Universitat Politècnia de Catalunya

ESEIAAT

DEGREE IN AEROSPACE TECHNOLOGY ENGINEERING

Assignment 2

COMPUTATIONAL HEAT CONDUCTION ANALYSIS

Author
Bernardo Márquez López

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

October 2022

List of Figures

1 Heat conduction problem geometry. 2
2 Definition of the surface in the statement using the finite elements software *GID CIMNE*. 3
3 Definition of the surface in the statement using the finite elements software *GID CIMNE* 3
4 Mesh generated for on the surface for 1 element discretization (1x1 quadrilateral elements mesh). . . 5
5 Simulation of the temperatures gradient on the surface for 1 discretization element (1x1 quadrilateral elements mesh). 5
6 Temperature distribution over the edge CD for 1 element discretization (1x1 quadrilateral elements mesh). 6
7 Mesh generated for on the surface for 25 elements discretization (5x5 quadrilateral elements mesh). . 6
8 Simulation of the temperatures gradient on the surface for 25 discretization elements (5x5 quadrilateral elements mesh). 7
9 Temperature distribution over the edge CD for 25 elements discretization (5x5 quadrilateral elements mesh). 7
10 Mesh generated for on the surface for 100 elements discretization (10x10 quadrilateral elements mesh). 8
11 Simulation of the temperatures gradient on the surface for 100 discretization elements (10x10 quadrilateral elements mesh). 8
12 Simulation of the temperatures gradient on the surface for 1 discretization element. 9
13 Mesh generated for on the surface for 900 elements discretization (30x30 quadrilateral elements mesh). 9
14 Simulation of the temperatures gradient on the surface for 900 discretization elements (30x30 quadrilateral elements mesh). 10
15 Temperature distribution over the edge CD for 900 elements discretization (30x30 quadrilateral elements mesh). 10
16 Caption 11
17 Simulation of the heat flux on the surface for 1 discretization element (1x1 quadrilateral elements mesh). 12
18 Simulation of the heat flux on the surface for 25 discretization elements (5x5 quadrilateral elements mesh). 12
19 Simulation of the heat flux on the surface for 100 discretization elements (10x10 quadrilateral elements mesh). 13
20 Simulation of the heat flux on the surface for 900 discretization elements (30x30 quadrilateral elements mesh). 13
21 Boundary conditions definition for the problem in part 2. 14

General statement of the assignment

The goal of this assignment is to develop a Matlab program able to solve any two dimensional heat conduction problem using bilinear quadrilateral elements.

To test the performance of the program, consider the heat conduction problem depicted in Figure 1. The coordinates are given in metres. The conductivity matrix is isotropic, with $\kappa = \kappa \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ *, and* $\kappa = 10W/^oC$*. The temperature* $u = 0$ *is prescribed along edges AB and AD. The heat fluxes* $q \cdot n = 0$ *and* $q \cdot n = 20W/m$ *are prescribed on edges BC and CD, respectively. A constant heat source* $f = 4W/m^2$ *is applied over the plate.*

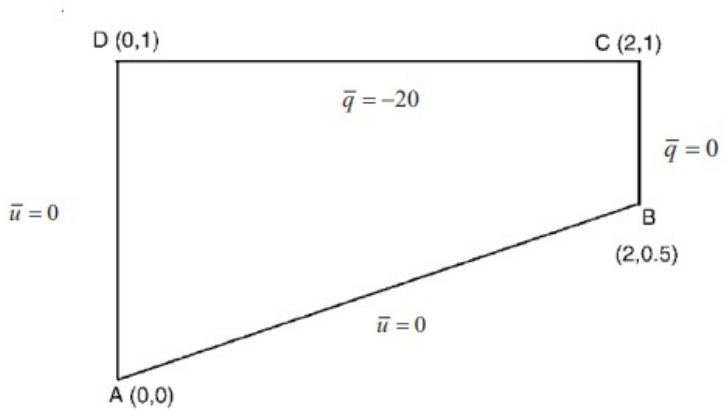

Figure 1: Heat conduction problem geometry.

To assess convergence upon mesh refinement, launch 4 different analysis with increasing number of finite elements. In particular, use structured meshes with:

- $n_{el} = 1$ *element*
- $n_{el} = 5$ *x* $5 = 25$ *elements*
- $n_{el} = 10$ *x* $10 = 100$ *elements*
- $n_e l = 30$ *x* $30 = 900$ *elements*

For these three cases, plot the distribution of temperature along the edge DC in the same graph.

1 Part 1 (Basic)

1.1 Pre-process

1.1.1 Mesh generation

First of all, the mesh of finite elements will be generated on the surface of the statement, assigning the geometrical an physical parameters and boundary conditions defined in the statement. The finite elements software *GID CIMNE* is used for this process.

The first step has been the geometrical definition of the surface, which is represented in figure 2.

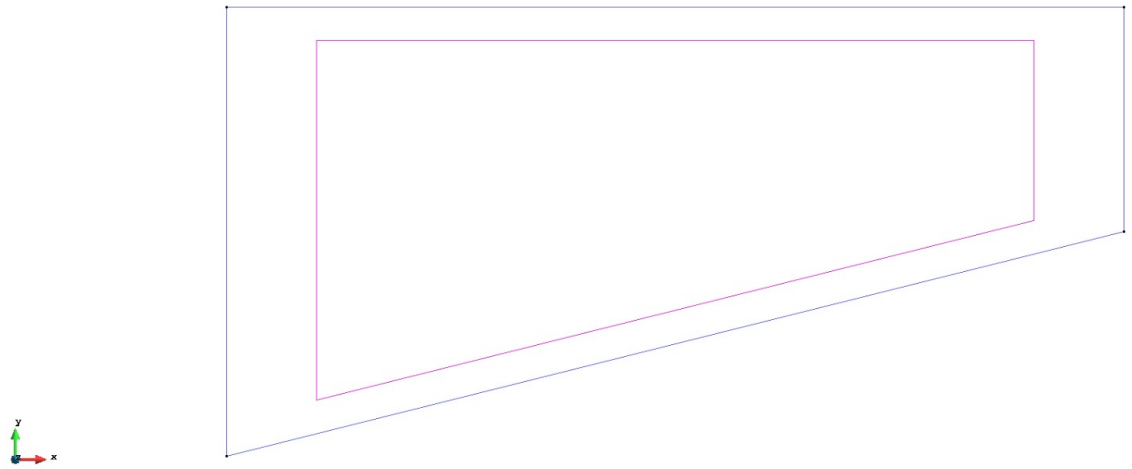

Figure 2: Definition of the surface in the statement using the finite elements software *GID CIMNE*.

Once the surface of the statement is defined, the next step is creating the mesh. The mesh will be structured and composed by quadrilateral elements. Figure 3 shows one of the meshes defined in the problem, with (n_{el} = 10 x 10 = 100 elements) once its elements and nodes have been properly labelled and identified, and material has been assigned.

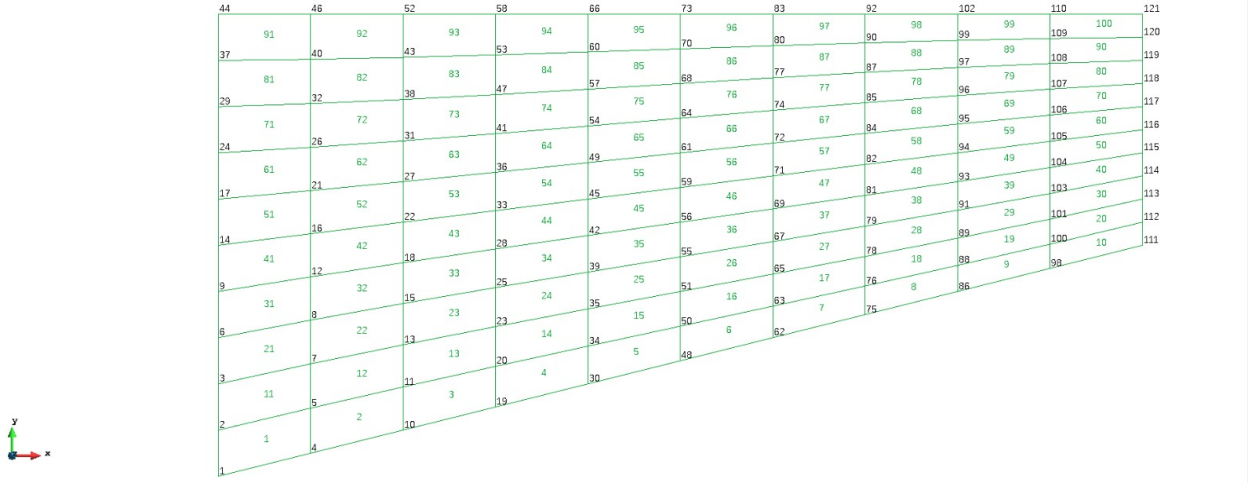

Figure 3: Definition of the surface in the statement using the finite elements software *GID CIMNE*

Once each mesh requested in the statement has been generated, and the element coordinates have been exported into a *.msh* file in order to be computed by Matlab, the next step will be developing the finite elements Matlab routines that will allow to study the heat transference by conduction over the surface for the specified conditions.

1.2 Development of finite element program

Most of the main routines needed to study a general 2D heat transfer system as the presented in the statement had already been given. However, some routines needed to be implemented in order to complete this analysis. In particular, the functions developed in this assignment were needed for the assembly of the global conductance matrix of the system, the global flux source vector, implementation of the shape functions routine, the solution of the final system of

equations and the determination of the heat flux vector at each Gauss point of the system. The development of these functions will be explained in detail in the following subsections, and they can be consulted in the Appendix of this report.

1.2.1 Assembly of the global conductance matrix K

The code implemented for this function can be seen in Appendix A.2. This function computes the global conductance matrix of the system by taking the conductance matrix of each element provided by the function *ComputeKeMatrix*, which was initially given. For the assembly, one loop is defined over the number of elements, and then K matrix stores the expressions of the conductances matrix corresponding to each element. It is important to remark that for the correct functionality of function *ComputeKeMatrix*, it was necessary to previously develop the function *Quadrilateral4NInPoints*, which determines the shape function routines for quadrilateral bilinear elements. The results provided by this function are used by function *ComputeElementShapeFun*, which finally is required for the computation of elemental conductance matrix K^e. In the next subsection, the implementation of the function *Quadrilateral4NInPoints* is explained in detail.

1.2.2 Implementation of a shape function routine for bilinear quadrilateral elements

The Matlab code developed for this function is show in Appendix A.4. This function determines the weights and the positions for the gaussian quadrature defined for 2D quadrilateral bilinear elements. Making use of the Gauss points position values, this function computes the elemental functions matrices N and B, which have the following structure:

$$N = \frac{1}{4} \cdot \begin{bmatrix} (1-\xi)\cdot(1-\eta) & (1+\xi)\cdot(1-\eta) & (1+\xi)\cdot(1+\eta) & (1+\xi)\cdot(1+\eta) \end{bmatrix} \tag{1}$$

$$B = \frac{1}{4} \cdot \begin{bmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi) \end{bmatrix} \tag{2}$$

1.2.3 Assembly of the global flux source vector F_s

This function has been implemented in the Matlab code shown in Appendix A.3. Following the same process as described for the assembly of the global conductance matrix, the flux source vector assembly is implemented in a loop over the number of elements, and storing the flux source values for each element, given by the function *ComputeFseVector*, which follows a similar process as the function *ComputeKeMatrix* explained.

1.2.4 Computation of the solution of the final system of equations

The function *SolveHe* (see Appendix A.5) operates and gives the solution of the final system of equations, which has the following matricial expression:

$$\begin{bmatrix} K_{RR} & K_{RL} \\ K_{LR} & k_{LL} \end{bmatrix} \cdot \begin{bmatrix} d_R \\ d_L \end{bmatrix} = \begin{bmatrix} f_R \\ f_L \end{bmatrix} \tag{3}$$

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

$$d_L = K_{LL}^{-1} \cdot (f_L - K_{LR} \cdot d_R) \tag{4}$$

1.2.5 Computation of the heat flux vector at each Gauss point of the system

The code developed for this function is shown in Appendix A.6. This function determines the heat flux vector at each Gauss point of the system.

$$c^T(Kd - F) = 0 \tag{5}$$

$$\nabla u^e = B^e d^e \tag{6}$$

Compute Ke matrix, diapo 52
Function QuadrilateralInPoints, diapo 65

1.3 Program results: Post-processing

Once the necessary functions have been implemented and 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 over the surface.

1.3.1 First Mesh (1 element 1x1 quadrilateral elements mesh)

In this first case, the mesh generation is shown in figure 4.

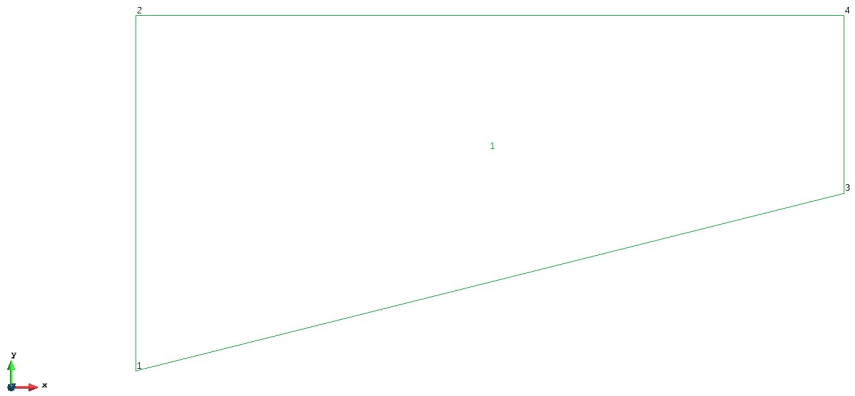

Figure 4: Mesh generated for on the surface for 1 element discretization (1x1 quadrilateral elements mesh).

By executing the program, the temperature field over the surface can be obtained from the GID post-processing feature. Figure 15 shows the temperatures field simulation obtained by the program.

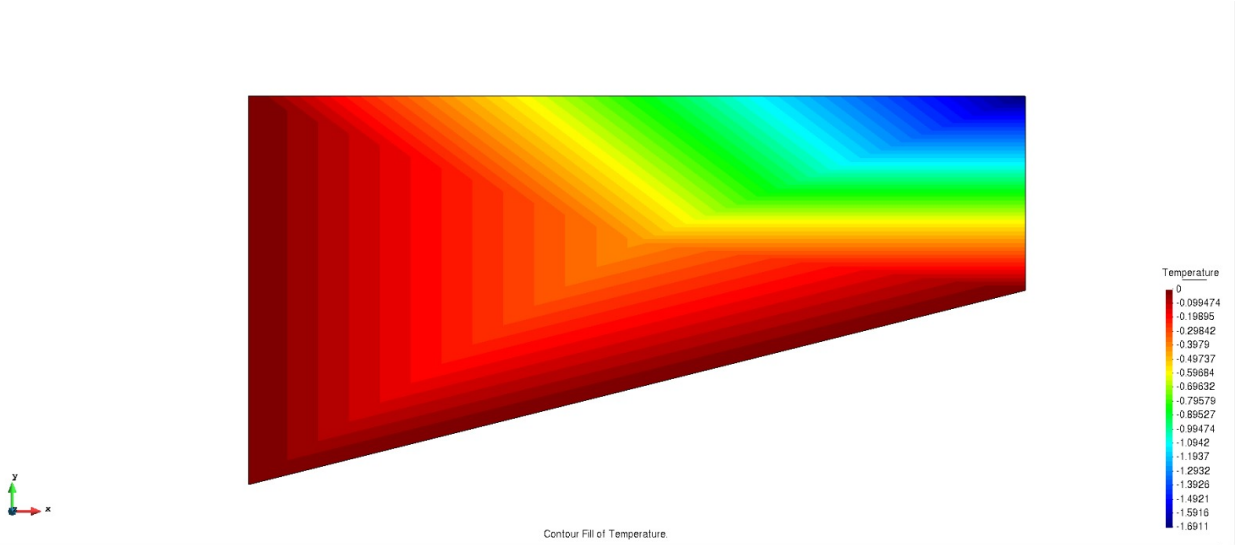

Figure 5: Simulation of the temperatures gradient on the surface for 1 discretization element (1x1 quadrilateral elements mesh).

Finally, the temperature distribution over the edge CD is analysed from a post-processing graph obtained from *GID* post-processing results window. This temperature distribution is shown in figure 6

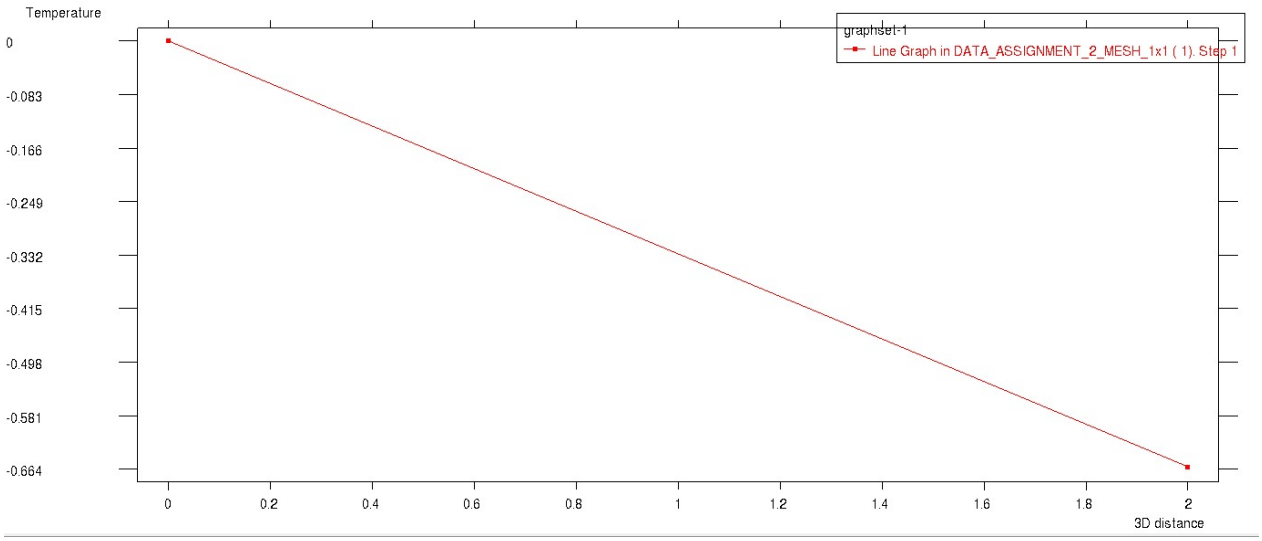

Figure 6: Temperature distribution over the edge CD for 1 element discretization (1x1 quadrilateral elements mesh).

1.3.2 Second Mesh (25 elements 5x5 quadrilateral elements mesh)

In this second analysis, a 25 elements mesh has been generated. As in the previous case, it is a structured mesh composed by quadrilateral elements, being 5 elements in contact with each edge of the surface. The final mesh generated using *GID* is shown in figure 7

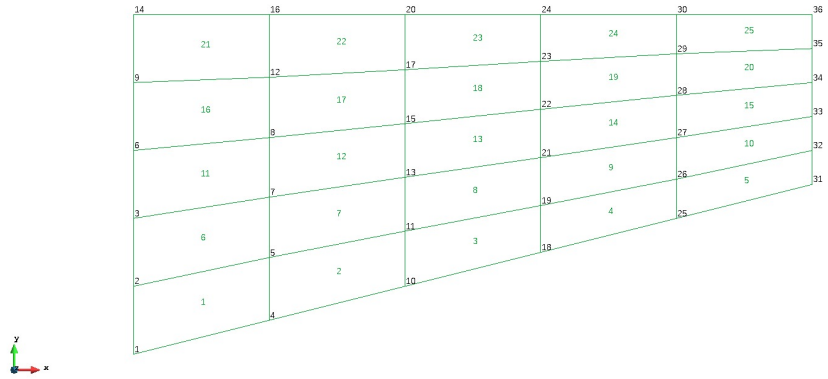

Figure 7: Mesh generated for on the surface for 25 elements discretization (5x5 quadrilateral elements mesh).

Ágain, compiling the finite elements program and post-processing the results for temperatures field using *GID* post-process results window, the following temperatures distribution on the surface is obtained:

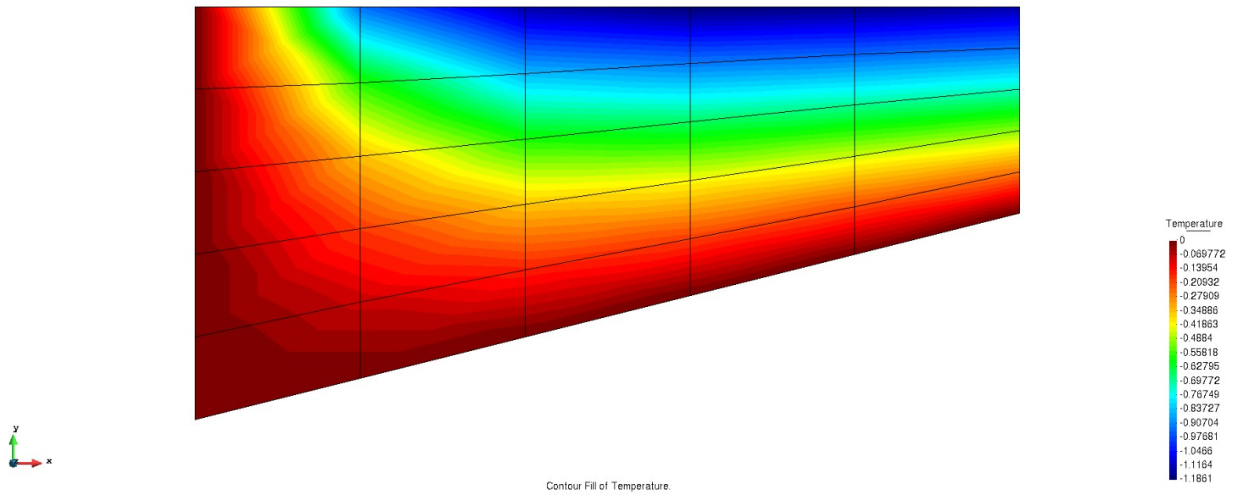

Figure 8: Simulation of the temperatures gradient on the surface for 25 discretization elements (5x5 quadrilateral elements mesh).

Regarding the analysis of the temperature distribution results over the edge CD for this new discretization, the following graph has been obtained from *GID* simulation interface.

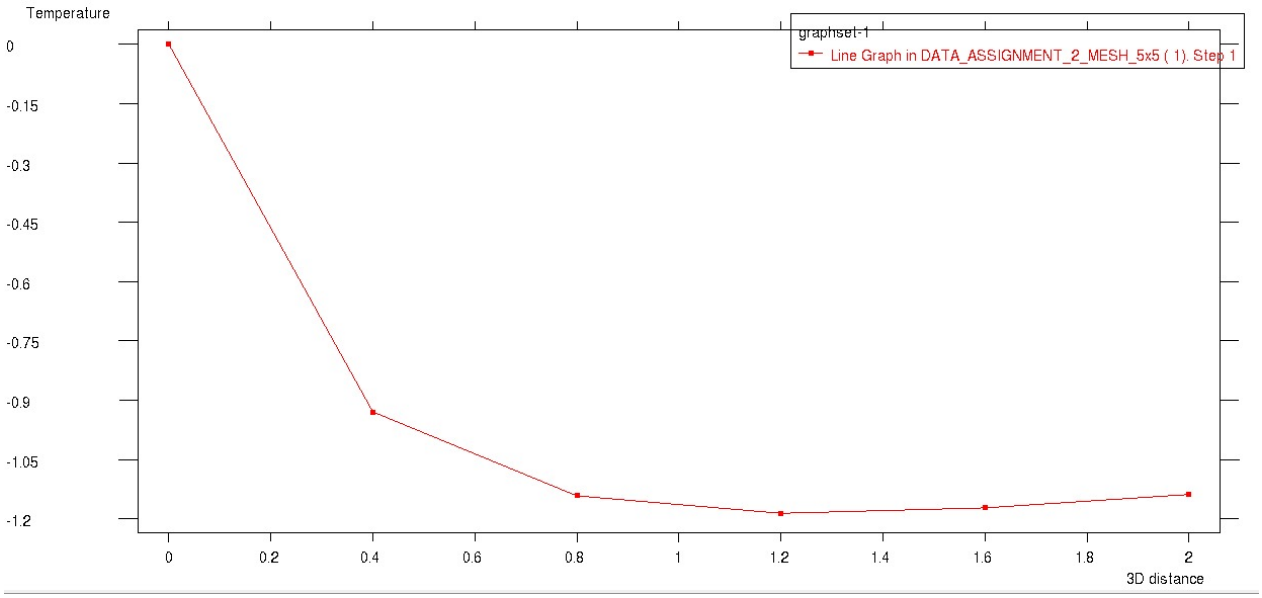

Figure 9: Temperature distribution over the edge CD for 25 elements discretization (5x5 quadrilateral elements mesh).

1.3.3 Third Mesh (100 elements 10x10 quadrilateral elements mesh)

The next mesh to analyse will be a structured quadrilateral elements mesh composed by 100 discretization elements, being 10 elements in contact to each edge of the surface. The generation of this mesh in *GID* can be seen figure 10

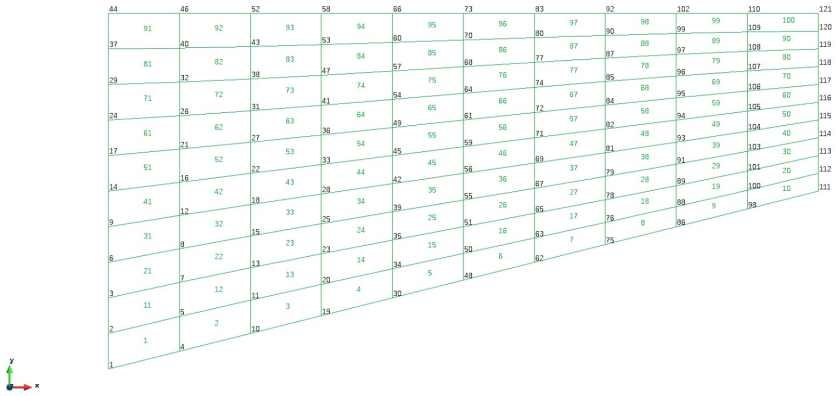

Figure 10: Mesh generated for on the surface for 100 elements discretization (10x10 quadrilateral elements mesh).

By executing the Matlab program and solving the system for this new mesh, a *.res* format file is obtained, which is computed by *GID* in its post-processing feature. The following temperature distribution is obtained:

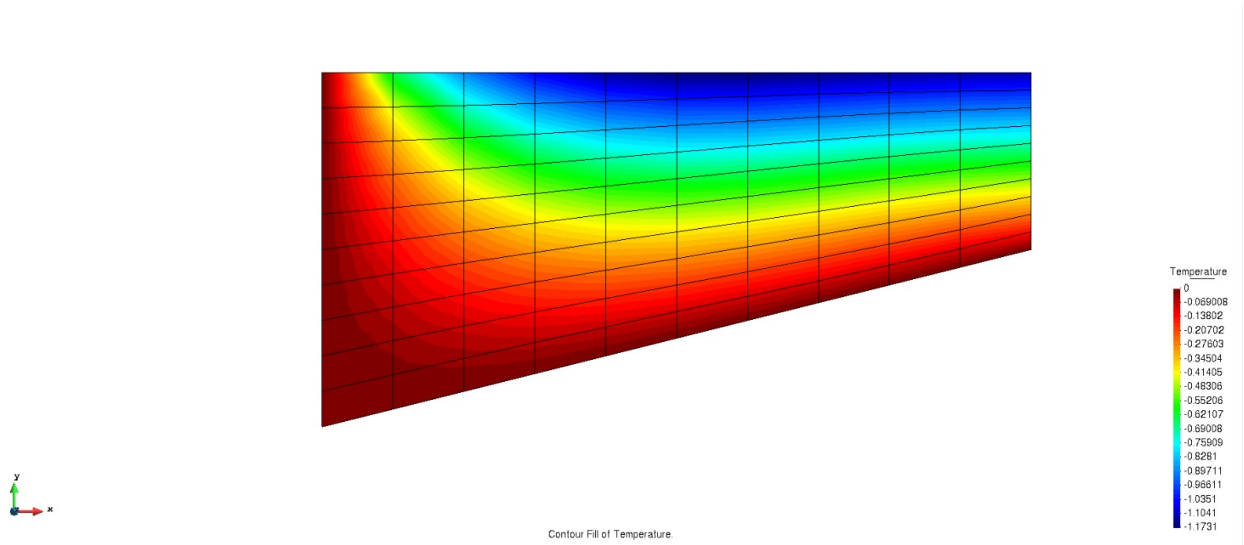

Figure 11: Simulation of the temperatures gradient on the surface for 100 discretization elements (10x10 quadrilateral elements mesh).

In this case, the simulation of the temperature gradient on the edge CD for 100 discretization elements mesh is computed by *GID* in the following graph:

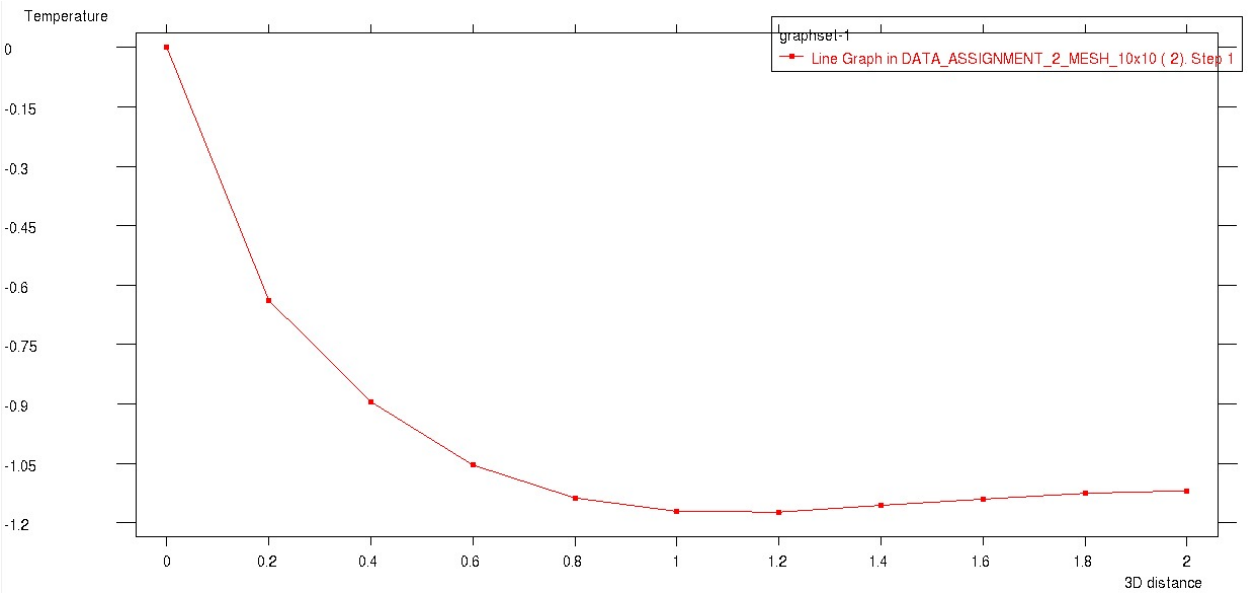

Figure 12: Simulation of the temperatures gradient on the surface for 1 discretization element.

1.3.4 Fourth Mesh (900 elements 30x30 quadrilateral elements mesh)

The last mesh to generate for the heat conduction analysis on the surface is a structured quadrilateral elements mesh, composed by 900 elements, being 30 elements in contact with each edge of the surface. The mesh generated in this case is shown in figure 13

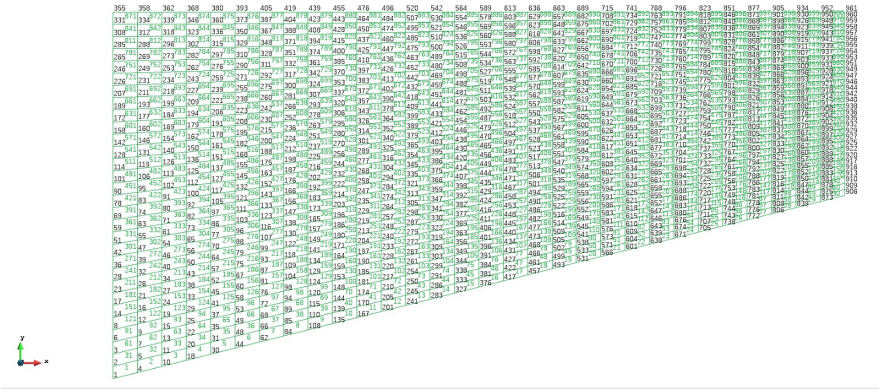

Figure 13: Mesh generated for on the surface for 900 elements discretization (30x30 quadrilateral elements mesh).

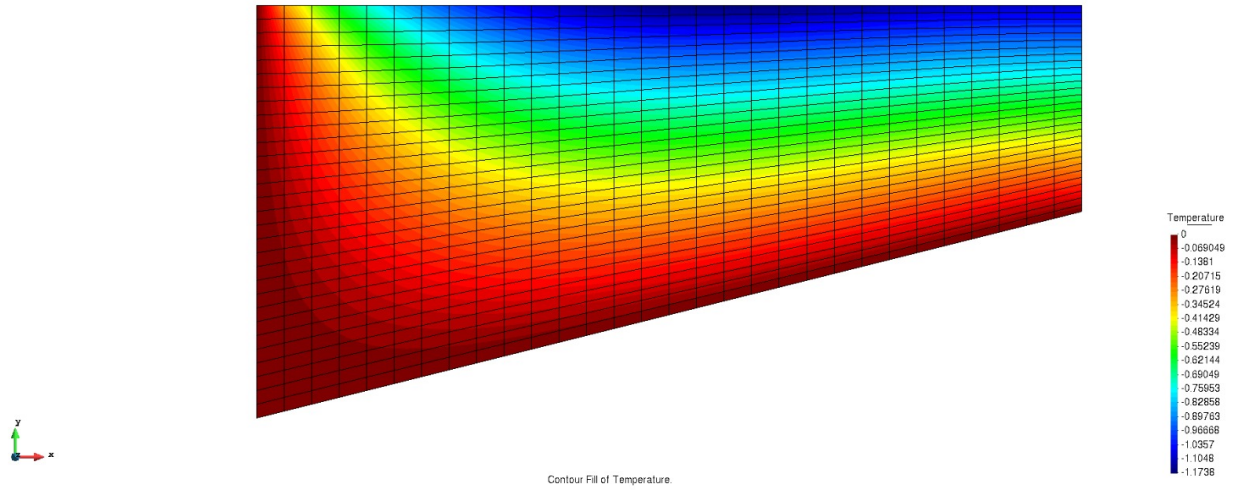

Figure 14: Simulation of the temperatures gradient on the surface for 900 discretization elements (30x30 quadrilateral elements mesh).

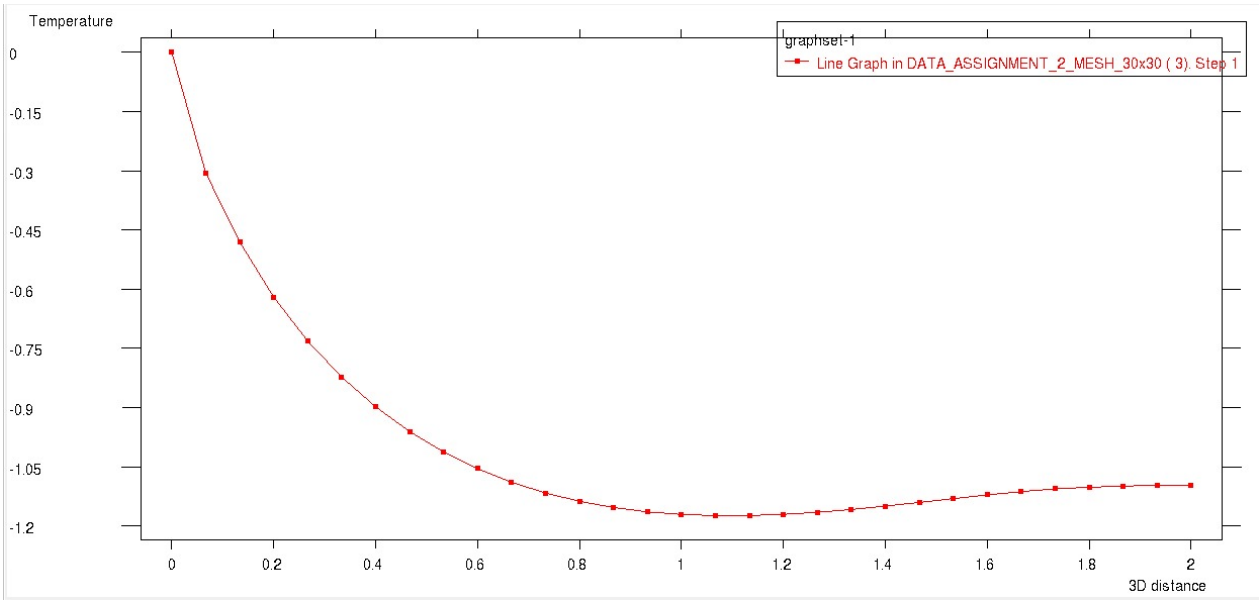

Figure 15: Temperature distribution over the edge CD for 900 elements discretization (30x30 quadrilateral elements mesh).

Repeating the same process as comented in the case of the previous meshes, the temperature field simulation over the surface is the following:

1.4 Analysis of the convergence in the finite elements program

From the analysis of the simulation results obtained in the previous section for different meshes discretizations, it has been checked that the temperatures distributions obtained have a significative dependence on the discretization characteristics. Comparing the results obtained for the different discretizations, it can be seen that there is a notable difference in precision between the first three meshes analysis, whereas in the case of the third and the fourth ones,

the difference is quite similar although there is a higher difference in the number of discretization elements. It shows that there is not a linear relationship between the number of discretization elements and the mesh precision, and results tend to converge for increasing number of finite elements discretization.

In order to assess this convergence more accurately, the results of the simulations for temperature distributions over the edge CD have been plotted in the same graph, as is shown in figure 16.

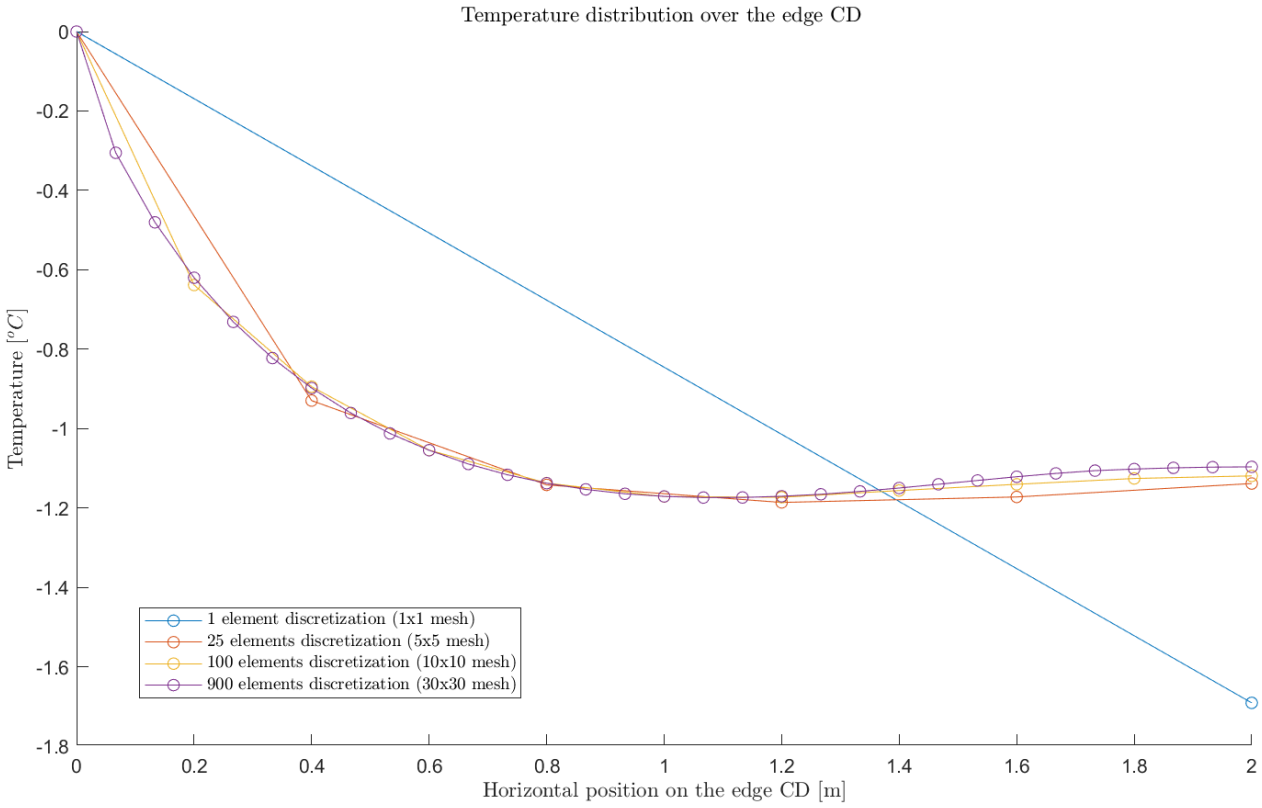

Figure 16: Caption

As commented, it is verified that the temperatures values tend to converge fot an increasing number of discretization elements, as the relative error between the results in discretizations minimizes when comparing results for discretizations with a higher number of elements.

1.5 Heat flux over the surface

In this section, the simulation results provided by *GID* for the heat flux over the surface are shown. These results have been obtained by implementation of the function *HeatFlux*, as explained in the first section of the report.
The simulation results for the different discretizations analysed are shown in the following figures:

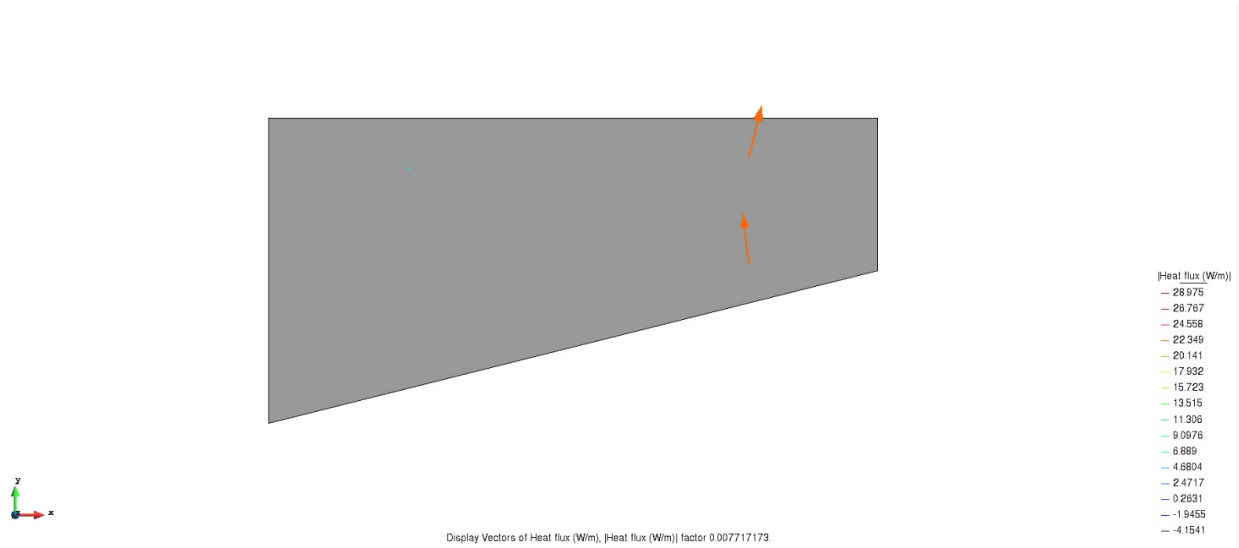

Figure 17: Simulation of the heat flux on the surface for 1 discretization element (1x1 quadrilateral elements mesh).

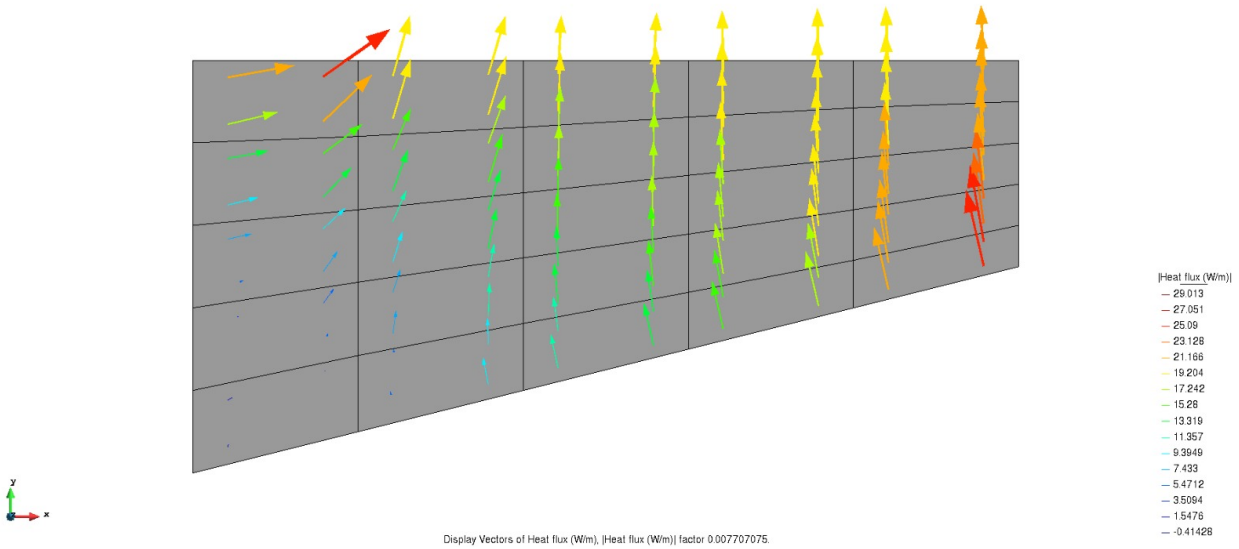

Figure 18: Simulation of the heat flux on the surface for 25 discretization elements (5x5 quadrilateral elements mesh).

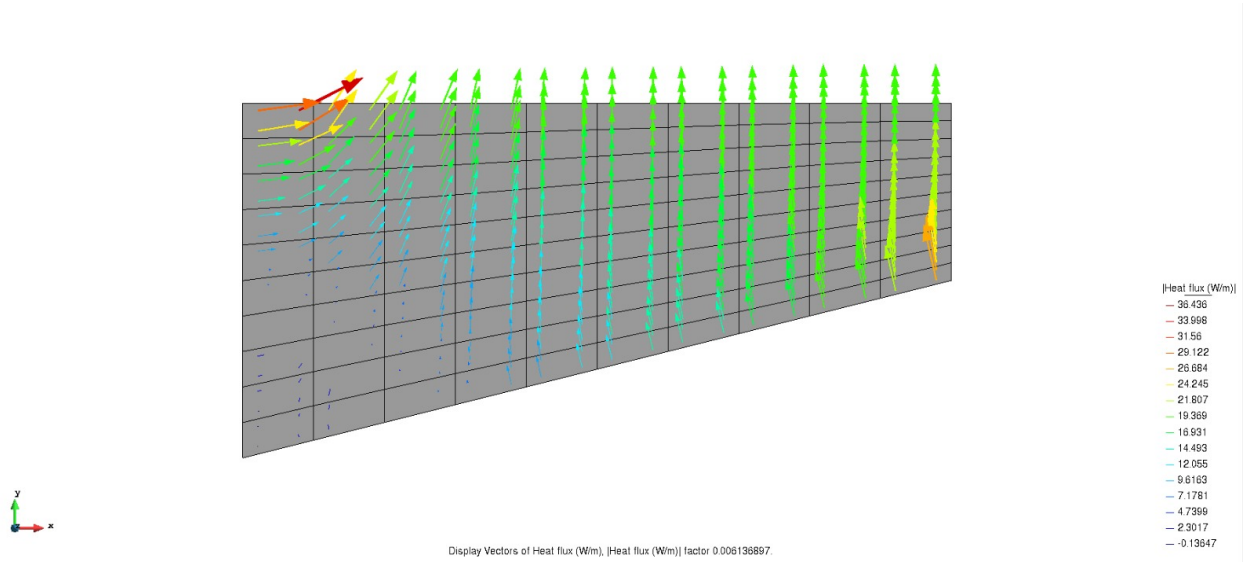

Figure 19: Simulation of the heat flux on the surface for 100 discretization elements (10x10 quadrilateral elements mesh).

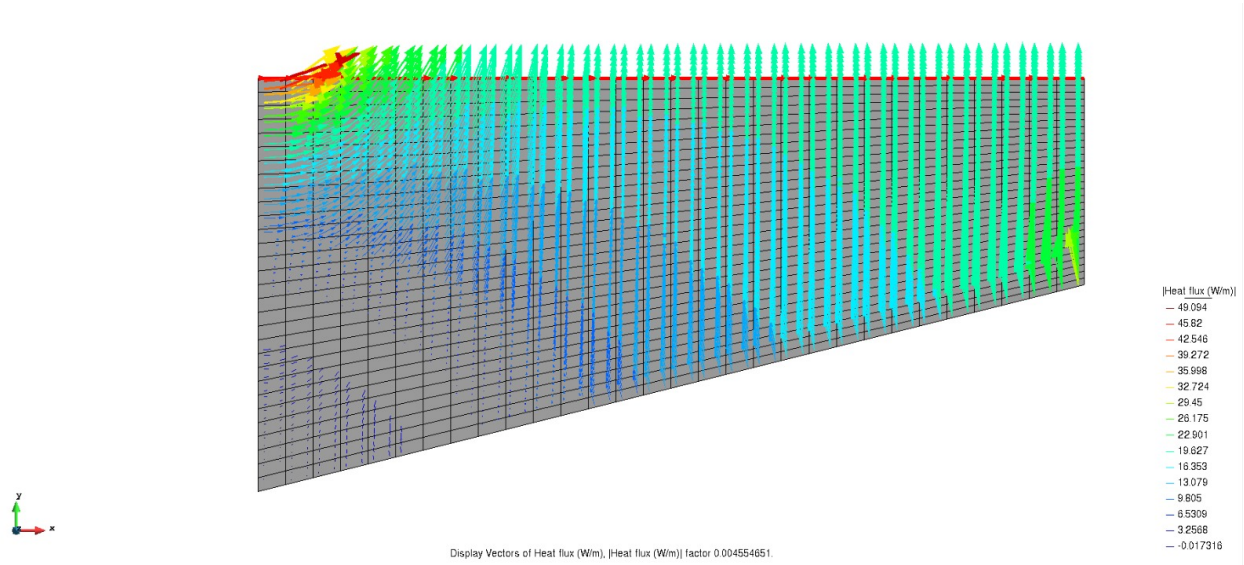

Figure 20: Simulation of the heat flux on the surface for 900 discretization elements (30x30 quadrilateral elements mesh).

From the simulation results obtained for the different discretizations, it is checked the heat flux vectors distribution over the surface is more accurate as the number of discretization elements is increased. Then, it has been figured out that the modulus and the direction of heat flux vectors over the surface has a higher precision with a increasing number of elements. As shown, the discretization for 900 elements is the one which provides a better description of the heat flux on the surface.

2 Part 2 (Advanced)

2.1 Convective heat transfer

In this advanced part, the problem in statement 1 is modified so that one of the has a heat transference through convection for being in contact with a fluid that creates a gradient of temperature on the upper side. These new boundary conditions are shown in figure 21.

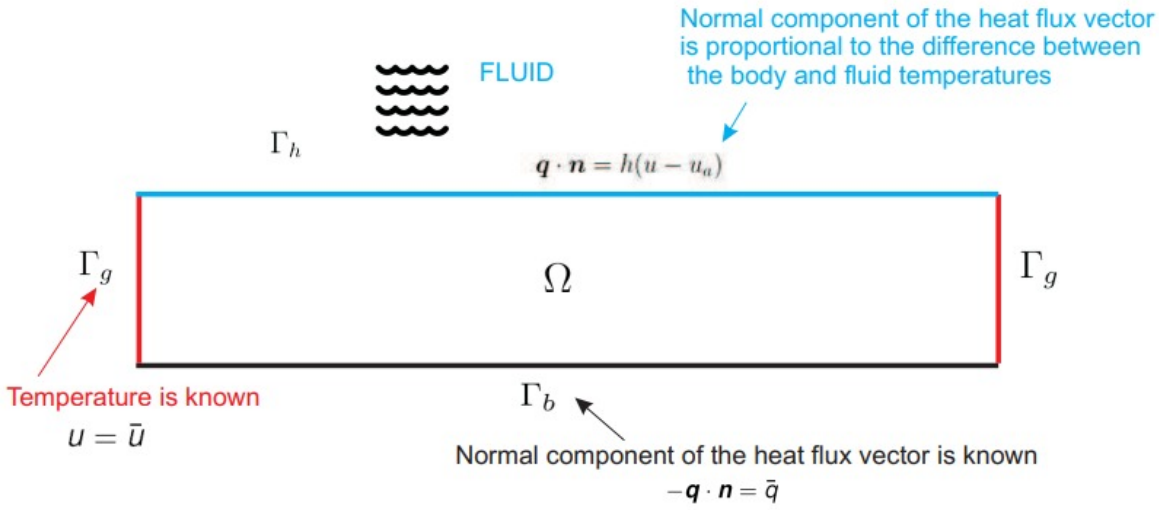

Figure 21: Boundary conditions definition for the problem in part 2.

In the statement, it is requested to solve the Weak form of this Boundary Value Problem in order to determine u such that:

$$\int_\Omega \nabla v^T \cdot \kappa \cdot \nabla u \cdot d\Omega = \int_\Omega v \cdot f \cdot d\Omega + \int_{\Gamma_b} v \cdot \overline{q} \cdot d\Gamma \quad f : \Omega \longrightarrow \mathbb{R}, \ \ \forall v \in V, \ \ u \in S \tag{7}$$

First of all, the changes in matrix K and vector F need to be studied from a simple heat conduction transference as the one analysed in the part 1 of the assignment, and a heat conduction problem with a new condition that implies heat transference by convection in one of its boundaries. For this analysis, the first step will be considering the formulation of the Weak Form of the BVP for a prototypical linear heat conduction problem, which is given by the expression in equation 7.

In the new conditions given in this statement, it is necessary to add a term $\overline{q} = h \cdot (u - u_a)$. Then, adding this term as an integral over the boundary surface, the new formulation of the Weak Form of the BVP in this case is the following:

$$\int_\Omega \nabla v^T \cdot \kappa \cdot \nabla u \cdot d\Omega = \int_\Omega v \cdot f \cdot d\Omega + \int_{\Gamma_b} v \cdot \overline{q} \cdot d\Gamma + \int_{\Gamma_a} v \cdot h \cdot (u - u_a) \cdot d\Gamma \quad f : \Omega \longrightarrow \mathbb{R}, \ \ \forall v \in V, \ \ u \in S \tag{8}$$

Where Γ_a is defined as the boundary surface where the heat convection between the solid and the fluid takes place, defining then that $\Gamma = \Gamma_a \cup \Gamma_b \cup \Gamma_g$; $\Gamma_a \cap \Gamma_b \cap \Gamma_g = \emptyset$, and u_a is defined as the values of the fluid temperature field over the Gauss points. The formulation in equation 8 can be expressed as follows:

$$\int_\Omega \nabla v^T \cdot \kappa \cdot \nabla u \cdot d\Omega - \int_{\Gamma_a} v \cdot h \cdot (u - u_a) \cdot d\Gamma = \int_\Omega v \cdot f \cdot d\Omega + \int_{\Gamma_b} v \cdot \overline{q} \cdot d\Gamma - \int_{\Gamma_a} v \cdot h \cdot u_a \cdot d\Gamma \tag{9}$$

This formulation can be expressed in terms of N and B shape functions matrices as:

$$v\left\{\left(\int_\Omega \nabla B^T\cdot\kappa\cdot B\cdot d\Omega-\int_{\Gamma_a} N^T\cdot h\cdot N\cdot d\Gamma\right)\cdot u=\int_\Omega N^T\cdot f\cdot d\Omega+\int_{\Gamma_b} N^T\cdot\overline{q}\cdot d\Gamma-\int_{\Gamma_a} N^T\cdot h\cdot u_a\cdot d\Gamma\right\} \tag{10}$$

From this formulation, matrix K and F vector can be deducted:

$$\begin{cases} K=\int_\Omega B^T\cdot\kappa\cdot B\cdot d\Omega-\int_{\Gamma_a} N^T\cdot h\cdot N\cdot d\Gamma \\ F=\int_\Omega N^T\cdot f\cdot d\Omega+\int_{\Gamma_b} N^T\cdot\overline{q}\cdot d\Gamma-\int_{\Gamma_a} N^T\cdot h\cdot u_a\cdot d\Gamma \end{cases} \tag{11}$$

Comparing this expression with the one given in the statement of the assignment, it is concluded that:

$$K_h=-\int_{\Gamma_a} N^T\cdot h\cdot N\cdot d\Gamma \tag{12}$$

$$F_h=-\int_{\Gamma_a} N^T\cdot h\cdot u_a\cdot d\Gamma \tag{13}$$

2.2 Time-dependent heat conduction

In this last section, it is requested to find the capacitance matrix M in terms of the shape functions employed in the interpolation of a time-dependent heat conduction problem.

$$c^T\cdot(M\cdot\dot{d}+L\cdot d-F)=0 \tag{14}$$

The Strong Form of the BVP in this case is the following:

$$\begin{cases} \nabla q-f=\rho\cdot c\cdot\frac{\partial u}{\partial t} & \text{in } \Omega\times[0,T] \\ u=\overline{u} & \text{on } \Gamma_g\times[0,T] \\ -q\cdot n=\overline{q} & \text{on } \Gamma_b\times[0,T] \\ u(x,0)=u_0(x) & \text{in } \Omega \end{cases} \tag{15}$$

First of all, in order to determine the Weak Form of the problem, the formulation in the Strong Form will be multiplied by an arbitrary function v and integrated over the domain Ω:

$$\int_\Omega v\cdot(\nabla q-f)d\Omega=\int_\Omega v\left(\cdot\rho\cdot c\cdot\frac{\partial u}{\partial t}\right)d\Omega \tag{16}$$

Applying integration by parts in this expression, yields:

$$\int_\Omega \nabla v\cdot q\cdot d\Omega=\int_\Omega v\cdot\left(\rho\cdot c\frac{\partial u}{\partial t}\right)\cdot d\Omega \tag{17}$$

$$\int_\Gamma v\cdot(-\overline{q})\cdot d\Gamma-\int_\Omega \nabla v\cdot q\cdot d\Omega-\int_\Omega v\cdot f\cdot\Omega=\int_\Omega v\cdot\left(\rho\cdot c\frac{\partial u}{\partial t}\right)\cdot d\Omega \tag{18}$$

Then, applying the definition $q=-\kappa\nabla u$ in the previous equation, the following formulation is obtained:
Expressing this equation in term of shape functions N and B the following formulation is achieved:

$$c^T\left[\left[\left(-\int_\Omega N^T\cdot\rho\cdot c\cdot d\Omega\right)\cdot\dot{d}+\left(\int_\Omega B^T\cdot\kappa\cdot B\cdot d\Omega\right)\cdot d-\left(\int_{\Gamma_b} N^T\cdot\overline{q}d\Gamma+\int_\Omega N^T\cdot f\cdot d\Omega\right)\right]=0 \tag{19}$$

Comparing the terms in this expression with the ones in equation , it can be concluded that the conductance matrix M will be defined by the following expression:

$$M=-\int_\Omega N^T\cdot\rho\cdot c\cdot N\cdot d\Omega \tag{20}$$

A Appendix

A.1 mainHEATC

```
  clc
clear all
% "Template" Finite Element Program for Heat Conduction Problems
% ECA.
% Technical University of Catalonia
% JoaquIn A. Hdez, October 8-th, 2015
% ------------------------------------------------

if exist('ElemBnd')==0
    addpath('ROUTINES_AUX') ;
end
%%% INPUT  %%%
% Input data file %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

NAME_INPUT_DATA = 'DATA_ASSIGNMENT_2_MESH_30x30';

%-------------------------------------------------------

% PREPROCESS
[COOR,CN,TypeElement,TypeElementB, ConductMglo,  rnod,dR, qFLUXglo,CNb,fNOD,NameFileMesh] ...
    = ReadInputDataFile(NAME_INPUT_DATA)  ;

% SOLVER
% ------------------------------------------
[d,qheatGLO,posgp] = ...
    SolveHeatFE(COOR,CN,TypeElement,TypeElementB,ConductMglo,rnod,dR,qFLUXglo,CNb,fNOD);

% POSTPROCESS
% ------------------------------------------
disp('POSTPROCESS....')
GidPostProcess(COOR,CN,TypeElement,d,qheatGLO,NAME_INPUT_DATA,posgp,NameFileMesh);
```

A.2 ComputeK Function

```
    function K = ComputeK(COOR,CN,TypeElement,ConductMglo);
%%%%
% This subroutine   returns the global conductance matrix K (nnode x nnode)
% Inputs
% --------------
% 1. Finite element mesh
% -------------------
% COOR: Coordinate matrix (nnode x ndim)
% CN: Connectivity matrix (nelem x nnodeE)
% TypeElement: Type of finite element (quadrilateral,...)
% -----------
% 2. Material
% -----------
%  ConductMglo (ndim x ndim x nelem)  % Array of conductivity matrices
%%%%
 if nargin == 0
     load('tmp1.mat')
 end

% Dimensions of the problem
nnode = size(COOR,1);  % Number of nodes
ndim = size(COOR,2);   % Spatial Dimension of the problem  (2 or 3)
nelem = size(CN,1);    % Number of elements
nnodeE = size(CN,2) ;  %Number of nodes per element
```

```
25
26 % Determine Gauss weights, shape functions and derivatives
27 TypeIntegrand = 'K';
28 [weig,posgp,shapef,dershapef] = ComputeElementShapeFun(TypeElement,nnodeE,TypeIntegrand) ;
29
30 % Assembly of matrix K
31 K=sparse(nnode,nnode);
32
33  for e = 1:nelem
34    NODOSe=CN(e,:);
35    Xe=COOR(NODOSe,:)';
36    ConductM=ConductMglo(:,:,e);
37    Ke=ComputeKeMatrix(ConductM,weig,dershapef,Xe);
38    K(NODOSe,NODOSe)=K(NODOSe,NODOSe)+Ke;
39    end
40  end
```

A.3 ComputeFs Function

```
1  function Fs = ComputeFs(COOR,CN,TypeElement, fNOD) ;
2  % This subroutine   returns the  heat source contribution (Fs)    to the
3  % global flux vector. Inputs
4  % --------------
5  % 1. Finite element mesh
6  % --------------------
7  % COOR: Coordinate matrix (nnode x ndim)
8  % CN: Connectivity matrix (nelem x nnodeE)
9  % TypeElement: Type of finite element (quadrilateral,...)
10 % -----------
11 % 2. Vector containing the values of the heat source function at the nodes
12 % of the mesh
13 % -----------
14 %  fNOD (nnode x 1)  %
15 %%%%
16
17 % Dimensions of the problem
18 nnode = size(COOR,1);
19 ndim = size(COOR,2);
20 nelem = size(CN,1);
21 nnodeE = size(CN,2) ;
22
23 TypeIntegrand='RHS';
24 [weig,posgp,shapef,dershapef]=ComputeElementShapeFun(TypeElement,nnodeE,TypeIntegrand);
25 Fs = zeros(nnode,1);
26
27     for e=1:nelem
28         NODOSe=CN(e,:);
29         fe=fNOD(NODOSe);
30         Xe=COOR(NODOSe,:)';
31         Fse=ComputeFseVector(fe,weig,shapef,dershapef,Xe);
32
33         % Fs Assembly
34         for f=1:nnodeE
35         Fs(e)=Fs(e)+Fse(f);
36         end
37
38     end
39
40 end
```

A.4 Quadrilateral4NInPoints Function

```
    function [weig,posgp,shapef,dershapef] = Quadrilateral4NInPoints()

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

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

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

end
```

A.5 SolveHE Function

```

    function[d,qheatGLO,posgp]=SolveHE(K,Fs,Fbnd,dR,rnod,COOR,CN,TypeElement,ConductMglo) ;

% This function returns  n returns the (nnode x 1) vector of nodal temperatures (d),
% as well as the vector formed by the heat flux vector at all gauss points  (qheatGLO)

%% Input data
% K = Global conductance matrix   (nnode x nnode)
% Fs = Global source flux vector  (nnode x 1)
% Fbnd = Global boundary flux vector  (nnode x 1)
% rnod = Set of nodes at which temperature is prescribed
% dR = Vector of prescribed displacements
% ----------------------
nnode = size(COOR,1);
ndim = size(COOR,2); nelem = size(CN,1);
nnodeE = size(CN,2);
posgp=[];

% Solution of the system of FE equation
% Right-hand side
F = Fs+Fbnd;
% Set of nodes at which temperature is unknown
lnod = 1:nnode ;
lnod(rnod) = [] ;
% dL =  Kll^{-1}*(Fl .Klr*dR)
dL=K(lnod,lnod)\(F(lnod)-K(lnod,rnod)*dR);

% Vector of   temperatures
d = zeros(nnode,1) ;
d(lnod)= dL ;
d(rnod) = dR ;

%%%% Computation of heat flux vector at each gauss point
disp('Computation of heat flux vector at each gauss point and elements')
```

```
ngauss = size(posgp,2);
qheatGLO = zeros(ngauss*ndim,nelem);
% Computing this array is not mandatory....
[qheatGLO posgp]= HeatFlux(COOR,CN,TypeElement,ConductMglo,d);
```

A.6 HeatFlux Function

```
function [qheatGLO posgp]= HeatFlux(COOR,CN,TypeElement,ConductMglo,d);


% Dimensions of the problem
nnodes = size(COOR,1);
ndimensions = size(COOR,2);
nelem = size(CN,1);
nnodeE = size(CN,2);

qheatGLO=zeros(8,nelem);

TypeIntegrand='K';
[weig,posgp,shapef,dershapef]=ComputeElementShapeFun(TypeElement,nnodeE,TypeIntegrand);


    for e=1:nelem

    NODOSe=CN(e,:);
    de=d(NODOSe);
    ConductM=ConductMglo(:,:,e);
    Xe=COOR(NODOSe,:)';
    ndimensions=size(Xe,1);
    nnodeE=size(Xe,2);
    ngauss=length(weig);

        for  g = 1:ngauss
            % Matrix of derivatives for Gauss point "g"
            BeXi = dershapef(:,:,g) ;
            % Jacobian Matrix
            Je = Xe*BeXi' ;
            % JAcobian
            detJe = det(Je) ;
            % Matrix of derivatives with respect to physical coordinates
            Be = inv(Je)'*BeXi ;
            % Heat Flux
            qheatflux=-ConductM*Be*de;
            first_el=(g-1)*ndimensions+1;
            last_el=g*ndimensions;

            qheatGLO(first_el:last_el,e)=qheatflux;

        end
    end
end
```