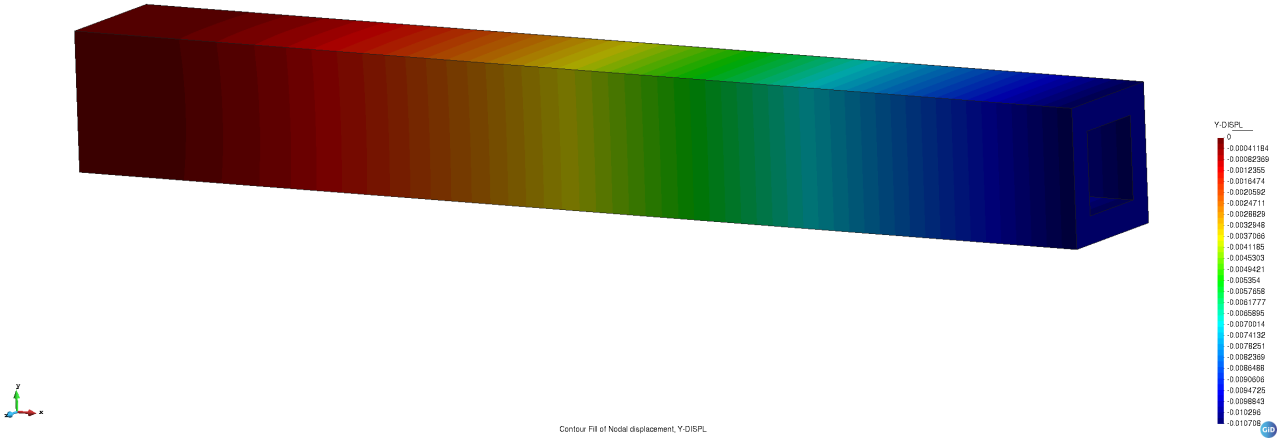

Figure 5: Vertical displacements distribution for the second mesh.

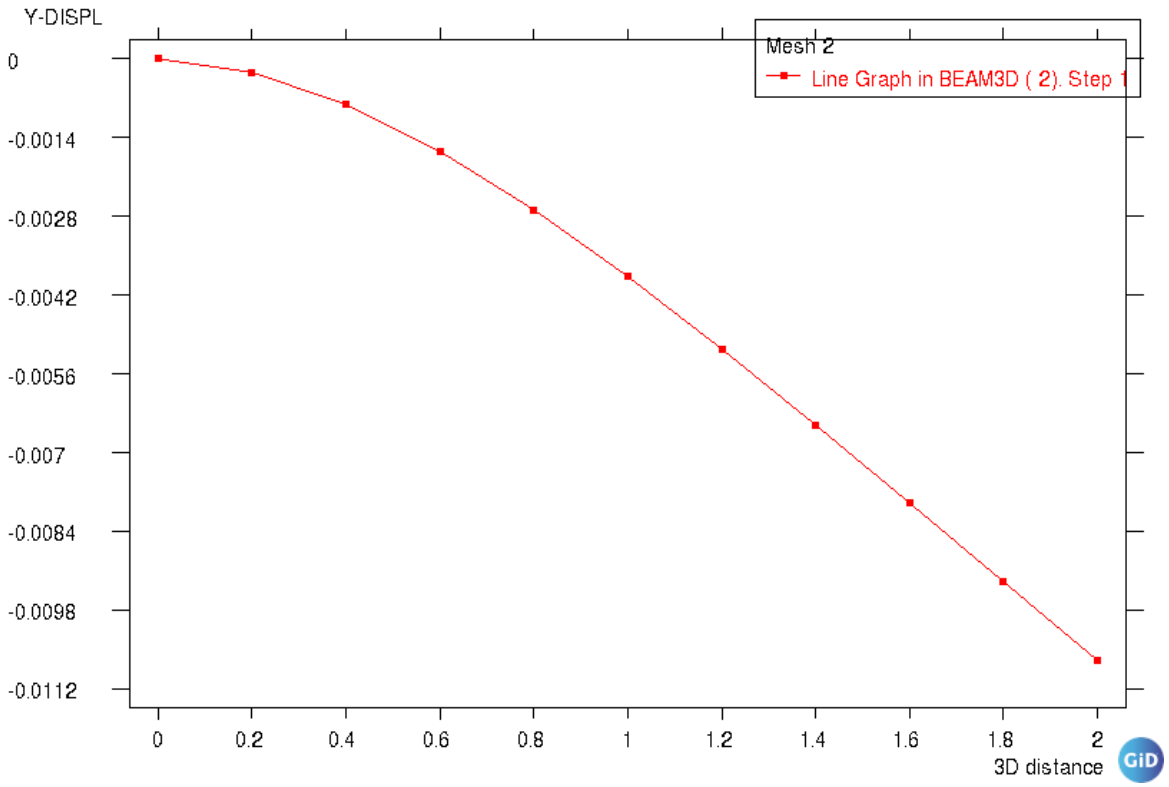

Figure 6: Vertical displacements (in the y-direction) along one of the edges on the top surface (parallel to the x-axis) for the second mesh.

2.3.3 Mesh 3

The third mesh consists in a mesh with 15 divisions and a typical size in the y-z plane of 0.05 m. For this specific mesh, the following results are obtained:

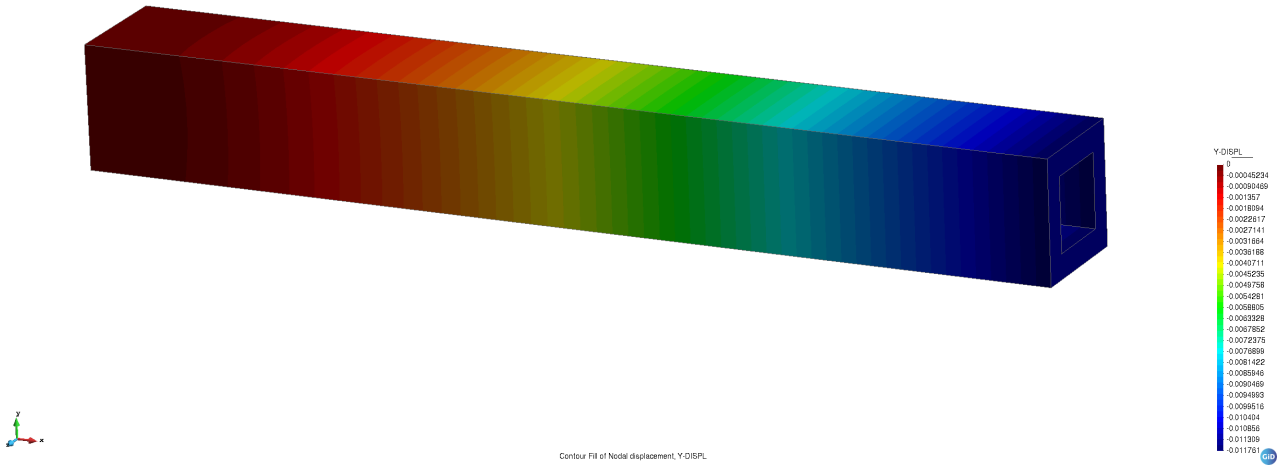

Figure 7: Vertical displacements distribution for the third mesh.

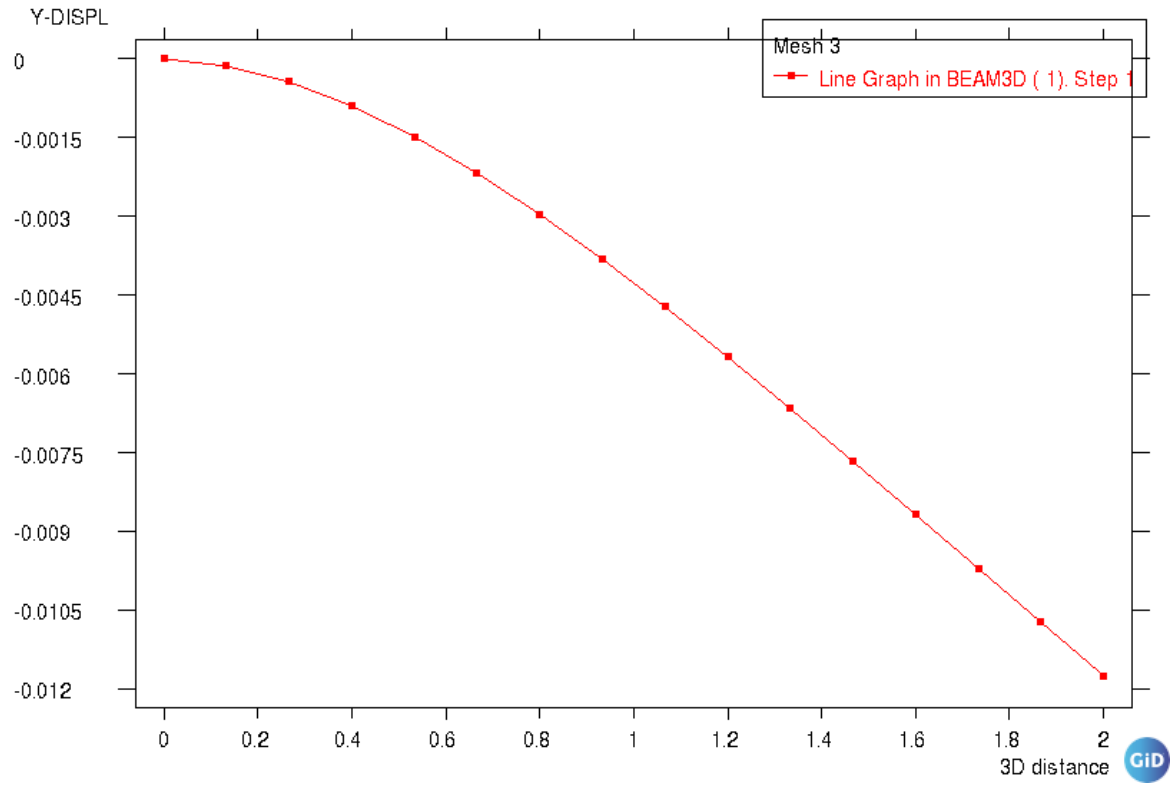

Figure 8: Vertical displacements (in the y-direction) along one of the edges on the top surface (parallel to the x-axis) for the third mesh.

2.3.4 Mesh 4

The fourth mesh consists in a mesh with 20 divisions and a typical size in the y-z plane of 0.025 m. For this specific mesh, the following results are obtained:

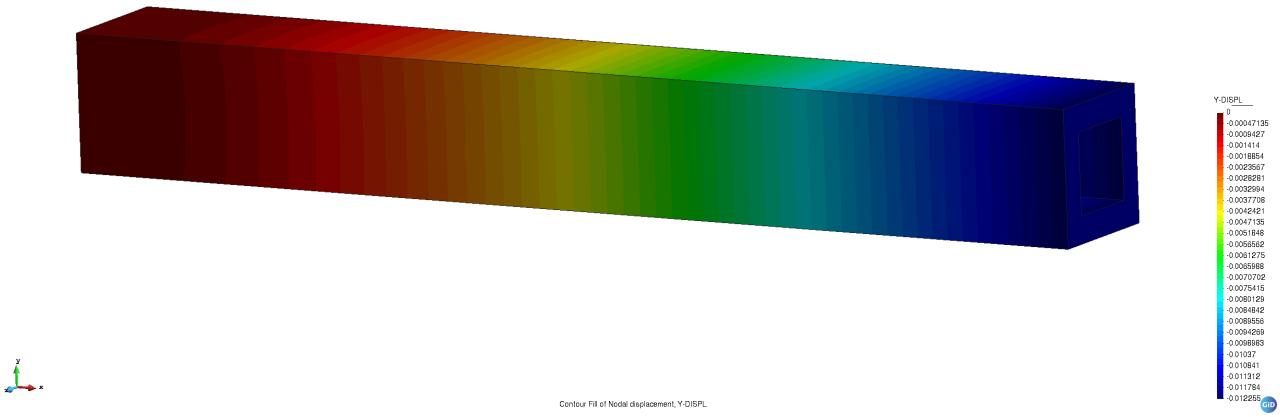

Figure 9: Vertical displacements distribution for the fourth mesh.

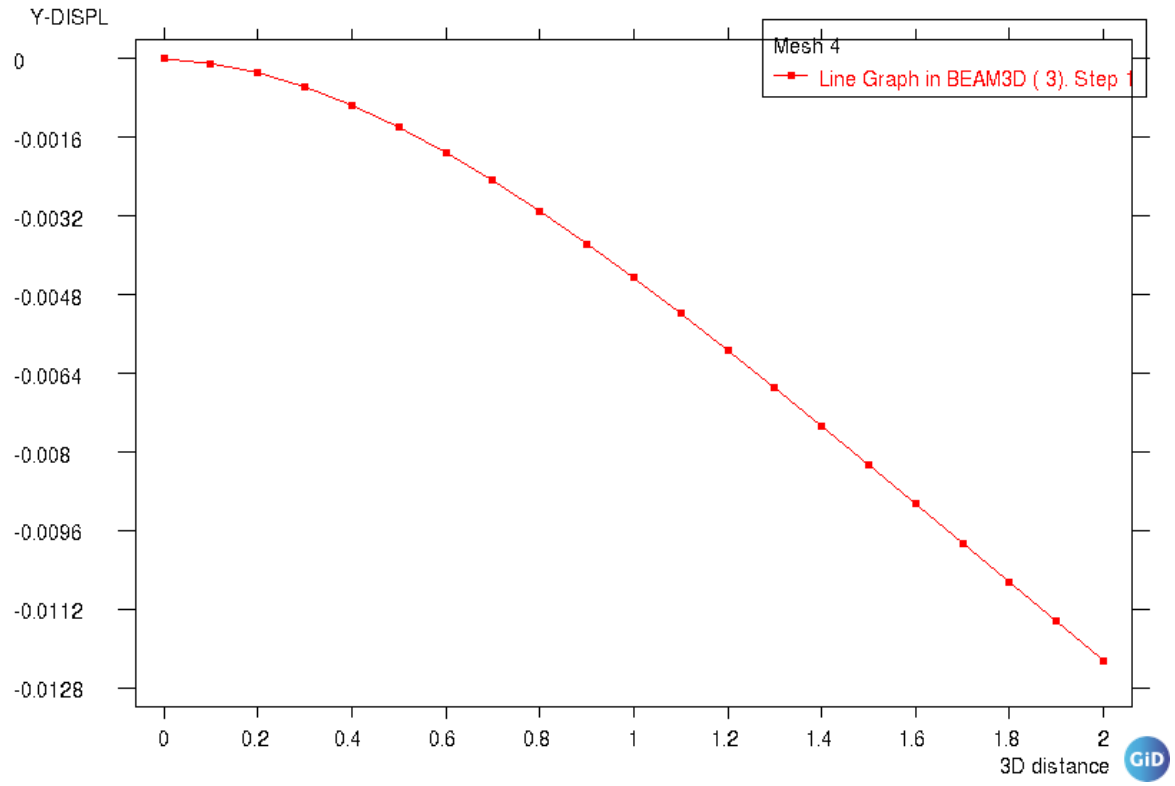

Figure 10: Vertical displacements (in the y-direction) along one of the edges on the top surface (parallel to the x-axis) for the fourth mesh.

2.3.5 Analytical prediction of the vertical displacement at the tip of the beam

To calculate the maximum displacement at the tip of the beam the analytical predictions provided by the theory of Strength of Materials will be used.

For a cantilever beam with uniformly distributed load, the deflection at any section in terms of x is:

$$y = \frac{\omega x^2}{24EI}(x^2 + 6l^2 - 4lx) \tag{2}$$

To calculate the deflection at the end of the beam, introduce the data at the end of the beam:

- x = L = 2 m
- E = 70 MPa
- I = $2.83 \cdot 10^{-4}$ m^4
- $\omega = h \cdot t = -125kN/m$

Introducing this values in equation 2, the value of the maximum deflection at the tip of the beam is obtained:

$$\delta(x = L) = -0.0126m$$

2.3.6 Convergence of the program

In order to visualize the program convergence, all the plots from the four meshes are represented in the same figure, as well as the plot for the analytical prediction. This way, it can be seen that by increasing the number of divisions in a mesh, the results tend to the anayltical expression.

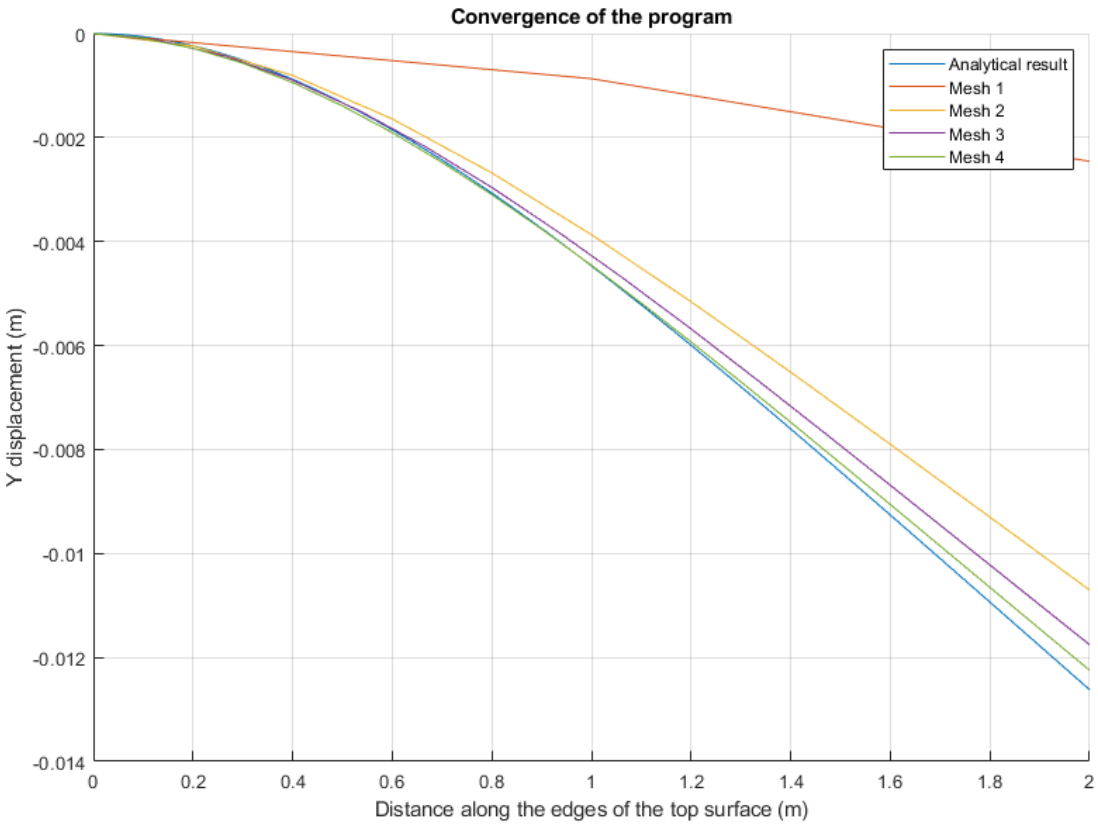

Figure 11: Convergence of the program with the different meshes.

2.4 Results applying a Torque T=100kN·m at the free end of the beam

In this section, a torque T=100 kN·m will be applied at the free end of the beam.

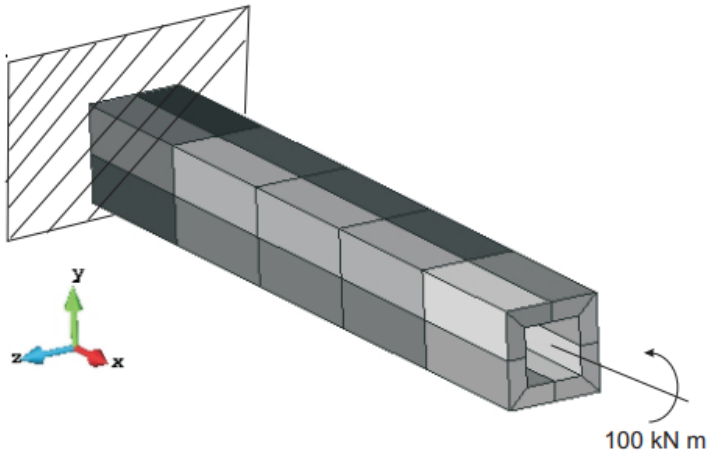

Figure 12: Convergence of the program with the different meshes.

The program allows the user to apply forces to the beam but not torques. In order to simulate this torque, it will be transformed into four different forces that will be applied at the center node of each edge of the free end of the beam. Since 4 different forces will be applied at the center node of each edge, the lever arm distance will be h/2 = 0.125m. To know the value of the force that will be applied:

$$F_{node} = \frac{Torque}{4 \cdot d} = \frac{100kNm}{4 \cdot 0.125m} = 200kN \tag{3}$$

With this information, the torque can be simulated in Gid, allowing us to see the displacements and the reaction forces at the fixed end.

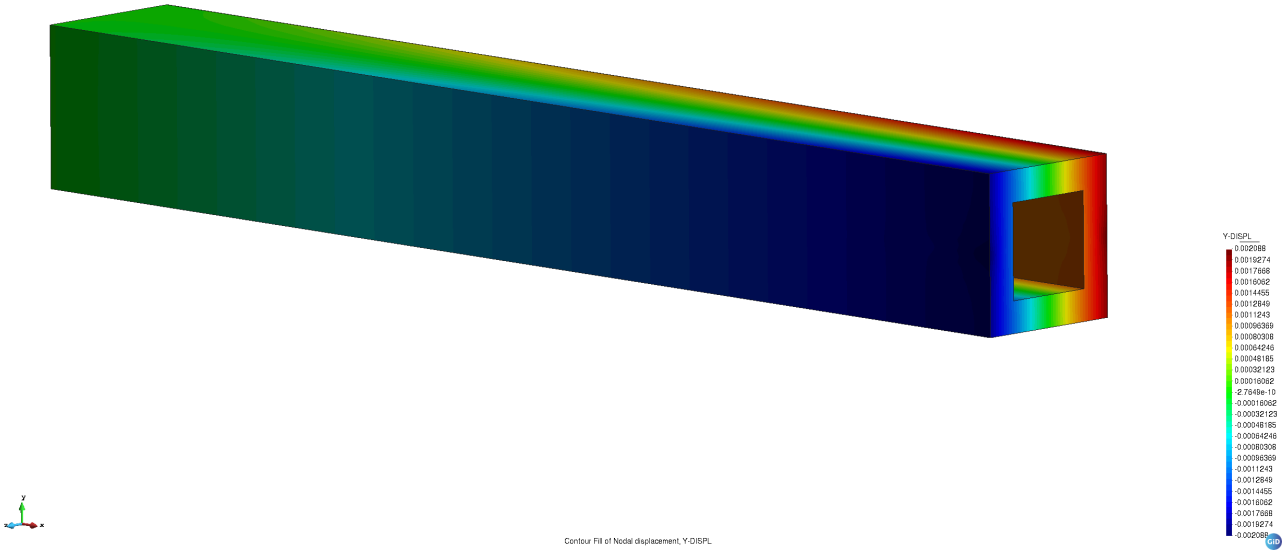

Figure 13: Y displacement of the beam.

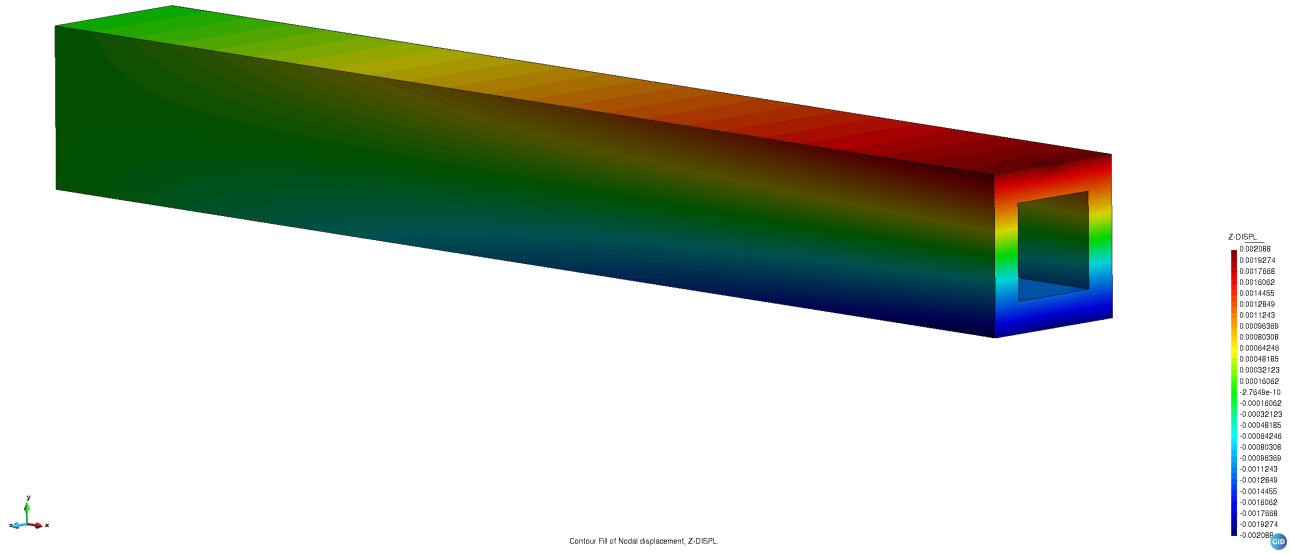

Figure 14: Z displacement of the beam.

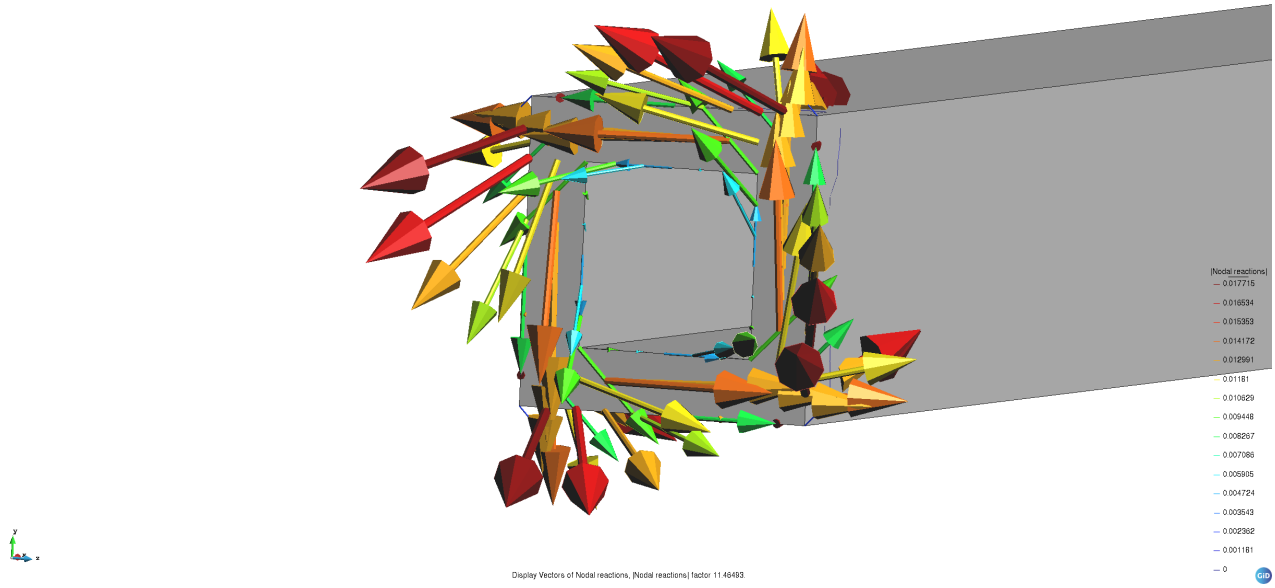

Figure 15: Reaction forces at the fixed end of the beam.

2.5 Reaction forces

A code to evaluate the reaction forces at the fixed end of the beam (for the first case) has been developed (see code in Appendix A4. The code shows the following values:

$$R_x = 0 \qquad R_y = 0.25MN \qquad R_z = 0$$

$$M_x = 0 \qquad M_y = 0 \qquad M_z = 0.25MNm$$

The prescribed forces are $-500kN/m^2$. Knowing this:

$$R_y = F \cdot h \cdot L = -500\frac{kN}{m^2} \cdot 0.25m \cdot 2m = -250kN = -0.25MN$$

And for the torque:

$$M_z = R_y \cdot d = R_y \cdot \frac{L}{2} = 0.25 \cdot \frac{2}{2} = 0.25MN \cdot m$$

So it can be concluded that the program calculates the reactions and torques correctly.

3 Part 2 (Advanced)

Let $\Omega = \cup_{e=1}^{n_{el}} \Omega^e$ be a given finite element discretization. Suppose we solve the heat conduction problem under given boundary conditions, and obtains the vector of temperatures, θ, at the nodes of the discretization (with respect to the reference temperature T_0). With this vector at our disposal, we can interpolate the relative temperature at any point within an element Ω^e by the corresponding shape functions:

$$\Delta T(x) = N^e(x)\theta^e, \qquad x \in \Omega^e \tag{4}$$

Where θ^e denotes the vector of nodal temperatures of element Ω^e. With this in consideration, derive the discrete system of equilibrium equations arising from the variational principle when the stresses depend on the temperature through the constitutive equation:

$$\sigma = C(\nabla^s u - \alpha \Delta T) = C\nabla^s u - \beta \Delta T \tag{5}$$

The weak form of this BVP takes the form:

$$\int_\Omega \nabla^s v^T \sigma \, d\Omega = \int_\Omega v^T f \, d\Omega + \int_{\Gamma_\sigma} v^T \bar{\mathrm{t}} \, d\Gamma \tag{6}$$

Which in this case will be:

$$\int_\Omega \nabla^s v^T (C\nabla^s u - \beta \Delta T) \, d\Omega = \int_\Omega v^T f \, d\Omega + \int_{\Gamma_\sigma} v^T \bar{\mathrm{t}} \, d\Gamma \qquad \forall v \in V \tag{7}$$

Now rearranging the terms:

$$\int_\Omega \nabla^s v^T C \nabla^s u \, d\Omega = \int_\Omega v^T f \, d\Omega + \int_\Omega \nabla^s v^T \beta \Delta T \, d\Omega + \int_{\Gamma_\sigma} v^T \bar{\mathrm{t}} \, d\Gamma \qquad \forall v \in V \tag{8}$$

Extracting from the theory that $\Delta T(x) = N^e(x) \cdot \theta^e$, remembering the relations $u = N \cdot d$ and $v = N \cdot c$; and defining B as $B = \nabla^s N$ the equation ends as:

$$c^T \int_\Omega \nabla^s N^T C \nabla^s (Nd) \, d\Omega = c^T \int_\Omega N^T f \, d\Omega + c^T \int_\Omega \beta N \theta \, d\Omega + c^T \int_{\Gamma_\sigma} N^T \bar{\mathrm{t}} \, d\Gamma \tag{9}$$

Simplifying the final expression is obtained:

$$d \cdot \int_\Omega B^T C B \, d\Omega = \int_\Omega N^T f \, d\Omega + \int_\Omega \beta N \theta \, d\Omega + \int_{\Gamma_\sigma} N^T \bar{\mathrm{t}} \, d\Gamma \tag{10}$$

Finally, knowing that $K \cdot d = F$, we can confirm that:

$$F_{th} = \int_\Omega N^T f \, d\Omega + \int_\Omega \beta N \theta \, d\Omega + \int_{\Gamma_\sigma} N^T \bar{\mathrm{t}} \, d\Gamma \tag{11}$$

4 Conclusions

To conclude this report, it can be clearly noted that by increasing the number of divisions in a mesh (and consequently, decreasing the typical size), the approximation of the experimental results obtained in the simulations tend to be more accurate to the analytic results obtained via equations. This can be clearly seen in figure 12, where the results in mesh 4 hardly differ from the ones obtained with the analytical procedure. The downside of increasing the number of divisions is the calculation time: the program experiments a substantial increase in calculation time when the mesh is more complex.
The program developed is useful to calculate situations with point forces or with load per unit surface. However, when it comes to torques, if the torque applied is complex (in terms of orientation) it can be difficult to transform it into point forces so that the program can evaluate its behaviour.

5 Bibliografia

[1] Hernández, J., *Chapter 3: Classical Linear Elastostatics*, Lecture notes (Aeroespace Computational Engineering), 2022.

Appendix A

Codes

1 Stiffness matrix

```
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 = sparse([],[],[],nnode*ndim,nnode*ndim,nnodeE*ndim*nelem) ;
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) ;

    for an=1:nnodeE
        a=Nod2DOF(an,ndim);
        An=CN(e,an);
        A=Nod2DOF(An,ndim);
        for bn=1:nnodeE
            b=Nod2DOF(bn,ndim);
            Bn=CN(e,bn);
            B=Nod2DOF(Bn,ndim);
            K(A,B)=K(A,B)+Ke(a,b);

        end
    end

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

end
```

2 Shape function routine

```
function [weig,posgp,shapef,dershapef] = Hexahedra8NInPoints

%code for the shape function routine for hexaedra

weig = [1 1 1 1 1 1 1 1];
posgp=1/sqrt(3)*[-1 -1 -1;
     1 -1 -1;
     1 1 -1;
     -1 1 -1;
     -1 -1 1;
     1 -1 1;
     1 1 1
     -1 1 1]';

ndim=3;
nnodeE=8;
ngaus=length(weig);
shapef=zeros(ngaus,nnodeE);
dershapef=zeros(ndim,nnodeE,ngaus);
for j=1:length(weig)
     xi = posgp(:,j).';
     [Ne, Bex]=Hexahedra8N(xi);
     shapef(j,:)=Ne;
     dershapef(:,:,j)=Bex;
end
end

function [Ne, Bex]=Hexahedra8N(x)
xi=x(1);
eta=x(2);
zeta=x(3);

Ne=1/8*[(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)];

Bex=1/8*[-(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
```

3 System of equations solver

```
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) = [] ;
d = zeros(nnode*ndim,1) ; % Nodal displacements (initialization)
```

```
React = zeros(size(d)) ;  %  REaction forces  (initialization)
d(DOFl,1)=K(DOFl,DOFl)\(F(DOFl,1)-K(DOFl,DOFr)*dR);
React(DOFr)=K(DOFr,DOFr)*d(DOFr,1)+K(DOFr,DOFl)*d(DOFl,1)-F(DOFr,1);

%%%% 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
```

4 Compute reactions forces

```
load("INFO_FE.mat") %info about COOR, DOFr and React

cg=[0 0.125 -0.125];
Re=React(DOFr); %save reaction values of each DOFr
R_total=reshape(Re,[length(DOFr)/3,3]);
fixed_nod_dim=size(R_total,1);
fixed_COOR=zeros(fixed_nod_dim,3);
fixed_nod=DOFr(end-fixed_nod_dim+1:end,:);
fixed_COOR(:,:)=COOR(fixed_nod/3,:);

Suma_total=sum(R_total,1);

Rx=Suma_total(1);
Ry=Suma_total(2);
Rz=Suma_total(3);
d=zeros(fixed_nod_dim,3);
for i=1:size(R_total,1)
    d(i,:)=[(fixed_COOR(i,1)-cg(1,1)) (fixed_COOR(i,2)-cg(1,2)) (fixed_COOR(i,3)-cg(1,3))];
end
M=cross(d,R_total); %perpendicular vector d x R_total
SumaM=sum(M,1);
Mx=SumaM(1);
My=SumaM(2);
Mz=SumaM(3);
```