

# Fiammetta, Finite Element Method Applied to Heat Transfer

## – Tutorials –

Benoit Beckers

*May 2023*

Urban Physics Joint Laboratory

Université de Pau et des Pays de l'Adour (France)

The six lectures on **Finite Element Method** and **Isoparametric Meshing Applied to Transient Thermal Analysis**, of which this document constitutes the tutorial, deal respectively with conduction, convection and radiation, first in steady state, then, with the fourth lecture, in transient state; the third lecture describes the isoparametric elements, which make it possible to generalize to any geometry what has been described before on the simplest shape: a rectangular mesh composed of square elements.

The aim of this course is to explain in a simple and concise way the basis of the finite element method for the study of thermal problems, including radiative exchanges, as in the case of buildings exposed to solar radiation, and finally to compute the surface temperature field, such as it could be captured, in the real world, by a thermal camera located in front of these buildings.

The tutorial presents short functions, written in Matlab<sup>©</sup>, which are progressively developed in conduction, convection, radiation and transient regime. Each step is completed by an exercise that enables the reader to become familiar with the main features presented in the lectures.

# 1. Tutorial I: Basic problem of thermal conduction

## 1.1 Basic finite element model

In each sub-domain or finite element numbered  $i$ , a Rayleigh-Ritz is applied. The temperature field  $\tau_i$  of element  $i$  is discretized by bilinear polynomial functions associated to the nodal temperatures  $T_{ij}$  of the four vertices (*Figure 2*).

$$\tau_i = \sum_{j=1}^4 T_{ij} f_{ij}(x, y) \quad (1)$$

Explicitly, for a rectangle of dimensions  $a \times b$ , we have:

$$\tau_i = T_{i1} \left(1 - \frac{x}{a}\right) \left(1 - \frac{y}{b}\right) + T_{i2} \frac{x}{a} \left(1 - \frac{y}{b}\right) + T_{i3} \frac{x}{a} \frac{y}{b} + T_{i4} \left(1 - \frac{x}{a}\right) \frac{y}{b} \quad (2)$$

The temperature gradients are obtained by derivation of the element temperature field  $\tau_i$  (*Figure 2*). It is easy to derive the polynomial expression of  $\tau_i$ :

$$\begin{aligned} \begin{bmatrix} \frac{\partial \tau_i}{\partial x} \\ \frac{\partial \tau_i}{\partial y} \end{bmatrix} &= \begin{bmatrix} \frac{1}{a} \left(\frac{y}{b} - 1\right) & \frac{1}{a} \left(1 - \frac{y}{b}\right) & \frac{y}{ab} & -\frac{y}{ab} \\ \frac{1}{b} \left(\frac{x}{a} - 1\right) & -\frac{x}{ab} & \frac{x}{ab} & \frac{1}{b} \left(1 - \frac{x}{a}\right) \end{bmatrix} \begin{bmatrix} T_{i1} \\ T_{i2} \\ T_{i3} \\ T_{i4} \end{bmatrix} \\ \begin{bmatrix} \frac{\partial \tau_i}{\partial x} \\ \frac{\partial \tau_i}{\partial y} \end{bmatrix} &= \frac{1}{ab} \begin{bmatrix} y - b & b - y & y & -y \\ x - a & -x & x & a - x \end{bmatrix} \begin{bmatrix} T_{i1} \\ T_{i2} \\ T_{i3} \\ T_{i4} \end{bmatrix} \end{aligned} \quad (3)$$

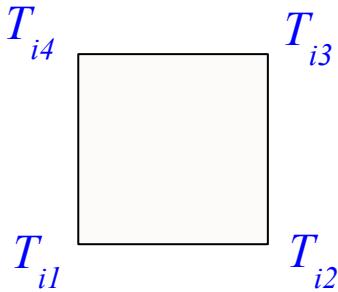
This evaluation of the temperature gradient allows computing the conduction matrix of the element (a square in *Figure 2*).

$$K_{el} = \int_{\omega_i} \frac{k_i e_i}{2} \begin{bmatrix} \frac{\partial \tau_i}{\partial x} & \frac{\partial \tau_i}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial \tau_i}{\partial x} \\ \frac{\partial \tau_i}{\partial y} \end{bmatrix} d\omega_i \quad (4)$$

This matrix does not depend on its geometrical size, but only on the conductivity coefficient  $k$ , the thickness  $e$  and the aspect ratio of the rectangle dimensions:  $a/b$ . *Figure 1* shows a rectangle its degrees of freedom (*DOF*) and its conductivity matrix. *Figure 2* shows the same for a square element. The element nodes are always presented in the counterclockwise sequence of the figure.

$$K_{el} = \frac{ke}{6ab} \begin{bmatrix} 2(a^2 + b^2) & (a^2 - 2b^2) & -(a^2 + b^2) & (b^2 - 2a^2) \\ (a^2 - 2b^2) & 2(a^2 + b^2) & (b^2 - 2a^2) & -(a^2 + b^2) \\ -(a^2 + b^2) & (b^2 - 2a^2) & 2(a^2 + b^2) & (a^2 - 2b^2) \\ (b^2 - 2a^2) & -(a^2 + b^2) & (a^2 - 2b^2) & 2(a^2 + b^2) \end{bmatrix}$$

*Figure 1: A rectangular element and its conductivity matrix*



$$K_{el} = \frac{ke}{6} \begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix}$$

Figure 2: A square element and its conductivity matrix

The contributions of the *nel* finite elements of the domain are added to build the discretized global functional  $I(T)$ :

$$\langle I(T) = \sum_{i=1}^{nel} \left( \int_{\Omega_i} \frac{1}{2} k_i \vec{\nabla} \sum_{j=1}^4 \mathbf{T}_{ij} f_{ij} \cdot \vec{\nabla} \sum_{j=1}^4 \mathbf{T}_{ij} f_{ij} d\Omega_i + \int_{S_{2i}} \bar{q}_n \sum_{j=1}^4 \mathbf{T}_{ij} f_{ij} dS_i \right) \rangle \quad (5)$$

After introducing the polynomial trial functions given in (2), we can write (5) in matrix form:

$$\langle I(T) = \sum_{i=1}^{nel} [\mathbf{T}_i]^T [K_i] [\mathbf{T}_i] + [\mathbf{T}_i]^T [F_i] \rangle \quad (6)$$

The last term  $[F_i]$  of (6) is the vector of generalized heat loads.

In the next step, we have to express the continuity of the temperature field across the domain. At each interface between two elements, it must be stated that the nodal temperature field is identical, which means that the nodal temperatures of the elements sharing a same node are the same. In the mesh of Figure 3, the third node of element 3, the fourth of element 4, the second of element 5 and the first of element 6 are assigned to the global node 8. The drawing of the mesh is achieved by the function *gra\_mnl.m* (Table 53).

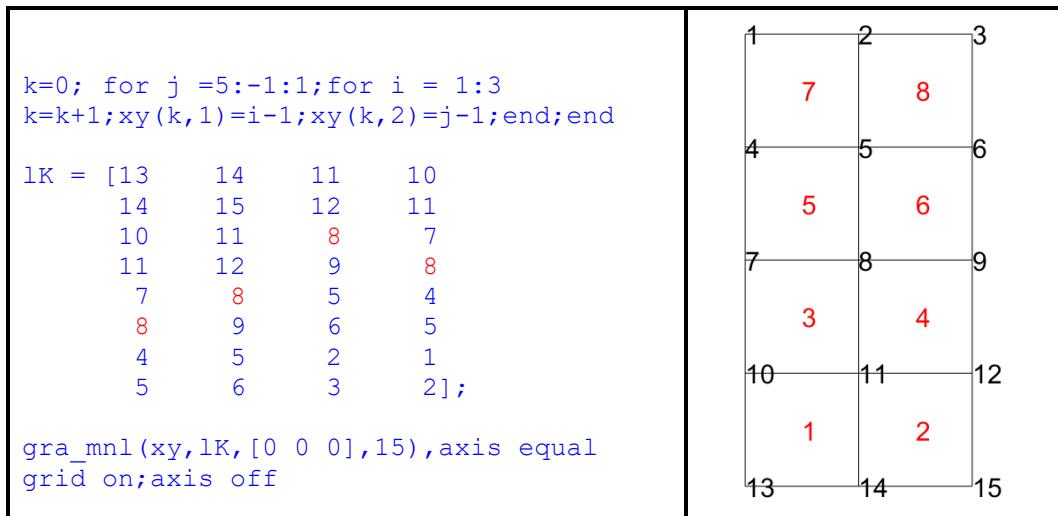


Figure 3: Mesh with nodes & elements labels

In the center ( $x = a/2$ ,  $y = b/2$ ) of the rectangular element we obtain:

$$\begin{aligned}
 \frac{\partial \tau_i}{\partial x} &= \frac{1}{2a} \left( (\mathbf{T}_{i2} + \mathbf{T}_{i3}) - (\mathbf{T}_{i1} + \mathbf{T}_{i4}) \right) \\
 \frac{\partial \tau_i}{\partial y} &= \frac{1}{2b} \left( (\mathbf{T}_{i3} + \mathbf{T}_{i4}) - (\mathbf{T}_{i1} + \mathbf{T}_{i2}) \right)
 \end{aligned} \quad (7)$$

The gradients are sensitive to the dimension of the element because they depend on the metrics. The heat flow is deduced from the gradient by the Fourier law.

$$\begin{bmatrix} q_{xi} \\ q_{yi} \end{bmatrix} = -k_i \begin{bmatrix} \frac{\partial \tau_i}{\partial x} \\ \frac{\partial \tau_i}{\partial y} \end{bmatrix} \quad (8)$$

The heat flow ( $Wm^{-2}$ ) is in the opposite direction of the gradient and, in homogeneous isotropic mediums, proportional to the conductivity coefficient  $k_i (WK^{-1}m^{-1})$ .

If all the data: nodes, elements and fixations are explicitly defined, the solution program is very short: 18 lines. In this procedure, the fixed nodes must be grouped at the end of the list. The results are shown in *Figure 4*.

If Dirichlet boundary conditions are imposed, we have to compute the values of the not imposed nodal temperatures. To solve the system of equations, we have to split the temperature vector in two groups: the free unknown temperatures [ $T_1$ ] and the imposed ones [ $T_2$ ]. The system is then:

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} F_1 \\ R_2 \end{bmatrix} \quad (9)$$

The solution is:

$$[T_1] = [K_{11}]^{-1} ([F_1] - [K_{12}][T_2]) \quad (10)$$

In the next example, the imposed second member [ $F_1$ ] corresponding to von Neumann boundary conditions is equal to zero. The solution (10) is written explicitly in *line 12* of the compact procedure 1 (*Table 1*).

Compact Matlab <sup>©</sup> procedure <i>Fia_20230130.m</i>	
Compact procedure 1	
1	k = 0; <b>for</b> i=1:3; <b>for</b> j=1:4; k=k+1; xy(k,:)=[i-1,j-1]; <b>end</b> ; <b>end</b> ;
2	xy(13,:)=[3 0]; xy(14,:)=[3 1]; nn=14; % size(xy) = (nn x 2)
3	lK = [1 5 6 2; 2 6 7 3; 3 7 8 4; 5 9 10 6; 7 11 12 8; 9 13 14 10];
4	ne = 6; % size(lK) = (4 x ne)
5	fi = [11 12 13 14]; fT=[270 270 300 300]'; nf = 4; % size(fi) = (1 x nf)
6	K = zeros(nn,nn); % Global conductivity matrix K initialization & assembly
7	Ke = [4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4]/6; % K mat. of a square
8	<b>for</b> n = 1:ne % Loop on ne elements
9	<b>for</b> i = 1:4
10	<b>for</b> j=1:4; K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+Ke(i,j); <b>end</b>
11	<b>end</b> ;
12	<b>end</b> % End of assembling the ne element matrices Ke in the global one: K
13	T = zeros(max(max(lK)),1); % Nodal temperatures vector
14	T(1:nn-nf)=K(1:nn-nf,1:nn-nf)\-K(1:nn-nf,nn-nf+1:nn)*fT; T(nn-nf+1:nn)=fT; % Isotherms of figure 4, left
15	figure
16	<b>for</b> i=1:ne; fill(xy(lK(i,:),1)',xy(lK(i,:),2)',T(lK(i,:))); hold on; <b>end</b>
17	colorbar
18	title(['Dissipation: ',num2str(T'*K*T/2,3), ' WK']); axis equal; axis off
19	disp(['Dissipation ..... : ',num2str(T'*K*T/2,3), ' WK'])
Compact procedure 2	
1	k = 0; <b>for</b> i=1:3; <b>for</b> j=1:4; k=k+1; xy(k,:)=[i-1,j-1]; <b>end</b> ; <b>end</b> ;
2	xy(13,:)=[3 0]; xy(14,:)=[3 1]; nn=14; % size(xy) = (nn x 2)
3	lK = [1 5 6 2; 2 6 7 3; 3 7 8 4; 5 9 10 6; 7 11 12 8; 9 13 14 10];
4	ne = 6; % size(lK) = (4 x ne)
5	fi = [11 12 13 14]; fT=[270 270 300 300]'; nf = 4; % size(fi) = (1 x nf)
6	K = zeros(nn,nn); % Global conductivity matrix K initialization & assembly
7	Ke = [4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4]/6; % K mat. of a square
8	<b>for</b> n = 1:ne % Loop on ne elements

```

9    for i = 1:4
10       for j=1:4; K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+Ke(i,j);end
11   end;
12 end % End of assembling the ne element matrices Ke in the global one: K
13 N = zeros(nf,nn);G=zeros(max(max(lK)),1,1);% Initializations
14 for i = 1:nf; N(i,fi(i))=1; G(nn+i)=fT(i) ;end % Building the equ. system
15 A = [K N';N zeros(nf,nf)];B = A\G;T = B(1:nn); % Solution
16 figure;colormap(gra_cob); % Isotherms
17 for i=1:ne;fill(xy(lK(i,:),1)',xy(lK(i,:),2)',T(lK(i,:)));hold on;end
18 colorbar;
19 title(['Dissipation: ',num2str(T'*K*T/2,3),' WK']);axis equal;axis off
20 disp(['Dissipation ..... : ',num2str(T'*K*T/2,3),' WK'])

```

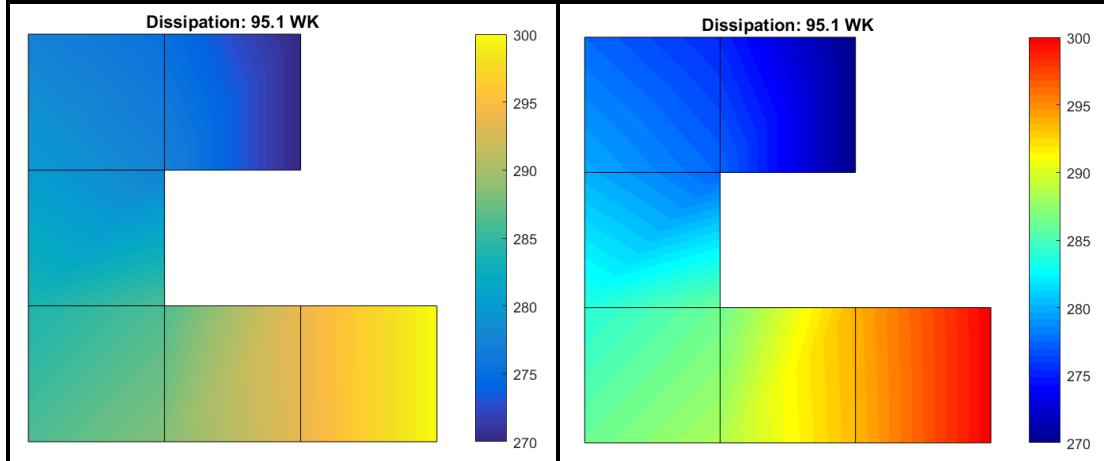
*Table 1: Compact Matlab<sup>®</sup> procedure Fia\_20230130.m*

The dissipation is equal to 95.1 *W*. With the compact procedure 1, we obtain the drawing of *Figure 4, left*, while, with the compact procedure 2, we obtain the result of *Figure 4, right*. To provide this color map, we use the function *gra\_cob.m* (*Table 59*). With this color bar, the quality of the output is significantly improved.

The conclusion of this first approach is twofold:

1. The only needed input are the node coordinates, the localization matrix and the fixed temperatures.
2. The procedure is quite rigid if we need to modify the Dirichlet boundary conditions.

It will then be welcome to ensure an easy way to introduce the Dirichlet boundary conditions by solving the equation system with a new method (procedure 2) presented in the next section.

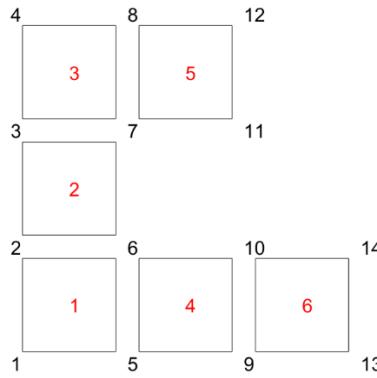


*Figure 4: Result of the compact procedures 1 & 2 with different color maps*

To obtain a visualization of the mesh, including node and element label, it is sufficient to use, after running the procedure, the *Matlab<sup>®</sup>* instruction:

```
gra_mnl(xy,lK,[0 0 0],15),axis equal;grid on;axis off
```

The first argument: *xy* is the matrix of nodal coordinates,. The second argument: *lK* is the localization matrix of the elements. The third argument will be described later. The last argument of the function is the font size specification, here, 15.



*Figure 5: Finite element mesh with nodes and elements labels*

## 1.2 Innovative handling of the Dirichlet boundary conditions

The Matlab<sup>©</sup> procedure of *Table 2* exhibits a more flexible finite element program. In the first example, we impose constant temperatures on the nodes located on both horizontal sides. The size of the domain is  $l \times h \times e = 1 \text{ m} \times 2 \text{ m} \times 1 \text{ m}$ . To run this example, we write that the concerned nodal temperatures are equal to some value. We have thus as many restraints as imposed temperatures which are added to the system with Lagrange multipliers.

The system of constraints is:

$$[N] [T] = [\bar{T}] \quad (11)$$

In this expression  $[T]$  is the uni-column matrix (or vector) of nodal temperatures and  $[N]$  a matrix with as many lines as fixed nodes and the same number of columns as the size of  $[T]$ . Each line contains only zero and one in the column corresponding to the fixed node. With the matrix of constraints we define a new term (blue) in the functional thanks to the Lagrange multipliers  $[\Lambda]$ :

$$\langle I(T) = [T]^T [K][T] + [\Lambda]^T ([N][T] - [\bar{T}]) \rangle \quad (12)$$

The Euler equations are obtained by derivations of the functional with respect to  $[\Lambda]$  and  $[T]$ :

1. derivative with respect to the Lagrange multipliers is restoring equation (9)
2. derivative with respect to  $[T]$ :

$$[K][T] + [N]^T [\Lambda] = [0] \quad (13)$$

The second member of this equation is equal to zero because the von Neumann boundary conditions are not considered (there are not nodal loads).

Combining (9) and (11), yields to the system:

$$\begin{bmatrix} K & N^T \\ N & 0 \end{bmatrix} \begin{bmatrix} T \\ \Lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{T} \end{bmatrix} \rightarrow [A][B] = [G] \quad (14)$$

The uni-column matrix  $[G]$  contains a sequence of the same size as  $[T]$  with zero values, followed by a sequence of the values of the imposed temperatures  $[\bar{T}]$ . The unknown vector  $[B]$  contains the full set of nodal temperature including the imposed ones followed by the Lagrange multipliers which represent the generalized heat flows through the fixed nodes. Unlike the matrix  $[K]$ , the matrix  $[A]$  is nonsingular if the constraints are independent.

Matlab <sup>©</sup> procedure <i>Fiammetta_33_20230130.m</i>	
1	r=6 ;nx=5*r;ny=2*nx;ii=0;nn=(nx+1)*(ny+1);xy=zeros(nn,2);CPU=tic;
2	for j = ny + 1:-1:1 % Geometry: definition of nodal coordinates
3	for i=1:nx+1; ii = ii+1; xy(ii,1) = i-1; xy(ii,2) = j-1;end
4	end % ;if nn<20;disp([(1:nn)' xy]);end % Display nodal coordinates
5	disp(['Number of nodes : ',num2str(nn)]) % End geometry definition
6	ne = nx*ny; lK = zeros(ne,4); m = 0; % Topology: mesh definition
7	for j = 1:ny
8	for i = 1:nx
9	m = m+1;lK(m,1) = nn-nx-1+i-(j-1)*(nx+1); lK(m,2) = lK(m,1) + 1;
10	lK(m,3) = lK(m,2)-nx-1; lK(m,4) = lK(m,3)-1;
11	end
12	end;disp(['Number of elements : ',num2str(ne)]) % End topology definition
13	n1 = (nx/r) + 1;nf = 2*n1; % Dirichlet boundary conditions
14	lfi = [nn:-1:nn-n1+1 1:n1];fT=[ones(1,n1)*270 ones(1,n1)*320];
15	disp(['Numb. of fix. temp. : ',num2str(nf)]);
16	gh = zeros(nn,1); % von Neumann boundary conditions
17	co = ones(ne,1);% co=mat_cok(nx,ny); % Element thickness * conductivity
18	K = zeros(nn,nn);% Global conductivity matrix K initialization & assembly
19	for n = 1:ne % Loop on ne elements
20	Ke = co(n)*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4]/6;

```

21 for i = 1:4;for j=1:4;K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+Ke (i,j);
22 end;end;           % End of assembling the ne conductivity matrices Ke
23 end
24 N      = zeros(nf,nn);G=[gh;zeros(nf,1)];% Initializing linear constraints
25 for i = 1:nf;N(i,lfi(i))=1;G(nn+i)=fT(i);end   % Building the equ. system
26 A      = [K N';N=zeros(nf,nf)];B = A\G;T = B(1:nn);          % Solution
27 figure;colormap(gra_cob)                                % Isotherms
28 for i=1:ne;fill(xy(lK(i,:),1)',xy(lK(i,:),2)',T(lK(i,:)));hold on;end
29 colorbar;axis equal;axis off
30 disp(['Dissipation ..... : ',num2str(T'*K*T/2,3),' WK'])
31 disp(['CPU ..... : ',num2str(toc(CPU),3),' sec. '])
32 disp(['Int work - ext work : ',num2str((dot(gh,T)-dot(G(nn+1:nn+nf),
33     B(nn+1:nn+nf))/2,3),' WK')])
34
35 % for i=1:ne;fill(xy(lK(i,:),1)',xy(lK(i,:),2)',T(lK(i,:)));hold on;end
36 % gra_ist(nx,ny,T,xy,[min(T) 2 max(T)])

```

Table 2: Matlab<sup>®</sup> procedure Fiammetta\_33\_20230130.m - Conduction problems

With the parameter  $r = 1$ , (line 1 of Table 2), the results displayed in Figure 6, left correspond to the exact solution, they do not depend on the mesh refinement. Enabling the use of function *gra\_cob.m* (Table 59) in line 27, we get better colors (Figure 6, center). Replacing line 28 by the enabled line 35, the use of the function *gra\_ist.m* (Table 51) instead of the Matlab<sup>®</sup> function *fill*, provides the graphics with isotherm lines (Figure 6, right).

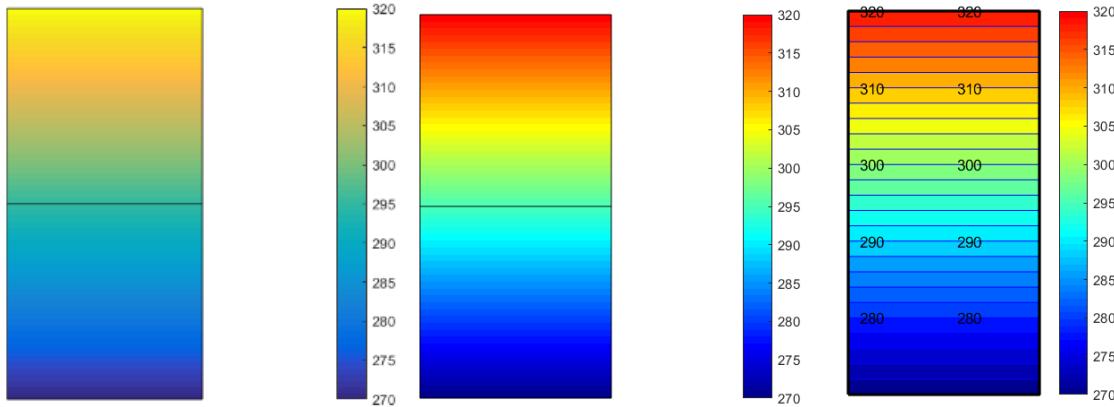


Figure 6: Example with imposed temperatures producing a vertical gradient

If the difference of temperatures is equal to 50  $K$  and if the temperatures are constant on both horizontal sides, the quantities of incoming heat on the top side and outgoing one in the base are identical and given by the heat flow integrated on a horizontal section:

$$le k \frac{\Delta T}{\Delta y} = k \frac{50}{2} = 25 \text{ W} \quad (15)$$

Added at the end of the procedure, the following line provides the nodal generalized heat flows associated to the fixed temperatures on the horizontal sides. The sum of the top flows is equal to the sum of the bottom ones and their modules are both equal to 25  $W$ .

```
hl=B(nn+1:nn+nf);disp(['Heat input & output : ',num2str(hl),' W'])
```

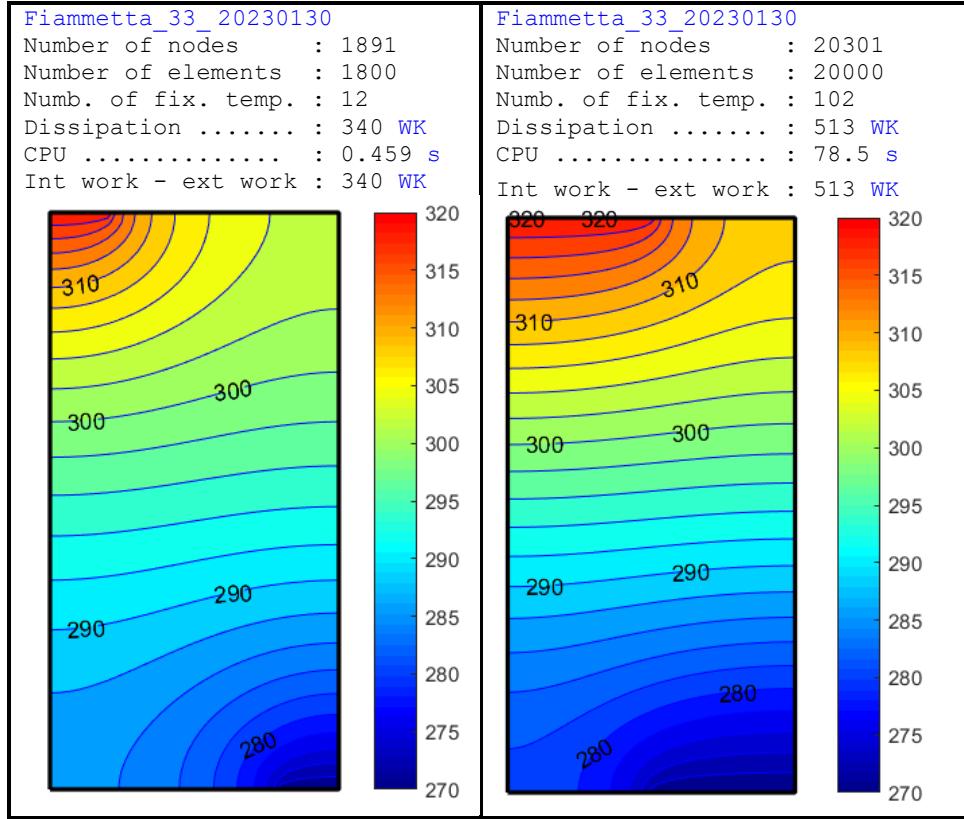
The input heat flows are the Lagrange multipliers of the top side, for instance the  $nf$  first terms of  $[A]$ ,  $nf$  is the number of fixations on a horizontal side,  $nf = nx/r$ .

In Figure 6, on the horizontal sides, each inner node links two element edges, but the outer ones only connect one. At the extremities of the sides, the second members are equal to half the others. Due to the homogeneity of the imposed temperatures, the nodal heat loads are equal to the total load computed in (9): 25  $W$  divided by the number of elements  $25/nx$   $W$  for inner nodes and half this value for the others:  $25/(2 nx)$   $W$ . If  $nx = 5*r$ , in Matlab<sup>®</sup> notation, the reactive generalized heat flows of the top horizontal side are given by:

```
disp(['hf = ',num2str(B(nn+1:nn+nf/2))]) → hf = 2.5 5 5 5 5 2.5 W
```

### 1.3 Dirichlet boundary conditions

With various definitions of variables *r* and *nx* in *line 1* of *Fiammetta\_33\_20230130.m*, we obtain the solutions shown in *Figure 7*.



*Figure 7: Imposed temp. on fragments of the horizontal faces ( $r = 6, nx = 5$  &  $r = 2, nx = 50$ )*

To perform tests on conduction, we use the procedure of *Table 2*, first in this simplified presentation, and later in its definitive form. The acronym **Fiammetta** means: **F**inite **E**lement **M**ethod and **I**soparametric **M**eshing **A**pplied to **T**ransient **T**hermal **A**nalysis.

In this chapter, the finite element model is limited to a rectangle composed of squares (*Figure 3*). The temperature is always expressed in Kelvin (*K*).

The Matlab<sup>©</sup> procedure of *Table 2* is providing one single output: the visualization of the domain with colored temperature levels (*r=3; nx=2\*r;*). The conductivity coefficients are specified in the vector *co* (*ne* components with *ne*, the number of elements) at *line 17* of the procedure. It is possible to define other distributions of conductivity (one per element).

The variables: *B*, *G*, *T* are uni-column matrices, while, *lf1* and *fT* are uni-line matrices. The drawings of *Figure 7* are obtained with a command calling *gra\_ist.m* (*Table 5I*), which is more efficient to represent isotherms than the original Matlab<sup>©</sup> command *fill*.

### 1.4 Neumann boundary conditions

To introduce the heat fluxes, we use the functional presented in the first lecture.

$$\langle I(\tau) = \int_{\Omega} \frac{1}{2} k \vec{\nabla} \tau \cdot \vec{\nabla} \tau d\Omega + \int_{S_2} \bar{q}_n \tau dS \rangle \quad \text{minimum} \quad (16)$$

Limiting the demonstration to one element edge, we can write that the second term of the above functional corresponds to the sum of products of generalized nodal heat flows  $g_i$  ( $\mathcal{W}$ ) by temperatures  $T_i$  ( $K$ ) and we can write it as follows:

$$\int_{S_{2\text{edge}}} \bar{q}_n \tau dS_{el} = g_1 T_1 + g_2 T_2 \quad (17)$$

We express the edge temperature in term of edge weight functions

$$\tau_{edge} = T_1 \left( 1 - \frac{x}{l} \right) + T_2 \frac{x}{l} \quad (18)$$

We can write the discretized functional:

$$\int_{S_{2\text{edge}}} \bar{q}_n \tau dS_{el} = \int_{S_{2\text{edge}}} \bar{q}_n \left( T_1 \left( 1 - \frac{x}{l} \right) + T_2 \frac{x}{l} \right) dS_{el} \quad (19)$$

In matrix form, we have:

$$\text{With : } [T] = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \text{ we have : } \int_{S_{2\text{edge}}} \bar{q}_n \tau dS_{el} = \int_{S_{2\text{edge}}} \bar{q}_n \begin{bmatrix} \left( 1 - \frac{x}{l} \right) & \frac{x}{l} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} dS_{el} \quad (20)$$

We can now get the nodal temperatures out of the integral:

$$\int_{S_{2\text{edge}}} \bar{q}_n \tau dS_{el} = \int_{S_{2\text{edge}}} \bar{q}_n \begin{bmatrix} \left( 1 - \frac{x}{l} \right) & \frac{x}{l} \end{bmatrix} dS_{el} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \quad (21)$$

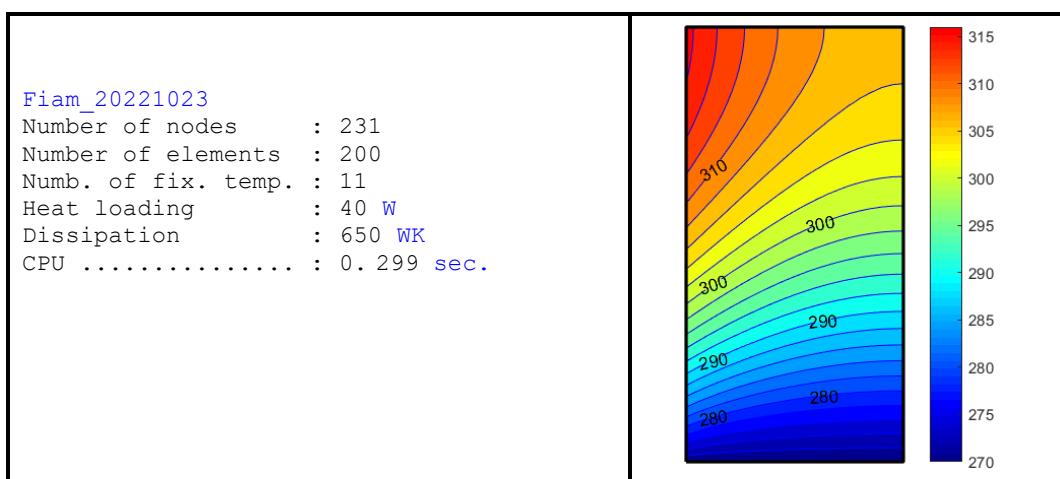
Finally, we can write the prescribed second member in matrix form:

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix}^T = \int_{S_{2\text{edge}}} \bar{q}_n \begin{bmatrix} \left( 1 - \frac{x}{l} \right) & \frac{x}{l} \end{bmatrix} dS_{el} \quad (22)$$

To run the example of *Figure 8*, the *lines 13* to *16* are replaced in *Fiammetta\_33\_20230130.m* (*Table 2*) by the instructions of *Table 3*

13	<code>nf = nx+1;lfi = nn:-1:nn-nf+1;fT=ones(1,nf)*270;</code>	% Dirichlet
14	<code>disp(['Numb. of fix. temp. : ',num2str(nf)]);</code>	
15	<code>gh = zeros(nn,1);gh(1) = 20/ny;gh(nx+2:(nx+1):(nx+1)*(ny+1)-nx)= 40/ny;...</code>	
16	<code>gh((nx+1)*(ny+1)-nx)=20/ny;</code>	% von Neumann boundary conditions
18	<code>disp(['Heat loading : ',num2str(sum(gh),3),' W'])</code>	

*Table 3: Instructions substituted to lines 13 to 16 in Fiammetta\_33\_20230130.m*



*Figure 8: Isocurves for the problem with prescribed heat flows*

We have obtained the general method to build the second members of the heat transfer equations corresponding to imposed heat flows. If the prescribed heat flow is constant on the edge, for an edge of length  $l$  and a thickness  $e$ , we obtain:

$$F_1 = \bar{q} \frac{el}{2} \quad ; \quad F_2 = \bar{q} \frac{el}{2} \quad (23)$$

## 1.5 Explanation of the Matlab<sup>©</sup> procedure *Fiammetta\_33\_20230130.m*

The Matlab<sup>©</sup> procedure *Fiammetta\_33\_20230130.m* is listed in *Table 2*.

**Line 1** : input data.

The variables  $nx$  and  $ny$  define the number of elements in the horizontal and vertical directions. The variable  $nx$  is imposed as a multiple of  $r$ .

**Lines 2 - 5** : are providing the grid of  $(nx + 1) (ny + 1)$  nodes.

**Lines 6 - 12** : localization matrix  $LK$  of the  $ne = nx \times ny$  elements.

**Lines 13 - 15** : Dirichlet boundary conditions: data for fixed temperatures.

In the examples proposed in this chapter, we fix some nodal temperatures on the horizontal sides, starting from opposite corners: left on the top, right in the bottom. The number  $nf$  of fixed temperatures is given by the relation:  $nf = 2 * ((nx / r) + 1)$ . Due to the definition of  $nx$ ,  $nx / r$  is an integer. As a consequence, it is easy to impose the proportion of fixed nodes on the horizontal sides.

**Line 16** : von Neumann bound. conditions,  $nn$  terms of the 2<sup>d</sup> member may be imposed.

**Line 17** : Material properties: product of thickness in  $m$  by conductivity in  $WK^{-1}m^{-1}$ .

**Lines 18 - 23** : Creation of the element conductivity of a square and assembly of the global conductivity matrix. The external loop is performed on the  $ne$  elements and the two internal loops are performed on the lines and columns of the element conductivity matrices. Each term  $(i, j)$  of element  $n$  is located at  $(LK(n, i), LK(n, j))$  in the global  $K$  matrix according to the  $LK$  matrix computed in the sequence of *lines 6 - 12*.

**Line 20** : conductivity matrix of a square element (*Figure 2*)

$$Ke = [4 -1 -2 -1; -1 4 -1 -2; -2 -1 4 -1; -1 -2 -1 4] / 6;$$

This matrix has to be multiplied by the thickness and the conductivity coefficient computed in *line 17* (vector  $co$ ). This matrix is expressed in  $W$ .

**Lines 24 - 26** : Solution of the system of equations involving the linear constraints. Its dimension is  $nn + nf$ . The uni-column matrix  $G$  contains the  $nn$  loads (here equal to zero) and the  $nf$  imposed temperatures  $fT$ .  $B$  is the uni-column matrix of the  $nn$  unknown nodal temperatures  $T$  and the  $nf$  reactive flows.

**Line 27** : Definition of a color bar with the Matlab function *gra\_cob.m* (*Table 59*).

**Line 28 - 33** : Drawing sequence using *gra\_ist.m* or *fill* sequence. Display of global results and processing time.

## 1.6 Exercises

1: The objective of this exercise is to manage finite element meshes and Dirichlet boundary conditions. The domain formed by the set of elements must be in one piece and the nodal coordinates must all be present in the finite elements' localization matrix. For this first test all the elements must be square.

For the explicit example shown below, the 3 first lines of the compact procedure 2 of *Table 1* are replaced by the 3 first lines of *Figure 9*. To be able to see the mesh and the node and element labels, we can use the function *gra\_mnl.m* (*Table 53*) that will be defined later. The Matlab<sup>©</sup> instruction used to produce this drawing is the last one of *Figure 9*.

```

xy = [0 3; 1 3; 2 3; 3 0; 2; 1 2; 2 2; 3 2; 0 1; 1 1; 2 1; 3 1; 0 0; 1 0; 2 0; 3 0]; nn = 16;
LK = [5 6 2 1; 6 7 3 2; 7 8 4 3; 10 11 7 6; 13 14 10 9; 14 15 11 10; 15 16 12 11]; ne = 7;
fi = [1 5 4 8 9 13 12 16]; fT = [270 270 270 270 300 300 300 300];
nf = 8;
K = zeros(nn, nn); % Global conductivity matrix K initialization & assembly
Ke = [4 -1 -2 -1; -1 4 -1 -2; -2 -1 4 -1; -1 -2 -1 4] / 6; % K mat. of a square
for n = 1:ne
    % Loop on ne elements

```

```

for i = 1:4
    for j=1:4; K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+Ke(i,j);end
end;
% End of assembling the ne element matrices Ke in the global one: K
N      = zeros(nf,nn);G=zeros(max(max(lK)),1,1);% Initializations
for i = 1:nf; N(i,fi(i))=1; G(nn+i)=fT(i) ;end % Building the equ. system
A     = [K N';N zeros(nf,nf)];B = A\G;T = B(1:nn);           % Solution
figure;colormap(gra_cob);
for i=1:ne;fill(xy(lK(i,:),1)',xy(lK(i,:),2)',T(lK(i,:)));hold on;end
colorbar;
title(['Dissipation: ',num2str(T'*K*T/2,3),' WK']);axis equal;axis off
gra_mnl(xy,lK,[ 0 0 0],15); axis equal; axis off

```

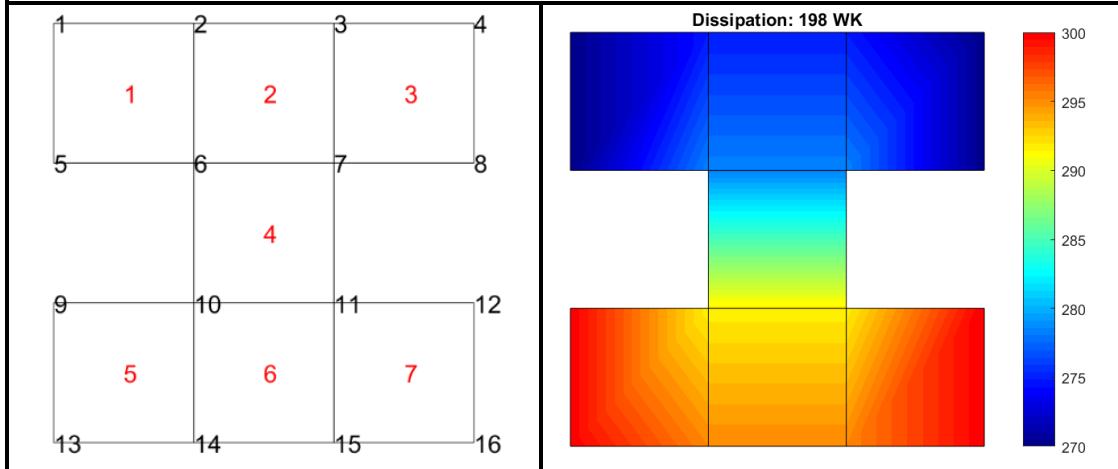


Figure 9: Exercise 1: mesh & isotherms

The exercise consists in adding or removing elements or fixed nodal temperature and to check that the results are consistent with the expected ones.

2: The fluxes are imposed on the four sides of a rectangle and a temperature is imposed on a single node. What happens if the fluxes are balanced (the flow coming in on the left and above is equal to the flow coming out from the right and from below)? What happens if the fluxes are no longer in equilibrium (for example: incoming flow on all four sides)? In both cases, vary the imposed temperature, and observe the generalized fluxes around the area. For this test, the procedure [Fiammetta\\_33\\_20230130.m](#) ([Table 2](#)) has to be used.

## Tutorial II: Convection

In this chapter, we consider that the conductive model may be immersed in a fluid, either liquid or gaz. Heat exchanges are assumed to act globally. In the previous chapter, the conductive medium was always limited by a perfectly reflecting surface (mirror) or adiabatic surface, except where temperatures are imposed.

### 2.1 Formulation of the convection

The partial differential equations of the convective heat transfer problem come from the stationarity conditions of the functional:

$$\langle I(\tau) = \int_{\Omega} \frac{1}{2} k \vec{\nabla} \tau \cdot \vec{\nabla} \tau d\Omega + \frac{1}{2} \int_{S_3} h (\tau - \tau_f)^2 dS + \int_{S_2} \bar{q}_n \tau dS \rangle \text{ minimum} \quad (24)$$

The Rayleigh Ritz procedure is the same as in the conduction problem, so we can directly examine how to compute the “conductivity” matrices of the convective elements.

$$\text{Functional at element level: } I_{el} = \frac{1}{2} \int_{S_3} h (\tau - \tau_f)^2 dS \quad (25)$$

In (24) and (25), the quantity  $h$  represents the convection coefficient. The fluid temperature  $\tau_f$  is assumed uniform. We study an element edge which is a line segment of length  $L$ . The edge temperature is discretized as follows:

$$\tau = T_0 \left(1 - \frac{x}{L}\right) + T_1 \frac{x}{L} \quad (26)$$

In this expression  $x$  varies between 0 and  $L$ . Replacing in the functional (25), we obtain:

$$\begin{aligned} I_{el} &= \frac{1}{2} \int_0^L h \left( \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} - t_f \right)^2 dx \\ &= \frac{1}{2} h \int_0^L \left\{ [T]^T \begin{bmatrix} 1 - \frac{x}{L} \\ \frac{x}{L} \end{bmatrix} \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} [T] - 2 \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} [T] t_f + t_f^2 \right\} dx \end{aligned} \quad (27)$$

With a new definition of the nodal temperatures vector including the fluid temperature, we obtain with the notation  $t_f = T_f$ , where we assimilate the fluid temperature to that of a virtual node<sup>1</sup>:

$$[T_{el}] = \begin{bmatrix} T_0 & T_1 & T_f \end{bmatrix}^T \quad (28)$$

$[T_{el}]$  is the vector containing the set of nodal temperatures related to one element. Replacing in (25), we obtain:

$$I_{el} = \frac{1}{2} h T_{el}^T \left( \int_0^L \begin{bmatrix} 1 - \frac{x}{L} \\ \frac{x}{L} \\ -1 \end{bmatrix} \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} & -1 \end{bmatrix} dx \right) T_{el} \quad (29)$$

Developing the expression:

$$I_{el} = \frac{1}{2} h T_{el}^T \int_0^L \begin{bmatrix} \left(1 - \frac{x}{L}\right)^2 & \left(1 - \frac{x}{L}\right)\frac{x}{L} & -\left(1 - \frac{x}{L}\right) \\ \left(1 - \frac{x}{L}\right)\frac{x}{L} & \left(\frac{x}{L}\right)^2 & -\frac{x}{L} \\ -\left(1 - \frac{x}{L}\right) & -\frac{x}{L} & 1 \end{bmatrix} dx T_{el} \quad (30)$$

After integrating and including the thickness  $e$  to ensure the coherence of units, we transform the functional (30) into an algebraic function of the nodal temperatures:

---

<sup>1</sup> The virtual nodes are not related to a position, they do not have any coordinate. However, to represent them like in [Figure 11](#), we give them an arbitrary position, only for the drawing.

$$I_{el} = \frac{1}{2} h e T_{el}^T \begin{bmatrix} \int_0^L \left(1 - \frac{x}{L}\right)^2 dx & \int_0^L \left(1 - \frac{x}{L}\right) \frac{x}{L} dx & -\int_0^L \left(1 - \frac{x}{L}\right) dx \\ \int_0^L \left(1 - \frac{x}{L}\right) \frac{x}{L} dx & \int_0^L \left(\frac{x}{L}\right)^2 dx & -\int_0^L \frac{x}{L} dx \\ -\int_0^L \left(1 - \frac{x}{L}\right) dx & -\int_0^L \frac{x}{L} dx & \int_0^L dx \end{bmatrix} T_{el} \quad (31)$$

The integrals present in the discretized functional are easily computed:

$$\int_0^L \left(1 - \frac{x}{L}\right)^2 dx = \int_0^L \left(1 - 2\frac{x}{L} + \frac{x^2}{L^2}\right) dx = L - L + \frac{L}{3} = \frac{L}{3} \quad (32)$$

$$\int_0^L \left(1 - \frac{x}{L}\right) \frac{x}{L} dx = \int_0^L \left(\frac{x}{L} - \frac{x^2}{L^2}\right) dx = \frac{L}{2} - \frac{L}{3} = \frac{L}{6} \quad (33)$$

$$-\int_0^L \left(1 - \frac{x}{L}\right) dx = -L + \frac{L}{2} = -\frac{L}{2} \quad (34)$$

The final expression of the integral related to convection is then:

$$I_{el} = \frac{1}{2} h e L T_{el}^T \begin{bmatrix} \frac{1}{3} & \frac{1}{6} & -\frac{1}{2} \\ \frac{1}{6} & \frac{1}{3} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix} T_{el} = \frac{1}{2} T_{el}^T K_h T_{el} \quad (35)$$

From this expression, we deduce the conductivity matrix for convection, so called because it is expressed in  $WK^{-1}$ .

$$K_h = \frac{ehL}{6} \begin{bmatrix} 2 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & 6 \end{bmatrix} \quad (36)$$

As well as the pure conduction matrix, this matrix is singular. It means that, with or without convection, it is necessary to fix at least one node (real or virtual) in order to make the conductivity matrix positive definite. To solve a problem including conduction and convection, we have to compute two kinds of conductivity matrices. We call the first one  $K_k$  and the second one  $K_h$ . Later, both matrices have to be added. The second one carries additional degrees of freedom corresponding to the virtual convective nodes.

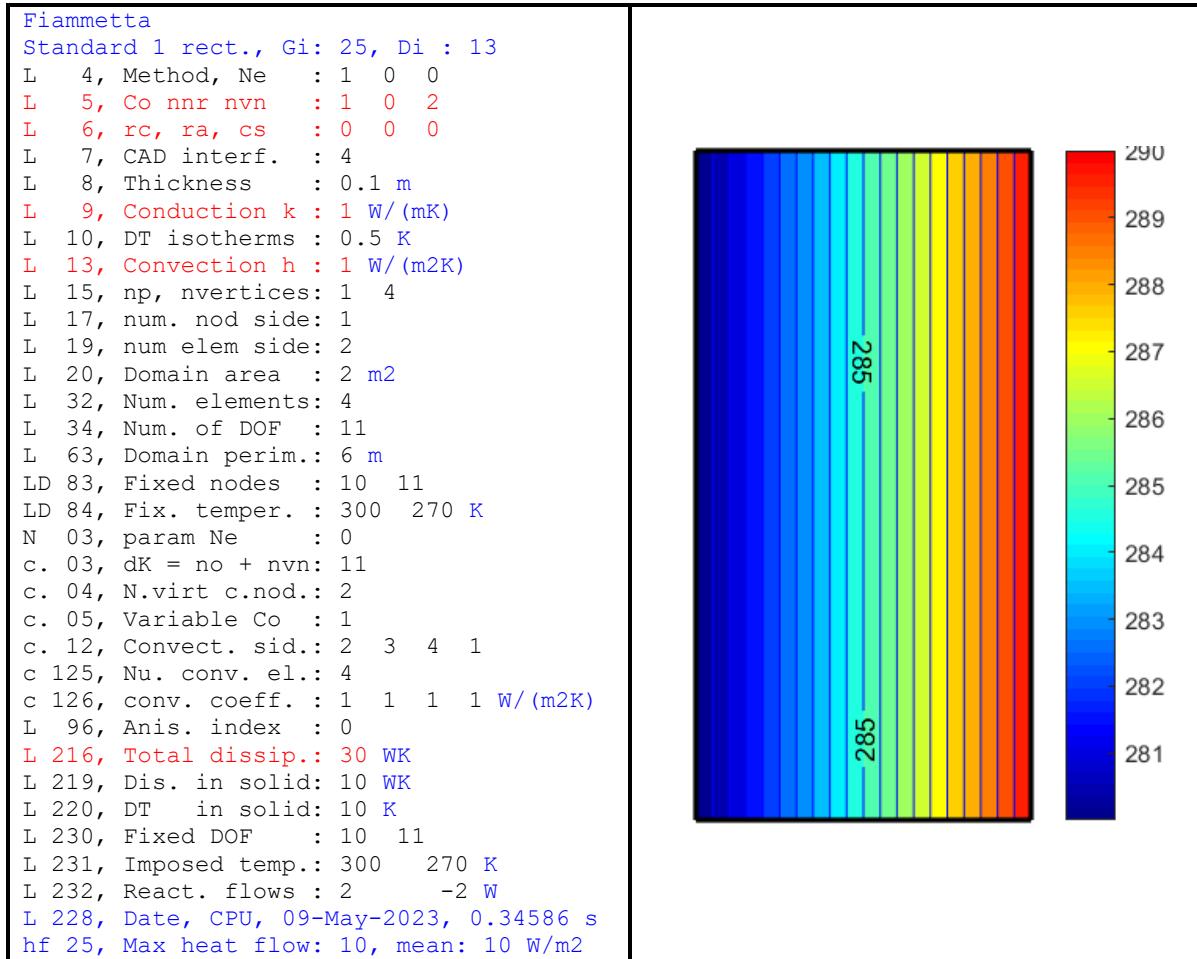
$$K = K_k + K_h \quad (37)$$

The convection virtual nodes may be free or fixed. The convection matrix defined in (34) is computed in the function `fem_Kcv.m` (*Table 45*). The input is the matrix `xyz` of node coordinates, the localization matrix `lc` of the convective element and the coefficient `hh`, product of the thickness by the convection coefficient.

The output is the 3 x 3 matrix of “*conduction – convection*” ( $WK^{-1}$ ). The length of a convection element is equal to `L(m)`, its thickness to `th(m)`, and the convection coefficient is `h(Wm-2K-1)`. The nodal sequence of an element starts with the two real nodes pertaining to the mesh and ends with the virtual one related to convection.

## 2.2 Two convective and two adiabatic faces

In the case of two convective and two adiabatic faces, we want to know what happens if the values of the convective and conductive coefficients are significantly modified. The markers used in the format of the displayed output are explained in [Table 4](#).



*Figure 10: Isotherms orthogonal to both adiabatic boundaries. Biot number = 1*

Matlab <sup>®</sup> function or procedure	Displayed output format, <i>n</i> is the line from which the command is issued
<i>Fiammetta.m</i>	['L n,
<i>cad_con.m</i>	['c. n,
<i>cad_Dir.m</i>	['LD n,
<i>cad_Neu.m</i>	['N n,
<i>fem_smd.m</i>	['sd n,
<i>fem_smt.m</i>	['st n,
<i>fem_smq.m</i>	['sq n,
<i>fem_smc.m</i>	['sc n,
<i>gra_ahf.m</i>	['hf 23,
<i>gra_chf</i>	['ch 04
<i>gra_atg.m</i>	['g 18,
<i>mat_cok.m</i>	['Lm n,

*Table 4: Format of displayed output*

[Figure 11](#) shows on the left the exploded view of the mesh with nodes and elements labels on the right, we see the visualization of the internal heat flows (in red) and the input convective heat flows (in green which are scaled with a coefficient equal to the mean internal heat flow (here, 10  $\text{Wm}^{-2}$ ). In the example of [Figure 11](#) (right), we observe that the lengths of the red arrows are equal to that of the green ones.

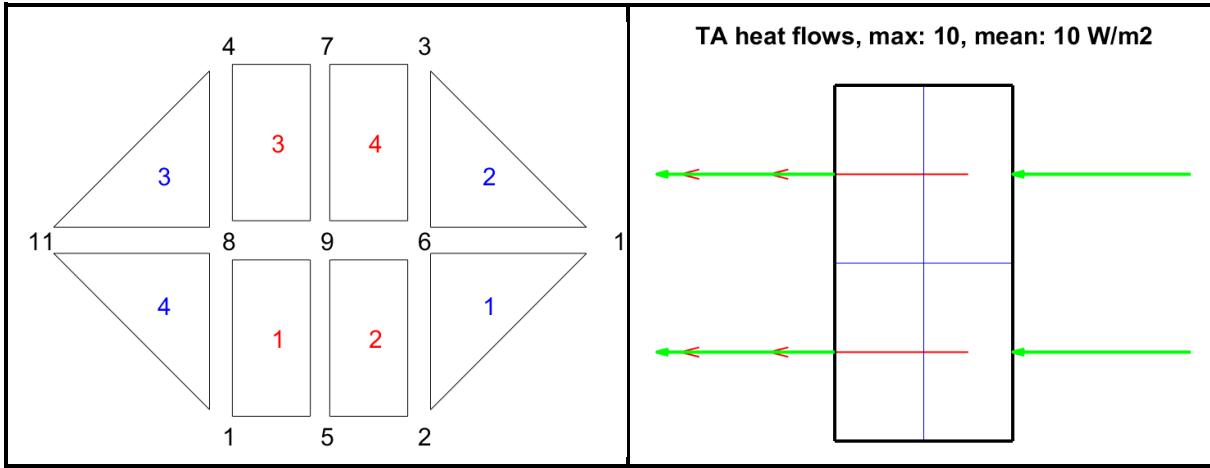


Figure 11: Nodes and elements numbering: heat flows with convective boundary conditions

It is proposed to express the difference of the fluid and the surface temperature as a function of the adimensional variable  $\beta = wh/k$  ( $w$  being the width of the domain,  $h$  and  $k$  respectively the convection and conduction coefficients) and to compare with the finite element model result. In this application, there is only one available characteristic length:  $w$ , which corresponds to the width (horizontal dimension) of the meshed domain.

This example deals with a very simple problem: evaluation of the temperature field in a wall submitted to convective heat transfers on both vertical sides. The horizontal sides are adiabatic. The solution is easily obtained explicitly. Let assume that the temperatures are defined as follows, from left to right:  $[t_0 \ t_1 \ t_2 \ t_3]$ . These variables correspond to the temperature  $t_0$  of the left virtual node, the surface temperature  $t_1$  of the left side, the surface temperature  $t_2$  of the right side and the temperature  $t_3$  of the right virtual node. Let assume that the convective coefficient is  $h$ , the conductive one  $k$ , the horizontal dimension of the domain  $w$  and the thickness,  $e$ . The continuity of the heat flux from left to right imposes the conditions:

$$\begin{array}{|c|c|c|} \hline t_0 < t_1 & \text{Conductive zone} & t_2 < t_3 \\ \hline \end{array} \quad q_x = h(t_0 - t_1) = k \frac{(t_1 - t_2)}{w} = h(t_2 - t_3) \quad (38)$$

The parameter  $w$ , which represents the width of the domain can be used as characteristic length  $L$  in the adimensional **Biot number** definition

$$\beta = \frac{h w}{k}, \quad \beta (t_0 - t_1) = t_1 - t_2 = \beta (t_2 - t_3) \quad (39)$$

From the first relation, we deduce:

$$t_1 = \frac{\beta t_0 + t_2}{1 + \beta} \quad (40)$$

Now, we develop the second one:

$$\frac{\beta t_0 + t_2}{1 + \beta} - t_2 = \beta t_2 - \beta t_3 \quad (41)$$

We obtain:

$$t_2 = \frac{t_0 + (1 + \beta)t_3}{2 + \beta} \quad (42)$$

Replacing (41) in (39), we have:

$$t_1 = \frac{\beta t_0}{1+\beta} + \frac{t_0 + (1+\beta) t_3}{(2+\beta)(1+\beta)} = \frac{t_0}{1+\beta} \left( \beta + \frac{1}{(2+\beta)} \right) + \frac{t_3}{(2+\beta)} = \frac{(1+\beta) t_0 + t_3}{(2+\beta)} \quad (43)$$

We also deduce the temperature gap in the wall as a function of the total temperature gap:

$$(t_2 - t_1) = (t_3 - t_0) \frac{\beta}{2+\beta} \quad (44)$$

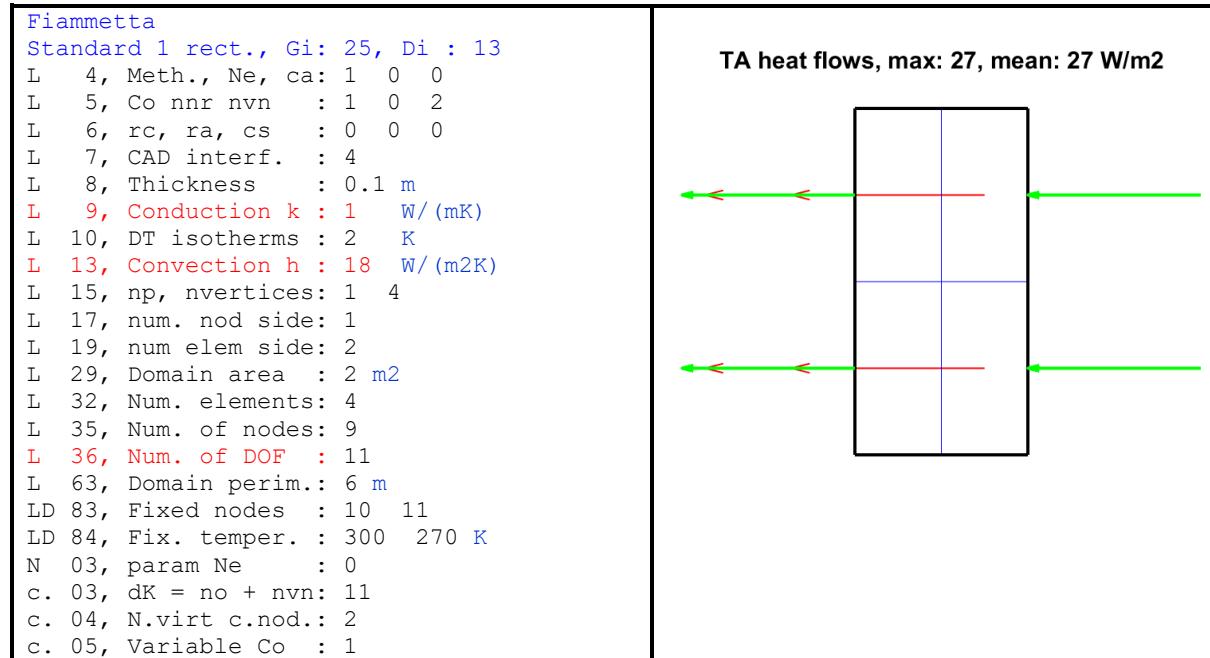
With  $t_0 = 270$  and  $t_3 = 300$ , we obtain both with formulas (42) and (43) and with the FEM simulation the results of *Table 5*.

Biot number $\beta$	$-q_x (\text{Wm}^{-2})$	$t_1 (\text{K})$	$t_2 (\text{K})$	$(t_1-t_0) (\text{K})$	$(t_2-t_1) (\text{K})$	$(t_3-t_2) (\text{K})$	Dissipation
.5	6	282	288	12	6	12	3.6 <i>WK</i>
1	10	280	290	10	10	10	10 <i>WK</i>
2	15	277.5	292.5	7.5	15	7.5	22.5 <i>WK</i>
18	27	271.5	298.5	1.5	27	1.5	72.9 <i>WK</i>
20	27.3	271.36	298.64	1.36	27.3	1.36	74.4 <i>WK</i>

*Table 5: Temperatures and heat flows as functions of Biot number*

Basically, we are working with four nodes: two virtual ones with indices 0 (left side) and 3 (right side) and two nodes situated on the surface of the conductive zone: one on the left side and two on the right one. Their corresponding temperature are:  $t_0$ ,  $t_1$ ,  $t_2$ ,  $t_3$ . For  $\beta = 1$ , the gap between the virtual convective nodes and their corresponding surface temperatures are the same as the gap in the conductive zone. As expected, higher is the Biot number, higher is the temperature gap inside the conductive zone. Because the bilinear quadrilateral finite element model is able to represent the exact solution, this analytical solution is obtained independently of the mesh refinement.

In the test of *Figure 12*, with  $\beta = 18$ , the temperature gradient in the conductive zone is equal to 27 *K/m*. The heat fluxes in the conductive and convective zones are the same: 27 *Wm<sup>-2</sup>*. The quantity of heat crossing the virtual nodes is the product of the heat flux by the section of the vertical side:  $27 \text{ Wm}^{-2} \times 0.2 \text{ m}^2 = 5.4 \text{ W}$ . The ratio between the temperature gap in the solid and the total gap is equal to  $27/30 * 100 = 90 \%$ .



```

c. 12, Convect. sid.: 2 3 4 1
c 179, Nu. conv. el.: 4
c 180, conv. coeff. : 18 18 18 18 W/ (m2K)
L 97, Anis. index : 0
L 216, Total dissip.: 81 WK
L 219, Dis. in solid: 72.9 WK
L 220, DT in solid: 27 K
L 230, Fixed DOF : 10 11
L 231, Imposed temp.: 300 270 K
L 232, React. flows : 5.4 -5.4 W
L 228, Date, CPU, 09-May-2023, 0.37034 s
hf 25, Max heat flow: 27, mean: 27 W/m2
ch 04, coef. red. dt: 27 W/m2
ch 21, Ave conv flow: -1 0 0 W/m2
g 18, Max temp grad: 27, mean: 27 K/m

```

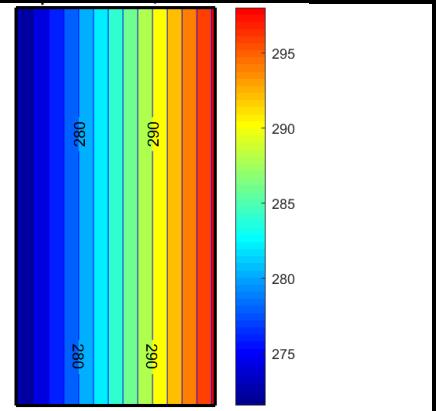


Figure 12: Example of heat transfer through a wall with a high Biot number  $\beta = 18$

## 2.3 Material anisotropy

The `mat_cok.m` function (Table 69) is subdivided into two sequences. The first one corresponds to horizontal strips ( $Ai = 1$ ), the second one to vertical strips ( $Ai = 3$ ). These sequences correspond to non-homogeneous materials. The coefficient applied to the conductivity coefficient is given by the variable `fa` defined in line 1 of *Fiammetta*.

We analyze a rectangular domain with part of the horizontal edges fixed either at values of 320 K or 270 K. On these edges, `nnc` nodes are fixed. To identify the *DOF* of a patch edge, we use an instruction giving the number label of the patch vertex, for instance `car_cao (1,3)`, which means vertex 3 of patch 1, and the characteristic of the edge given in a line of the matrix `bor` prevised by the reference to matrix `pbo`, which is giving for each patch the number of the line of `bor` where its sides are stored. (Sequence corresponding to *Di* = 3 in `cad_Dir.m`)

```

Fiammetta
Standard 1 rect., Gi: 9, Di : 3
L 4, Meth., Ne, ca: 1 0 0
L 5, Co nnr nnv : 0 0 0
L 6, rc, ra, cs : 0 0 0
L 7, CAD interf. : 4
L 8, Thickness : 1 m
L 9, Conduction k : 1 W/ (mK)
L 10, DT isotherms : 2 K
L 15, np, nvertices: 1 4
L 17, num. nod side: 16
L 19, num elem side: 17
L 29, Domain area : 2 m2
L 32, Num. elements: 289
L 34, Num. of DOF : 324
L 63, Domain perim.: 6 m
LD 23, N. fix. nodes: 10
N 03, param Ne : 0
L 96, Anis. index : 0
L 216, Total dissip.: 390 WK
L 219, Dis. in solid: 390 WK
L 220, DT in solid: 50 K
L 229, Date, CPU, 09-May-2023, 0.3631 s
hf 25, Max heat flow: 66, mean: 18 W/m2
g 18, Max temp grad: 66, mean: 18 K/m

```

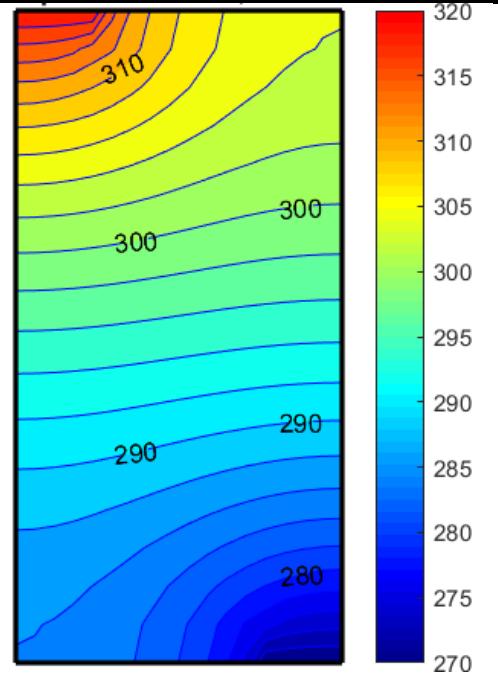


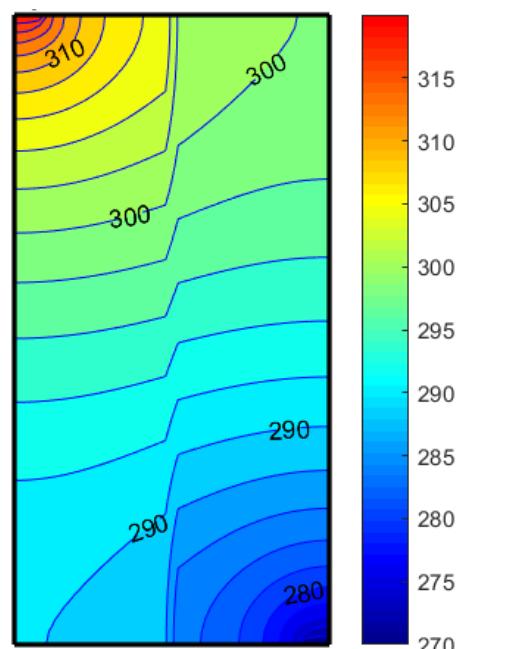
Figure 13: Isocurves in a rectangle with fixed DOF on horizontal edges, isotropic material

It is proposed to examine the effects of a modification of the conductivity coefficients. Let us try, for instance, to introduce a thermal bridge by increasing the conductivity along a vertical or a horizontal strip. This modification has to be performed by modifying the function `mat_cok.m` (Table 69). The elements are numbered from left to right and from bottom to top.

```

Fiammetta
Standard 1 rect., Gi: 9, Di : 3
L 4, Meth., Ne, ca: 1 0 0
L 5, Co nnr nnv : 0 0 0
L 6, rc, ra, cs : 0 0 0
L 7, CAD interf. : 4
L 8, Thickness : 1 m
L 9, Conduction k : 1 W/(mK)
L 10, DT isotherms : 2 K
L 15, np, nvertices: 1 4
L 17, num. nod side: 49
L 19, num elem side: 50
L 29, Domain area : 2 m2
L 32, Num. elements: 2500
L 34, Num. of DOF : 2601
L 63, Domain perim.: 6 m
LD 23, N. fix. nodes: 10
N 03, param Ne : 0
L 97, Anis. index : 3
Lm 48, k & coef*k : 1 0.1 W/(m K)
Lm 49, Thickn. ratio: 0.04
L 216, Total dissip.: 259 WK
L 219, Dis. in solid: 259 WK
L 220, DT in solid: 50 K
L 228, Date, CPU, 09-May-2023, 1.3461 s
hf 25, Max heat flow: 130, mean: 12 W/m2
g 18, Max temp grad: 130, mean: 14 K/m

```



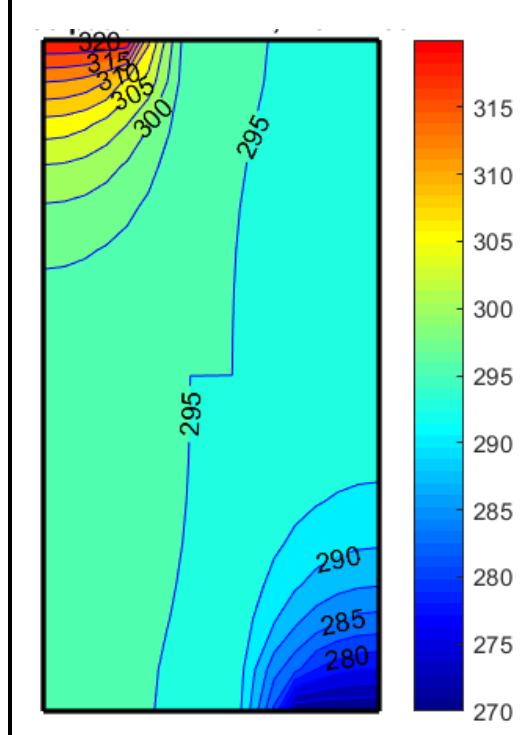
*Figure 14: Isocurves for material with vertical strip, 0.1 k (variable fa = .1)*

In the example shown in *Figure 15*, we have tested the function on a domain involving a vertical strip in which the ratio of conductivities is equal to 10000. In *Figure 16* we represent the vertical bridge when the number of elements per patch side is even (16) so that the width of the bridge is equal to two elements. At the right of the figure, diagrams of heat flows and temperature gradients are obtained thanks to the Matlab<sup>©</sup> procedure *P\_flgr.m* (*Table 64*), executed after the main procedure *Fiammetta.m*.

```

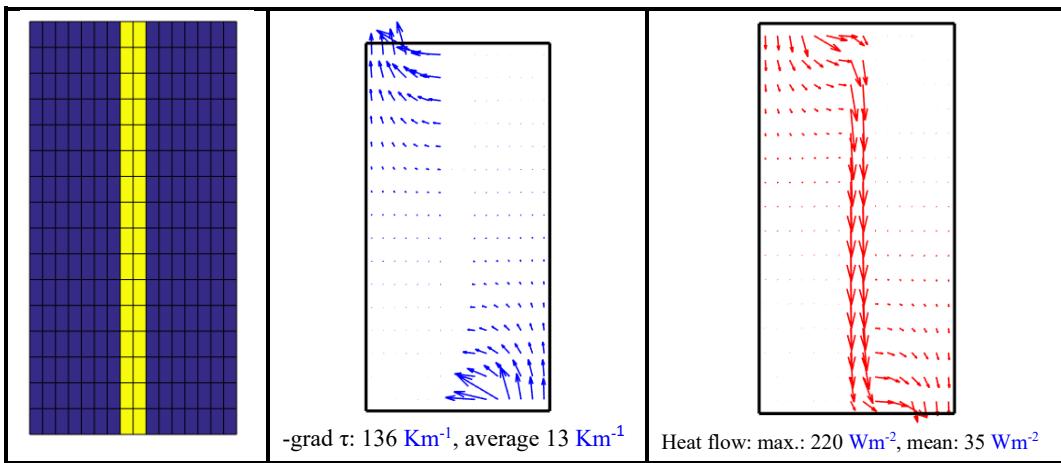
Fiammetta
Standard 1 rect., Gi: 9, Di : 3
L 4, Meth., Ne, ca: 1 0 0
L 5, Co nnr nnv : 0 0 0
L 6, rc, ra, cs : 0 0 0
L 7, CAD interf. : 4
L 8, Thickness : 1 m
L 9, Conduction k : 1 W/(mK)
L 10, DT isotherms : 2.5 K
L 15, np, nvertices: 1 4
L 17, num. nod side: 15
L 19, num elem side: 16
L 29, Domain area : 2 m2
L 32, Num. elements: 256
L 34, Num. of DOF : 289
L 63, Domain perim.: 6 m
LD 23, N. fix. nodes: 10
N 03, param Ne : 0
L 96, Anis. index : 3
Lm 48, k & coef*k : 1 10000 W/(m K)
Lm 49, Thickn. ratio: 0.125
L 216, Total dissip.: 741 WK
L 219, Dis. in solid: 741 WK
L 220, DT in solid: 50 K
L 228, Date, CPU, 09-May-2023, 0.27143 s
hf 23, Max heat flow: 220, mean: 35 W/m2
g 18, Max temp grad: 140, mean: 13 K/m

```



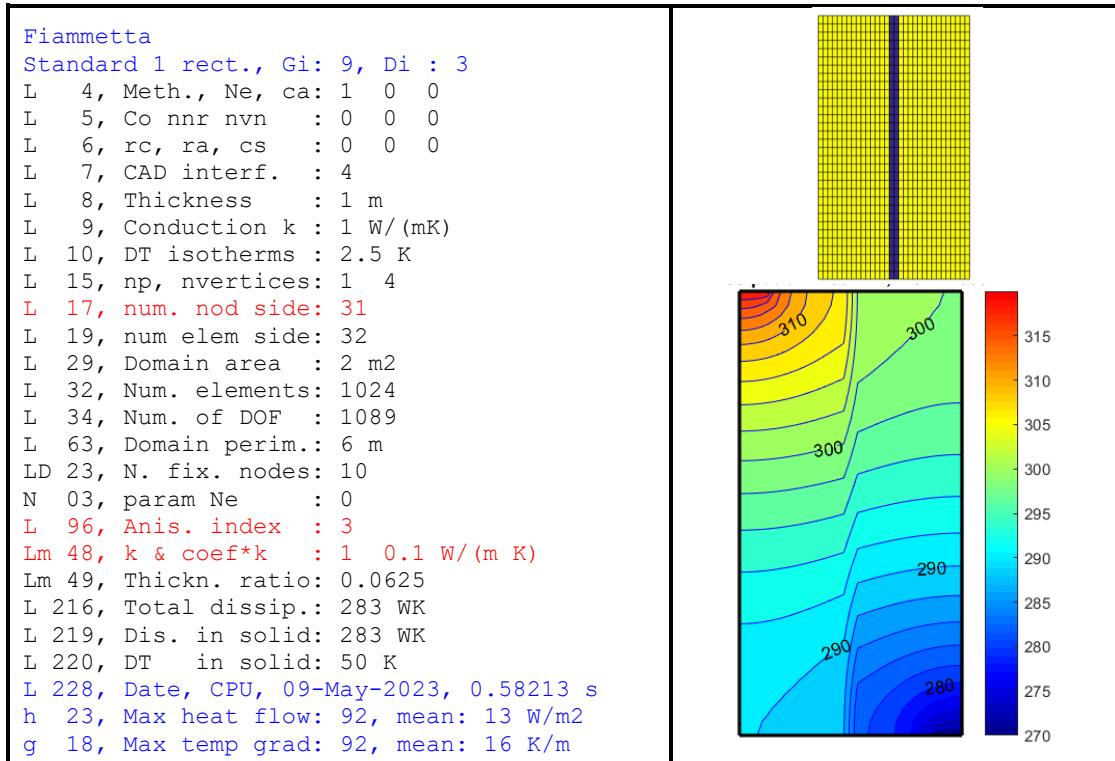
*Figure 15: Isocurves for the vertical strip 2 elements wide, 16 x 16 mesh*

The output of this program corresponds to both last lines of *Figure 15*, *Figure 17*, etc.



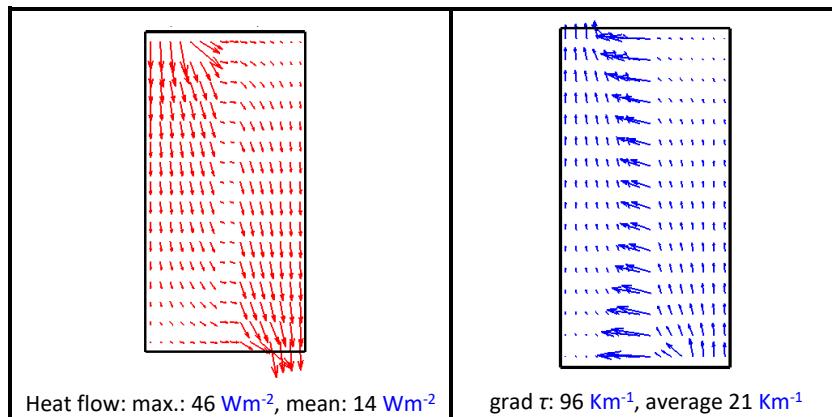
*Figure 16: Heat flows and temperature gradients for a vertical thermal bridge*

In *Figure 17*, we test the same mesh with a vertical strip of insulating material for which the conductivity is ten times smaller than the general one and the width of the bridge equal to two elements.

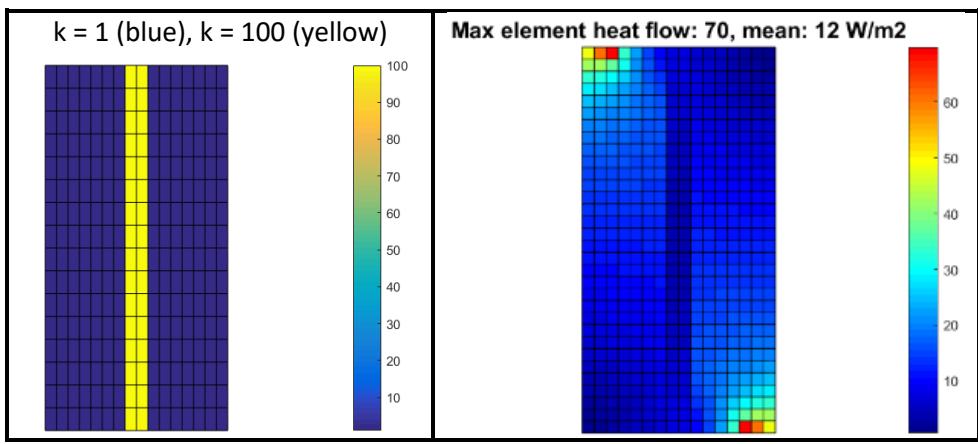


*Figure 17: Isocurves in presence of a vertical small conductivity strip (0.1 k)*

With 16 elements per patch side we obtain the pictures of the heat flows and the temperature gradients shown in *Figure 16*.



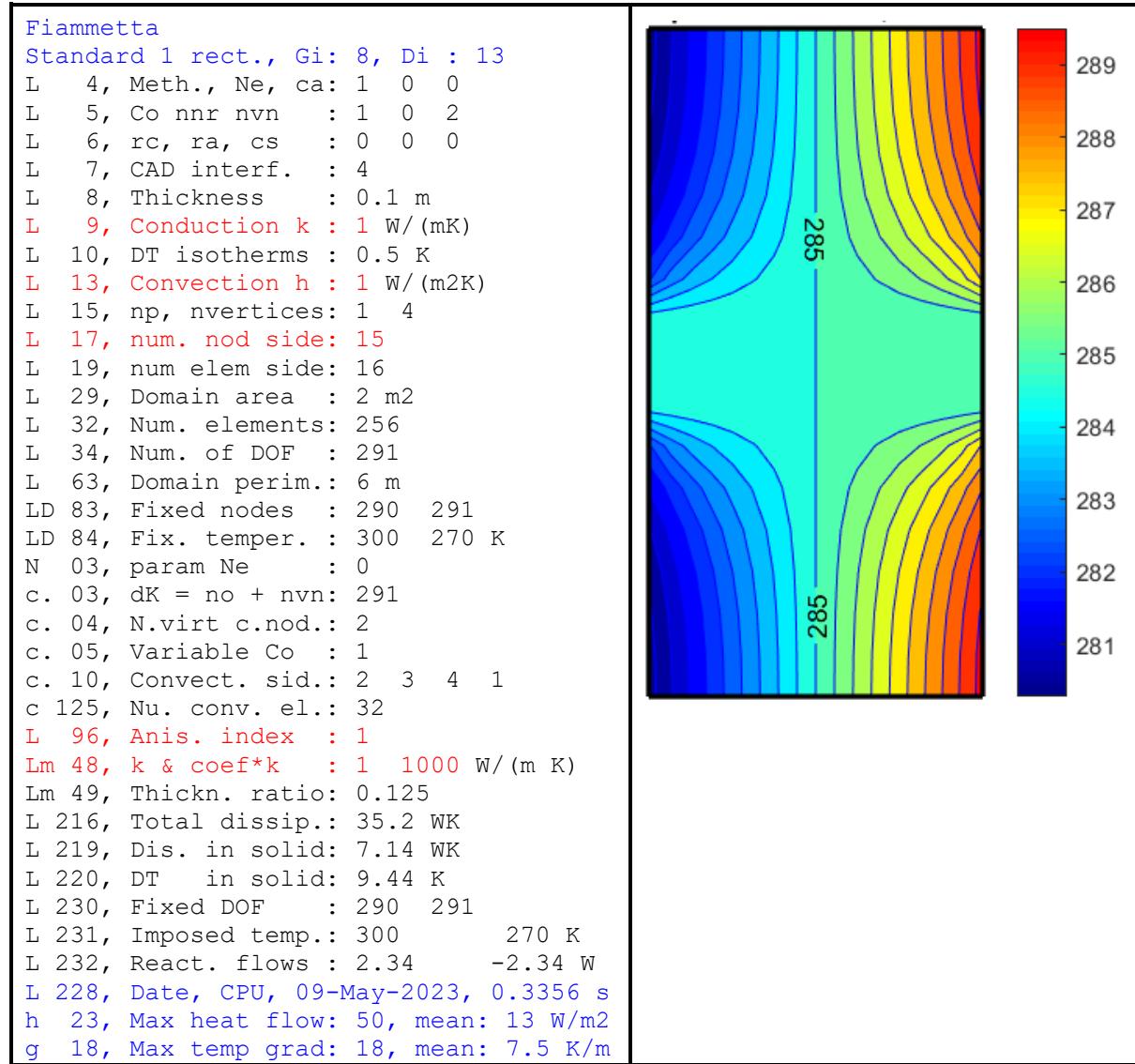
*Figure 18: Heat flows and temperature gradients (vertical strip with small conductivity)*



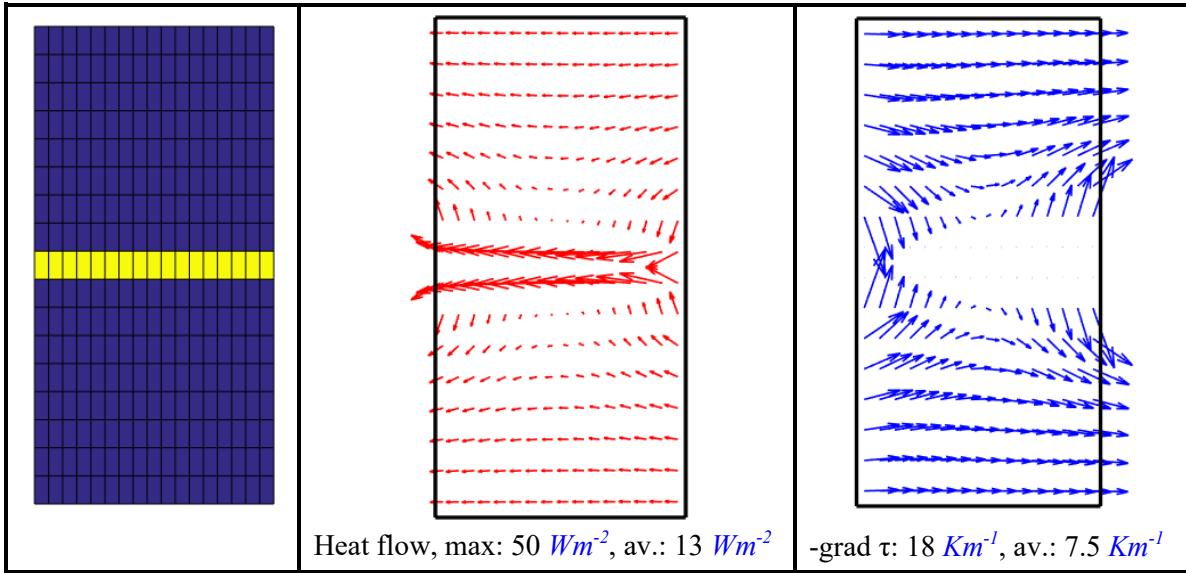
*Figure 19: Visualizations of scalars defined element by element*

It is possible to use two other visualizations (*Figure 19*), the first concerns the anisotropy of material characteristics (conductivity coefficient and thickness): This illustration is created in *mat\_coK.m* (*Table 69*). The second represents scalars defined element by element, like the module of the heat flow.

## 2.4 Thermal horizontal bridge



*Figure 20: Isocurves - horizontal thermal bridge with high conductivity (1000 k)*



*Figure 21: Heat flows and temperature gradients (horizontal strip with high conductivity)*

With respect to the example of [Figure 10](#), we simply modify the function [mat\\_cok.m](#) ([Table 69](#)). The effect of the thermal bridge is important when, as in [Figure 20](#), we impose a conductivity ratio of 1000. This test shows the importance of thermal bridges in building design ([Figure 20](#)). We also observe that the heat rate crossing the domain is equal to 3.6  $W$  when the conductivity is uniform. It reaches the value of 50  $W/m^2$  if the conductivity in the thermal bridge is 1000 times the conductivity in the other part of the domain.

## 2. Tutorial III: Structured mesh based on Coons' patch

Two authors contributed to the second generation of finite element models based on numerical integration techniques. The isoparametric element technique [[Iron 1966](#)] is based on the Coons patch developed in the frame of Computed Aided Design (CAD) [[Coons 1967](#)].

### 3.1 Numerical evaluation of the temperature gradient in a Coons patch

To simplify the subsequent development dedicated to the explanation on how to represent temperature gradients and heat flows, we limit ourselves to **two dimensions** by modeling elements and fields in the plane. We start by rewriting the nodes definition of the Coons patch (quadrilateral) in 2D. The patch is defined by its four vertices  $[Q]$ .

$$[Q] = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix} = [X \quad Y] \quad (45)$$

Any point pertaining to the patch is expressed as a function of the four vertices  $[Q]$  and the blending functions  $f(s, t)$  stored in the vector  $[F]$ :

$$\begin{aligned} x(s, t) &= [F][X] = [(1-s)(1-t) \quad s(1-t) \quad st \quad (1-s)t][X] \\ y(s, t) &= [F][Y] = [(1-s)(1-t) \quad s(1-t) \quad st \quad (1-s)t][Y] \end{aligned} \quad (46)$$

Cartesian space $x, y$ $dS = dx dy$	Parametric space $s, t$ $dS = J(s, t) ds dt$
---	--

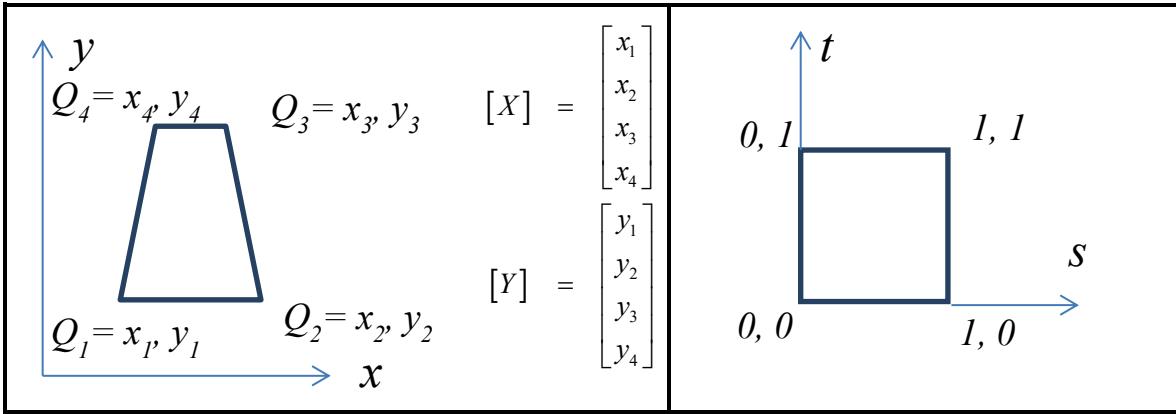


Table 6: Coons patch definition in Cartesian and parametric spaces

The barycenter of the four vertices is the point situated at  $s = \frac{1}{2}$ ,  $t = \frac{1}{2}$ .

$$\begin{aligned} x(s=1/2, t=1/2) &= [1/4 \quad 1/4 \quad 1/4 \quad 1/4][X] = (x_1 + x_2 + x_3 + x_4)/4 \\ y(s=1/2, t=1/2) &= [1/4 \quad 1/4 \quad 1/4 \quad 1/4][Y] = (y_1 + y_2 + y_3 + y_4)/4 \end{aligned} \quad (47)$$

The derivatives of the  $x$  and  $y$  cartesian coordinates expressed in parametric coordinates  $s$  and  $t$  are:

$$\begin{aligned} \frac{\partial x(s,t)}{\partial s} &= \frac{\partial [F]}{\partial s}[X] = [-(1-t) \quad (1-t) \quad t \quad -t][X] \\ \frac{\partial x(s,t)}{\partial t} &= \frac{\partial [F]}{\partial t}[X] = [-(1-s) \quad -s \quad s \quad (1-s)][X] \\ \frac{\partial y(s,t)}{\partial s} &= \frac{\partial [F]}{\partial s}[Y] = [-(1-t) \quad (1-t) \quad t \quad -t][Y] \\ \frac{\partial y(s,t)}{\partial t} &= \frac{\partial [F]}{\partial t}[Y] = [-(1-s) \quad -s \quad s \quad (1-s)][Y] \end{aligned} \quad (48)$$

For the bilinear element of equation (1.51), the Jacobian matrix  $[J]$  is equal to:

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{bmatrix} = \begin{bmatrix} [-(1-t) \quad (1-t) \quad t \quad -t][X] & [-(1-t) \quad (1-t) \quad t \quad -t][Y] \\ [-(1-s) \quad -s \quad s \quad (1-s)][X] & [-(1-s) \quad -s \quad s \quad (1-s)][Y] \end{bmatrix} \quad (49)$$

Its determinant  $J$  is called the **jacobian of the transformation**. In the center of the square representing the patch in parametric coordinates,  $s = 0.5$ ,  $t = 0.5$ , we have:

$$[J]_{s=t=\frac{1}{2}} = \frac{1}{2} \begin{bmatrix} [-1 \quad 1 \quad 1 \quad -1][X] & [-1 \quad 1 \quad 1 \quad -1][Y] \\ [-1 \quad -1 \quad 1 \quad 1][X] & [-1 \quad -1 \quad 1 \quad 1][Y] \end{bmatrix} \quad (50)$$

Writing this relation explicitly in terms of the cartesian coordinates of the vertices, we obtain:

$$[J]_{s=t=\frac{1}{2}} = \frac{1}{2} \begin{bmatrix} x_2 + x_3 - x_1 - x_4 & y_2 + y_3 - y_1 - y_4 \\ x_4 + x_3 - x_1 - x_2 & y_4 + y_3 - y_1 - y_2 \end{bmatrix} \quad (51)$$

At the barycenter of the element, the jacobian of the transformation, which is the scalar function corresponding to the determinant of the jacobian matrix, is then:

$$J_{s=t=\frac{1}{2}} = \frac{1}{2} ((x_2 + x_3 - x_1 - x_4)(y_4 + y_3 - y_1 - y_2) - (x_4 + x_3 - x_1 - x_2)(y_2 + y_3 - y_1 - y_4)) \quad (52)$$

Finally:

$$\mathbf{J}_{s=t=\frac{1}{2}} = \frac{1}{2}((x_2 - x_4)(y_3 - y_1) + (x_3 - x_1)(y_4 - y_2)) \quad (53)$$

The gradient of a scalar function, for instance the temperature  $\tau(s, t)$ , is computed as follows. After expressing it in parametric coordinates, it is converted in Cartesian ones ([the real world](#)).

$$\begin{bmatrix} \frac{\partial \tau}{\partial s} \\ \frac{\partial \tau}{\partial t} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \end{bmatrix} = [\mathbf{J}] \begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \end{bmatrix} = [\mathbf{J}] \vec{\nabla} \tau \quad (54)$$

After inverting (52), we obtain:

$$\vec{\nabla} \tau = \begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \end{bmatrix} = [\mathbf{J}]^{-1} \begin{bmatrix} \frac{\partial \tau}{\partial s} \\ \frac{\partial \tau}{\partial t} \end{bmatrix} \quad (55)$$

Because the temperature field is defined in the parametric coordinates with the same blending functions as the geometry:  $x(s, t)$  and  $y(s, t)$ , these elements are named isoparametric:

$$\tau = [(1-s)(1-t) \quad s(1-t) \quad st \quad (1-s)t][T] \quad (56)$$

We can easily compute the temperature derivatives with respect to  $s$  and  $t$ :

$$\begin{bmatrix} \frac{\partial \tau}{\partial s} \\ \frac{\partial \tau}{\partial t} \end{bmatrix} = \begin{bmatrix} -(1-t) & (1-t) & t & -t \\ -(1-s) & -s & s & (1-s) \end{bmatrix} [T] \quad (57)$$

In the barycenter:

$$\begin{bmatrix} \frac{\partial \tau}{\partial s} \\ \frac{\partial \tau}{\partial t} \end{bmatrix}_{s=t=\frac{1}{2}} = \frac{1}{2} \begin{bmatrix} -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix} [T] \quad (58)$$

The two components of the following equation correspond to the [lines 11 & 12](#) of the function [gra\\_atg.m](#) ([Table 56](#)). They represent the temperature gradient

$$\vec{\nabla} \tau = \begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \end{bmatrix} = [\mathbf{J}]^{-1} \frac{1}{2} \begin{bmatrix} -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix} [T] \quad (59)$$

Being able to calculate the temperature gradients, it is now possible to compute the conductivity matrices. The Matlab<sup>©</sup> function [fem\\_Kco.m](#) ([Table 44](#)) allows computing the conductivity matrix  $[\mathbf{K}]$  of an isoparametric quadrilateral element with a bilinear temperature field. To obtain the effective element conductivity matrix, the output of the function has to be multiplied by the conductivity coefficient  $k$  (expressed in  $WK^{-1}m^{-1}$  and stored in the vector  $co$  because it may vary from element to element) and the constant thickness  $th$ .

To compute a conductivity matrix, we need the matrix  $[xyz]$  of element coordinates (first argument of the function) and the localization  $lo$  of the element, for instance, the positions of its four nodes in the coordinate matrix (second argument of the function [fem\\_Kco.m](#)). A direct Matlab evaluation of the conduction matrix of a square is given in [Table 7](#), using explicit definitions of both the coordinates and the localization vector. As noted before in the explicit

analytical calculation of the conductivity matrix, it is easy to check that the result does not depend on the scale of the geometry.

Matlab input	<code>xyz = [0 0 0;1 0 0;1 1 0;0 1 0];lo=[1 2 3 4]; [K] = fem_Kco (xyz,lo)*6</code>
Matlab Output	<code>K = 4 -1 -2 -1 -1 4 -1 -2 -2 -1 4 -1 -1 -2 -1 4</code>

Table 7: Numerically integrated conductivity matrix

To obtain the effective conductivity matrix, the result displayed in *Table 7* is multiplied by  $k \text{ th} / 6$ , where  $k$  is the conductivity coefficient and  $\text{th}$  the thickness.

### 3.2 CAD model of the domain to be analyzed

The inputs of a CAD model involve three kinds of data. The `xyz_cao` matrix contains the coordinates of the nodes, `car_cao` is giving the patches definition and `nbo` is the number of interfaces limiting the patches. The size of the matrix `xyz_cao` must be the maximum numbering of the nodes defined in the matrix `car_cao`. Because these numbers represent *DOF*, they must all be present in the matrix `car_cao`. For the example of *Figure 30*, we have, on the left of *Table 8*, the node numbering `[(1: npv)' xyz_cao]` and the matrix of 2D nodal coordinates and, on the right, `[(1:np)' car_cao]`, the patch numbering and the patch matrix. The line numbering of both matrices appears in blue on the left. Note that `npv` is the number of patch vertices and `np`, the number of patches.

<code>npv=size(xyz_cao,1);[(1:npv)' xyz_cao]</code>	<code>np=size(xyz_cao,1);[(1:np)' car_cao]</code>
1 2 2	
2 2 3	
3 1 2	
4 0 3	1 1 2 4 3
5 0 0	2 3 4 5 6
6 1 1	3 7 8 6 5
7 3 0	
8 3 1	

Table 8: Instructions used to display input data of *Figure 30*

*Lines 16 - 19* of *Table 31* are generating the Coons patches displayed in *Table 8* and drawn in *Figure 30*.

### 3.3 Identification of the *DOF* pertaining to a patch side

The introduction of fixations or distributed loads on a patch side needs the identification of the concerned *DOF*. This detection is obtained through a single instruction. In *Table 9*, we see the four instructions used to determine the *DOF* of the cavity of *Figure 22*. The cavity is defined by *lines 1 - 4* of *Table 31*. The four *CAD* patches and their related data are shown in *Figure 22*. The matrix `bor` corresponds to a mesh of 100 elements counting four nodes on each patch side.

The matrix `car_cao` defines the four patches, the matrix `pbo` indicates in which line of `bor` the sides of the patches are described. For instance, the second side of patch 1 connecting node 5 to node 1 is described in *line 2* of matrix `bor`. The cavity side of patch 1 is connecting node 6 (`car_cao (1,1)`) to node 5 (`car_cao (1,2)`), it is the first side (`pbo (1,1)`) of the patch and its description is in *line 1* of `bor` (*columns 5 and 6* are giving the sequence of side nodes).

In the *line 50* shown in *Table 9*, we select the nodes of the second side of the third patch (*Figure 22*). According to the second column of *line 3* of `pbo`, the side is described in *line 8* of matrix `bor`. The result is displayed in *Figure 23*.

<pre> with nni = 4,[ (1:size(bor,1))' bor ] → 1   6      5      1      0      9      12 2   5      1      1      4     13      16 3   1      2      1      0     17      20 4   2      6      1      2     21      24 5   2      3      2      0     25      28 6   3      7      2      3     29      32 7   7      6      2      0     33      36 8   3      4      3      0     37      40 9   4      8      3      4     41      44 10  8      7      3      0     45      48 11  5      8      4      0     49      52 12  4      1      4      0     53      56 </pre>	
<pre> [(1:np)' car_cao] → 1   6      5      1      2 2   6      2      3      7 3   7      3      4      8 4   1      5      8      4 </pre>	<pre> [(1:np)' pbo] → 1   1      2      3      4 2   4      5      6      7 3   6      8      9      10 4   2      11     9      12 </pre>

Figure 22: CAD data defining a domain surrounding a cavity

The sequence for selecting DOF along a patch side is:

- 1: `car_cao` (patch number, numbering of first vertex of the concerned side),
- 2: `bor` (`pbo` (patch number, side number), 5),
- 3: `bor` (`pbo` (patch number, side number), 6),
- 4: `car_cao` (patch number, number of second vertex of the side).

```

50 lg  = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
62 bc = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
63 [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
64 [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
65 [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)]];

```

Table 9: Instructions to identify nodes along patch sides

Listing of boundary nodes of the cavity.  
Each line of the matrix `bc` contains  
the nodes pertaining to a cavity side,  
output of [lines 62-65 \(Table 9\)](#):

Nodes pertaining to the cavity boundary =  
6     9     10    11    12    5  
7    33    34    35    36    6  
8    45    46    47    48    7  
5    49    50    51    52    8

Loaded nodes on the top side, output of  
[line 50 \(Table 9\)](#):

`lg` =      3    37    38    39    40    4

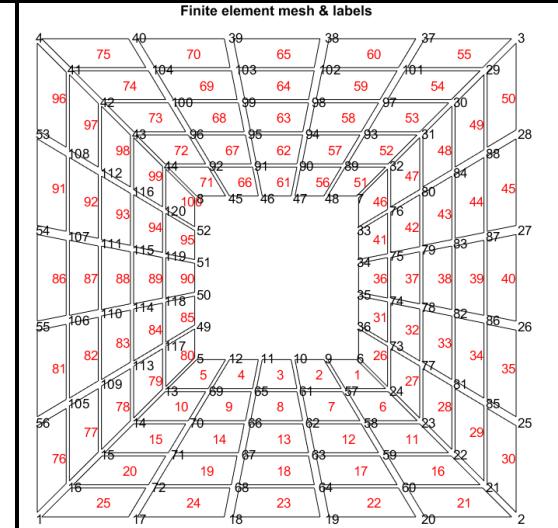


Figure 23: Finite element mesh corresponding to the CAD definition of Figure 22

### 3.4 Cavity with adiabatic hole

The temperatures of the external horizontal borders are fixed to 270 K and 300 K. The first and natural method to handle adiabatic border is to let the temperatures free on it ([Figure 24](#)). Another way to impose this condition is to impose that the border is perfectly reflective ( $\rho = 1$ , [Figure 25](#)).

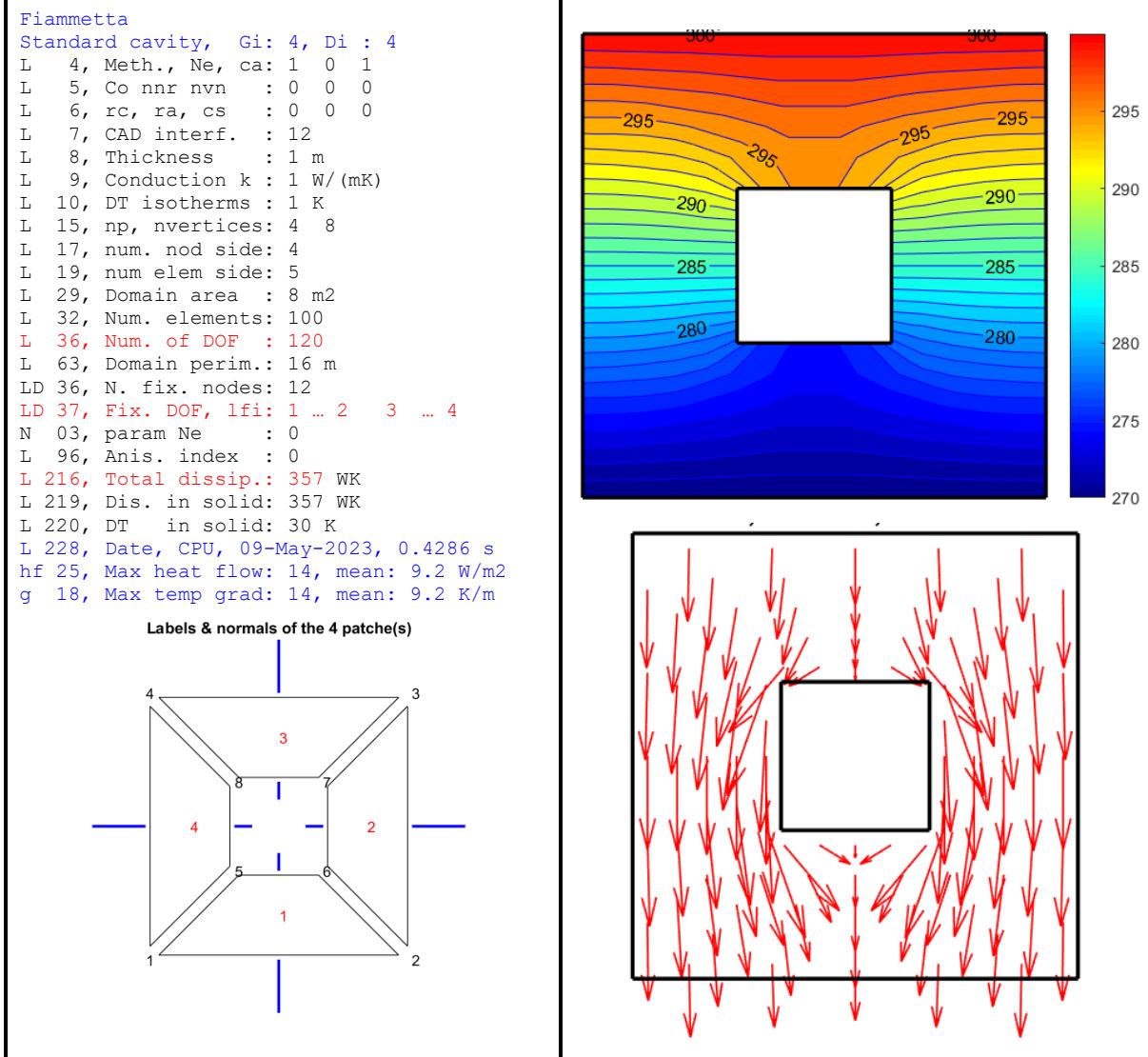
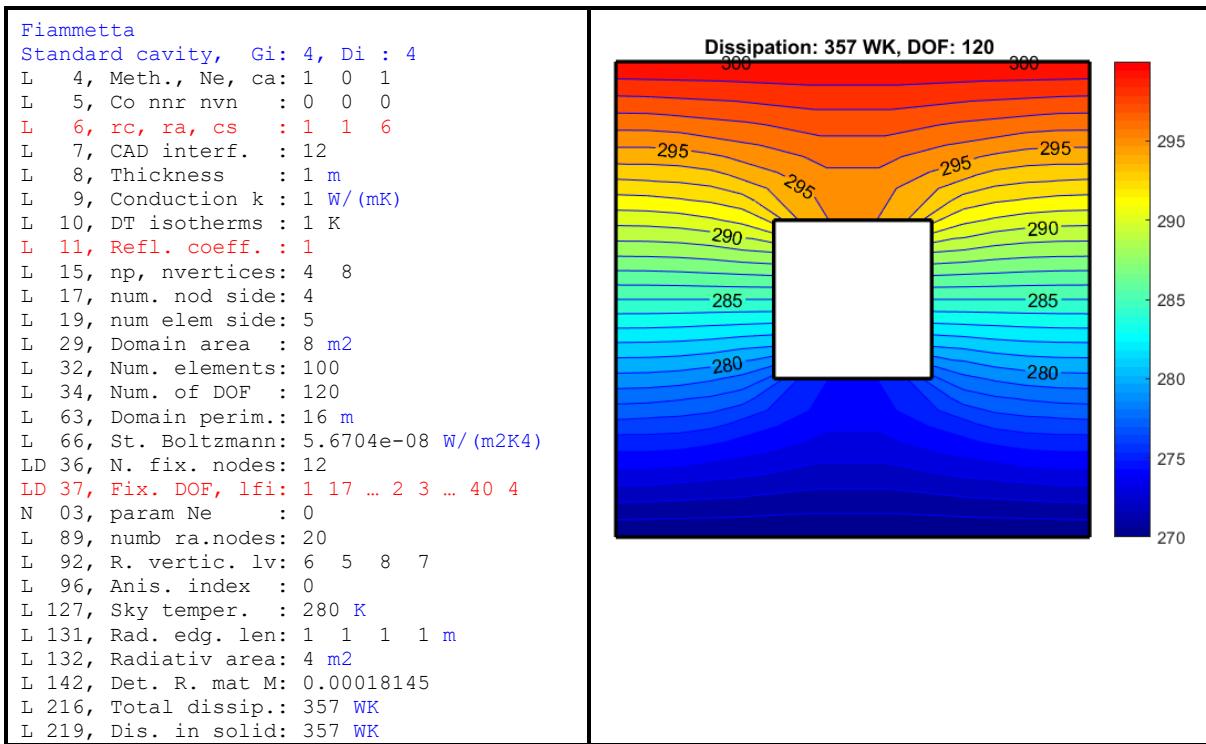


Figure 24: Heat flow around an adiabatic square cavity

As expected, the result of the same problem, but with pure reflective cavity borders, is identical to the previous one.



```

L 220, DT in solid: 30 K
L 228, Date, CPU, 09-May-2023, 0.36452 s
g 18, Max temp grad: 14, mean: 9.2 K/m
hf 25, Max heat flow: 14, mean: 9.2 W/m2

```

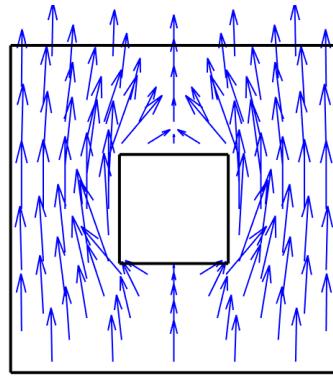


Figure 25: Heat flow around a perfectly reflective cavity

With the radiative boundary conditions taken into account in the procedure:  $rc = 1$ ,  $cs = 6$ , the function `geo_yfc.m` (*Table 67*) is called at *line 140* of `Fiammetta.m` to compute the matrix  $Fs$  of view factors of the cavity.

### 3.5 C shaped domain

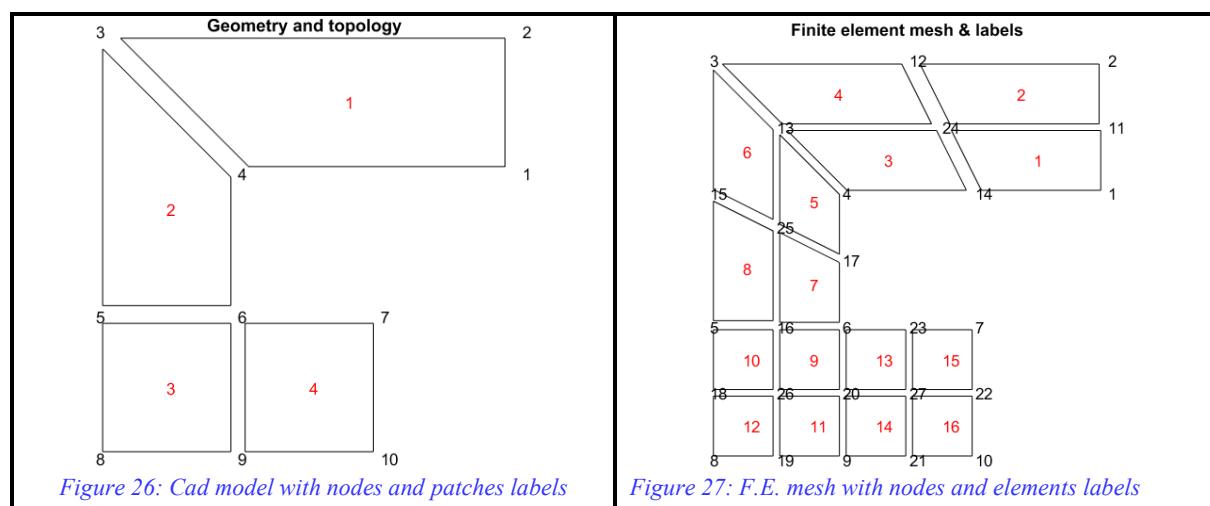


Figure 26: Cad model with nodes and patches labels

Figure 27: F.E. mesh with nodes and elements labels

The `Fiammetta.m` procedure starts generating the *CAD* model: shrunk *CAD* mesh with nodes and patches labels (function `gra_mel.m` of *Table 52* & *Figure 26*). Hereafter, the same domain will be defined with different *CAD* models: the long or the short horizontal parts of the model are trapezoidal in *Figure 28*, the domain is only composed of rectangular patches in *Figure 29* and, finally, it is composed of three trapezoidal patches. In these four situations, the dissipative function is respectively equal to 0.993 *WK*, 0.991 *WK*, 0.987 *WK* and, finally, 0.979 *WK*. Due to the convergence property of a pure conduction model with Dirichlet boundary conditions, the lowest value of the dissipative function is the best one [Debongnie, Zhong & Beckers 1995]. With the last model, it converges to 0.979 *WK* when we have 8241 *DOF*. So, the dissipation is decreasing when the mesh is finer [Debongnie & Beckers 2001]. The Matlab<sup>®</sup> functions `gra_mel.m` (*Table 52*) and `gra_mnl.m` (*Table 53*) enable the visualization of the finite element mesh (*Figure 27*).

Two functions are fundamental in `Fiammetta.m`: the function `cad_mes.m` (*Table 35*) and the function `cad_edg.m` (*Table 36*), which is called in `cad_mes.m`. Both allow defining the topology of the *CAD* model through the construction of matrices `bor` and `pbo` that describe the patch interfaces.

```

Fiammetta
C shape 4 patches Gi: 5, Di : 5
L 4, Method, Ne : 1 0 0
L 5, Co nnr nvn : 0 0 0
L 6, rc, ra, cs : 0 0 0
L 7, CAD interf. : 13
L 8, Thickness : 0.1 m
L 9, Conduction k : 1 W/(mK)
L 10, DT isotherms : 0.5 K
L 15, np, nvertices: 4 10
L 17, num. nod side: 4
L 19, num elem side: 5
L 29, Domain area : 6 m2
L 32, Num. elements: 100
L 34, Num. of DOF : 126
L 63, Domain perim.: 14 m
N 03, param Ne : 0
L 96, Anis. index : 0
L 216, Total dissip.: 0.993 WK
L 219, Dis. in solid: 0.993 WK
L 220, DT in solid: 10 K
L 228, Date, CPU, 09-May-2023, 0.34111 s
g 18, Max temp grad: 3.2, mean: 1.7 K/m
hf 25, Max heat flow: 3.2, mean: 1.7 W/m2

```

Labels & normals of the 4 patch(es)

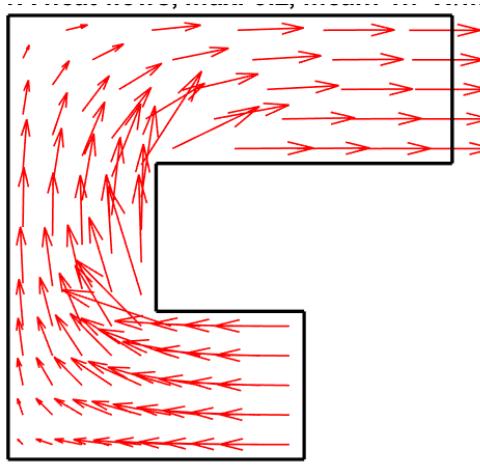
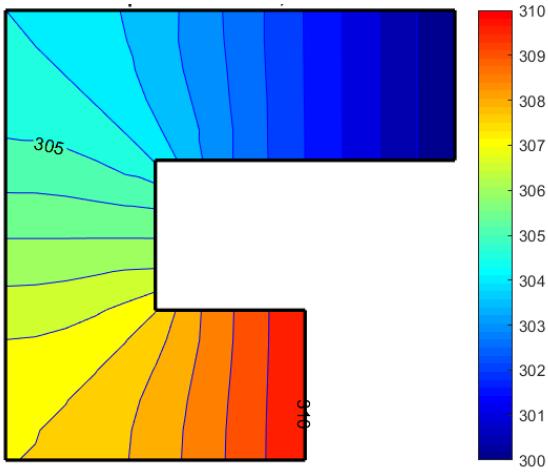
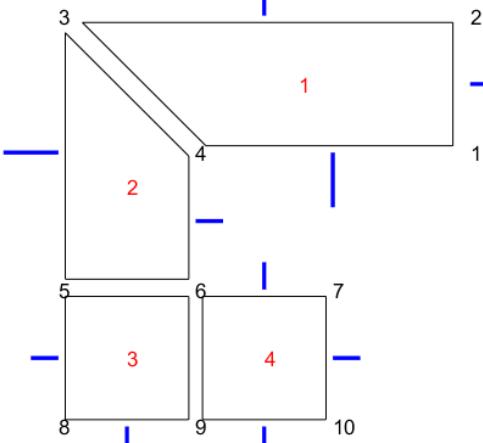


Figure 28: CAD model with diagonal on the long horizontal side

In the next test, we avoid the distorted shapes of the first patches by replacing them by 3 squares.

```

7 xyz_cao = [2 2;2 3;1 2;1 3;0 2;0 3;0 0;0 1;1 0;1 1;3 0;3 1];%CAD geometry
8 car_cao = [1 2 4 3;3 4 6 5;3 5 8 10;10 8 7 9 ;10 9 11 12];% nbo=16; % patch

31 th = .1; disp(['Thickness : ',num2str(th),' m'])

55 lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2) ...
56 car_cao(5,3) bor(pbo(5,3),5):bor(pbo(5,3),6) car_cao(5,4)];
57 nfi = size(lfi,2); % Dirichlet boundary conditions .....
58 if nfi > 2 ;fix = [ones(1,nfi/2)*300 ones(1,nfi/2)*310 ] ;end

69 gh = zeros(ndK,1); % Second member initialization always necessary
83 h = 18;hh = h*th;nec = nni+1;

122 K = zeros(ndK,ndK);% co = mat_cok (nni+1,nni+1); % Initializations
123 co = ones(1,nel)*th;

```

The result is given for two meshes:

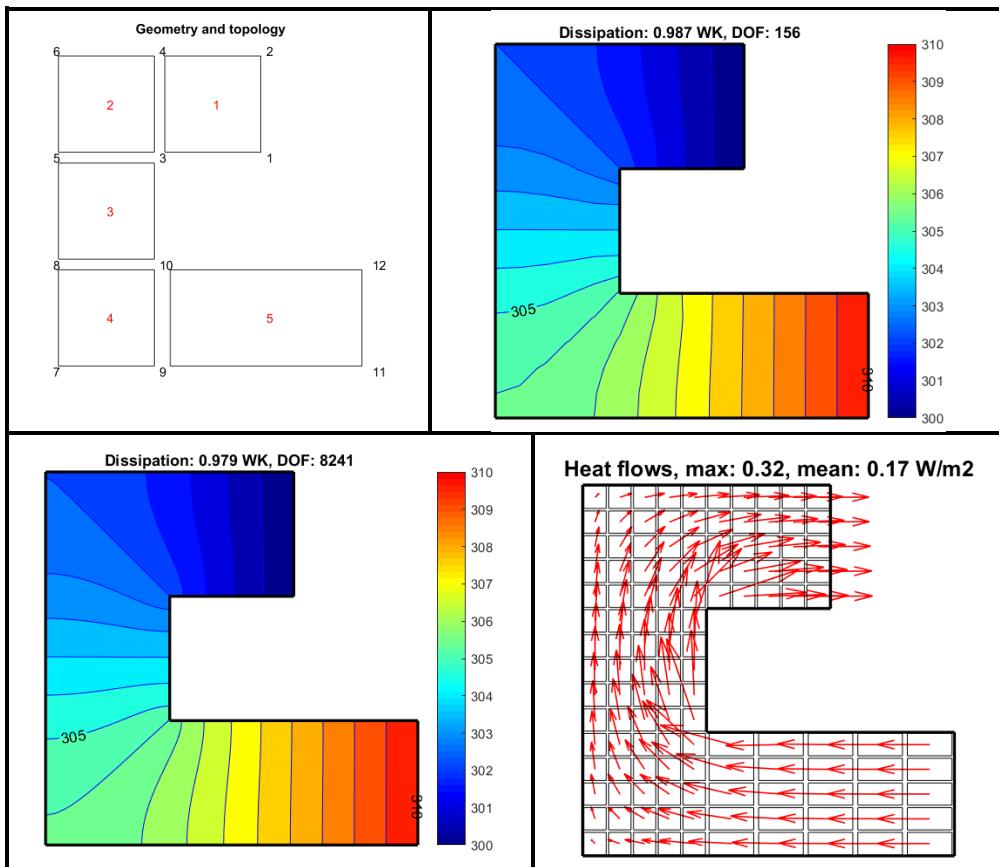


Figure 29: CAD model fully based on rectangular patches

In the last test, we reproduce the same conditions as in [Figure 4](#).

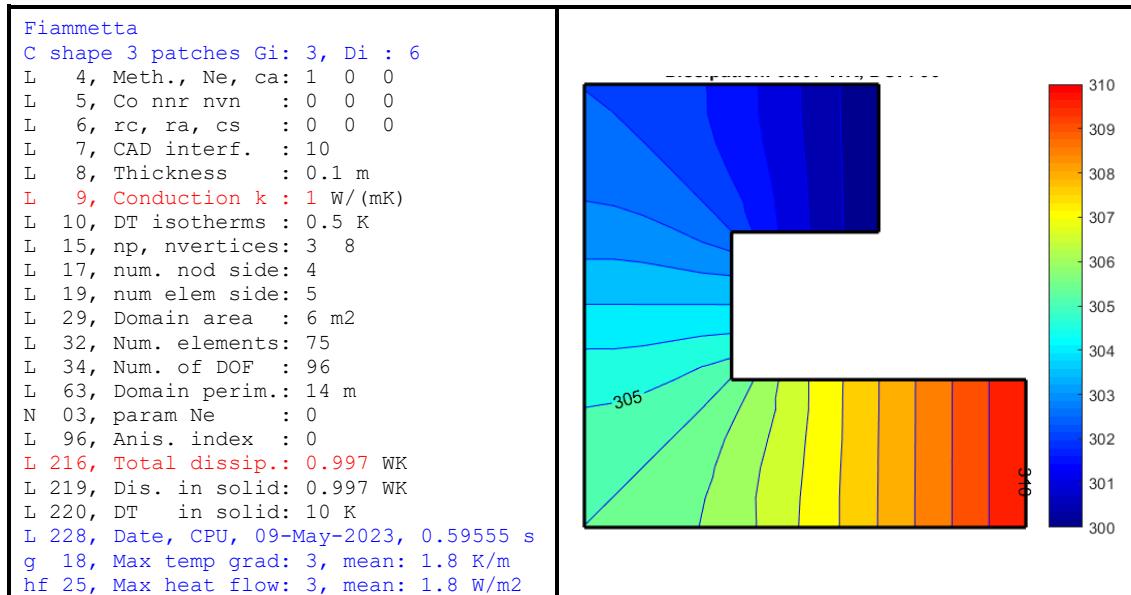


Figure 30: CAD model based on 3 trapezoidal patches, 75 elements

### 3.6 Boundary conditions

We show here how we can introduce Dirichlet ([Table 32](#)), von Neumann ([Table 33](#)) and convective boundary conditions ([Table 34](#)). For Dirichlet boundary conditions, we have only to give a list and the values of the fixed nodes. For the second, we give a list of nodes and the values of the corresponding second members of the equations. In the case of convection, we give the localizations of the  $3 \times 3$  convective matrices ([34](#)) and for each element the convection coefficient (uni-line matrix [he](#)).

### 3.7 Basic isotherm drawing

In a Coons patch, the isolines of a scalar quantity of a finite element solution are displayed in the `gra_ipa.m` function (Table 60). Another method for drawing the levels of a scalar function consists in working independently in each element with the function `gra_lin.m` (Table 61).

The gradient of the temperature in the barycenter of the elements (the barycenter corresponds to a low integration with only 1 Gauss point) are computed in `gra_atg.m` (Table 56). According to the property of super convergence of the Gauss integration points [Barlow 1976], we state that the gradient evaluated at this point is suitable for the representation using arrows symbol. To obtain the heat flow, the gradient is multiplied by  $-k \times th$ , with  $k$ , the element conductivity stored in the vector `co` and  $th$  the global thickness. The function `gra_atg.m` needs the nodal coordinates computed previously, for instance, in the Matlab<sup>©</sup> function `cad_mes.m` (Table 35).

### 3.8 Thermal bridge in a trapezoidal domain

Now, we modify the shape of the rectangular domain analyzed in Figure 10 and check the consequence of introducing an horizontal thermal bridge.

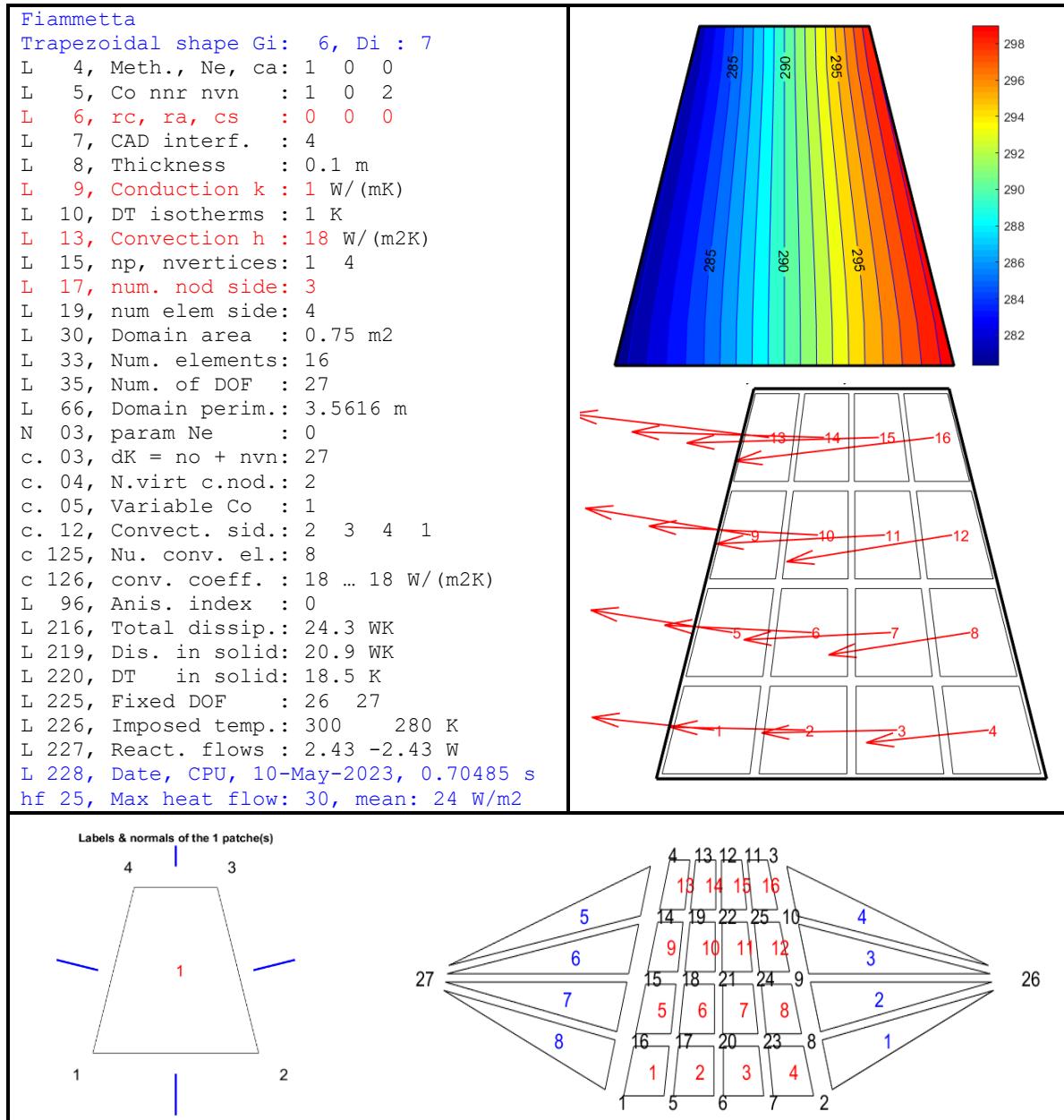
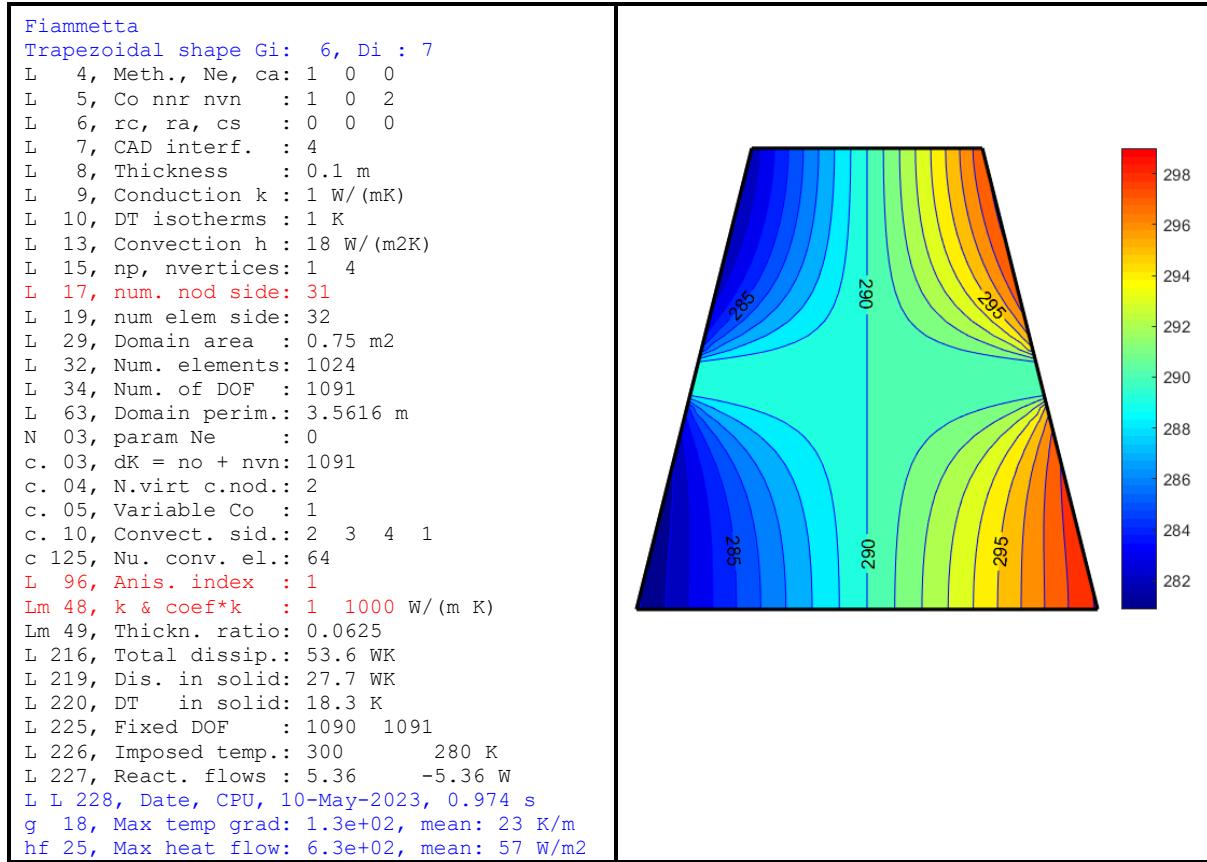
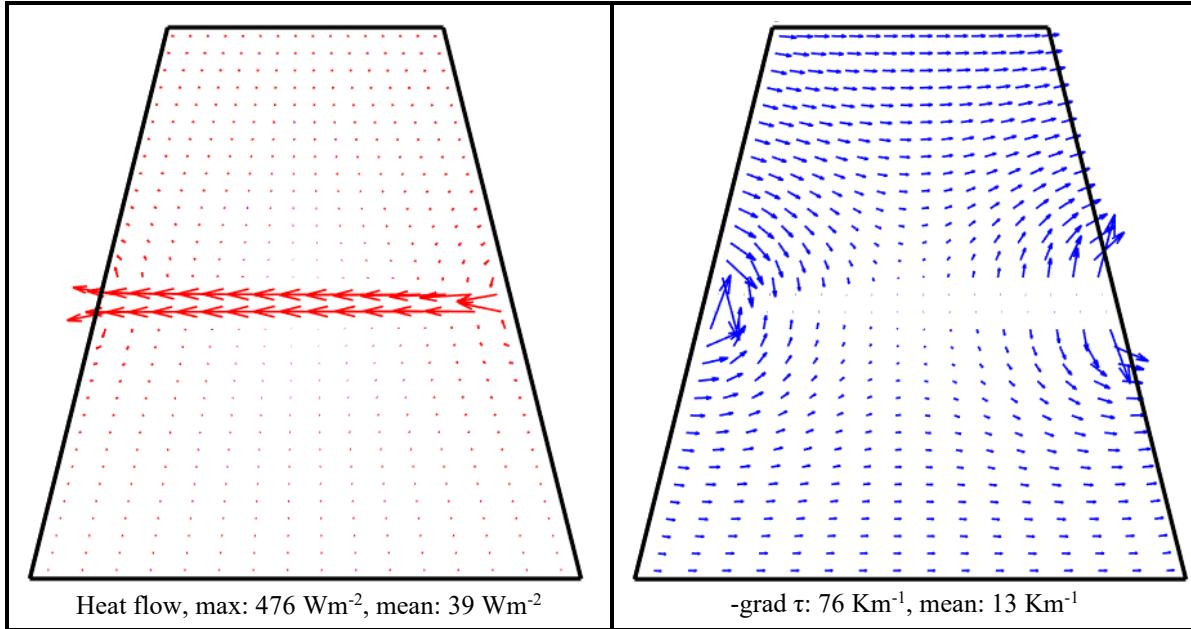


Figure 31: 2 imposed temperatures, 2 adiabatic faces in a trapezoidal domain

The introduction of non-homogeneous material is performed using the definition, element by element, of the conductivity coefficient. This function is written with the hypotheses that the numbers of elements in the  $x$  and  $y$  directions satisfy certain conditions that can be checked in the listing of the function ([Table 69](#)). In this situation of anisotropic material defining an horizontal thermal bridge, the heat flow picture is dominated by the arrows in the central zone, while in the temperature gradient one, the same zone of high flows is disappearing due to the small value of the gradients.



*Figure 32: Non homogeneous trapezoidal domain*



*Figure 33: Horizontal strip with high conductivity in a trapezoidal domain*

### 3. Tutorial IV: Transient heat transfer

To carry out the transient studies, new physical quantities are introduced, such as the density of the material and its heat capacity. The specific heat capacity ( $Jkg^{-1}K^{-1}$ ) corresponds to a system defined per unit of mass ( $kg$ ) of a compound (the term 'specific heat' is sometimes used). The thermal capacity ( $JK^{-1}$ ) is an extensive scalar quantity.

The thermal diffusivity  $\alpha$  of a material, expressed in  $m^2s^{-1}$ , represents its tendency to facilitate the heat diffusion.

$$\alpha = \frac{k}{\rho c_p} \quad (60)$$

Useful references: [Lee & Jackson 1976], [Lee 1977], [Lee & Mason 2008], [Siemens 2017].

#### 4.1 Solution of the transient problem

To introduce the time variation in the heat equations, a new matrix  $[C]$  ( $JK^{-1}$ ) is introduced.

$$([C] + \theta \Delta t [K]) [T^{n+1}] = ([C] - (1-\theta) \Delta t [K]) [T^n] + \Delta t (\theta [f^{n+1}] + (1-\theta) [f^n]) \quad (61)$$

The value  $\theta = 1$  corresponds to the implicit scheme, which is considered as unconditionally stable. We write:

$$([C] + \Delta t [K]) [T^{n+1}] = [C] [T^n] + \Delta t [f^{n+1}] \quad (62)$$

The uni-column matrix  $[T]$  is divided into two parts:

1. the unknown nodal temperatures  $T_I$  and
2. the fixed and therefore, constant temperatures  $T_f$

$$[T] = \begin{bmatrix} T_I \\ T_f \end{bmatrix} \quad (63)$$

Equation (62) becomes:

$$\left( \begin{bmatrix} C_{11} & C_{1f} \\ C_{f1} & C_{ff} \end{bmatrix} + \Delta t \begin{bmatrix} K_{11} & K_{1f} \\ K_{f1} & K_{ff} \end{bmatrix} \right) \begin{bmatrix} T_I^{n+1} \\ T_f^{n+1} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{1f} \\ C_{f1} & C_{ff} \end{bmatrix} \begin{bmatrix} T_I^n \\ T_f^n \end{bmatrix} + \Delta t \begin{bmatrix} f^{n+1} \\ r^{n+1} \end{bmatrix} \quad (64)$$

The superscripts  $n$  and  $n+1$  indicate the iteration number. The variable  $f^{n+1}$  expressed in  $W$ , represents the heat loading at step  $n+1$ , while  $r^{n+1}$  represents the reactions on the fixed DOF at the same step. The first group of (64) is:

$$\left( \begin{bmatrix} C_{11} & C_{1f} \end{bmatrix} + \Delta t \begin{bmatrix} K_{11} & K_{1f} \end{bmatrix} \right) \begin{bmatrix} T_I^{n+1} \\ T_f^{n+1} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{1f} \end{bmatrix} \begin{bmatrix} T_I^n \\ T_f^n \end{bmatrix} + \Delta t \begin{bmatrix} f^{n+1} \\ r^{n+1} \end{bmatrix} \quad (65)$$

Developing this relation gives:

$$([C_{11}] + \Delta t [K_{11}]) [T_I^{n+1}] = [C_{11}] [T_I^n] + \Delta t ([f^{n+1}] - [K_{1f}] [T_f]) \quad (66)$$

If fixations are present, the solution of this equation is:

$$[T_I^{n+1}] = ([C_{11}] + \Delta t [K_{11}])^{-1} \{ [C_{11}] [T_I^n] + \Delta t ([f^{n+1}] - [K_{1f}] [T_f]) \} \quad (67)$$

Without fixation:

$$[T^{n+1}] = ([C] + \Delta t [K])^{-1} \{ [C] [T^n] + \Delta t [f^{n+1}] \} \quad (68)$$

If there are no loads nor fixations, we can remove the indices and we obtain the very simple relation:

$$[T^{n+1}] = ([C] + \Delta t [K])^{-1} [C] [T^n] \quad (69)$$

The second line of (64), where the unknowns are the outgoing heat flows  $[r^{n+1}]$ , is decomposed as follows:

$$\begin{aligned} ([C_{f1} \ C_{ff}] + \Delta t [K_{f1} \ K_{ff}]) \begin{bmatrix} T_1^{n+1} \\ T_f \end{bmatrix} &= [C_{f1} \ C_{ff}] \begin{bmatrix} T_1^n \\ T_f \end{bmatrix} + \Delta t [r^{n+1}] \\ [r^{n+1}] &= \frac{1}{\Delta t} [C_{f1}] ([T_1^{n+1}] - [T_1^n]) + [K_{f1}] [T_1^{n+1}] + [K_{ff}] [T_f] \end{aligned} \quad (70)$$

## 4.2 Capacity matrix

### 4.2.1 Quadrilateral

The capacity matrix  $[C]$  is a function of the density  $\rho$  of the material, its heat capacity  $c_p$  and its volume  $V$ .

$$\tau = T_1 \left(1 - \frac{x}{a}\right) \left(1 - \frac{y}{b}\right) + T_2 \frac{x}{a} \left(1 - \frac{y}{b}\right) + T_3 \frac{x}{a} \frac{y}{b} + T_4 \left(1 - \frac{x}{a}\right) \frac{y}{b} \quad (71)$$

$$\begin{aligned} \tau &= [F][T] \\ [F] &= \begin{bmatrix} \left(1 - \frac{x}{a}\right) \left(1 - \frac{y}{b}\right) & \frac{x}{a} \left(1 - \frac{y}{b}\right) & \frac{x}{a} \frac{y}{b} & \left(1 - \frac{x}{a}\right) \frac{y}{b} \\ [T]^T &= [T_1 \ T_2 \ T_3 \ T_4] \end{bmatrix} \quad (72) \end{aligned}$$

$$[C] = \int_V \rho c_p [F]^T [F] dV \quad (73)$$

To show the process of integration, we compute the term  $C_{33}$  of the capacity matrix  $[C]$ . The volume  $V$  is the product of the area  $ab$  by the thickness  $e$ .

$$\begin{aligned} C_{33} &= e \int_0^a \left( \int_0^b \rho c_p \frac{x^2 y^2}{a^2 b^2} dy \right) dx \\ &= \rho e c_p \int_0^a \frac{b^3 x^2}{3a^2 b^2} dx = \rho e c_p \frac{b}{3} \int_0^a \frac{x^2}{a^2} dx = \rho e c_p \frac{ab}{9} = \frac{\rho V c_p}{9} \quad (74) \end{aligned}$$

In (74), we observe that the sum of the terms is equal to 36, and, therefore, that, concentrated in one point, the capacity is equal to  $\rho V c_p$ . Matrix  $C$  is expressed in  $JK^T$ .

$$[C] = \frac{\rho V c_p}{36} \begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix} \quad (75)$$

The capacity matrix of an element given by its localization vector *lo* and the set of nodal coordinates *xyz* is computed in the Matlab<sup>©</sup> function *fem\_Cae.m* (*Table 43*).

*Table 10* shows Matlab<sup>©</sup> instructions allowing to compute capacity matrices in various geometrical situations. To obtain the effective capacity matrix, the output of *fem\_Cae.m* is multiplied by the thickness *th* (m), the capacity *cp* (Jkg<sup>-1</sup>K<sup>-1</sup>) and the specific mass *ro* (kgm<sup>-3</sup>). The total capacity for the cases presented in *Table 10* is equal to *th* (0.1) x *cp* (1000) x *ro* (2500) x *sum (sum (C)) / 36* = 250000 JK<sup>-1</sup> for the four first cases and 10<sup>6</sup> JK<sup>-1</sup> for the last one.

Input	<code>xyz =[0 0 0;1 0 0;1 1 0;0 1 0];lo=[1 2 3 4]; [C] = fem_Cae(xyz,lo)*36</code>
Matlab Output	C = 4 2 1 2 sum(sum(C)) = 36. 2 4 2 1 total capacity: 250000 JK <sup>-1</sup> 1 2 4 2 2 1 2 4
Input	<code>xyz=[0 0 0;2 0 0;2 .5 0;0 .5 0];lo=[1 2 3 4];[C] = fem_Cae(xyz,lo)*36</code>
Matlab Output	C = 4 2 1 2 sum(sum(C)) = 36. 2 4 2 1 total capacity: 250000 JK <sup>-1</sup> 1 2 4 2 2 1 2 4
Input	<code>xyz=[0 0 0;.5 0 0; .5 2 0;0 2 0];lo=[1 2 3 4];[C] = fem_Cae(xyz,lo)*36</code>
Matlab Output	C = 4 2 1 2 sum(sum(C)) = 36. 2 4 2 1 total capacity: 250000 JK <sup>-1</sup> 1 2 4 2 2 1 2 4
Input	<code>xyz=[0 0 0;1.5 0 0;1 1 0; .5 1 0];lo=[1 2 3 4];[C] = fem_Cae(xyz,lo)*36</code>
Matlab Output	C = 5 2.5 1 2 sum(sum(C)) = 36. 2.5 5 2 1 total capacity: 250000 JK <sup>-1</sup> 1 2 3 1.5 2 1 1.5 3
Input	<code>xyz =[0 0 0;2 0 0;2 2 0;0 2 0];lo=[1 2 3 4];[C] = fem_Cae(xyz,lo)*36</code>
Matlab Output	C = 16 8 4 8 sum(sum(C)) = 144 8 16 8 4 total capacity: 1000000 JK <sup>-1</sup> 4 8 16 8 8 4 8 16

*Table 10: Numerical integration of the capacity matrix*

#### 4.2.2 Triangle

The capacity matrix [*C*] is a function of the density *p* of the material, its heat capacity *cp* and its volume *V*. The temperature field in the triangle *S<sub>1</sub>* - *S<sub>2</sub>* - *S<sub>3</sub>* is a first-degree polynomial:

$$\tau = \alpha_0 + \alpha_1 x + \alpha_2 y \quad (76)$$

This field can also be defined by three nodal temperatures at the three vertices of the triangle:

$$T' = \begin{bmatrix} T_1 & T_2 & T_3 \end{bmatrix} \quad (77)$$

Let the [*R*] matrix relate the coefficients of the polynomial to the nodal temperatures:

$$[R] = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \quad (78)$$

The determinant of  $[R]$  is equal to twice the area  $A$  of the triangle

$$A = \frac{1}{2}(x_1y_2 + x_2y_3 + x_3y_1 - x_2y_1 - x_3y_2 - x_1y_3) \quad (79)$$

The relation is now:

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = [R] \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \quad (80)$$

And, by inverting this relation:

$$[R]^{-1} = \frac{1}{2A} \begin{bmatrix} x_2y_3 - x_3y_2 & x_3y_1 - x_1y_3 & x_1y_2 - x_2y_1 \\ y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{bmatrix} = [F] \quad (81)$$

We can now express the quantities in term of nodal temperatures

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = [F] \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \quad (82)$$

$$\tau = [1 \ x \ y] \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = [1 \ x \ y] [F] \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \quad (83)$$

The capacity matrix is obtained by integration of the following expression where we assume that the density  $\rho$  of the material, the thickness  $e$  of the element and its heat capacity  $c_p$  are constant:

$$[C] = e\rho c_p [F]^T \int_S \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} [1 \ x \ y] dS [F] \quad (84)$$

### 4.3 Temperature evolution

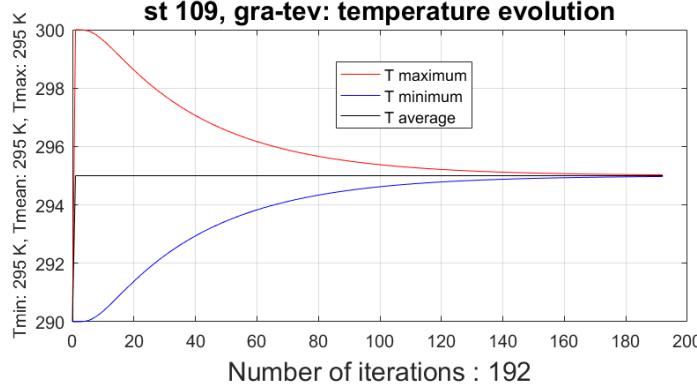
The procedure `gra_tev.m` (*Table 63*) is used to draw the time evolution of quantities that are functions of the time, like nodal temperatures of the model or any other quantity derived from the temperature field, which is the primary output of an analysis. These temperatures have to be collected at each iteration step and stored in vectors whose length is the number of iterations more one. An example of drawing produced with this procedure is given in *Figure 34*.

### 4.4 Temperature homogenization in an insulated solid

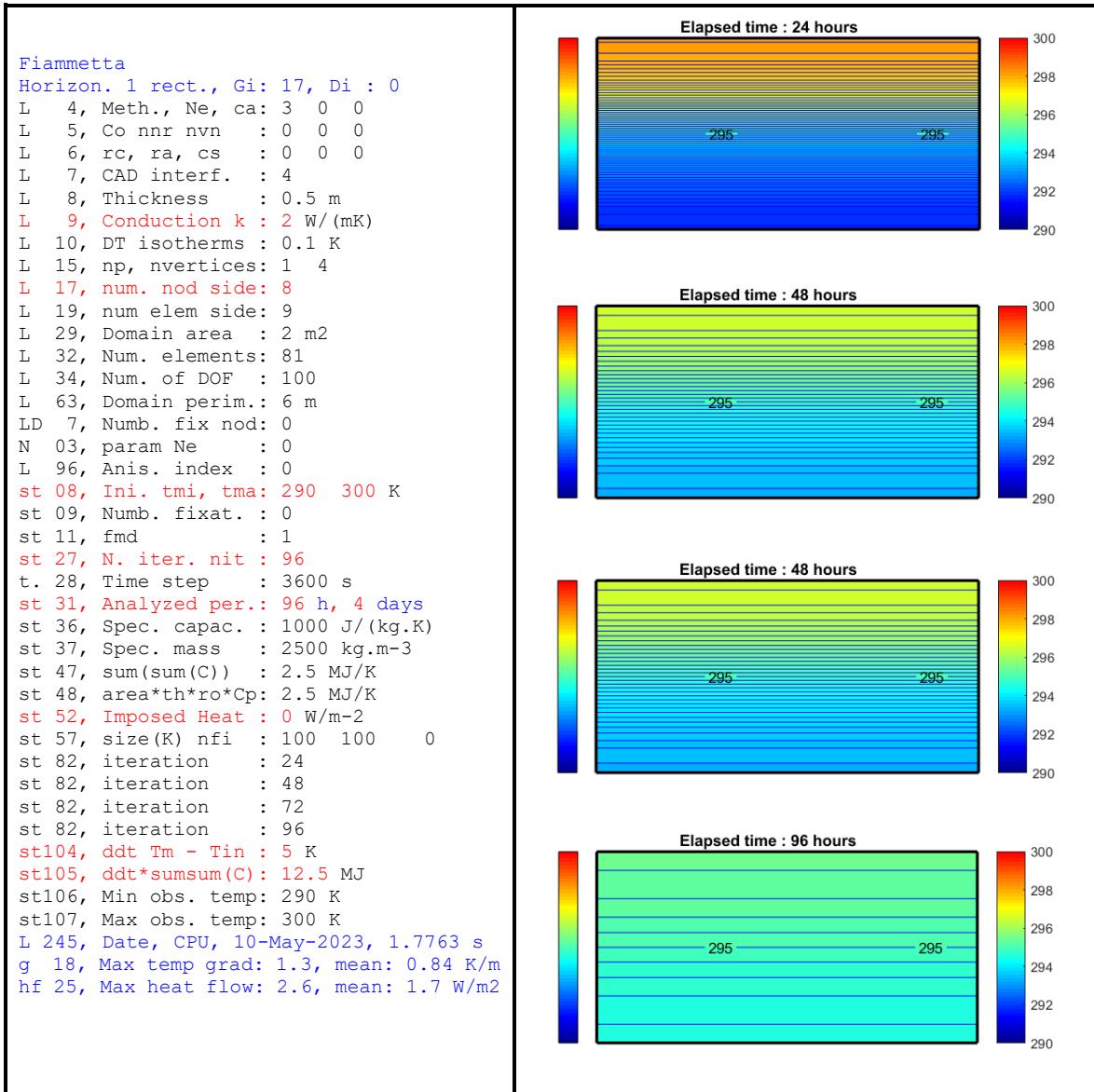
A uniform temperature test is carried out on a domain of dimensions (2 m x 1m x 0.5 m) whose walls are adiabatic. Half the area is at 300 K, half at 290 K (*line 5* to *line 12* in `fem_til.m`, *Table 28*, *Table 39*). The time evolution and the moment at which the temperature becomes uniform are examined. The homogenization process depends on the diffusivity.

Input data: conductivity coefficient = 2  $Wm^{-1}K^{-1}$ , specific capacity = 1000  $Jkg^{-1}K^{-1}$ , specific mass = 2500  $kgm^{-3}$ , the heat capacity of the domain is obtained with the Matlab<sup>®</sup> instruction: `sum (sum (C))`. For this example, it is equal to  $2.5 \cdot 10^6 JK^{-1}$  (verified with the explicit formula:

`cap = area*th*Cp*ro * 1e-6;`). After 33.2 *hours*, the temperature gap is reduced by half: 5 *K*. Homogeneity of the temperature is obtained after 180 *hours*, when the temperature gap is equal to 0.1 *K* (*Figure 34*). At the displayed time step of 24, 48, 72 and 96 *hours*, the temperature gap is decreasing in the following sequence: 6.5 *K*, 3.3 *K*, 1.7 *K*, and finally 0.8 *K*. To ensure the coherence of the data with the mesh, it is mandatory to impose an even number of nodes per patch side. Here, the flag *Gi = 17* is used in the function *fem\_til.m* (*lines 8 to 14*) to specify these unusual initial conditions: half of the domain at 290 *K* and half at 300 *K*.



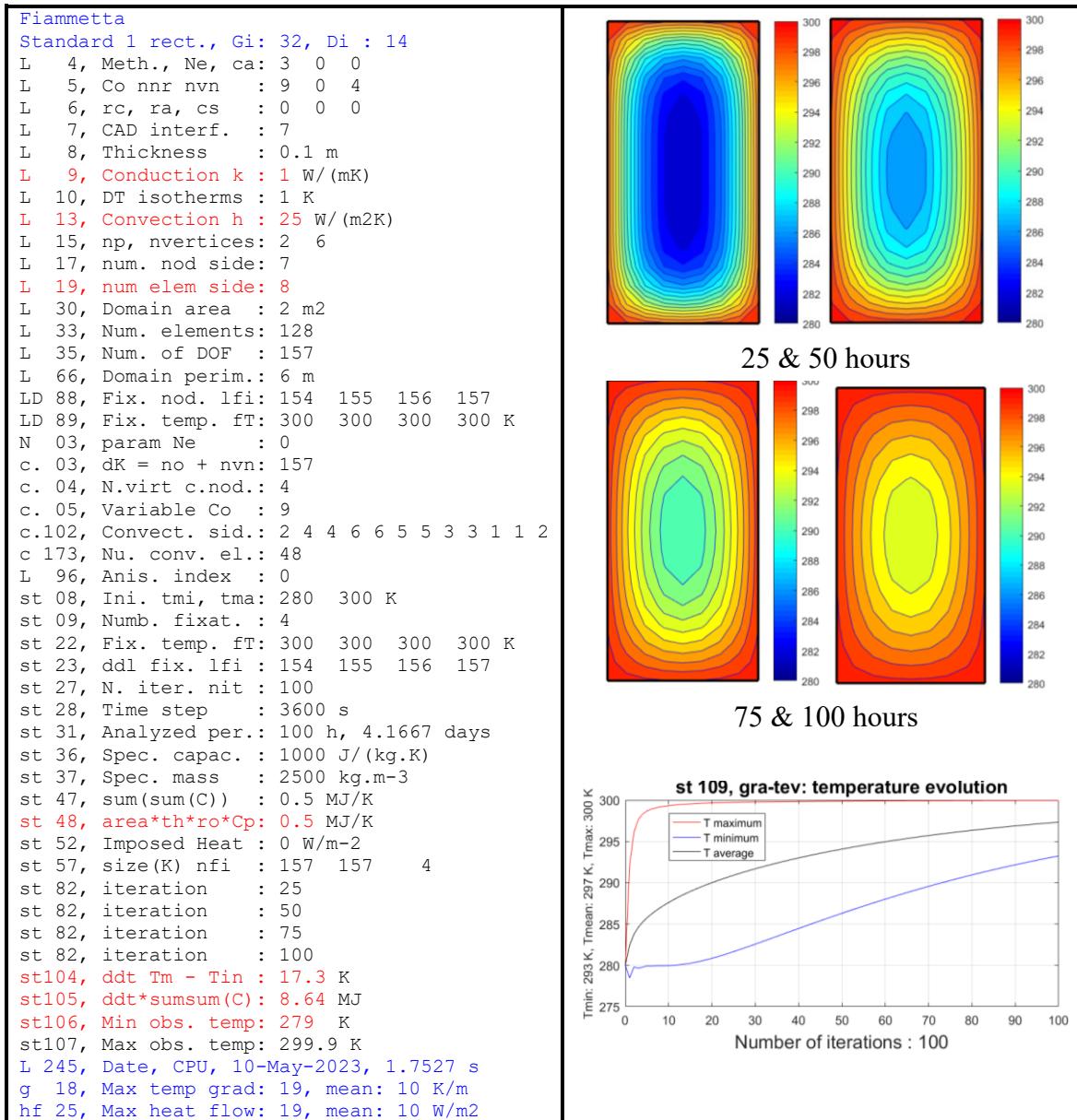
*Figure 34: Smoothing process convergence in temperature homogenization*



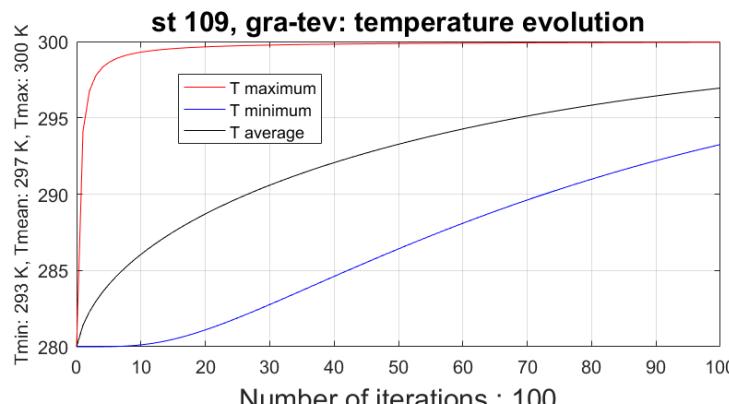
*Figure 35: Evolution of the temperature field in a smoothing process*

## 4.5 Heating of a solid at initial uniform temperature

The next example ([Figure 36](#), [Figure 38](#)) relates to the heating of a solid immersed in a fluid at 300 K, with convective heat exchanges on the four sides of the solid. At the beginning (initial boundary conditions), the temperature of the solid is 280 K. After 100 hours, the mean temperature is 296 K. The quantity of exchanged heat is equal to the product of the temperature growth by the specific heat and by the mass of the solid (output [st105](#)).



*Figure 36: Evolution of the temperature field in a heating process*

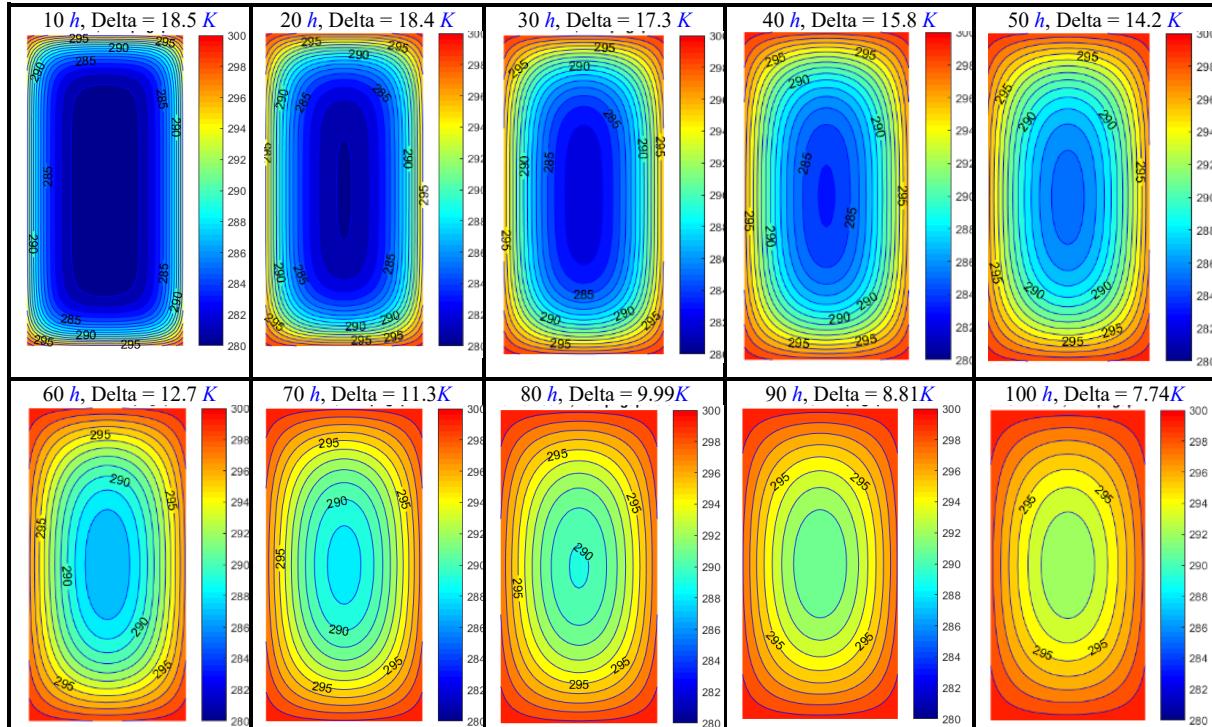


*Figure 37: Evolution of specific temperatures in a heating operation*

With the above relatively coarse mesh ( $8 \times 8 \times 2$  elements), we obtain some unforeseen results in the picture of temperatures evolution. In *Figure 36*, at the beginning of the process, the minimum temperature inside the mesh is decreasing; five iterations are needed for the temperature to become again greater than the initial one. According to the second law of thermodynamics, this behavior is not physically acceptable. However, the convergence is very similar to that of *Figure 37*; in both cases, the final temperatures are the same.

In *Figure 37* and *Figure 38*, the mesh involves 3200 conductive elements, 3325 *DOF*. Some outputs are given here:

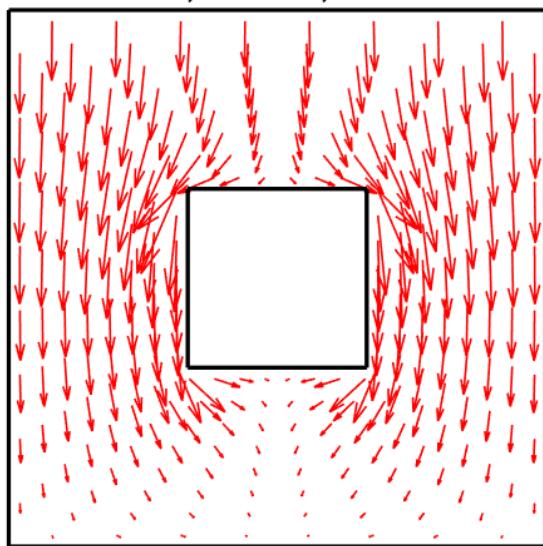
```
st104, ddt Tm - Tin : 16.9 K
st105, ddt*sumsum(C) : 8.47 MJ
st106, Min obs. temp: 280 K
st107, Max obs. temp: 299.9 K
L 248, Date, CPU, 05-Feb-2023, 72.8782 s
```



*Figure 38: Evolution of isotherms in a heating operation: 100 h*

#### 4.6 Adiabatic cavity with imposed temperatures on the top of the domain

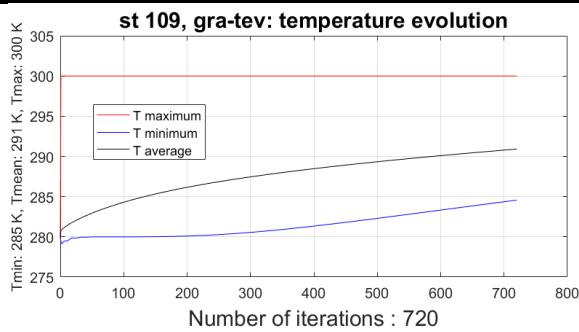
```
Fiammetta
Standard cavity, Gi: 1, Di : 1
L 4, Me, Ne : 3 0 1
L 5, Co nnr nvn : 0 0 0
L 6, rc, ra, cs : 0 0 0
L 7, CAD interf. : 12
L 8, Thickness : 0.1 m
L 9, Conduction k : 2 W/(mK)
L 10, DT isotherms : 1 K
L 15, np, nvertices: 4 8
L 17, num. nod side: 7
L 19, num elem side: 8
L 29, Domain area : 8 m2
L 32, Num. elements: 256
L 34, Num. of DOF : 288
L 63, Domain perim.: 16 m
ID 12, Fixed nodes : 3 58 ... 64 4
N 03, param Ne : 0
L 96, Anis. index : 0
st 08, Ini. tmi, tma: 280 300 K
st 09, Numb. fixat. : 9
st 27, N. iter. nit : 720
t. 28, Time step : 3600 s
st 31, Analyzed per.: 720 h, 30 days
```



```

st 36, Spec. capac. : 1000 J/(kg.K)
st 37, Spec. mass   : 2500 kg.m-3
st 47, sum(sum(C)) : 2 MJ/K
st 48, area*th*ro*Cp: 2 MJ/K
st 52, Imposed Heat : 0 W/m2
st 57, size(K) nfi : 288 288 9
st 82, iteration    : 180
st 82, iteration    : 360
st 82, iteration    : 540
st 82, iteration    : 720
st104, ddt Tm - Tin : 10.9 K
st105, ddt*sumsum(C) : 21.8 MJ
st106, Min obs. temp: 280 K
st107, Max obs. temp: 300 K
L 245, Date, CPU, 10-May-2023, 3.3639 s
hf 25, Max heat flow: 20, mean: 9.3 W/m2

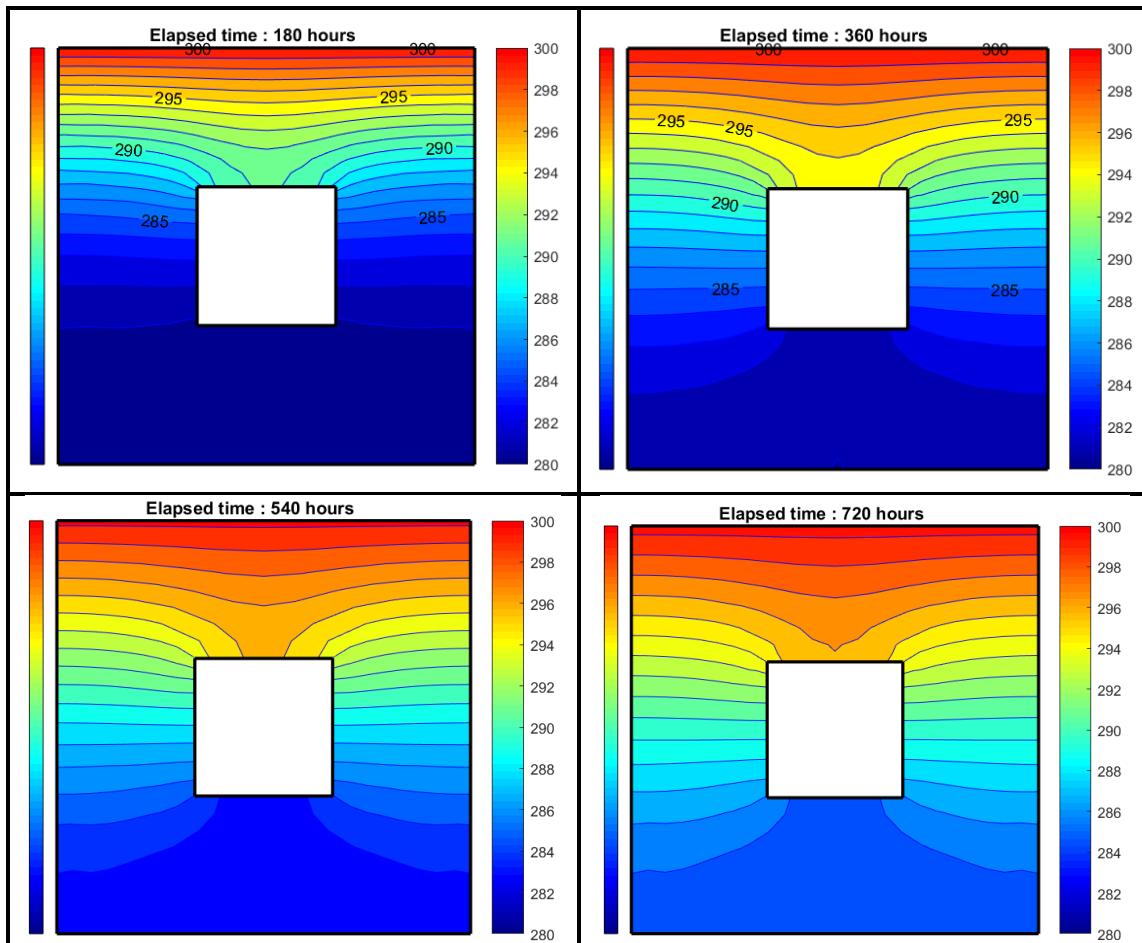
```



*Figure 39: Adiabatic cavity, 1 month,  $T_{ini} = 280 \text{ K}$ ,  $T_{top} = 300 \text{ K}$*

The results of *Figure 39* confirms that the stored heat (*t.101*) is equal to  $21.8 \text{ MJ}$  which is the product of the total capacity (*t.47*) of  $2 \text{ MJ/K}$  by the difference of temperature of  $10.9 \text{ K}$  (*t.100*). To display the evolution of the temperature field in the domain, it is mandatory to use the same color bar for all the drawings. It is obtained thanks to the definition and the display of the thin vertical bar at the left of the drawing. In *Figure 40*, all the illustrations correspond to the temperature gap:  $280 - 300 \text{ K}$ .

To enforce the colorbar to act always between the initial temperature *tmi* and  $300 \text{ K}$ , the Matlab<sup>®</sup> instructions *80 & 83* are enabled in *fem\_til.m*.



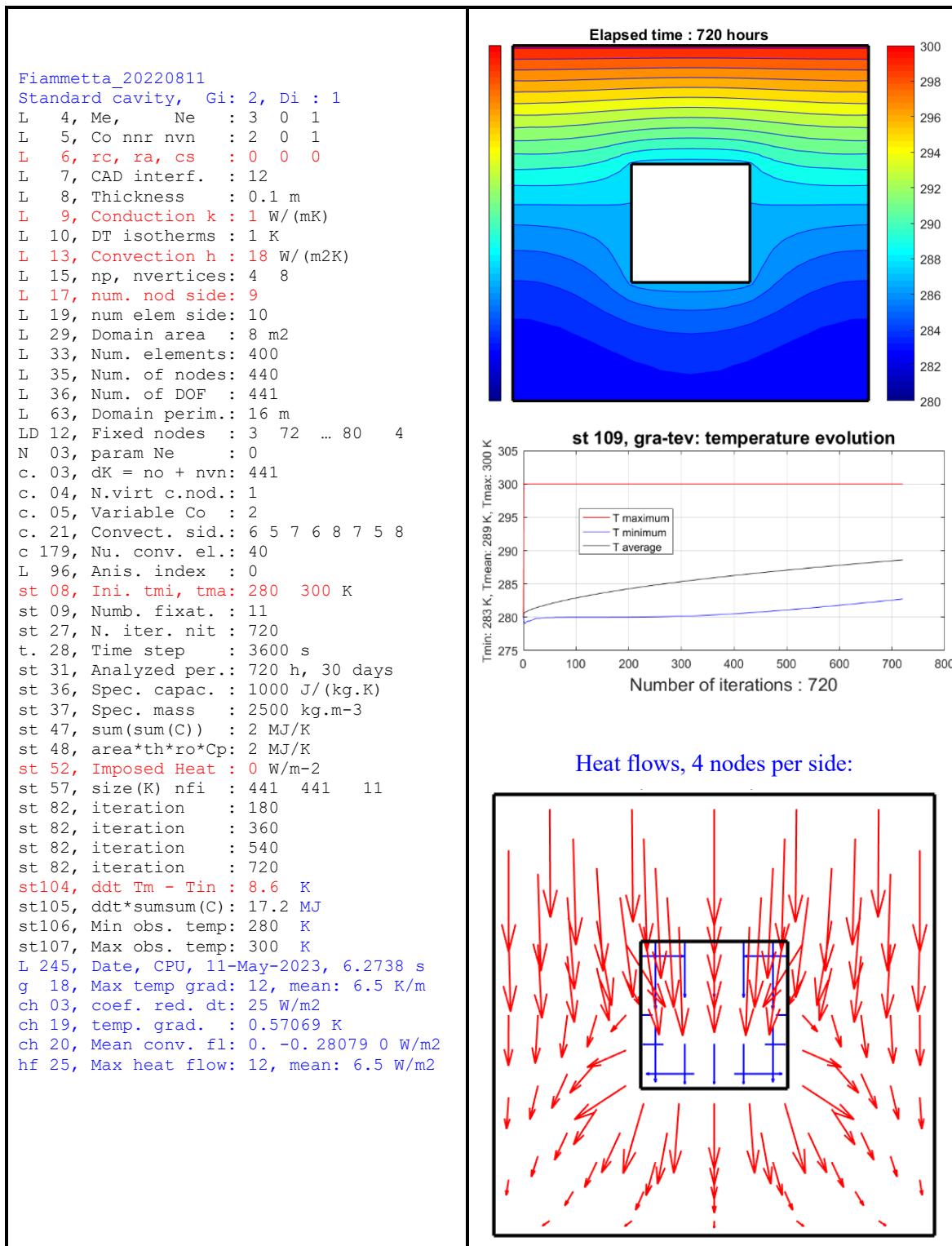
*Figure 40: Adiabatic cavity, evolution of the temperature field*

## 4.7 Square cavity with convection

The interaction of the internal fluid contained in a cavity with the solid continuum is determined by the convection laws.

The heat flow in the conductive medium appears more clearly in the element-by-element heat flows drawing. The flow enters in the cavity basically from above and leaves it on almost all of the other three sides. The arrows representing the flows are orthogonal to the isotherms and their lengths are inversely proportional to the distances between successive isothermal lines.

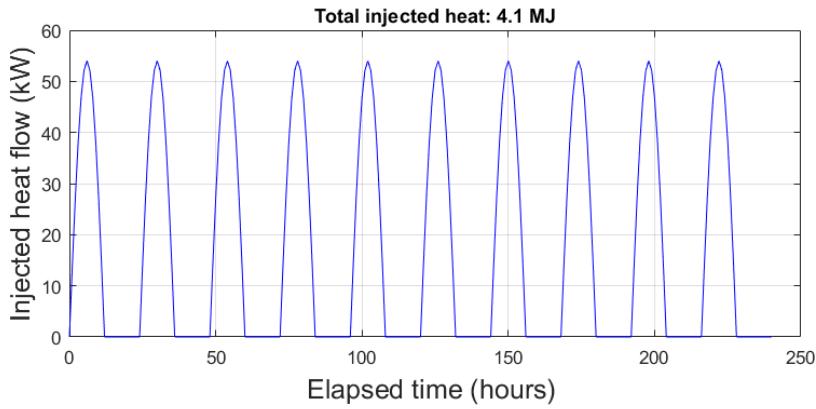
An additional possibility is to show the heat flows in the convective elements as seen in blue in the next table. This action is performed in the `gra_chf.m` function (*Table 58*).



*Figure 41: Isotherms, convection in the cavity, 1 month*

## 4.8 Periodic imposed heat flow

The problem addressed here is the heating of a domain from its upper horizontal border. The periodic heating function can be applied on any patch side ([Table 9](#)).



*Figure 42: Thermal loading over a period of 10 days (240 hours)*

The drawing of *Figure 42* is created only if the time step (variable *dth*) is equal to 1 hour ([line 233](#) in *Fiammetta.m*, [Table 28](#)). The function *gra\_hie.m* ([Table 62](#)) is called at [line 334](#) of *Fiammetta.m*. The time function *f(t)* used to weight the heat flow input is given by:

$$f(t) = \begin{cases} \sin\left(\frac{2t}{p}\pi\right) & \text{if } 0 \leq t \leq \frac{p}{2} \\ 0 & \text{if } \frac{p}{2} \leq t \leq p \end{cases} \quad (85)$$

This function is adimensional, *p* is the period of the sinusoidal function and *t* the time, expressed in the same unit as the period *p*. Over the semi-period *p/2*, the average of the function is:

$$\frac{2}{p} \int_0^{p/2} \sin\left(\frac{2t}{p}\pi\right) dt = -\frac{2}{p} \left[ \cos\left(\frac{2t}{p}\pi\right) \frac{p}{2\pi} \right]_0^{p/2} = \frac{-1}{2\pi} (-1 - 1) = \frac{2}{\pi} \approx 0.6366 \quad (86)$$

Then, over the period *p*, the average of the time weighting function *f(t)* is  $1/\pi$  or 0.3183. If the intensity of the imposed heat flow is *i<sub>h</sub>* = 50  $\text{Wm}^{-2}$  and if the time is expressed in seconds, the heat flow is:

$$\bar{q}_n = i_h f(t) \text{ Wm}^{-2} = i_h \sin \frac{t \pi}{12 * 3600} \text{ Wm}^{-2} = i_h \sin \frac{t \pi}{12 * 3600} \text{ Wm}^{-2} \quad (87)$$

The area of the boundary where heat is injected is:

$$s_b = e \times L = 0.1 * 3 = 0.3 \text{ m}^2 \quad (88)$$

In this expression, *e* is the thickness of the solid, *L* the length of the border and *i<sub>h</sub>* is the imposed heat flow density. The imposed heat flow (*W*) is:

$$\bar{q}_n b_s = i_h e L f(t) \text{ W} = i_h e L f(t) \text{ W} = i_h e L \sin \frac{t \pi}{12 * 3600} \text{ W} = i_h e L \sin \frac{t \pi}{43200} \text{ W} \quad (89)$$

In one day, the injected heat is equal to *h<sub>d</sub>*, now expressed in joules:

$$h_d = \int_0^{43200} \bar{q}_n b_s dt \quad \text{J} = i_h e L \int_0^{43200} \sin \frac{t\pi}{43200} dt \quad \text{J} = i_h e L \left[ -\frac{43200}{\pi} \cos \frac{t\pi}{43200} \right]_0^{43200} \quad \text{J}$$

$$h_d = \frac{i_h e L 43200 * 2}{\pi} \quad \text{J} = 0.2475 i_h \quad \text{MJ}$$
(90)

With a heat flow density  $i_h = 50 \text{ Wm}^{-2}$  and a loaded area  $eL = 0.3 \text{ m}^2$ , the total incoming heat flow after 30 days is:  $h_d * 30 = 12.4 \text{ MJ}$ .

According to the input data of [Figure 62](#), the instruction “tt = 720\*3600; ih = 50; bos = .3; hdm = tt\*ih/pi\*bos\*1e-6; disp ( [‘Lt142, Ex.heat input:’, num2str(hdm,3), ‘MJ’] )” gives the result: “Lt142, Ex.heat input: 12.4 MJ”

## 4. Tutorial V: Radiosity

### 5.1 Theoretical background

This chapter refers to longwave radiative exchanges [\[Beckers 2013\]](#). The variables are both radiosities and surface temperatures. In radiative heat transfers, the first step is to compute the topological and geometrical relations between radiating surfaces. The view factor or form factor  $F_{ij}$  represents the fraction of radiant energy leaving surface  $I$  and impinging on surface  $j$  [\[Goral et al 1984\]](#).

In 2D [\[Beckers 2011\]](#), the “*point – segment*” view factor relates a point to a segment ([Figure 43](#)) according to the formula:

$$F_{dL-j} = \int_{y \in L_j} \frac{\cos \theta_{dL} \cos \theta_j}{2r} dy \quad (91)$$

The explicit expression of the view factor is computed via the Nusselt analogy ([Figure 43](#)) [\[Nusselt 1928, Beckers et al, 2009, Beckers & Beckers 2014\]](#): it is the projection of the circular arc intercepted by the two rays on the base diameter of the semi-circle. This projection must then be divided by the area of the base circle or, here, by the length of the base diameter (= 2, because the radius of the circle = 1). When calculating the upper semicircle on a polygon spanning, the sum of the view factors is equal to the diameter of the semicircle. The view factor is therefore given by:

$$F_{dL-j} = \frac{1}{2} \left( \frac{x_1}{r_1} - \frac{x_0}{r_0} \right) \quad (92)$$

To obtain the view factors of the elements of a surrounding square from a differential element  $dL$  ([Figure 43](#)) situated on one of its sides, we perform a numerical integration on the other sides of the square. This is completed using Gaussian quadrature. The point-segment view factors are evaluated at the various Gauss points of the square sides, the number of which is determined by the desired precision. The weighted sum of these values constitutes the approximation of the integral. The *closure* condition expresses that the sum of the view factors is equal to 1 for each row of the matrix  $[F]$ :

$$\sum_{j=1}^N F_{ij} = 1 \quad \text{closure} \quad (93)$$

The view factors must satisfy a second condition: the *reciprocity*

$$A_i F_{ij} = A_j F_{ji} \quad \text{reciprocity} \quad (94)$$

In (95), the terms  $A_i$  and  $A_j$  define the areas of the patches linked by the view factors (here, the lengths of the segments times their thicknesses). If the considered segment is not horizontal,

the form factor is calculated with a generalization of formula (92) to the side on which the element  $dL$  is located. Assuming that  $\vec{r}_i / |\vec{r}_i|$  is the unit vector joining the studied segment to the extremities  $l = 0$  and  $l = 1$  of the target segments and  $\vec{t}$  the tangent to the studied segments. The “*point – segment*” view factor is:

$$F_{dL-j} = \frac{1}{2} \left( \frac{\vec{r}_1}{|\vec{r}_1|} - \frac{\vec{r}_0}{|\vec{r}_0|} \right) \cdot \vec{t} \quad (95)$$

The view factor  $F_{ij}$  is obtained by integrating this “*point – segment*” view factor over patch  $i$ :

$$F_{ij} = \frac{1}{A_i} \int_i F_{dL-j} dA_j \quad (96)$$

In this relation,  $dA_j$  is the product of the differential length  $dL$  of element  $j$  by its thickness while  $A_i$  is the product of the length of element  $i$  by its thickness.

The next development is based on [Lobo & Emery 1995], [Rupp & Péniguel 1999], [Coulon 2006], [Beckers 2020 a, b, c] and [van Eekelen 2012]. This formulation facilitates the transcription to *Matlab*<sup>©</sup> code. Capital letters represent vectors (seen as uni-column matrices) and capital letters between brackets represent matrices.

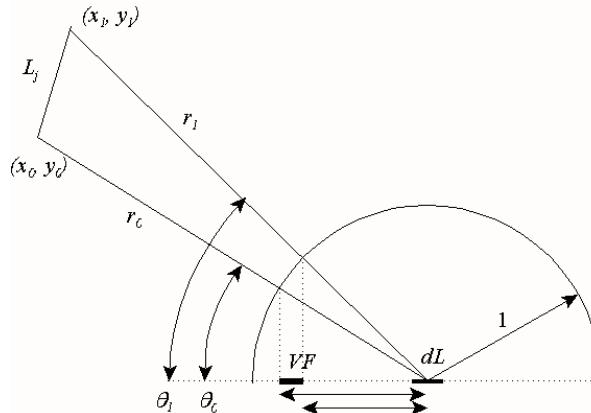


Figure 43: 2D “*point – segment*” view factor [Beckers 2011]

The equilibrium of the radiative heat exchanges on the surface of the solid expresses that the *radiosity*  $B$  is the sum of the *exitance*  $E$  and the *reflected irradiance*  $[R] J$ , [Goral et al 1984]. The three quantities  $B$ ,  $E$  and  $J$  are expressed in  $Wm^{-2}$

$$B = E + [R] J \quad (97)$$

In (98), the adimensional diagonal matrix  $[R]$  contains the reflection coefficients. They satisfy the Kirchhoff law relating, in each element  $i$ , the emissivity  $\varepsilon_i$ , the absorptivity  $\alpha_i$ , and the reflectivity  $\rho_i$ :

$$R_{ii} = \rho_i \quad ; \quad \varepsilon_i = \alpha_i \quad ; \quad \rho_i = 1 - \varepsilon_i \quad (98)$$

As shown in (98), the radiosity component  $B_i$  is the radiant flux leaving the surface  $i$  of the solid domain, it is the sum of the emitted  $E_i$  and the reflected  $\rho_i J_i$  fluxes.

As shown in (100), the incoming irradiance vector  $J$  is connected to the other elements of the domain by the view factor matrix  $[F]$  defined in (97). When they are visible, it is also related to the sky by the sky view factor uni-column matrix  $F_{sky}$  and to the infinite horizontal plane assimilated to the ground, by the ground view factor uni-column matrix  $F_{gr}$  (nearby, we use magenta color for the quantities related to the sky and the ground):

$$J = [F]B + F_{sky}E_{sky} + F_{gr}E_{gr} \quad (99)$$

Introducing the incoming irradiance (100) in (98), we have:

$$B = E + [R]\{[F]B + F_{sky}E_{sky} + F_{gr}E_{gr}\} \quad (100)$$

The radiosity matrix  $[M]$  depends on the view factor and the reflection matrices ( $[I]$  is the identity matrix)

$$[M] = [I] - [R][F] \quad (101)$$

The exitance vector  $E$  can be expressed as a function of the radiosity vector  $B$ :

$$\begin{aligned} E &= B - [R][F]B - [R]\{F_{sky}E_{sky} + F_{gr}E_{gr}\} \\ &= \{[I] - [R][F]\}B - [R]\{F_{sky}E_{sky} + F_{gr}E_{gr}\} \\ &= [M]B - [R]\{F_{sky}E_{sky} + F_{gr}E_{gr}\} \end{aligned} \quad (102)$$

From (102), we deduce:

$$B = [M]^{-1}\{E + [R]\{F_{sky}E_{sky} + F_{gr}E_{gr}\}\} \quad (103)$$

The *radiative load* vector  $Q (Wm^{-2})$  is obtained by subtracting the *incoming* flux  $J (Wm^{-2})$  to the *radiosity outgoing* flux  $B (Wm^{-2})$ :

$$Q = B - J \quad (104)$$

Replacing in (105)  $J$  by the result of (100), we obtain:

$$\begin{aligned} Q &= B - [F]B - \{F_{sky}E_{sky} + F_{gr}E_{gr}\} \\ &= \{[I] - [F]\}B - \{F_{sky}E_{sky} + F_{gr}E_{gr}\} \end{aligned} \quad (105)$$

Replacing  $B$  by the result of (101), leads to:

$$Q = \{[I] - [F]\}[M]^{-1}E + \{[I] - [F]\}[M]^{-1}[R]\{F_{sky}E_{sky} + F_{gr}E_{gr}\} - \{F_{sky}E_{sky} + F_{gr}E_{gr}\} \quad (106)$$

This solution involves two contributions, the first one is related to the *exitance*  $E$  and the second one to  $E_{sky}$  and  $E_{gr}$

$$Q = \{[I] - [F]\}[M]^{-1}E + \{[I] - [F]\}[M]^{-1}[R] - [I]\{F_{sky}E_{sky} + F_{gr}E_{gr}\} \quad (107)$$

The exitances components,  $E_i (Wm^{-2})$ , of the vector  $E$  introduced in (102) may be calculated as functions of the surface temperature field components  $T_i$  of the boundary elements  $i$ , the emissivities  $\varepsilon_i$  and the Stefan-Boltzmann constant  $\sigma (Wm^{-2}K^{-4})$ :

$$E \rightarrow E_i = \varepsilon_i \sigma T_i^4 \quad (108)$$

**Mirror.** If the surfaces are perfectly reflective (like mirrors):  $[R] = [I]$ , the emissivities  $\varepsilon_i$  are equal to zero and the vector  $E$  (109) is also equal to zero. Moreover, according to the definition of the radiosity matrix (102), the coefficient related to sky and ground is transformed as follows. However; we have to note that the radiosity matrix becomes singular and that the expression (110) is satisfied with  $\rho = .9999$

$$\{\{[I] - [F]\}[M]^{-1}[R] - [I]\} = \{[I] - [F]\}\{[I] - [F]\}^{-1} - [I] = [I] - [I] = 0 \quad (109)$$

As a consequence, the equation (108) becomes  $Q = E = 0$ . It means that the surface is adiabatic and thus does not emit nor absorb any heat flow.

**Black body.** For a black body,  $[R] = 0$ , the emissivities  $\varepsilon_i$  are equal to 1 and the vector  $E$  (109) is proportional to the Stefan-Boltzmann constant and the fourth power of the surface temperatures. Equation (108) becomes:

$$Q = E - \{F_{sky}E_{sky} + F_{gr}E_{gr}\} \quad (110)$$

**Gray body.** The first step is to express equation (108) as a function of the temperatures. As seen in (109), the vector  $E$  is containing components in  $t^4$ . So, we will write this vector with the following notation:

$$E = S^T T \text{ where } S_i T_i = (\varepsilon_i \sigma T_i^3)^{it-1} (T_i)^{it} \quad (111)$$

If present, the sky and ground exitances are obtained in the same way as (109):

$$\begin{aligned} E_{sky} &\rightarrow E_{sky} = \varepsilon_{sky} \sigma T_{sky}^4 = \sigma T_{sky}^4 \text{ (if sky emissivity = 1)} \\ E_{gr} &\rightarrow E_{gr} = \varepsilon_{gr} \sigma T_{gr}^4 = \sigma T_{gr}^4 \text{ (if sky emissivity = 1)} \end{aligned} \quad (112)$$

$$Q = \{[I] - [F]\}[M]^{-1} E + \left\{ \{[I] - [F]\}[M]^{-1}[R] - [I] \right\} \{F_{sky}E_{sky} + F_{gr}E_{gr}\} \quad (113)$$

Throughout (105) to (114),  $Q$  is a function of the surface temperature of the radiating solid and can therefore be injected in the finite element model.

$$Q = \{[I] - [F]\}[M]^{-1} S^T T + \left\{ \{[I] - [F]\}[M]^{-1}[R] - [I] \right\} \{F_{sky} \varepsilon_{sky} \sigma T_{sky}^4 + F_{gr} \varepsilon_{gr} \sigma T_{gr}^4\} \quad (114)$$

In (114), the vector  $E$  is changing at each iteration step. The last two terms of (114) correspond to the sky and ground radiations which have both imposed temperatures. These terms are classical second members of the system. In (114), all the quantities are expressed in variables resulting from a discretization based not on the finite element mesh, but on element interfaces (surfaces in 3D and edges in 2D).

To insert the relation (114) in the finite element model, we have to distribute the mid-edge loads on the adjacent nodes. This process is not trivial, except if the number of radiative nodes is equal to the number of radiative edges, which is the case if the radiative contour is closed.

In many other situations, we have more nodes than edges, as it is the case in 3D models (for instance, a cube has 6 faces and 8 vertices) and with open contours. In this situation we have to define a rule that allows a uniform distribution and respects the total flow.

We have thus to compute the nodal *radiant* fluxes on the two nodes (0 & 1) surrounding the element  $i$ :  $h_{0i}$  and  $h_{1i}$ . The fluxes satisfy the relation:  $h_{0i} + h_{1i} = a_i q_i$ , where  $a_i$  is the area of the edge  $i$  (length time thickness) and  $q_i$  the heat flow on the edge. Because these edges are on the boundary of the mesh, each side  $i$  appears only once and the connected nodes no more than twice. Finally, the nodal radiant fluxes  $h_j$  are weighted averages of the edge fluxes  $q_i$ . Formally, we can write:

$$H = [W] Q \quad (115)$$

We have still to overcome a last problem: the vector  $E$  concerns surface elements involving temperatures  $T_i$  at the fourth power, with the consequence that the finite element problem is highly nonlinear. To overcome this problem, we replace the temperatures of iteration  $it$  by those of iteration  $it-1$ :

$$E_i = \varepsilon_i \sigma (\tau_i^4)_{it-1} \rightarrow \varepsilon_i \sigma (\tau_i^3)_{it-1} (\tau_i)_{it} \quad (116)$$

For the sky and the ground, the relations are:

$$e_{sky} = \sigma \tau_{sky}^4 \quad \& \quad e_{gr} = \sigma \tau_{gr}^4 \quad (117)$$

Equation (113) becomes:

$$\begin{aligned} Q = & \left[ \{[I] - [F]\} [M]^{-1} \varepsilon_i \sigma (\tau_i^3)_{it-1} \right] (\tau_i)_{it} \\ & + \left\{ \{[I] - [F]\} [M]^{-1} [R] - [I] \right\} \{F_{sky} \sigma \tau_{sky}^4 + F_{gr} \sigma \tau_{gr}^4\} \end{aligned} \quad (118)$$

This relation exhibits one matrix (first line, inside blue brackets) that must be added to the conductivity one, and two second member vectors. Both have to be expressed in nodal variables and inserted in the global finite element system.

**Closed radiative enclosure.** This is the typical situation of the ovens where radiative exchanges are dominating. The combination of (112) and (119), gives:

$$Q = F_{sid} = \{[I] - [F]\} [M]^{-1} S^T T_{sid} \quad (119)$$

To introduce these equations in the global system, we have to transform the side variables in nodal variables, both for heat flows and temperatures. Assuming that we have  $n$  nodes or sides on the enclosure boundary and if the elements of the domain are isoparametric bilinear elements, we have:

$$\begin{bmatrix} T_{sid}(1) \\ T_{sid}(2) \\ T_{sid}(3) \\ T_{sid}(4) \\ \vdots \\ T_{sid}(n) \end{bmatrix} = 0.5 \begin{bmatrix} 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & 1 & 1 & & \\ & & & 1 & 1 & \\ & & & & 1 & 1 \\ 1 & & & & & 1 \end{bmatrix} \begin{bmatrix} T(1) \\ T(2) \\ T(3) \\ T(4) \\ \vdots \\ T(n) \end{bmatrix} \text{ or } [T_{sid}] = [L_c][T] \quad (120)$$

For the heat flows, we have the relation:

$$[F] = [L_c]^T [F_{sid}] \quad (121)$$

This relation is expanded according to (120).

$$[F] = [L_c]^T F_{sid} = [L_c]^T \{[I] - [F]\} [M]^{-1} S^T [L_c][T] \quad (122)$$

Finally, the “conductivity” matrix related to the radiative exchanges is:

$$[K_{rad}] = [L_c]^T \{[I] - [F]\} [M]^{-1} S^T [L_c] \quad (123)$$

According to its definition given in (112), the vector  $S$  involves coefficients of third power of side temperatures computed in a previous iteration. The final expression is:

$$[F] = [K_{rad}] T \quad (124)$$

**Open radiative zone.** This is the typical situation of a street section where radiative exchanges exist between the street elements and also with the sky or the ground. If we have only to consider the sky, the equation becomes:

$$[K_{rad}] = [L_o]^T \{[I] - [F]\} [M]^{-1} S^T [L_o] \quad (125)$$

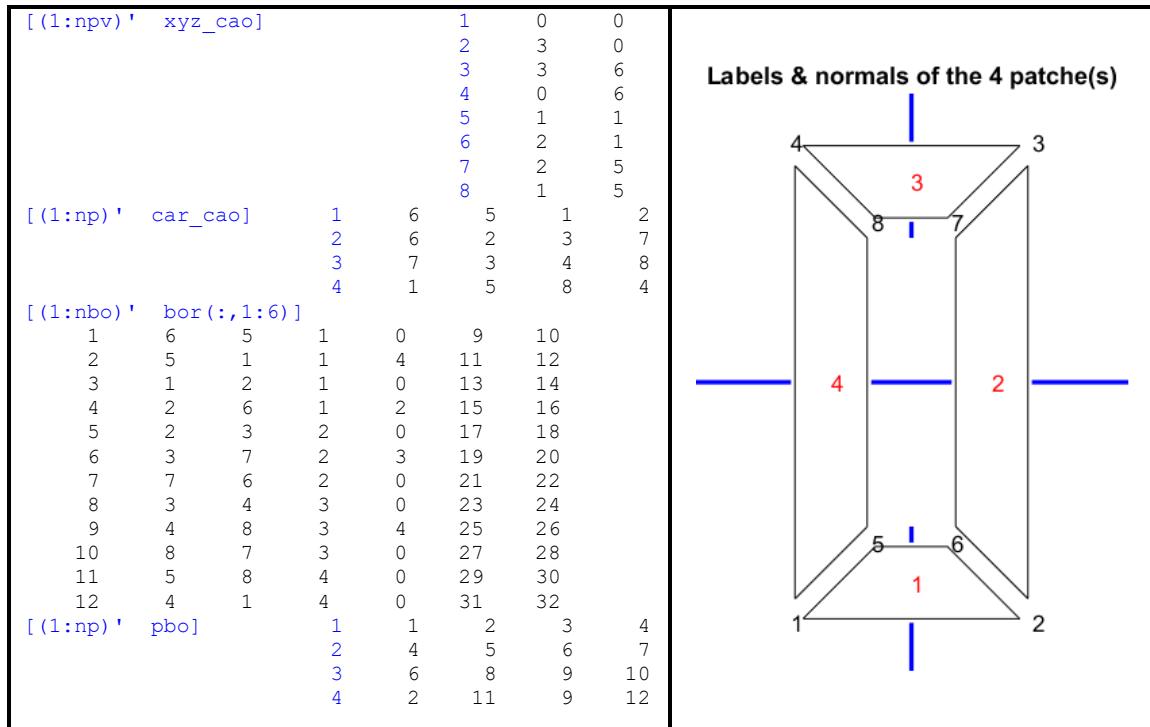
$$[F] = [K_{rad}][T] + L_{sky}^T \left[ \{[I] - [F]\} [M]^{-1} [R] - [I] \right] F_{sky} \sigma \tau_{sky}^4 \quad (126)$$

$$\begin{bmatrix} T_{sid}(1) \\ T_{sid}(2) \\ T_{sid}(3) \\ \vdots \\ T_{sid}(n) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & & & & & \\ & 1 & 1 & & & & \\ & & 1 & 1 & & & \\ & & & & 1 & 1 & \\ & & & & & 1 & 1 \end{bmatrix} \begin{bmatrix} T(1) \\ T(2) \\ T(3) \\ T(4) \\ \vdots \\ T(n) \end{bmatrix} \text{ or } [T_{sid}] = [L_o][T] \quad (127)$$

## 5.2 View factor matrix

*Table 37* gives the instructions for the generation of the list of nodes concerned with radiative heat exchange in three situations: cavity, street section and balcony located on the left side of the meshed domain. The vector *lv* provides the listing of the patch vertices while the vector *lcont* is giving the listing of the nodes involved in the radiative exchanges. Data of a rectangular cavity are shown in *Figure 44*.

### → a) Quadrilateral cavities



*Figure 44: Rectangular cavity with the display of four matrices describing the geometry*

The view factors are calculated using the “*point – segment*” method (96). In *Figure 44*, the cavity is oriented with the sides parallel to the global axes. If all the elements have the same length, the view factor matrix is symmetric according to the *reciprocity property* (95) and both the

sums of columns and lines are equal to 1 (*closure property* (94)). *Table 11* shows the instructions used to visualize the view factor matrix and the *closure property* when there is only 1 element per square cavity side. The view factor matrix is computed in *geo\_vfc.m* (*Table 67*). This function allows computing the view factor matrix for a rectangle whose length and height are stored in the 2 x 1 uni-column matrix *Lel* () .

F = geo_vfc(1, [1 1], 4); disp(F); disp(sum(F)); disp(sum(F, 2)')			
0 0.2764 0.4472 0.2764	Closure tests		
0.2764 0 0.2764 0.4472			
0.4472 0.2764 0 0.2764			
0.2764 0.4472 0.2764 0			
n=1;Lel=[1 1 1 1];lon=zeros(1,n*4);k=0;for j=1:4;for i=1:n;k = k+1;lon(k)=Lel(j)/n;end;end;ns=size(lon,2);Reci=zeros(ns,ns);for i=1:ns;for j=1:ns;Reci(i,j)=lon(i)*F(i,j)-lon(j)*F(i,j);end;end;disp(Reci);			
0 0 0 0			
0 0 0 0			
0 0 0 0			
0 0 0 0			

*Table 11: Matrix F & reciprocity in a square cavity – 4 segments, 1 Gauss point*

From the middle of the bottom side of a square cavity, the “*point – segment*” view factor of the top face is equal to  $\sqrt{5}/5 = 0.4472$ ; the view factors related to the adjacent sides are then equal to  $(1 - \sqrt{5}/5) / 2 = 0.2764$ . We observe that only these two values are present in the matrix of *Table 11*. With numerical integration and 1 integration point (method used in *geo\_vfc.m*), the view factor of the opposite sides of a square is equal to 0.4472 (*Table 11*), and to 0.4130 with 2 Gauss points (*Table 12*). The exact value of 0.4142 is obtained with 3 integration points.

F = geo_vfr (1, [1 1]); disp(F); disp(sum(F)); disp(sum(F, 2)')							
0 0.2935 0.4130 0.2935					1.0000 1.0000 1.0000 1.0000		
0.2935 0 0.2935 0.4130							
0.4130 0.2935 0 0.2935							
0.2935 0.4130 0.2935 0					1.0000 1.0000 1.0000 1.0000		
n=1;Lel=[1 1 1 1];lon=zeros(1,n*4);k=0;for j=1:4;for i=1:n;k = k+1;lon(k)=Lel(j)/n;end;end;ns=size(lon,2);Reci=zeros(ns,ns);for i=1:ns;for j=1:ns;Reci(i,j)=lon(i)*F(i,j)-lon(j)*F(i,j);end;end;disp(Reci);							
0 0 0 0							
0 0 0 0							
0 0 0 0							
0 0 0 0							

*Table 12: Matrix F & reciprocity in a square cavity – 4 segments, 2 Gauss points*

F = geo_vfc (2, [1 1], 4); disp(F); disp(sum(F)); disp(sum(F, 2)')								
0 0 0.1023 0.1160 0.1787 0.2425 0.0840 0.2764								
0 0 0.2764 0.0840 0.2425 0.1787 0.1160 0.1023								
0.0840 0.2764 0 0.1023 0.1160 0.1787 0.2425 0.2425								
0.1160 0.1023 0 0 0.2764 0.0840 0.2425 0.1787								
0.1787 0.2425 0.0840 0.2764 0 0 0.1023 0.1160								
0.2425 0.1787 0.1160 0.1023 0 0 0.2764 0.0840								
0.1023 0.1160 0.1787 0.2425 0.0840 0.2764 0 0								
0.2764 0.0840 0.2425 0.1787 0.1160 0.1023 0 0								
disp(sum(F)); 1 1 1 1 1 1 1 1								
disp(sum(F, 2)')								
F = geo_vfc (2, [1 1], 4); round((F-F')/0.0184)								
0 0 1 0 0 0 -1 0								
0 0 0 -1 0 0 0 1								
-1 0 0 0 1 0 0 0								
0 1 0 0 0 -1 0 0								
0 0 -1 0 0 0 1 0								
0 0 0 1 0 0 0 -1								
1 0 0 0 -1 0 0 0								
0 -1 0 0 0 1 0 0								

*Table 13: Matrix F & reciprocity in a square cavity – 8 segments, 1 Gauss point*

F = geo_vfr (2,[1 1]); disp(F); disp(sum(F)); disp(sum(F,2)')								
0	0	0.0885	0.1148	0.1782	0.2360	0.0890	0.2935	
0	0	0.2935	0.0890	0.2360	0.1782	0.1148	0.0885	
0.0890	0.2935	0	0	0.0885	0.1148	0.1782	0.2360	
0.1148	0.0885	0	0	0.2935	0.0890	0.2360	0.1782	
0.1782	0.2360	0.0890	0.2935	0	0	0.0885	0.1148	
0.2360	0.1782	0.1148	0.0885	0	0	0.2935	0.0890	
0.0885	0.1148	0.1782	0.2360	0.0890	0.2935	0	0	
0.2935	0.0890	0.2360	0.1782	0.1148	0.0885	0	0	
disp(sum(F));		1	1	1	1	1	1	1
disp(sum(F,2)')		1	1	1	1	1	1	1
F = geo_vfr (2,[1 1]); round((F-F')/.0004719)								
0	0	-1	0	0	0	1	0	
0	0	0	1	0	0	0	-1	
1	0	0	0	-1	0	0	0	
0	-1	0	0	0	1	0	0	
0	0	1	0	0	0	-1	0	
0	0	0	-1	0	0	0	1	
-1	0	0	0	1	0	0	0	
0	1	0	0	0	-1	0	0	

Table 14: Matrix F & reciprocity in a square cavity – 8 segments, 2 Gauss points

With 8 segments, (Table 13, Table 14), the *reciprocity property* (95) is not perfectly satisfied, but *closure properties* are satisfied both for lines and columns. When the number of Gauss points is increasing, this lack of precision is decreasing. The Matlab functions *geo\_vfr.m* and *geo\_vfc.m* are not giving identical results, but they are compatible.

Similar tests are now performed in a rectangular cavity (Table 15)

F = geo_vfc(1,[1 4],4); disp(F); disp(sum(F,1)); disp(sum(F,2)'); lon = [1 4 1 4];								
0	0.0528	0.1240	0.0528					
0.4380	0	0.4380	0.8944					
0.1240	0.0528	0	0.0528					
0.4380	0.8944	0.4380	0					
disp(sum(F))	;columns	1	1	1	1			
disp(sum(F,2)');	lines	0.2296	1.7704	0.2296	1.7704			
ns=size(lon,2); Reci=zeros(ns,ns); for i=1:ns,for j=1:								
ns; Reci(i,j)=lon(i)*F(i,j)-lon(j)*F(j,i); end; end; disp(round(Reci/1.6991))								
0	-1	0	-1					
1	0	1	0					
0	-1	0	-1					
1	0	1	0					

Table 15: Rectangular cavity: F and reciprocity test– 4 segments, 1 Gauss point

F = geo_vfr (1,[1 4]); disp(F); disp(sum(F)); disp(sum(F,2)'); Lel=[1 4 1 4];								
0	0.1003	0.1231	0.1003					
0.4385	0	0.4385	0.7994					
0.1231	0.1003	0	0.1003					
0.4385	0.7994	0.4385	0					
disp(sum(F));				1	1	1	1	
disp(sum(F,2)');				0.3237	1.6763	0.3237	1.6763	
n=1;k=0;lon = zeros(1,n*4);ns=size(lon,2);Reci=zeros(ns,ns);								
for j=1:4;for i=1:n;k=k+1;lon(k)=Lel(j)/n;end;end;								
for i=1:ns;for j=1:ns;Reci(i,j)=lon(i)*F(i,j)-lon(j)*F(j,i);end;end;								
disp(round(Reci/1.6535))								
0	-1	0	-1					
1	0	1	0					
0	-1	0	-1					
1	0	1	0					

Table 16: Rectangular cavity: F and reciprocity test– 4 segments, 2 Gauss points

F = geo_vfc (2,[1 4],4); lon=[.5 .5 2 2 .5 .5 2 2]; ;disp(F); disp(sum(F)); disp(sum(F,2)'); disp(sum([sum(F,2)'])/8);								
0	0	0.0937	0.0189	0.0610	0.0624	0.0068	0.0528	
0	0	0.0528	0.0068	0.0624	0.0610	0.0189	0.0937	
0.3244	0.4380	0	0	0.0308	0.0834	0.1208	0.7071	
0.0834	0.0308	0	0	0.4380	0.3244	0.7071	0.1208	
0.0610	0.0624	0.0068	0.0528	0	0	0.0937	0.0189	
0.0624	0.0610	0.0189	0.0937	0	0	0.0528	0.0068	

0.0308	0.0834	0.1208	0.7071	0.3244	0.4380	0	0
0.4380	0.3244	0.7071	0.1208	0.0834	0.0308	0	0
<code>disp(sum(F)) ; columns 1 1 1 1 1 1 1 1</code>							
<code>disp(sum(F,2)'); lines 0.2954 0.2954 1.7046 1.7046 0.2954 0.2954 1.7046 1.7046</code>							
<code>disp(sum([sum(F,2)'])/8): 1</code>							
<code>ns=size(lon,2);Reci=zeros(ns,ns);</code>							
<code>for i=1:ns;for j=1:ns;Reci(i,j)=lon(i)*F(i,j)-lon(j)*F(i,j);end;end;</code>							
<code>disp(Reci); disp(sum(Reci));disp(sum(Reci,2)')</code>							
0	0	-0.1405	-0.0283	0	0	-0.0102	-0.0792
0	0	-0.0792	-0.0102	0	0	-0.0283	-0.1405
0.4867	0.6570	0	0	0.0462	0.1251	0	0
0.1251	0.0462	0	0	0.6570	0.4867	0	0
0	0	-0.0102	-0.0792	0	0	-0.1405	-0.0283
0	0	-0.0283	-0.1405	0	0	-0.0792	-0.0102
0.0462	0.1251	0	0	0.4867	0.6570	0	0
0.6570	0.4867	0	0	0.1251	0.0462	0	0
<code>disp(sum(Reci)); 1.3150 1.3150 -0.2582 -0.2582 1.3150 1.3150 -0.2582 -0.2582</code>							
<code>disp(sum(Reci,2)'); -0.2582 -0.2582 1.3150 1.3150 -0.2582 -0.2582 1.3150 1.3150</code>							

Table 17: Rectangular cavity:  $F$  and reciprocity test - 8 segments, 1 Gauss point

<code>F = geo_vfr(2,[1 4]);disp(F); disp(sum(F)); disp(sum(F,2)'); Lel=[1 4 1 4];</code>							
0 0 0.0912 0.0206 0.0608 0.0623 0.0076 0.1003							
0 0 0.1003 0.0076 0.0623 0.0608 0.0206 0.0912							
0.3255 0.4384 0 0 0.0304 0.0825 0.1634 0.6169							
0.0825 0.0304 0 0 0.4384 0.3255 0.6169 0.1634							
0.0608 0.0623 0.0076 0.1003 0 0 0.0912 0.0206							
0.0623 0.0608 0.0206 0.0912 0 0 0.1003 0.0076							
0.0304 0.0825 0.1634 0.6169 0.3255 0.4384 0 0							
0.4384 0.3255 0.6169 0.1634 0.0825 0.0304 0 0							
<code>disp(sum(F)) lines 1 1 1 1 1 1 1 1</code>							
<code>disp(sum(F,2)) columns 0.3428 0.3428 1.6572 1.6572 0.3428 0.3428 1.6572 1.6572</code>							
<code>n=2;lon = zeros(1,n*4);k=0;for j = 1:4;for I = 1:n;k = k+1; lon(k) = Lel(j)/n; end; end; ns=size(lon,2);Reci=zeros(ns,ns);for i=1:ns;for j=1: ns;Reci(i,j) = lon(i)*F(i,j)-lon(j)*F(i,j); end;end;disp(Reci);disp(sum(Reci));disp(sum(Reci,2)');</code>							
0 0 -0.1369 -0.0309 0 0 -0.0114 -0.1504							
0 0 -0.1504 -0.0114 0 0 -0.0309 -0.1369							
0.4882 0.6577 0 0 0.0456 0.1238 0 0							
0.1238 0.0456 0 0 0.6577 0.4882 0 0							
0 0 -0.0114 -0.1504 0 0 -0.1369 -0.0309							
0 0 -0.0309 -0.1369 0 0 -0.1504 -0.0114							
0.0456 0.1238 0 0 0.4882 0.6577 0 0							
0.6577 0.4882 0 0 0.1238 0.0456 0 0							
<code>disp(sum(Reci)); 1.3153 1.315 -0.3295 -0.3295 1.3153 1.3153 -0.3295 -0.3295</code>							
<code>disp(sum(Reci,2)'); -0.3295 -0.3295 1.3153 1.3153 -0.3295 -0.3295 1.3153 1.3153</code>							

Table 18: Rectangular cavity:  $F$  and reciprocity test - 8 segments, 2 Gauss points

## → b) Street section

After the cavities, we analyze how radiative exchanges behave in an open space like a street section ([Figure 45](#)). We define a domain with 8 vertices, 3 patches and 10 interfaces. This domain has an area of 19  $m^2$  and a perimeter of 40  $m$ .

Displayed output: L 24, Domain area : 19 m2

Displayed output: L 60, Domain perim.: 40 m

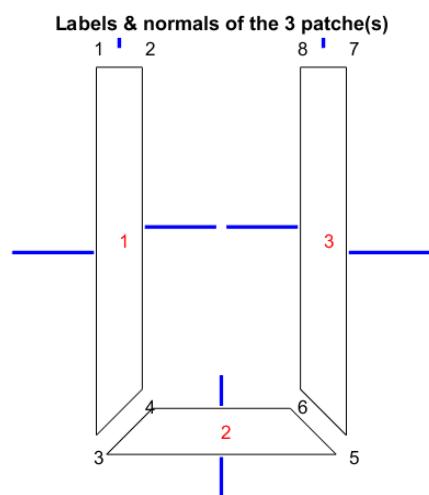
The matrix of  $CAD$  vertices is displayed with the Matlab<sup>®</sup> instruction: `[(1: npv)' xyz_cao]`. The first column of the four tables contains the numbering (in blue) of their lines. The matrix of  $CAD$  patches is displayed with the Matlab<sup>®</sup> instruction: `[(1: np)' car_cao]`.

These matrices and the variables `rc`, `ra` and `cs` are the input data visible in [line 3](#) of the procedure `Fiammetta.m`. After running this procedure, the matrix of interfaces `bor` is displayed with the Matlab<sup>®</sup> instruction: `[(1: nbo)' bor]`. In this example, we imposed two nodes per interface.

Displayed output: L 11, num. nod side: 2

The domain is represented at the right of [Figure 45](#) with node (black) and patch (red) numbering. On the boundary of the domain, the outward normal vectors (blue) are also represented.

The street section analyses use the parameters  $Gi = 14$  when radiation is present and  $Gi = 15$  for the linear transient analyses. (Matlab<sup>©</sup> function [cad\\_gin.m](#), [Table 31](#)).

<code>[(1:npv)' xyz_cao]</code>	1 0 8 2 1 8 3 0 0 4 1 1 5 5 0 6 4 1 7 5 8 8 4 8	<b>Labels &amp; normals of the 3 patch(es)</b> 
<code>[(1:np)' car_cao]</code>	1 2 1 3 4 2 4 3 5 6 3 6 5 7 8	
<code>[(1:nbo)' bor]</code>	1 2 1 0 9 10 2 1 3 1 0 11 12 3 3 4 1 2 13 14 4 4 2 1 0 15 16 5 3 5 2 0 17 18 6 5 6 2 3 19 20 7 6 4 2 0 21 22 8 5 7 3 0 23 24 9 7 8 3 0 25 26 10 8 6 3 0 27 28	
<code>[(1:np)' pbo]</code>	1 1 2 3 4 2 3 5 6 7 3 6 8 9 10	

[Figure 45: Data –  \$Gi = 14\$ ,  \$Gi = 15\$ : Street section, aspect ratio  \$dx/dy = 3/7\$](#)

Radiative heat transfers need the evaluation of the view factors of the boundary elements of the model. For a street canyon or a rectangle open on the top side, they are calculated with the function [geo\\_stf.m](#) of [Table 66](#).

A simple instruction is used to display the view factor matrices both for the rectangular cavity ([Figure 44](#)) using the Matlab<sup>©</sup> function [geo\\_vfc.m](#) of [Table 67](#) and the street section ([Figure 45](#)) using the Matlab<sup>©</sup> function [geo\\_stf.m](#) of [Table 66](#). The arguments of both functions are the number  $n$  of elements per side and the vector  $Lel$  containing the lengths of the horizontal and the vertical sides. The [geo\\_vfc.m](#) function needs one more argument defining the number of sides of the cavity.

<code>F = geo_stf (2, [3 7]); disp(F); disp(sum(F,1)); disp(sum(F,2)')</code>						
0	0	0.0192	0.0466	0.1822	0.5039	
0	0	0.1204	0.1277	0.5039	0.1822	
0.0515	0.3952	0	0	0.2296	0.1174	
0.1174	0.2296	0	0	0.3952	0.0515	
0.1822	0.5039	0.1277	0.1204	0	0	
0.5039	0.1822	0.0466	0.0192	0	0	
0.8549	1.3109	0.3139	0.3139	1.3109	0.8549	
0.7519	0.9341	0.7937	0.7937	0.9341	0.7519	
<code>F = geo_vfc (2, [3 7], 4); disp(F); disp(sum(F,1)); disp(sum(F,2)')</code>						
0	0	0.1277	0.0466	0.0997	0.1065	0.0192 0.1204
0	0	0.1204	0.0192	0.1065	0.0997	0.0466 0.1277
0.2296	0.3952	0	0	0.0515	0.1174	0.1822 0.5039
0.1174	0.0515	0	0	0.3952	0.2296	0.5039 0.1822
0.0997	0.1065	0.0192	0.1204	0	0	0.1277 0.0466
0.1065	0.0997	0.0466	0.1277	0	0	0.1204 0.0192
0.0515	0.1174	0.1822	0.5039	0.2296	0.3952	0 0
0.3952	0.2296	0.5039	0.1822	0.1174	0.0515	0 0
1	1	1	1	1	1	
0.5202	0.5202	1.4798	1.4798	0.5202	0.5202	1.4798 1.4798

[Table 19: View factor matrices for street section \(top\) and rectangular cavity \(bottom\)](#)

<code>F = geo_stf (2, [3 7]); disp ([F 1-sum(F,2)]); disp (sum ([F 1-sum(F,2)],2)')</code>						
0	0	0.0192	0.0466	0.1822	0.5039	0.2481
0	0	0.1204	0.1277	0.5039	0.1822	0.0659

0.0515	0.3952	0	0	0.2296	0.1174	0.2063
0.1174	0.2296	0	0	0.3952	0.0515	0.2063
0.1822	0.5039	0.1277	0.1204	0	0	0.0659
0.5039	0.1822	0.0466	0.0192	0	0	0.2481

Table 20: View factor matrix and sky view factor vector - street section (6 segments)

F = geo_stf (3, [3 7]); disp ([F 1-sum(F,2)]);disp (sum ([F 1-sum(F,2)],2)')									
0	0	0	0.0072	0.0198	0.0283	0.0650	0.1984	0.3624	0.3188
0	0	0	0.0192	0.0466	0.0545	0.1984	0.3624	0.1984	0.1204
0	0	0	0.1204	0.1277	0.0707	0.3624	0.1984	0.0650	0.0554
0.0176	0.0515	0.3952	0	0	0	0.1345	0.1294	0.0679	0.2038
0.0482	0.1174	0.2296	0	0	0	0.2296	0.1174	0.0482	0.2095
0.0679	0.1294	0.1345	0	0	0	0.3952	0.0515	0.0176	0.2038
0.0650	0.1984	0.3624	0.0707	0.1277	0.1204	0	0	0	0.0554
0.1984	0.3624	0.1984	0.0545	0.0466	0.0192	0	0	0	0.1204
0.3624	0.1984	0.0650	0.0283	0.0198	0.0072	0	0	0	0.3188
1	1	1	1	1	1	1	1	1	1

Table 21: View factor matrix and sky view factor vector - street section (9 segments)

### → c) L shape configurations

We now examine two situations containing both vertical and horizontal connected edges looking at sky, ground and the domain itself. In the first configuration (Figure 46), the vertical wall is always seeing half the sky vault while the horizontal one is seeing more than half the sky vault. In the second configuration (Figure 47), the vertical wall is always seeing half the ground, while the horizontal one is seeing more than half the ground.

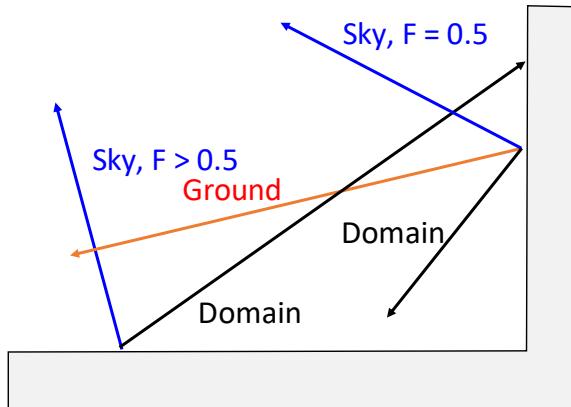


Figure 46: Vertical wall above horizontal edge

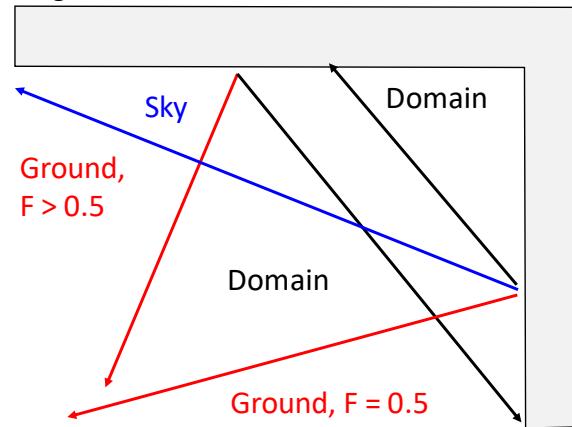


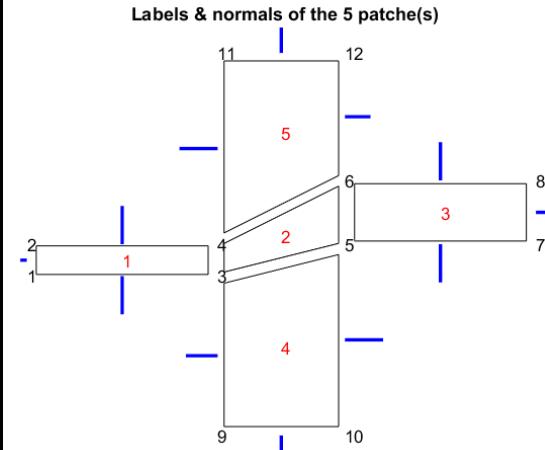
Figure 47: Vertical wall below horizontal edge

### → d) Thermal bridge

In this section we define the thermal bridge:

The data of the parameter  $Gi$  are defined in the Matlab<sup>©</sup> function [cad\\_gin.m](#), [Table 31](#)

<code>[ (1:npv)' xyz_cao]</code>	1	-2	5												
	2	-2	6												
	3	4	5												
	4	4	6												
	5	8	6												
	6	8	8												
	7	14	6												
	8	14	8												
	9	4	0												
	10	8	0												
	11	4	12												
	12	8	12												
<code>[ (1:np)' car_cao]</code>	1	1	3	4	2										
	2	3	5	6	4										
	3	5	7	8	6										
	4	9	10	5	3										
	5	4	6	12	11										
<code>[ (1:nbo)' bor]</code>	1	1	3	1	0	13	14								
	2	3	4	1	2	15	16								
	3	4	2	1	0	17	18								
	4	2	1	1	0	19	20								
	5	3	5	2	4	21	22								
	6	5	6	2	3	23	24								
	7	6	4	2	5	25	26								
	8	5	7	3	0	27	28								
	9	7	8	3	0	29	30								
	10	8	6	3	0	31	32								
	11	9	10	4	0	33	34								
	12	10	5	4	0	35	36								
	13	3	9	4	0	37	38								
	14	6	12	5	0	39	40								
	15	12	11	5	0	41	42								
	16	11	4	5	0	43	44								



[ (1:np)' pbo]

Figure 48: Thermal bridge data:  $Gi = 16, 18, 19, 20, 21, 22$

View factor matrix of the left side of a building involving a balcony												
0	0	0.0840	0.1160	0	0	0	0	0	0	0.3000	0.5000	
0	0	0.2764	0.1023	0	0	0	0	0	0	0.1213	0.5000	
0.1023	0.2764	0	0	0	0	0	0	0	0	0	0.6213	
0.1160	0.0840	0	0	0	0	0	0	0	0	0	0.8000	
0	0	0	0	0	0	0	0	0	0	0.5000	0.5000	
0	0	0	0	0	0	0	0	0	0	0.5000	0.5000	
0	0	0	0	0	0	0	0	0	0.0629	0.1026	0.8345	0
0	0	0	0	0	0	0	0	0.2428	0.1136	0.6437	0	
0	0	0	0	0	0	0.0903	0.3077	0	0	0.5000	0.1020	
0	0	0	0	0	0	0	0.1254	0.1096	0	0	0.5000	0.2650
0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	

Table 22: View factor matrix  $F$  around a balcony – 10 segments +ground + sky

Because the third segment of the left side connecting nodes 2 and 1 does not see any other part of the model, its view factors are equal to zero ([Table 22](#) & [Figure 49](#)), with the consequence that the corresponding lines and columns of matrix  $F$  are also equal to zero.

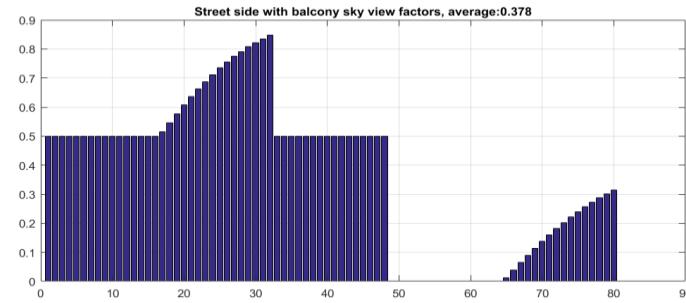


Figure 49:  $F_{sky}$  in a street with balcony – 16 segments / patch side

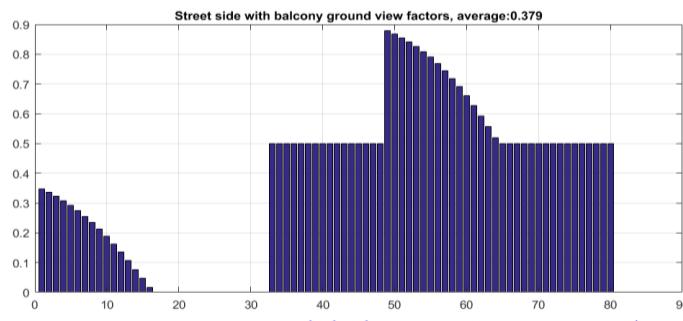


Figure 50: Fgr in a street with balcony – 16 segments / patch side

## 5.3 Radiosity matrix

### → a) Rectangular cavity

According to [Goral *et al* 1984], the radiosity of a surface  $j$  is the total rate at which radiant energy leaves the surface in terms of energy per unit time and per unit area ( $Wm^{-2}$ ). The concept of radiosity is fundamental in radiative heat exchanges [Sillion & Puech 1994].

For a gray body of emissivity  $\varepsilon = 1 - \rho = 0.5$  and, thus, reflectivity  $\rho = 0.5$  (in the Matlab<sup>©</sup> procedures presented in this document,  $\rho$  is noted *re*), the radiosity matrix  $[M]$  is computed according to the sequence of Matlab<sup>©</sup> instructions in the heading of *Table 23*. Two extreme situations are identified:

- For a black body,  $\rho = 0$ , the emissivity is equal to 1 and, then, the radiosity matrix is equal to the identity matrix [7].
- For a mirror (adiabatic cavity),  $\rho = 1$ , the emissivity  $\varepsilon = 0$ . Therefore, the radiosity matrix is singular: the column and line sums are both equal to 0 as confirmed by the result shown in *Table 23*, by using the instructions *disp (sum(M)); disp(sum(M'))*.

re = 1; F = geo_vfs(2, [1 1], 4); M = eye(size(F,1))-re*F							
1.0000	0	-0.1023	-0.1160	-0.1787	-0.2425	-0.0840	-0.2764
0	1.0000	-0.2764	-0.0840	-0.2425	-0.1787	-0.1160	-0.1023
-0.0840	-0.2764	1.0000	0	-0.1023	-0.1160	-0.1787	-0.2425
-0.1160	-0.1023	0	1.0000	-0.2764	-0.0840	-0.2425	-0.1787
-0.1787	-0.2425	-0.0840	-0.2764	1.0000	0	-0.1023	-0.1160
-0.2425	-0.1787	-0.1160	-0.1023	0	1.0000	-0.2764	-0.0840
-0.1023	-0.1160	-0.1787	-0.2425	-0.0840	-0.2764	1.0000	0
-0.2764	-0.0840	-0.2425	-0.1787	-0.1160	-0.1023	0	1.0000
<i>disp(det(M)); 8.7522e-17</i>							
<i>disp (sum(M)) :</i> 1.0e-15 *							
-0.1110 -0.0555 0 0 -0.0555 -0.0833 0 0							
<i>disp(sum(M, 2)') :</i> 1.0e-15 *							
-0.0555 -0.0833 -0.1110 -0.0555 0.0278 -0.0555 0 0							

Table 23: Radiosity matrix of an adiabatic ( $\rho = 1$ ) square cavity – 8 segments,

### → b) Street section

Interesting geometric properties of the street section example are the matrices related to the view factors. Using the Matlab<sup>©</sup> function *geo\_stf.m*, we can compute them directly. The first argument of this function is the number of segments on each side of the street section. The second argument is a vector containing the width and the height of the street. With one segment per side, we obtain a 3 x 3 matrix, with two segments per side, we obtain a 6 x 6 matrix (*Table 19*, *Table 20*, *Table 21*).

The radiosity matrix  $M$  is deduced from the view factor matrix (92). If the reflection coefficient  $\rho$ , noted *re* in Matlab<sup>©</sup> instructions, is equal to zero, it reduces to the identity matrix (*Table 24*).

For  $\rho=1$ , examples of radiosity matrices are given in *Table 25*.

F = geo_stf(2, [3 7])						F = geo_stf(1, [3 7])		
0	0	0.0192	0.0466	0.1822	0.5039			
0	0	0.1204	0.1277	0.5039	0.1822			
0.0515	0.3952	0	0	0.2296	0.1174			
0.1174	0.2296	0	0	0.3952	0.0515			
0.1822	0.5039	0.1277	0.1204	0	0			
0.5039	0.1822	0.0466	0.0192	0	0			
re = 0.; M = eye(size(F,1)) - re*F								
1	0	0	0	0	0			
0	1	0	0	0	0			
0	0	1	0	0	0			
0	0	0	1	0	0			
0	0	0	0	1	0			
0	0	0	0	0	1			

Table 24: View factor and radiosity matrices of a street section,  $\rho = 0$

re=1; F = geo_stf(2, [3 7]); M = eye(size(F,1)) - re*F; disp(det(M))	re=1; F = geo_stf(1, [3 7]); M = eye(size(F,1)) - re*F; disp(det(M))
1.0000	1.0000
0 1.0000 -0.1204 -0.1277 -0.5039 -0.1822	-0.3952 1.0000 -0.1204 -0.7593
-0.0515 -0.3952 1.0000 0 -0.2296 -0.1174	-0.1174 -0.2296 0 1.0000 -0.0515
-0.1174 -0.2296 0 1.0000 -0.3952 -0.0515	-0.1822 -0.5039 -0.1277 -0.1204 1.0000
-0.1822 -0.5039 -0.1277 -0.1204 1.0000 0	-0.5039 -0.1822 -0.0466 -0.0192 0 1.0000
-0.5039 -0.1822 -0.0466 -0.0192 0 1.0000	disp(det(M)); 0.2570
disp(det(M)); 0.2561	disp(det(M)); 0.2561

Table 25: Radiosity matrices of a street section,  $\rho = 1$

The sky view factor uni-column matrix  $F_{sky}$  is obtained as the complement to unity of the view factor matrix. For a street section it is computed in [geo\\_stf.m](#) (Table 66).

n=2; F=geo_stf(n, [3 7]); Fsky=(1-eye(size(F,2))*(sum(F')))'
n = 2: Fsky = [0.2481 0.0659 0.2063 0.2063 0.0659 0.2481]'
n = 1: Fsky = [0.1204 0.2095 0.1204]'
Generation of <a href="#">Figure 51</a>
F = geo_stf(200, [3 7])*100; Fsky =(100-eye(size(F,2))*(sum(F')))'; figure('Position',[100 100 1200 600]);hold on;grid on;bar(Fsky); title(['SVF in a street section, max, min, mean in %: ', num2str([max(Fsky) min(Fsky) mean(Fsky)],2)],'fontsize', 20) ylabel('%', 'fontsize', 20)

Table 26: Sky view factor – uni-column matrix  $F_{sky}$  of a street section

The distribution of the sky view factor along the street walls ([Figure 51](#)) is computed with the Matlab<sup>®</sup> instructions of [Table 26](#). As expected, the sky view factor is equal to 50 % on the top of the street walls. Going down, it is continuously decreasing until 4 %. On the road surface, it is more or less constant and close to 20 %.



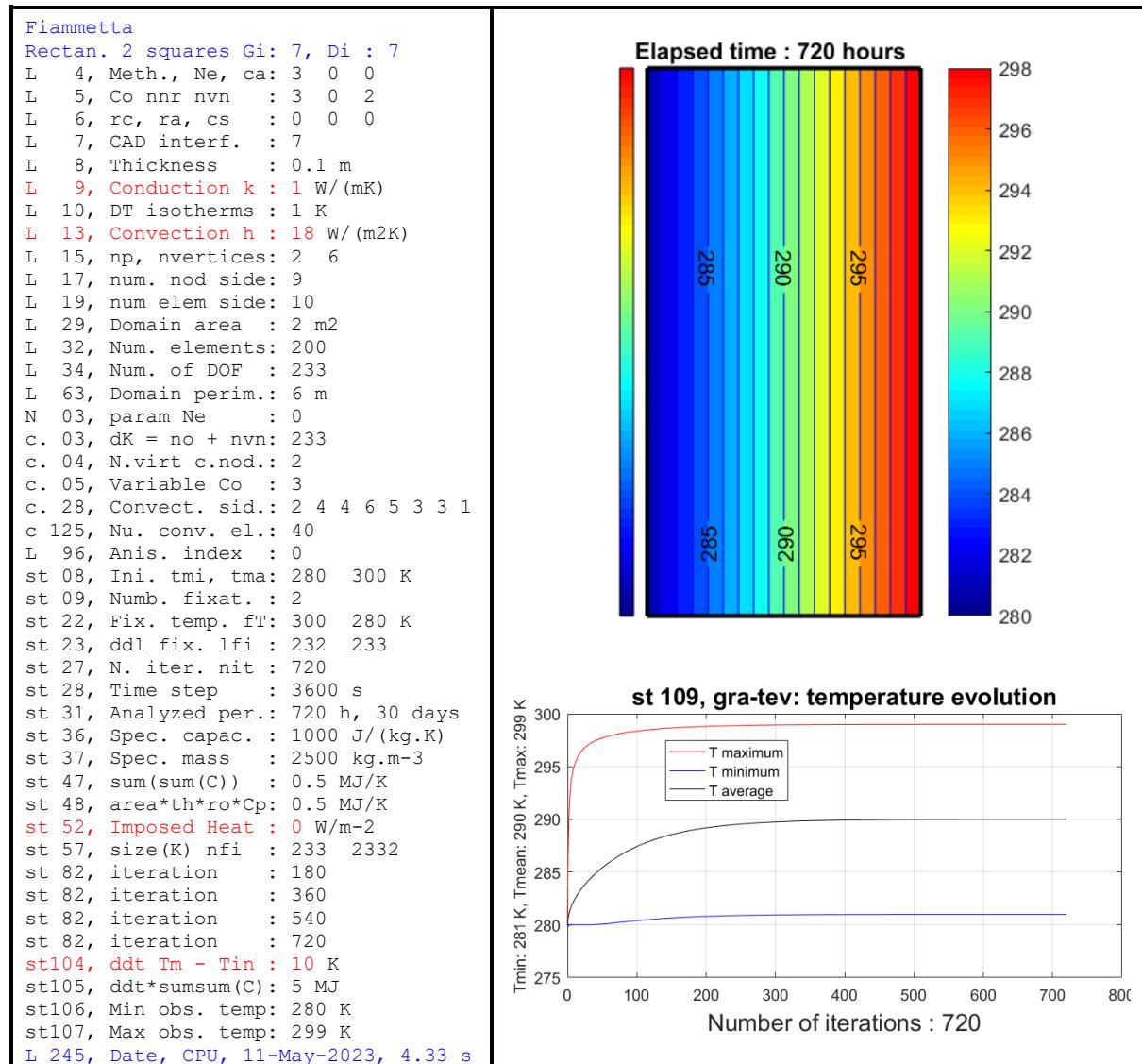
Figure 51: Sky View Factors in the street section – 200 radiative segments per side

## 5. Tutorial VI: Radiative heat exchanges

In the simple situation where the domain edge submitted to radiation is either horizontal with only sky above or vertical with facing only sky and ground, the computation of the view factors is obvious. For the horizontal edge, the sky view factor is constant and equal to 1. For the vertical edge, the sky and ground view factors are both equal to 0.5. Heat exchanges between ground and sky are very important, but in this study, we do not matter them.

### 6.1 A simple convex domain

Before solving a problem where radiative heat exchanges are present, we propose a case with virtual convective nodes ([fem\\_Kcv.m](#), [Table 45](#)) on both vertical sides of a rectangular domain ([Figure 53](#)). The display output of Fiammetta refers to the Cad definition of the problem and to the finite element mesh generated



[Figure 52: Transient heat transfer in rectangular domain with two convective walls.](#)

The difference of temperature between both sides expressed in the Matlab<sup>©</sup> notations of the procedure [Fiammetta.m](#) ([Table 28](#)) is given by:  $\max(tca(1:15)) - \min(tca(1:15)) = 18 \text{ K}$ , while the differences of temperature between the walls and the virtual nodes are equal to 1  $\text{K}$  on both sides. Stable values of temperature components are obtained after about 300 iterations ([Figure 52](#)).

We compare this problem to another one ([Figure 54](#)) including the same convective wall on the right and a radiative one defined by a virtual radiative node on the left ([fem\\_Kcr.m](#), [Table 46](#)).

The number of nodes pertaining to the domain is equal to 15, the minimum nodal temperature in the solid is min (tca (1:15)) = 280 K, the maximum: max (tca (1:15)) = 298.95 K and the difference: (max (tca (1:15)) - min (tca (1:15))) = 18.9424 K  $\approx$  19 K. In the solid, the temperature gradient is equal to -19 K and consequently, the heat flow =  $-(19 \text{ K} * 1 \text{ Wm}^{-1}\text{K}^{-1}) = 19 \text{ Wm}^{-1}$ . In the previous example (Figure 52), the difference was only equal to 18 K. As a conclusion, the heat flow is higher with radiative than with convective boundary.

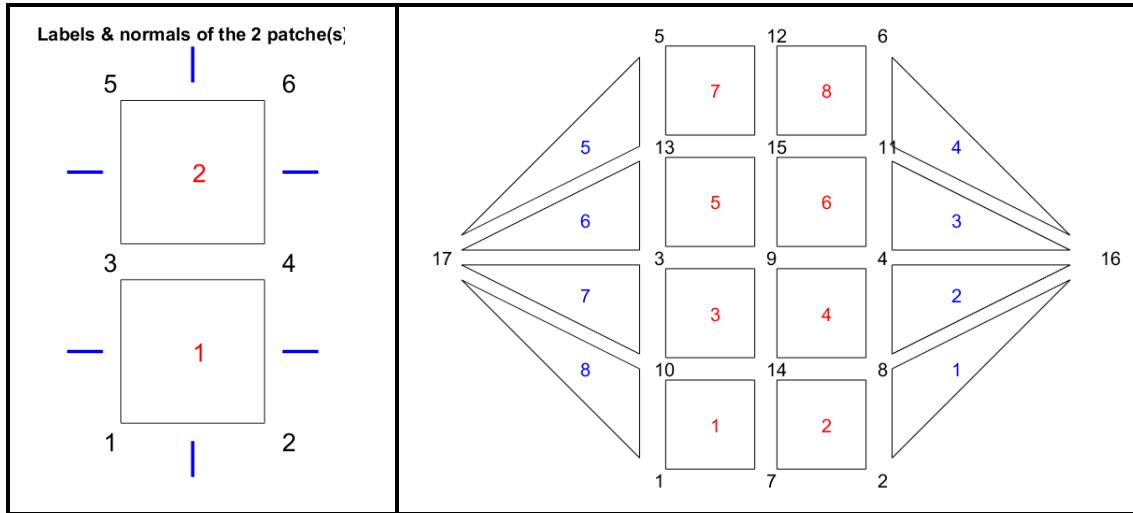
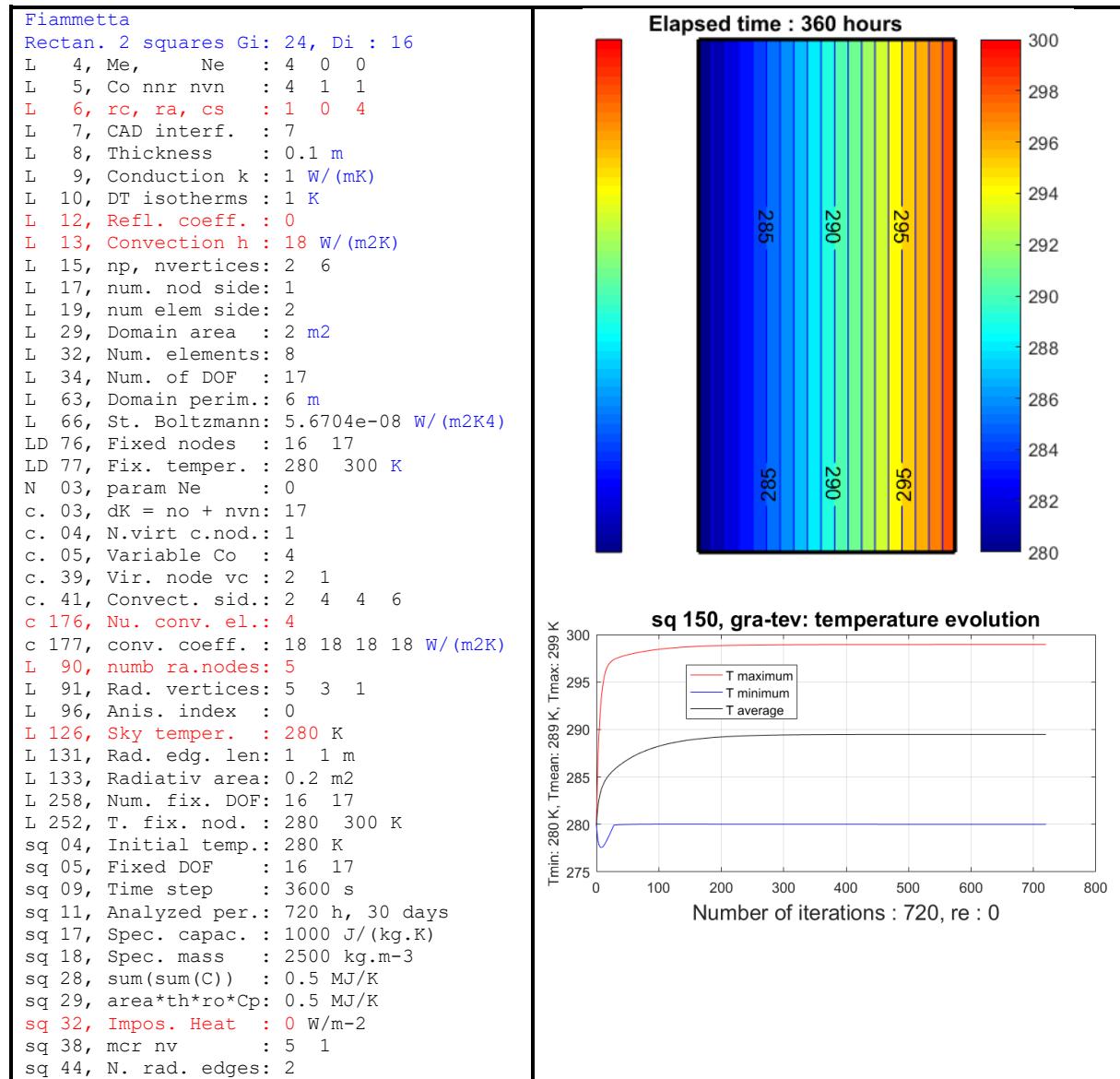


Figure 53: A simple convex domain, CAD definition and finite element mesh



```

sq 45, rad seg leng.: 1 1 m
sq 46, N. rad. elem.: 4
sq 48, rad sid leng.: 0.5 0.5 0.5 0.5 m
sq122, Iteration : 360 gt: 280 1 300 K
sq122, Iteration : 720 gt: 280 1 300 K
sq143, Tmean - Tini : 9.48 K
sq144, Stored heat : 4.74 MJ
sq146, Min it+1 temp: 280 K
sq147, Max it+1 temp: 289 K
sq148, Max it+1 temp: 299 K
sq154, Ejected heat : 0.202 MJ
L 260, Diss in solid: 35.9 WK
L 257, Total dissip.: 37.9 WK
L 260, Date, CPU, 11-May-2023, 1.2096 s
hf 25, Max heat flow: 19, mean: 19 W/m2
ch 03, coeff. red. dt: 25 W/m2
ch 19, temp. grad. : -1.0523 K
ch 20, Mean conv. fl: -1.0523 0 0 W/m2
hf 25, Max heat flow: 19, mean: 19 W/m2

```

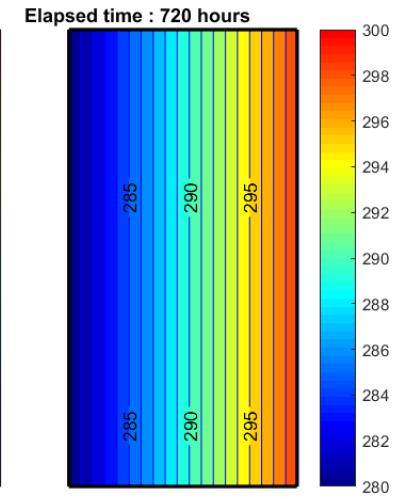


Figure 54: Rectangle with one radiative wall (left) and one convective wall (right)

## 6.2 Square cavity

a)

### Virtual radiative node concept

In the next test, radiative heat exchanges are analyzed in a cavity, using the concept of virtual radiative node. The domain is submitted to a vertical uniform temperature gradient. Without the cavity, the dissipation should be equal to 100 *WK* and the mean heat flow to 10 *Wm<sup>-1</sup>*.

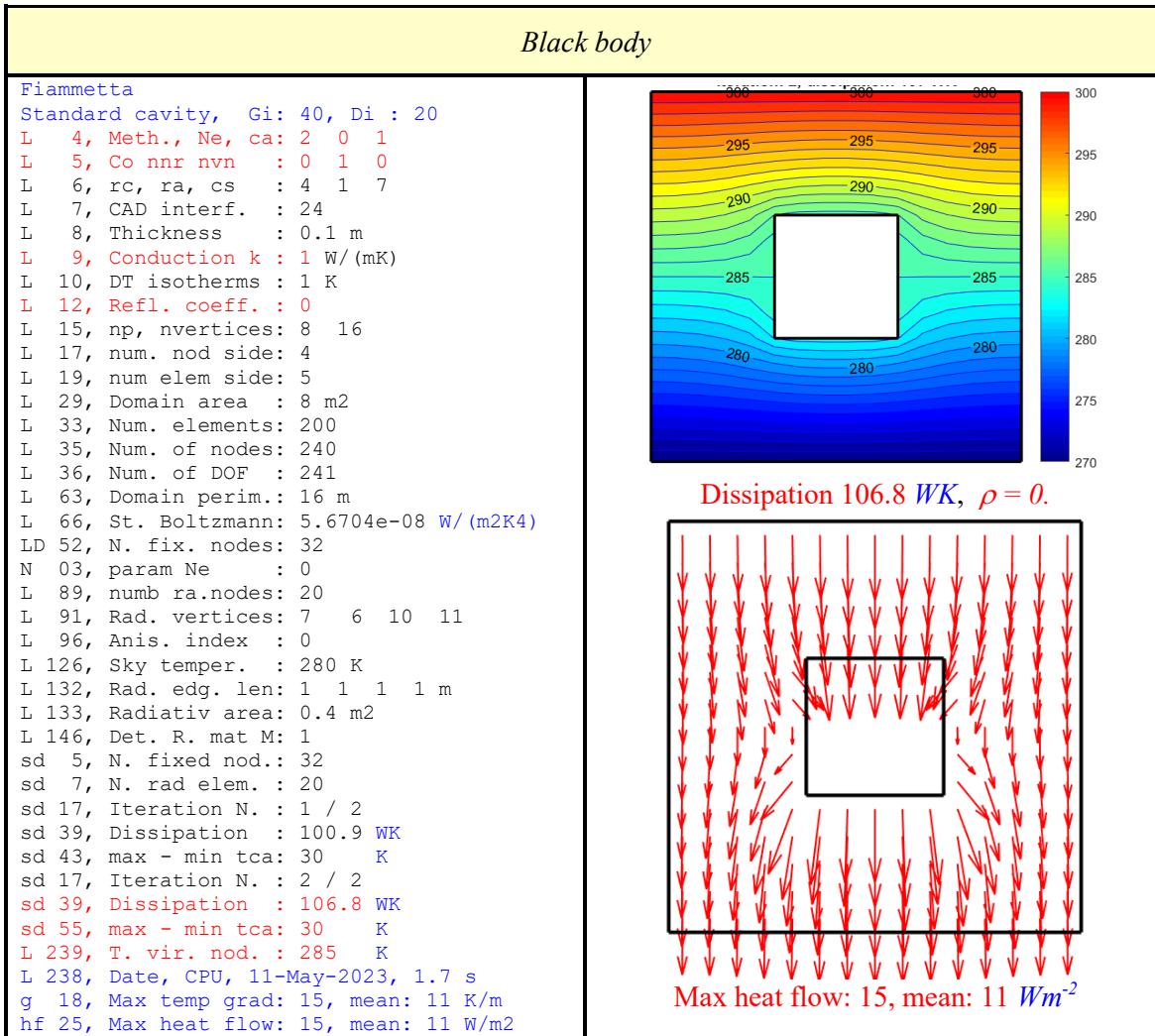


Figure 55: Isotherms and heat flows through a cavity, 1 virtual radiative node, black body

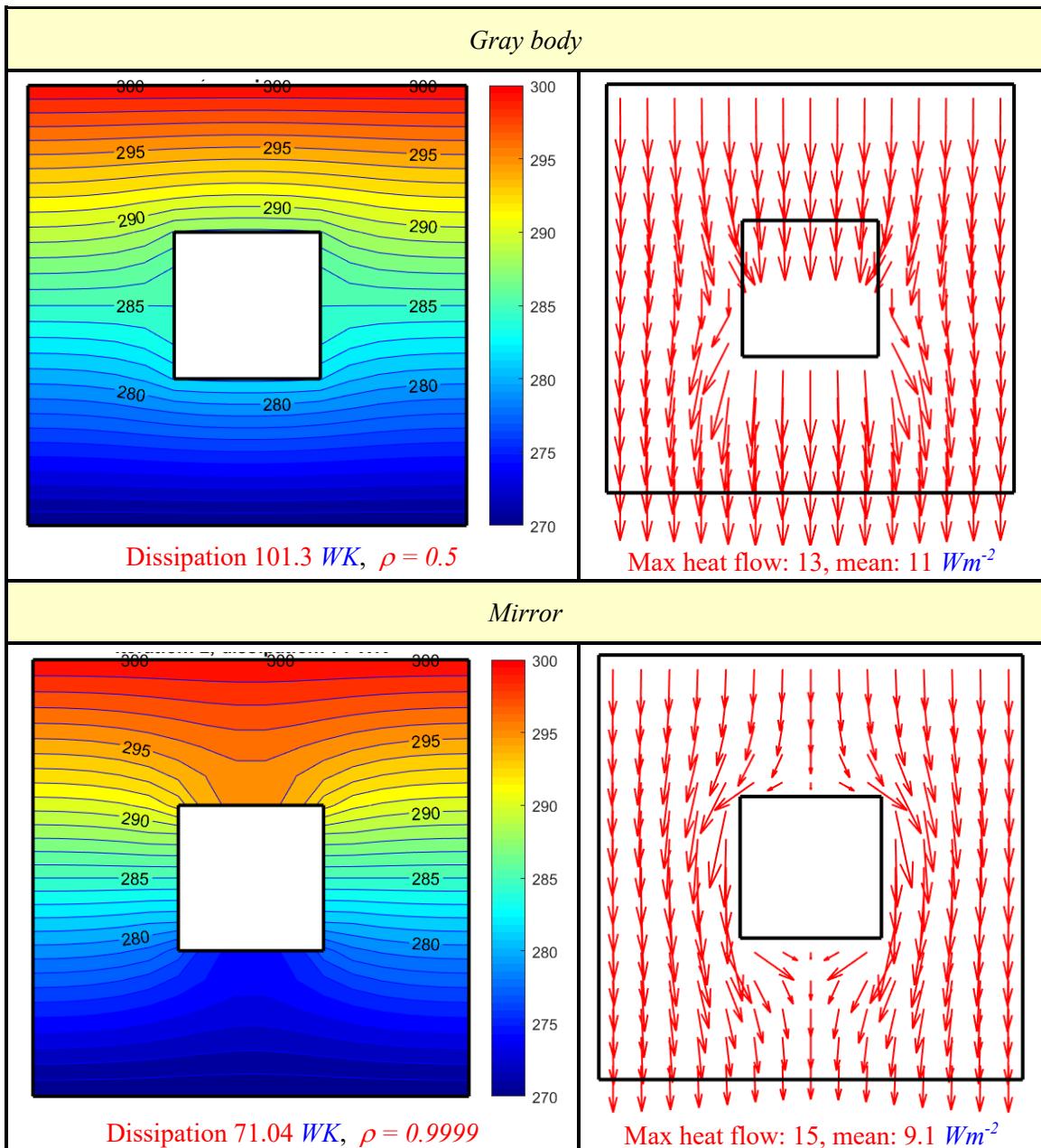


Figure 56: Isotherms and heat flows, gray body & mirror, 30 days, 1 virtual radiative node

When using a radiative virtual node for the square cavity model, the variation of dissipation and heat flows as functions of the reflection coefficient  $\rho$  are noticeable (Table 27). The dissipation and the mean heat flows are both decreasing with the emissivity  $\epsilon = (1 - \rho)$ .

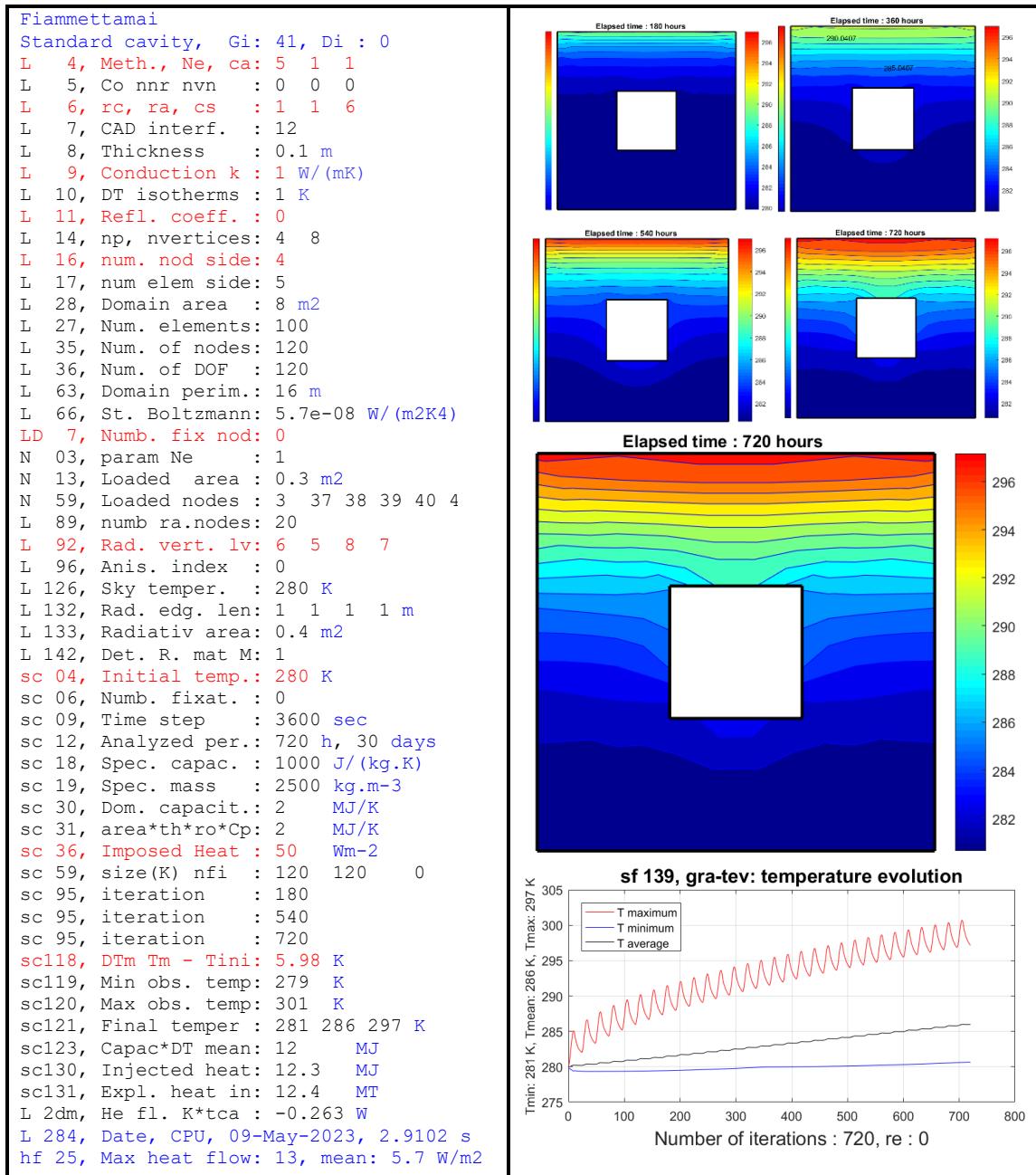
Emissivity $\epsilon = (1 - \rho)$	$\rho$	Dissipation	Heat flow, max, mean
1.	0.	106.8 WK	15 $Wm^{-2}$ 11 $Wm^{-2}$
.8	.2	105.2 WK	15 $Wm^{-2}$ 11 $Wm^{-2}$
.5	.5	101.3 WK	13 $Wm^{-2}$ 11 $Wm^{-2}$
.2	.8	91.77 WK	11 $Wm^{-2}$ 10 $Wm^{-2}$
.1	.9	84.67 WK	12 $Wm^{-2}$ 9.7 $Wm^{-2}$
0.0001	0.9999	71.04 WK	15 $Wm^{-2}$ 9.1 $Wm^{-2}$

Table 27: Influence of emissivity when using one virtual radiative node,

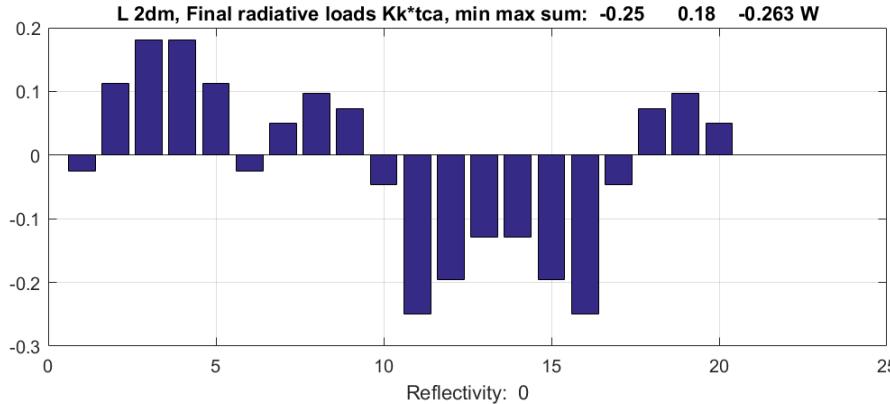
b)

## Radiosity method, black body square cavity $\rho = 0$

A heat flow of  $50 \text{ Wm}^{-2}$  is applied on the upper horizontal border of the domain ([Figure 57](#), display output: sc 36, Imposed Heat :  $50 \text{ Wm}^{-2}$ ). The cavity is acting as a black body and some results are compared to a cavity acting as a mirror ([Figure 59](#)).



[Figure 57: Isotherms of radiative exchanges, 256 elements, 1-month, black body,  \$\rho = 0\$](#)



[Figure 58 : Flux de chaleur nodaux le long de la cavité \(gra\\_2dm.m, Table 55\)](#)

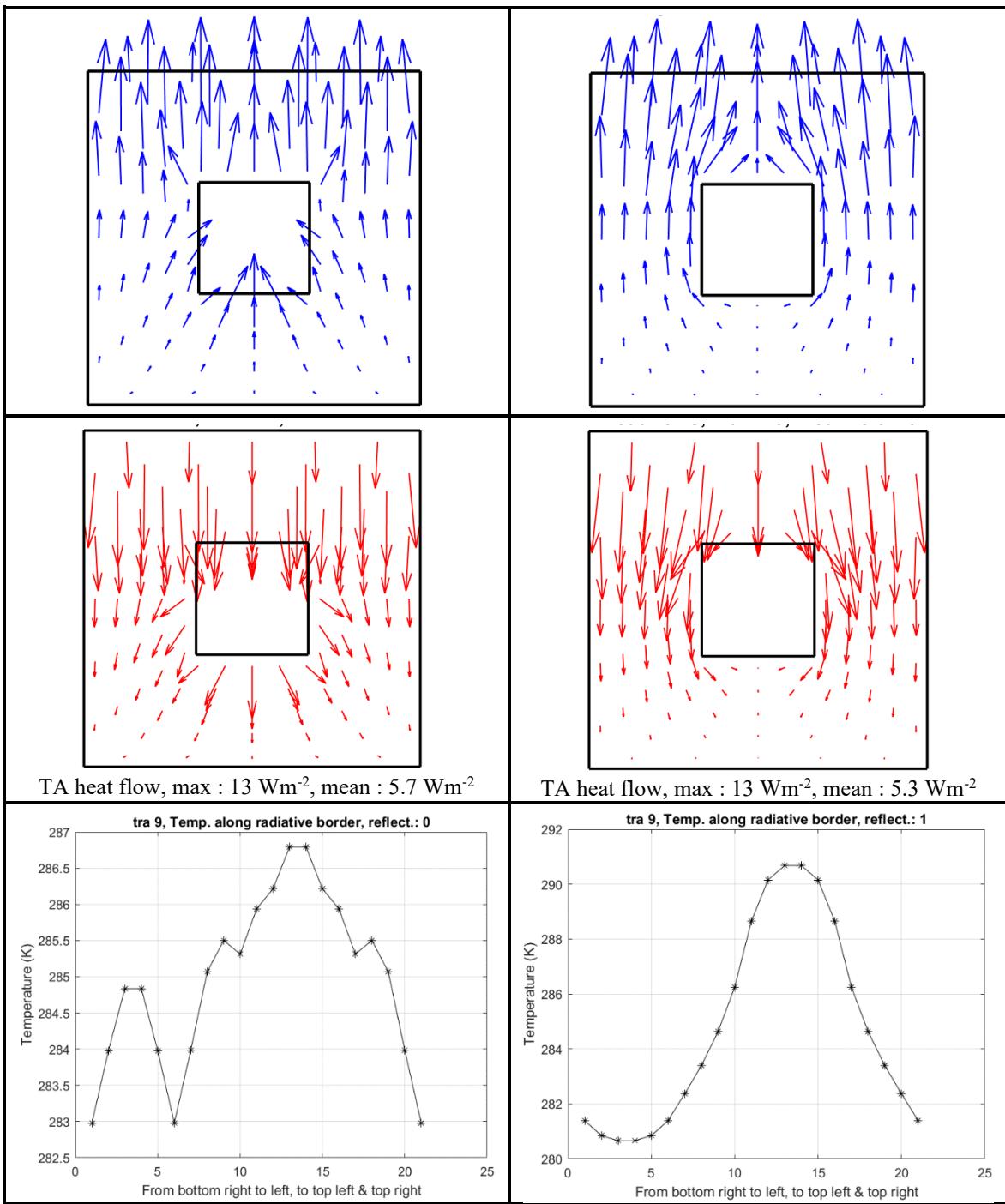


Figure 59: Grad., H.F. & T. after 30 days, left: black body,  $\rho = 0$ ; right: mirror,  $\rho = 1$

With 1088 DOF, the solution of Figure 59 becomes that of Figure 60.

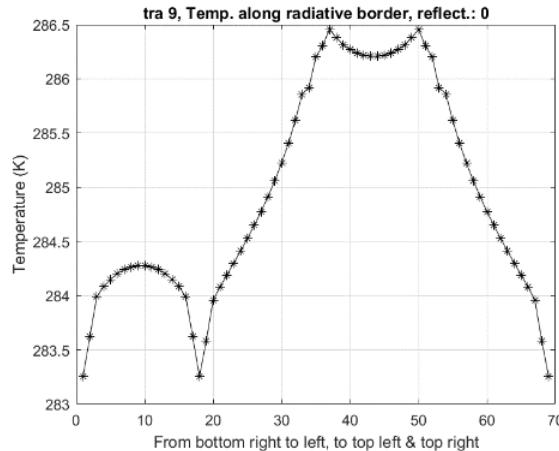


Figure 60: Cavity,  $\rho = 0$  (Figure 57), but 1224 DOF, nni = 16, (gra\_tra.m, Table 54)

The description of the cavity boundary is given in the array *lcont* computed in *cad\_ban.m*, *Table 37*. The cavity boundary nodes are ordered from bottom right vertex and following the border, cavity area right. This list is completed by the list of the four vertices *lv* ordered in the same way as the side nodes (*cad\_ban.m*, *Table 37*). This description is using the matrices *bor* and *pbo* computed in *cad\_mes.m* (*Table 35*).

Input of *fem\_rsm.m* (*Table 48*) called in *line 64* of *fem\_smc.m* (*Table 41*):

- temperature vector *tca* computed in a previous iteration,
- reflection coefficient *re*,
- product *SBr* of the Stefan-Boltzmann by the thickness *th* of the radiative element,
- matrix  $M_{pr} = (I - F) M^{-1}$  defined in equation (93) and computed from the view factor matrix *F* and the radiosity matrix *M*,
- *it* is the iteration number
- *Ms* is the vector of sky loads of the edges
- list *lcont* of the *DOF* on the border of the cavity (computed in *cad\_ban.m*, *Table 37*),
- lengths *lon* of the elements of the radiating sides.
- *DK* is the dimension of the system of equations

Output of *fem\_rsm.m*:

- radiative contribution *Kr* to the global matrix, this matrix is used in *fem\_smc.m*.
- Vector *Mn*

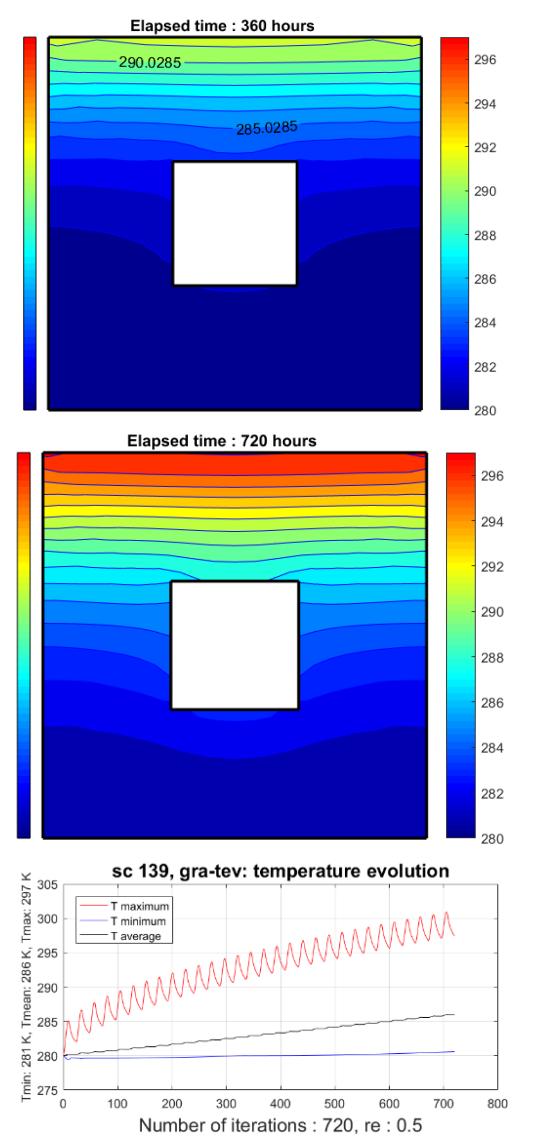
c)

### Radiosity method, gray body square cavity

```

Fiammettamai
Standard cavity, Gi: 41, Di : 0
L 4, Meth., Ne, ca: 5 1 1
L 5, Co nnr nvn : 0 0 0
L 6, rc, ra, cs : 1 1 6
L 7, CAD interf. : 12
L 8, Thickness : 0.1 m
L 9, Conduction k : 1 W/(mK)
L 10, DT isotherms : 1 K
L 11, Refl. coeff. : 0.5
L 14, np, nvertices: 4 8
L 16, num. nod side: 4
L 17, num elem side: 5
L 28, Domain area : 8 m2
L 27, Num. elements: 100
L 33, Num. of DOF : 120
L 63, Domain perim.: 16 m
L 66, St. Boltzmann: 5.6e-08 W/(m2K4)
LD 7, Numb. fix nod: 0
N 03, param Ne : 1
N 13, Loaded area : 0.3 m2
N 58, Loaded nodes : 3 37 38 39 40 4
L 89, numb ra.nodes: 20
L 91, Rad. vertices: 6 5 8 7
L 96, Anis. index : 0
L 126, Sky temper. : 280 K
L 131, Rad. edg. len: 1 1 1 1 m
L 132, Radiativ area: 0.4 m2
L 142, Det. R. mat M: 0.74107
sc 04, Initial temp.: 280 K
sc 06, Numb. fixat. : 0
sc 09, Time step : 3600 sec
sc 12, Analyzed per.: 720 h, 30 days
sc 18, Spec. capac. : 1000 J/(kg.K)
sc 19, Spec. mass : 2500 kg.m-3
sc 30, Dom. capacit.: 2 MJ/K
sc 31, area*th*ro*Cp: 2 MJ/K
sc 36, Imposed Heat : 50 Wm-2
sc 61, size(K) nfi : 120 120 0
sc 95, iteration : 180
sc 95, iteration : 360
sc 95, iteration : 540
sc 95, iteration : 720
sc132, DTM Tm - Tini: 5.97 K
sc133, Min obs. temp: 279 K
sc134, Max obs. temp: 301 K
sc135, Final temper.: 280 286 298
sc123, Capac*DT mean: 11.9 MJ

```



```

sc130, Injected heat: 12.3 MJ
sc131, Ex.heat input: 12.4 MJ
L 2dm, He fl. K*tca : -0.263 W
L 284, Date, CPU, 09-May-2023, 3.0294 s
hf 25, Max heat flow: 13, mean: 5.5 W/m2

```

Figure 61: Isotherms, radiation, 4 patches, 800 elements, 1 month,  $\rho = 0.5$

The test of [Figure 62](#) is the same as that of [Figure 61](#), but with more patches. The elements are rectangular. The goal is to show the heat fluxes on a regular mesh like in [Figure 64](#).

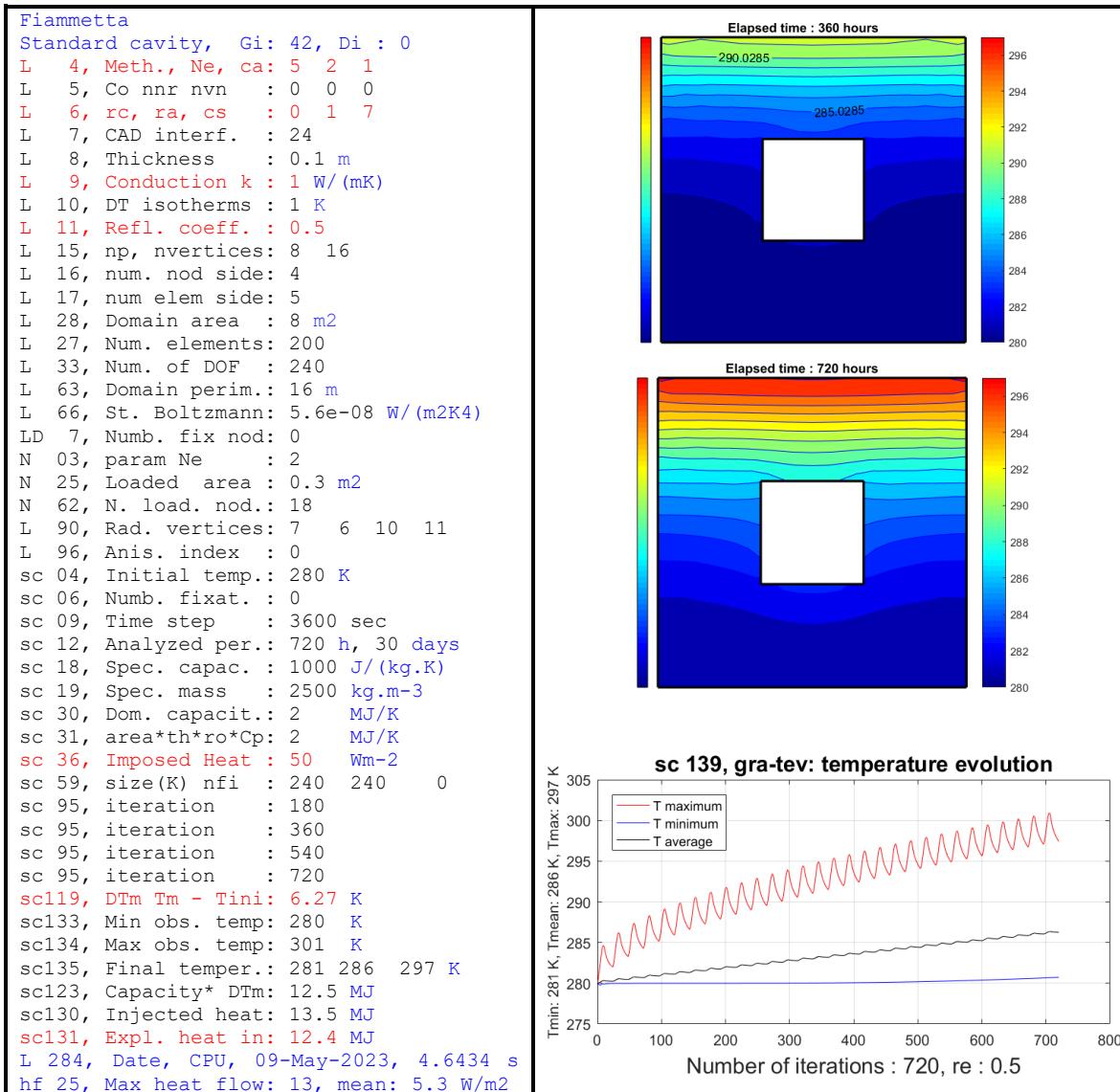


Figure 62: Isotherms, radiation, 8 patches, 200 elements, 1 month,  $\rho = 0.5$

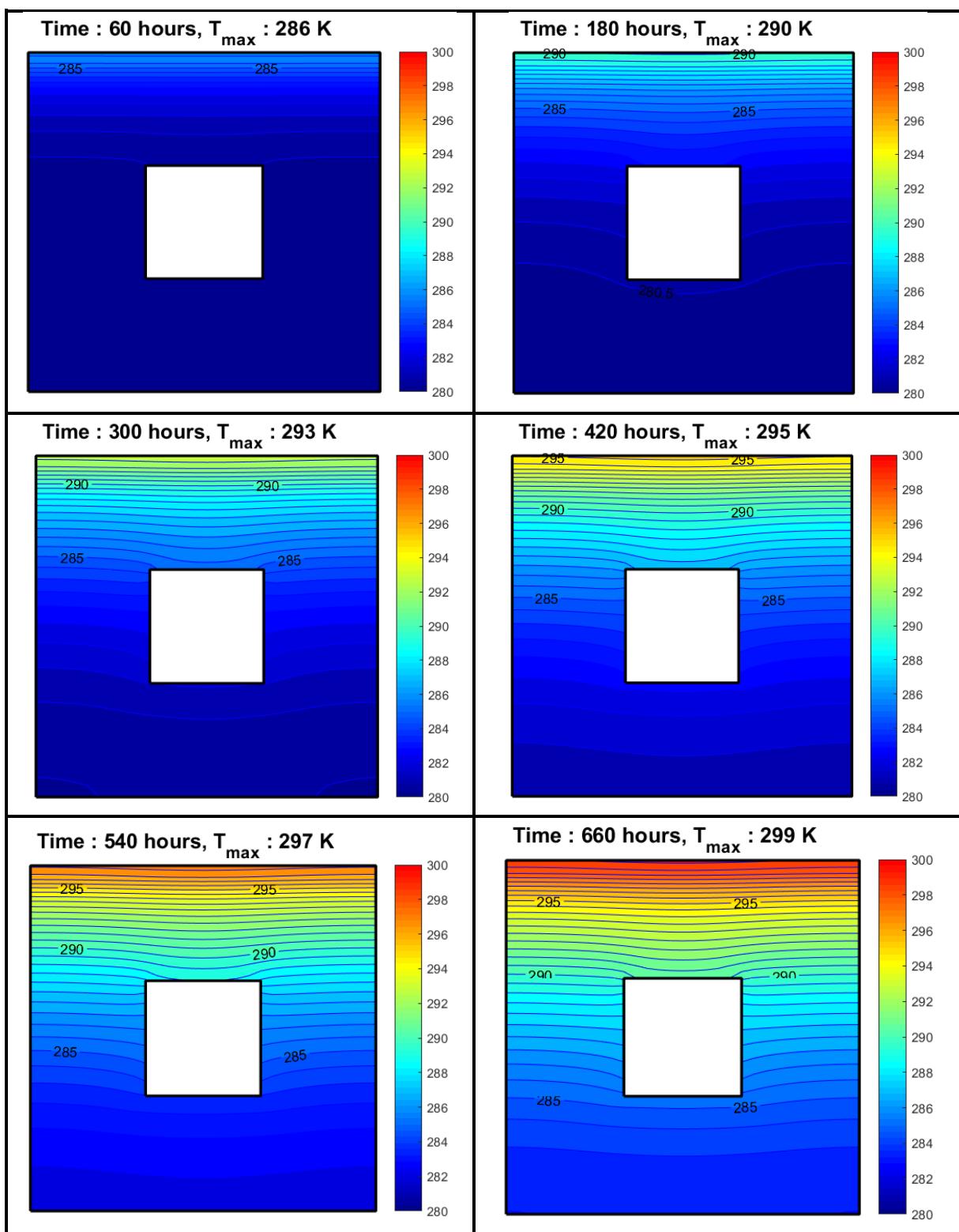


Figure 63: Isotherms, gray body, evolution after 60, 120, ..., 660 hours,  $\rho = 0.5$

d)

#### Radiosity method, comparison of gray cavities

In [Figure 64](#), tests are carried out with reflection coefficients equal to 0, 0.5 and 1, over a period of 30 days. The differences are well marked on the isothermal diagrams and even better on the representations of heat fluxes. Presented in the form of graphical animations, these results should help the user to better understand these physical phenomena in their four dimensions (space and time).

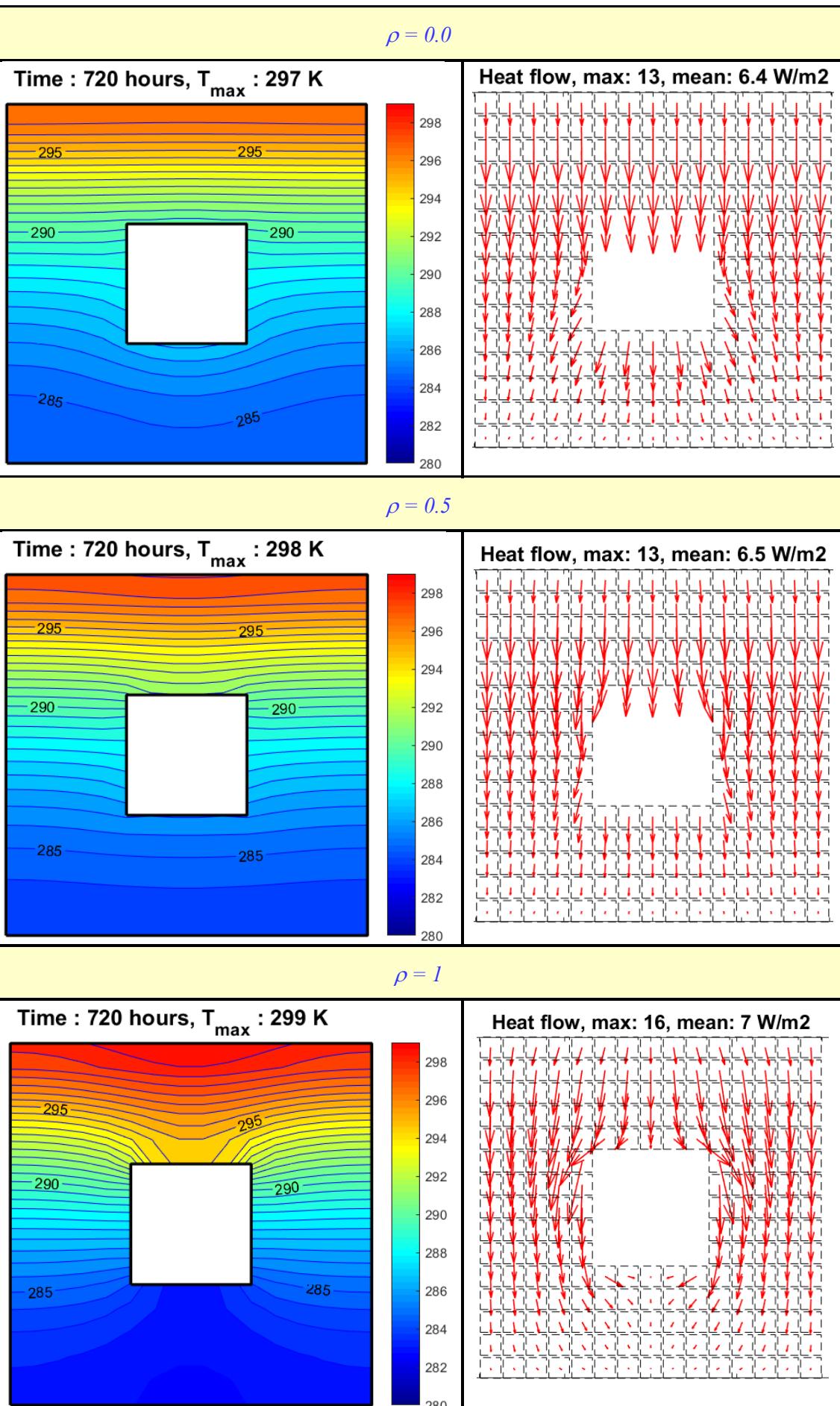


Figure 64: Isotherms and heat flows, black, gray body & mirror, 30 days

### 6.3 Rectangular cavity

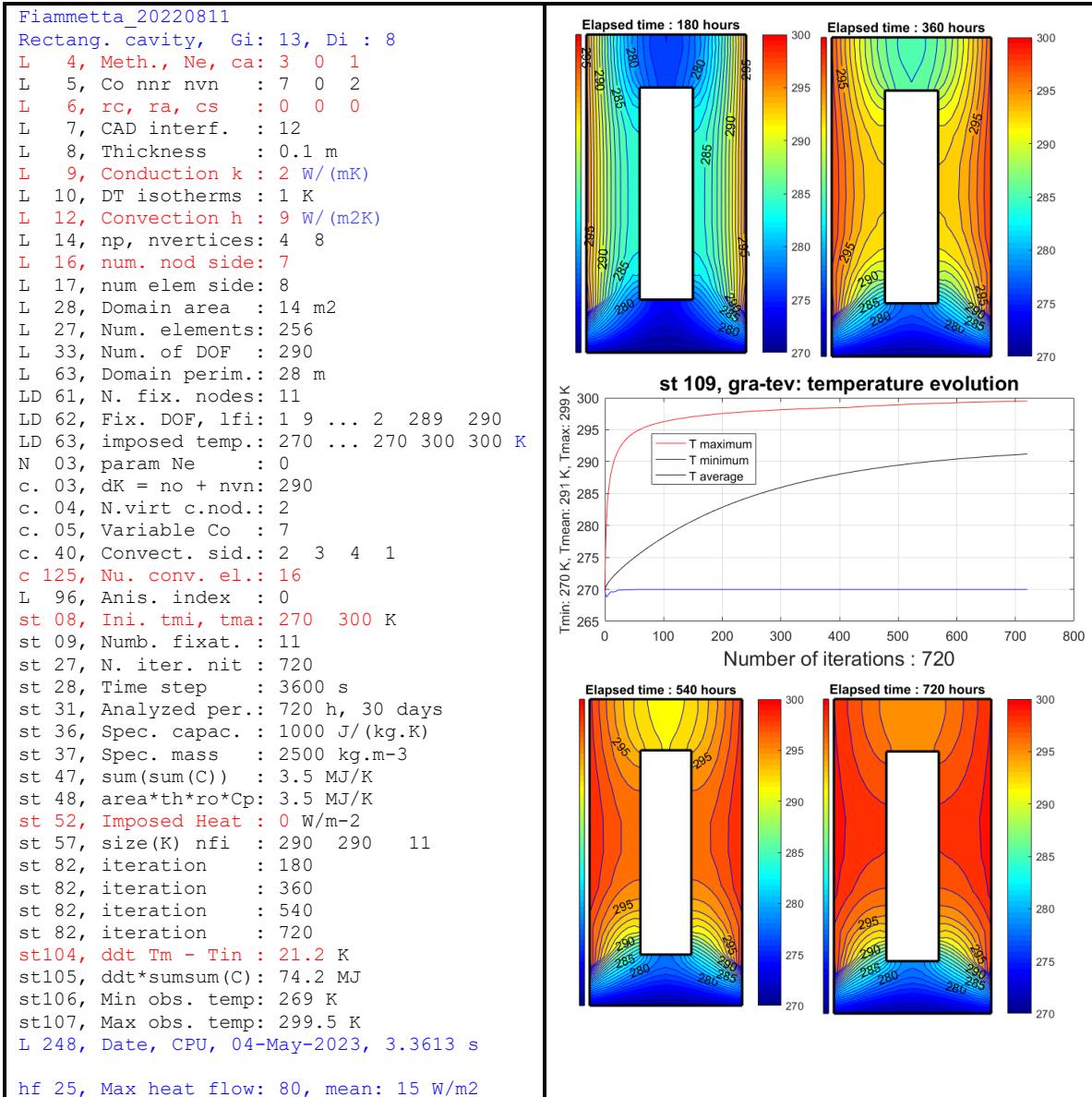


Figure 65: Adiabatic rectangular cavity, method 3, 2 virt. conv. nodes

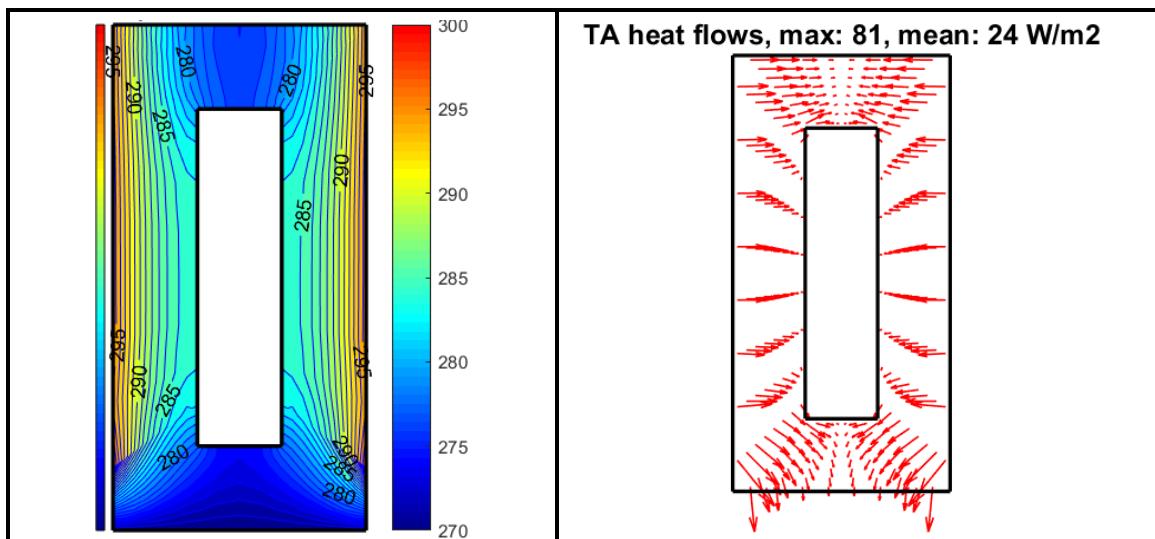
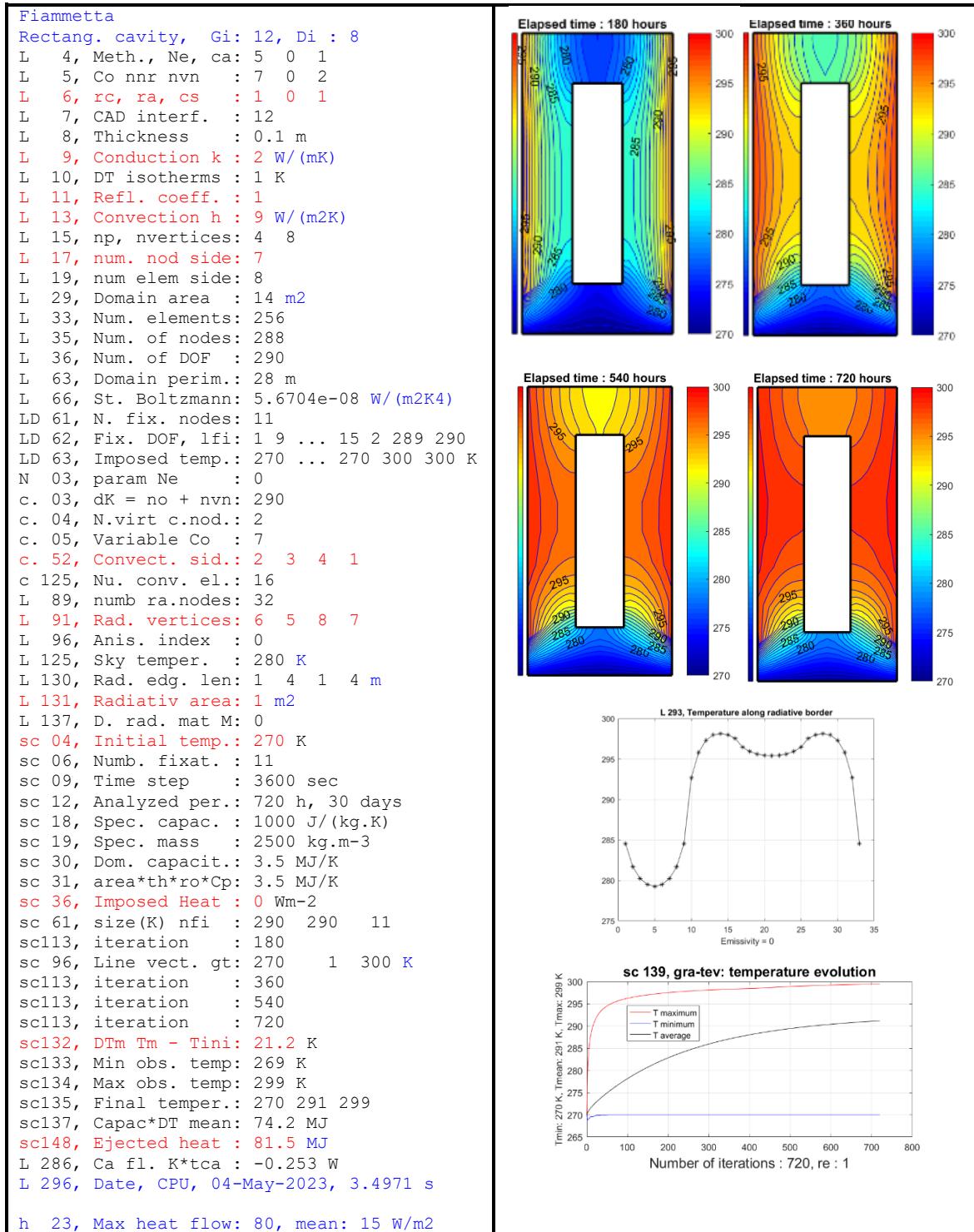


Figure 66: Adiabatic rectangular cavity, isotherms & heat flows after 180 hours

We observe (Figure 66) that after 180 hours, the heat flow is mainly horizontal meaning that the heat is inflowing in the domain, thanks to the convective vertical sides.

Before analyzing the radiative cavity, we give the result for an adiabatic cavity which is the limit situation of a cavity with perfect reflection coefficient,  $\rho = 1$  (or emissivity  $\varepsilon = 0$ ). The temperature of the inferior horizontal side of the domain is fixed to 270 K while the fluid temperature of the virtual convection nodes of both external vertical sides is equal to 300 K (see the output [LD 63](#), [Figure 65](#), where output labelled LD means that it is issued from the function `cad_Dir.m` ([Table 32](#))).

To test the radiative exchanges between the boundary elements of the rectangular cavity, we first consider a boundary whose reflection coefficient is equal to 1 ( $\rho = 1$  or emissivity  $\varepsilon = 0$ ), which means that the boundary is adiabatic. The results shown in [Figure 67](#) are the same as those of [Figure 65](#). Note the settings of [lines 94 & 101](#) in `fem_smc.m`



*Figure 67: Cavity with adiabatic boundary  $\varepsilon = 0$ ,  $\rho = 1$*

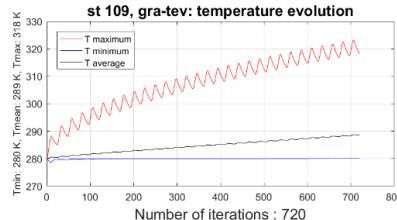
When we introduce some emissivity on the cavity boundary, the behavior of the solution is changing.

## 6.4 Street section

### 6.4.1 Periodic heat loading

A periodic heat input is introduced, the imposed heat is increasing during 6 hours, decreasing to 0 during 6 hours and absent during 12 hours (*Figure 42*). It is similar to the situation on Equator at equinoxes time.

### 6.4.2 Adiabatic or pure reflective walls, $\rho = 1$



*Figure 68: Street section – adiabatic walls - 225 DOF*

In an adiabatic rectangular street section, with a finite element model of 225 *DOF*, the temperature is always greater than the initial one ( $> 280 \text{ K}$ ). At the end of the integration process of 720 hours,  $T_{min} = 280 \text{ K}$ ,  $T_{max} = 318 \text{ K}$  and  $T_{mean} = 288 \text{ K}$ . This example is shown in *Figure 68*, etc. The method number 3 (Matlab<sup>®</sup> function *fem\_smt.m*) is used in *Fiammetta* for this problem (Solution of transient linear heat transfer problems, *line 251, Table 28*).

For  $nni = 1$ , the uni-line matrix *lg* computed in *cad\_Neu.m* (*Table 33*) contains the loaded *DOF* [7 17 8 2 9 1] (Definition of the *CAD* patches in *Figure 45*).

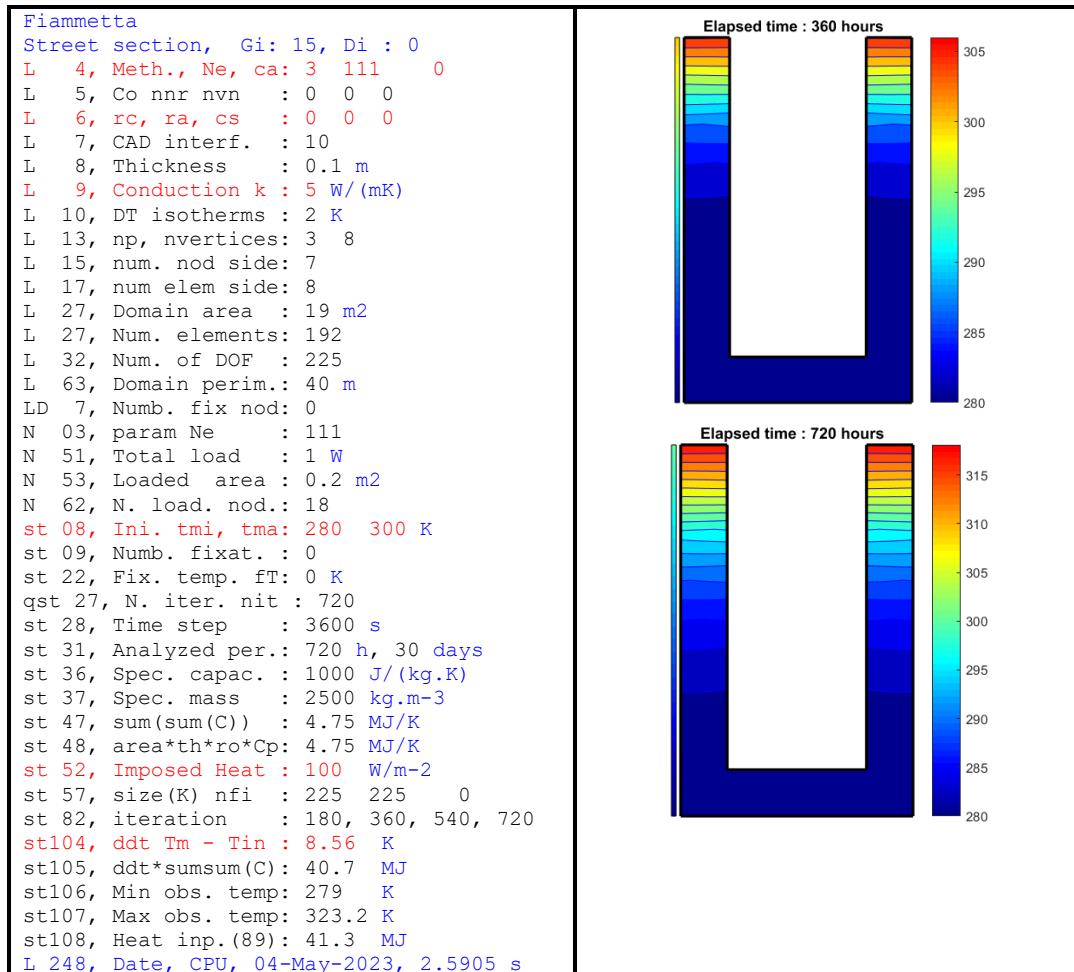


Figure 69: Adiabatic Street Section

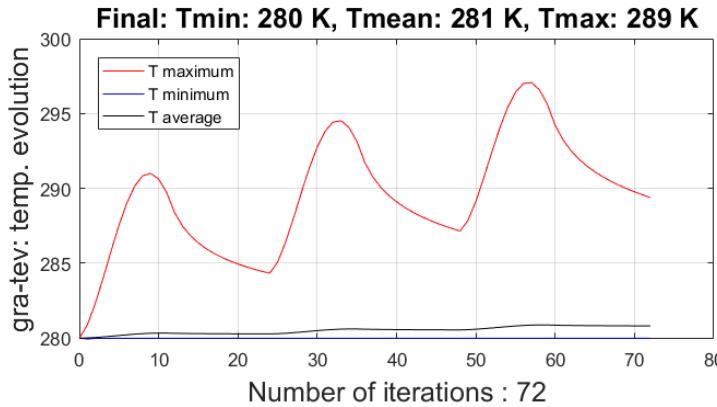


Figure 70: Temperature evolution, adiabatic walls, 3 days, 32 elements per side

The color scales of both drawings of *Figure 69* are the same when the instructions 78 - 80 of the Matlab<sup>®</sup> function *fem\_smt.m* (*Table 39*) are enabled. These results are obtained with adiabatic street boundaries. At least 8 segments per patch side are necessary to get acceptable results. The amplitudes of the oscillations of the maximum temperature are growing with the number of elements per patch side. In a short period of integration of 3 days and a fine mesh of 32 elements per patch side, the minimum temperature is always and everywhere greater than the initial one of 280 K. The amplitude of the oscillations of the maximum temperature is approximatively equal to 7 K (*Figure 70*).

A perfectly reflective wall is identical to an adiabatic one (*Figure 71*).The latter needs less data and is easier to run because it does not use radiative exchanges parameters or methods. The method used in *Fiammetta* for this problem is the number 5 (*line 266-298, Table 28*), entitled: “Solution of nonlinear transient radiative heat transfer problems”

Adiabatic wall	Perfectly reflective wall ( $\rho = 1$ )
<pre> Fiammetta Street section, Gi: 35, Di : 0 L 4, Meth., Ne, ca: 5 111 0 L 5, Co nnr nvn : 0 0 0 L 6, rc, ra, cs : 0 0 0 L 7, CAD interf. : 10 L 8, Thickness : 0.1 m L 9, Conduction k : 5 W/(mK) L 10, DT isotherms : 2 K  L 14, np, nvertices: 3 8 L 16, num. nod side: 15 L 17, num elem side: 16 L 28, Domain area : 19 m<sup>2</sup> L 27, Num. elements: 768 L 33, Num. of DOF : 833 L 63, Domain perim.: 40 m  LD 7, Numb. fix nod: 0 N 03, param Ne : 111 N 51, Total load : 1 W N 53, Loaded area : 0.2 m<sup>2</sup> N 62, N. load. nod.: 34  L 96, Anis. index : 0  sc 04, Initial temp.: 270 K sc 06, Numb. fixat. : 0 sc 09, Time step : 3600 sec sc 12, Analyzed per.: 720 h, 30 days sc 18, Spec. capac. : 1000 J/(kg.K) sc 19, Spec. mass : 2500 kg.m<sup>-3</sup> sc 30, Dom. capacit.: 4.75 MJ/K </pre>	<pre> Fiammetta Street section, Gi: 35, Di : 0 L 4, Method, Ne : 5 111 L 5, Co nnr nvn : 0 0 0 L 6, rc, ra, cs : 1 0 2 L 7, CAD interf. : 10 L 8, Thickness : 0.1 m L 9, Conduction k : 5 W/(mK) L 10, DT isotherms : 2 K L 11, Refl. coeff. : 0.9999 L 14, np, nvertices: 3 8 L 16, num. nod side: 15 L 17, num elem side: 16 L 28, Domain area : 19 m<sup>2</sup> L 27, Num. elements: 768 L 33, Num. of DOF : 833 L 63, Domain perim.: 40 m L 66, St. Boltzmann: 5.67e-08 W/(m<sup>2</sup>K<sup>4</sup>) LD 7, Numb. fix nod: 0 N 03, param Ne : 111 N 51, Total load : 1 W N 53, Loaded area : 0.2 m<sup>2</sup> N 62, N. load. nod.: 34 L 89, N. r.nodes ca: 49 L 91, Rad. vertices: 2 4 6 8 L 96, Anis. index : 0 L 125, Sky temper. : 280 K L 130, Rad. edg. len: 7 3 7 m L 131, Radiativ area: 1.7 m<sup>2</sup> L 150, Det. R. mat M: 0.2129 L 152, SVF sum(Fsky): 8.73 L 166, Sum sky loads: -0.05922 W sc 04, Initial temp.: 270 K sc 06, Numb. fixat. : 0 sc 09, Time step : 3600 sec sc 12, Analyzed per.: 720 h, 30 days sc 18, Spec. capac. : 1000 J/(kg.K) sc 19, Spec. mass : 2500 kg.m<sup>-3</sup> sc 30, Dom. capacit.: 4.75 MJ/K </pre>

```

sc 31, area*th*ro*Cp: 4.75 MJ/K
sc 36, Imposed Heat : 100 Wm-2
sc 61, size(K) nfi : 833 833 0
sc113, iteration : 180
sc113, iteration : 360
sc113, iteration : 540
sc113, iteration : 720
sc132, Tmean - Tini : 3.18 K
sc133, Min obs. temp: 270 K
sc134, Max obs. temp: 288 K
sc135, Final temper.: 270 273 285
sc137, Capac*DT mean: 15.1 MJ
sc144, Injected heat: 16.4 MJ
sc145, Ex.heat input: 16.5 MJ

L 296, Date, CPU, 04-May-2023, 11.2512 s
hf 23, Max heat flow:26, mean:6.6 W/m2

```

```

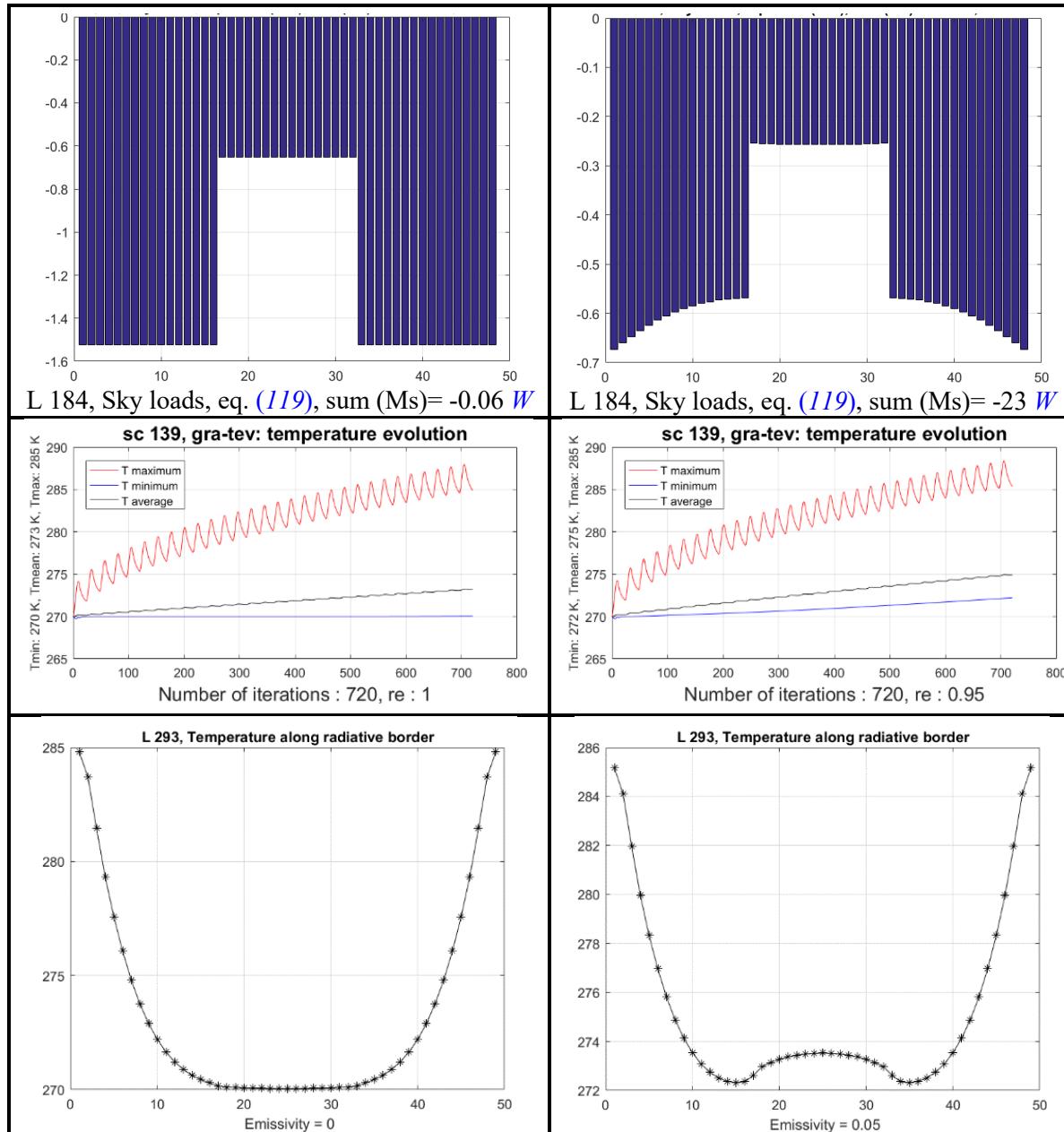
sc 31, area*th*ro*Cp: 4.75 MJ/K
sc 36, Imposed Heat : 100 Wm-2
sc 61, size(K) nfi : 833 833 0
sc110, iteration : 180
sc110, iteration : 360
sc110, iteration : 540
sc110, iteration : 720
sc118, DTm Tm - Tini: 3.18 K
sc133, Min obs. temp: 270 K
sc134, Max obs. temp: 298 K
sc135, Final temper.: 270 273 285 K
sc137, Capac*DT mean: 15.1 MJ
sc144, Injected heat: 16.4 MJ
sc145, Ex.heat input: 16.5 MJ
L 286, Ca fl. K*tca : 0.00317 W
L 296, Date, CPU, 04-May-2023, 11.5325 s
hf 23, Max heat flow: 26, mean: 6.6 W/m2

```

Figure 71: Identity between adiabatic and perfectly reflective walls ( $\rho = 1$ )

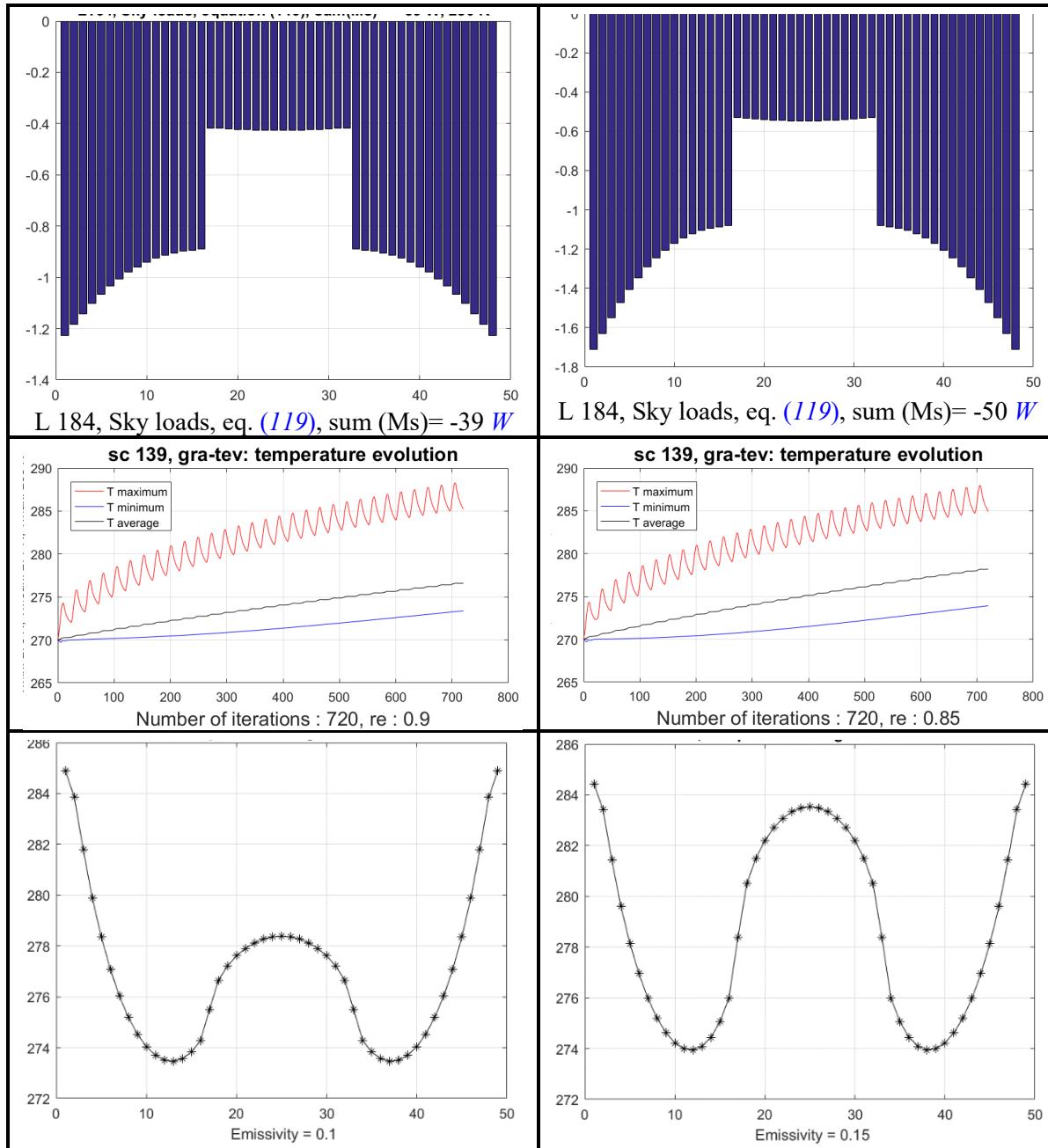
#### 6.4.3 Variation of the street walls emissivity

In the next example dealing with the same conditions as the previous one, we show three typical graphical outputs related to either an emissivity equal to zero (left) or an emissivity equal to 0.05 (right). The principal comment is that the temperature distribution on the street walls is very sensitive to their emissivity.



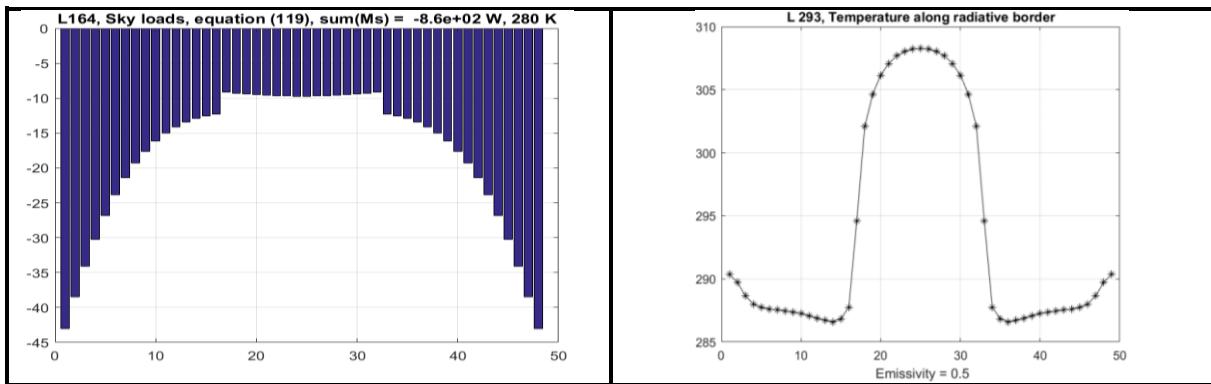
*Figure 72: Street section, comparison of two low emissivities: 0 (left), 0.05 (right)*

In the next test, we examine the situations with higher, but still small emissivities equal to 0.1 and 0.15 and, again, the results are very sensitive to the street emissivities.



*Figure 73: Street section, comparison of two low emissivities: 0.1 (left), 0.15 (right)*

With higher emissivity, but in a short period of three days, we obtain the following results:



*Figure 74: Street section,  $\rho = 0.5$ , sky loads and wall temperatures after 72 hours*

If we impose gray street walls  $\rho = 0.5$  and 280 K sky temperature, we observe in [Figure 75](#) that the heat is sucked mainly on the street walls. At the end of a cooling process of 72 hours, the temperatures are:  $T_{min72} = 287 \text{ K}$ ,  $T_{max72} = 308 \text{ K}$ ,  $T_{mean72} = 292 \text{ K}$ , (*bottom right*).

```

Fiammetta
Street section, Gi: 35, Di : 0
L 4, Method, Ne : 5 111
L 5, Co nnr nvn : 0 0 0
L 6, rc, ra, cs : 1 0 2
L 7, CAD interf. : 10
L 8, Thickness : 1 m
L 9, Conduction k : 5 W/(mK)
L 10, DT isotherms : 1 K
L 11, Refl. coeff. : 0.5
L 14, np, nvertices: 3 8
L 16, num. nod side: 15
L 17, num elem side: 16
L 28, Domain area : 19 m2
L 27, Num. elements: 768
L 33, Num. of DOF : 833
L 63, Domain perim.: 40 m
L 66, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD 7, Numb. fix nod: 0
N 03, param Ne : 111
N 61, Total load : 1 W
N 53, Loaded area : 2 m2
N 62, N. load. nod.: 34
L 89, Numb. r.nodes: 49
L 92, Rad. vert. lv: 2 4 6 8
L 97, Anis. index : 0
L 127, Sky temper. : 280 K
L 130, Rad. edg. len: 7 3 7 m
L 131, Radiativ area: 17 m2
L 152, Det. R. mat M: 0.77088
L 154, SVF sum(Fsky): 8.73
L 166, Sum sky loads: -855 W
sc 04, Initial temp.: 290 K
sc 06, Numb. fixat. : 0
sc 09, Time step : 3600 sec
sc 12, Analyzed per.: 72 h, 3 days
sc 18, Spec. capac. : 1000 J/(kg.K)
sc 19, Spec. mass : 2500 kg.m-3
sc 30, Dom. capacit.: 47.5 MJ/K
sc 31, area*th*ro*Cp: 47.5 MJ/K
sc 36, Imposed Heat : 100 Wm-2
sc 61, size(K) nfi : 833 833 0
sc110, iteration : 18, 36, 54
rs 25, sum(Mss) : -855 W
sc110, iteration : 72
sc132, Tmean - Tini : 2.11 K
sc133, Min obs. temp: 287 K
sc134, Max obs. temp: 308 K
sc135, Final temper.: 287 292 308 K
sc137, Capac*DT mean: 100 MJ
sc144, Injected heat: 16.4 MJ
sc145, Ex.heat input: 16.5 MJ
L 286, Radiative fl.: 74.9 W
L 296, Date, CPU, 04-May-2023, 2.8192 s
hf 25, Max heat flow: 160, mean: 25 W/m2

```

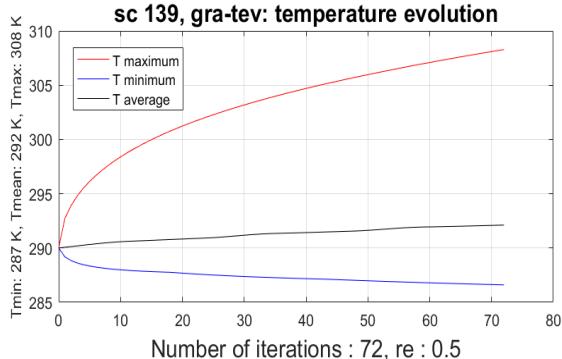
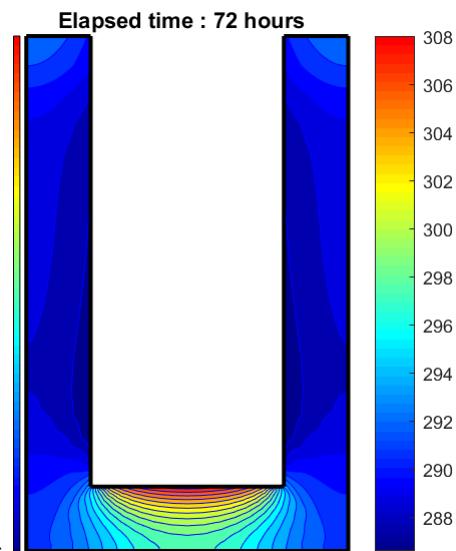
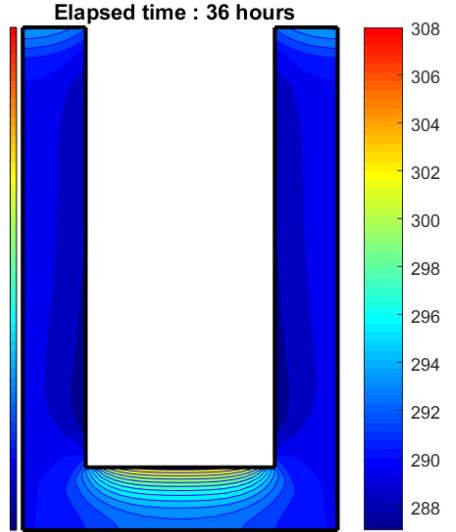


Figure 75: Street section,  $\rho = 0.5$ , injected heat: 16.4 MJ, sky temperature = 280 K

#### 6.4.4 Black body walls, $\rho = 0$

```

Fiammetta
Street section, Gi: 35, Di : 0
L 4, Meth., Ne, ca: 5 111 0
L 5, Co nnr nvn : 0 0 0
L 6, rc, ra, cs : 1 0 2
L 7, CAD interf. : 10
L 8, Thickness : 0.1 m
L 9, Conduction k : 5 W/(mK)
L 10, DT isotherms : 2 K
L 11, Refl. coeff. : 0
L 15, np, nvertices: 3 8
L 17, num. nod side: 15
L 19, num elem side: 16
L 29, Domain area : 19 m2
L 32, Num. elements: 768
L 34, Num. of DOF : 833
L 63, Domain perim.: 40 m
L 66, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD 7, Numb. fix nod: 0
N 03, param Ne : 111
N 51, Total load : 1 W
N 53, Loaded area : 0.2 m2
N 62, N. load. nod.: 34
L 89, numb ra.nodes: 49
L 90, Rad. vertices: 2 4 6 8
L 96, Anis. index : 0
L 125, Sky temper. : 280 K
L 130, Rad. edg. len: 7 3 7 m
L 131, Radiativ area: 1.7 m2
L 150, Det. R. mat M: 1
L 152, SVF sum(Fsky): 8.73
L 166, Sum sky loads: -104.5 W
sc 04, Initial temp.: 270 K
sc 06, Numb. fixat. : 0
sc 09, Time step : 3600 sec
sc 12, Analyzed per.: 720 h, 30 days
sc 18, Spec. capac. : 1000 J/(kg.K)
sc 19, Spec. mass : 2500 kg.m-3
sc 30, Solid heat C.: 4.75 MJ/K
sc 31, area*th*ro*Cp: 4.75 MJ/K
sc 36, Imposed Heat : 100 Wm-2
sc 59, size(K) nfi : 833 833 0
sc 95, iteration : 180
sc 95, iteration : 360
sc 95, iteration : 540
rs 25, sum(Mss) : -104.5 W
sc 95, iteration : 720
sc118, DTm Tm - Tini: 21.3 K
sc119, Min obs. temp: 266 K
sc120, Max obs. temp: 326 K
sc121, Final temper.: 275 291 326
sc123, Capacity* DTm: 101 MJ
sc130, Injected heat: 16.4 MJ
sc131, Ex.heat input: 16.5 MJ
L 287, Ca fl. K*tca : 14.2 W
L 296, Date, CPU, 04-May-2023, 11.5299 s
hf 25, Max heat flow: 330, mean: 39 W/m2

```

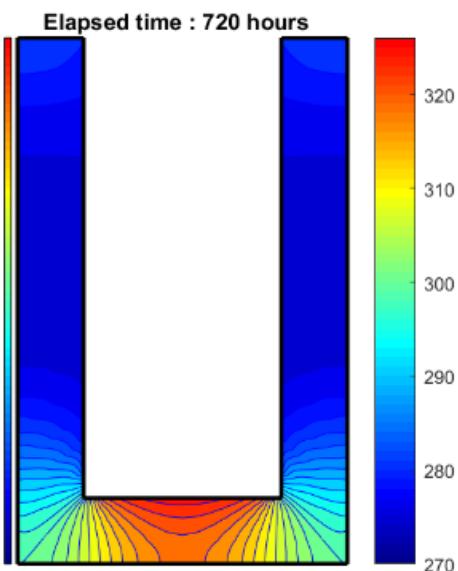
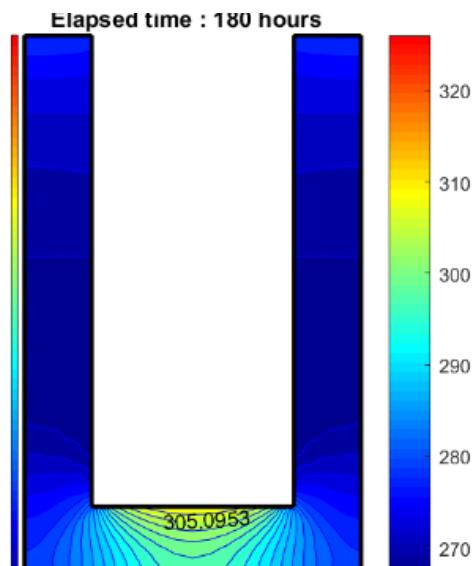
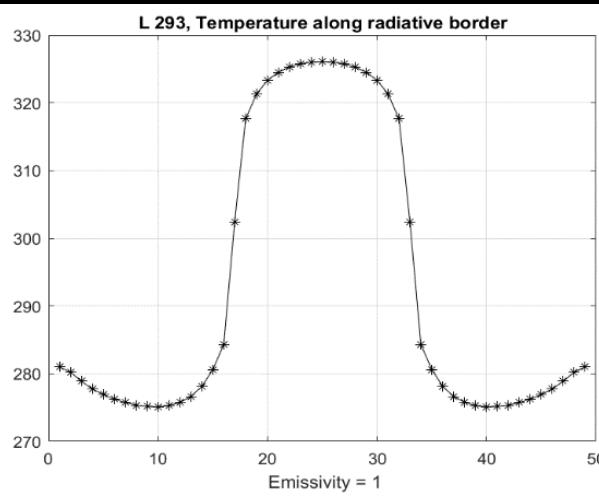
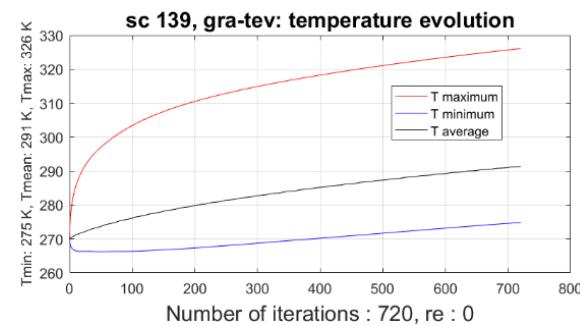


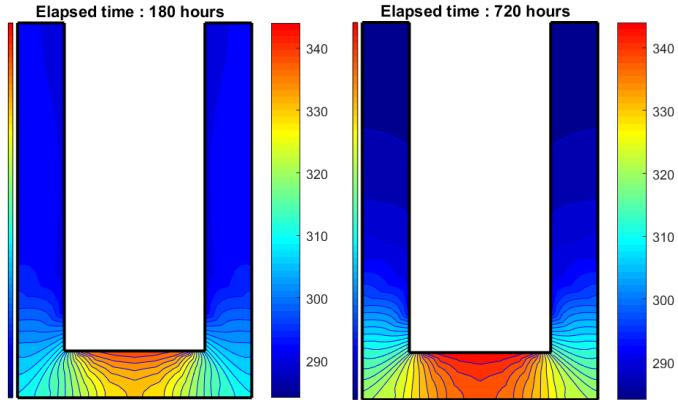
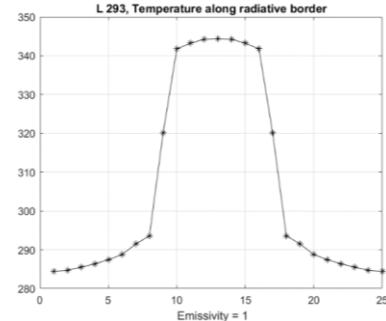
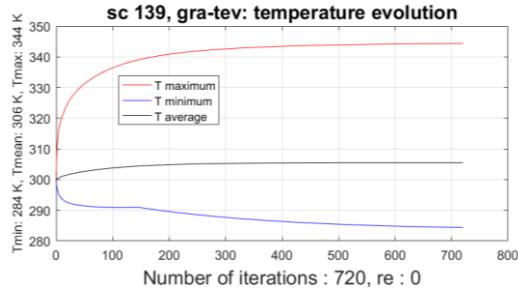
Figure 76: Street section with black body walls and injected heat of 100 Wm<sup>-2</sup>, 16.4 MJ

If we impose again black body street walls and remove the imposed heat load, the solution becomes:

```

Fiammetta
Street section, Gi: 14, Di : 0
L 4, Meth., Ne, ca: 5 0 0
L 5, Co nnr nvn : 0 0 0
L 6, rc, ra, cs : 1 0 2
L 8, Thickness : 1 m
L 9, Conduction k : 5 W/ (mK)
L 10, DT isotherms : 2 K
L 11, Refl. coeff. : 0
L 14, np, nvertices: 3 8
L 16, num. nod side: 7
L 17, num elem side: 8
L 28, Domain area : 19 m2
L 27, Num. elements: 192
L 33, Num. of DOF : 225
L 63, Domain perim.: 40 m
L 66, St. Boltzmann: 5.67e-08 W/ (m2K4)
LD 7, Numb. fix nod: 0
N 03, param Ne : 0
L 89, numb ra.nodes: 25
L 91, Rad. vertices: 2 4 6 8
L 96, Anis. index : 0
L 125, Sky temper. : 280 K
L 131, Rad. edg. len: 7 3 7 m
L 132, Radiativ area: 17 m2
L 152, SVF sum(Fsky): 4.36
L 166, Sum sky loads: -1042 W
sc 04, Initial temp.: 300 K
sc 06, Numb. fixat. : 0
sc 09, Time step : 3600 sec
sc 12, Analyzed per.: 720 h, 30 days
sc 18, Spec. capac. : 1000 J/(kg.K)
sc 19, Spec. mass : 2500 kg.m-3
sc 30, Dom. capacit.: 47.5 MJ/K
sc 31, area*th*ro*Cp: 47.5 MJ/K
sc 36, Imposed Heat : 0 Wm-2
sc 61, size(K) nfi : 225 225 0
sc110, iteration : 180,360, 540
rs 44, sum(Mss) : -1042 W
sc113, iteration : 720
sc118, DTm Tm - Tini: 5.51 K
sc133, Min obs. temp: 284 K
sc134, Max obs. temp: 344 K
sc135, Final temper.: 284 306 344 K
sc123, Capacity* DTm: 262 MJ
L 287, He fl. K*tca : -4.97 W
L 296, Date, CPU, 04-May-2023, 2.4296 s
hf 25, Max heat flow: 230, mean: 41 W/m2

```

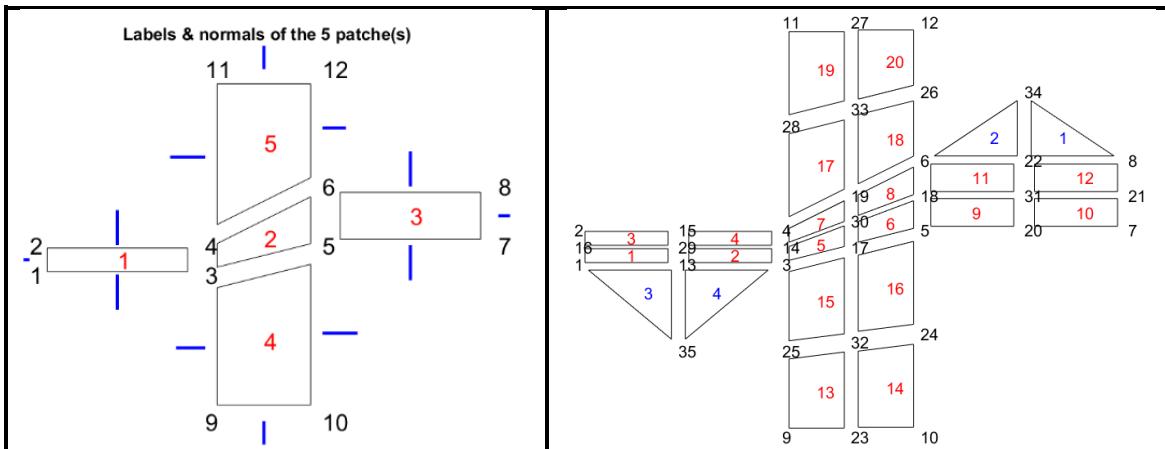


*Figure 77: Street section with black body walls and without injected heat*

From the pure geometric characteristic [ $F_{sky}$ ], we deduce the impact of the radiation emitted by the sky, which is proportional to its temperature.

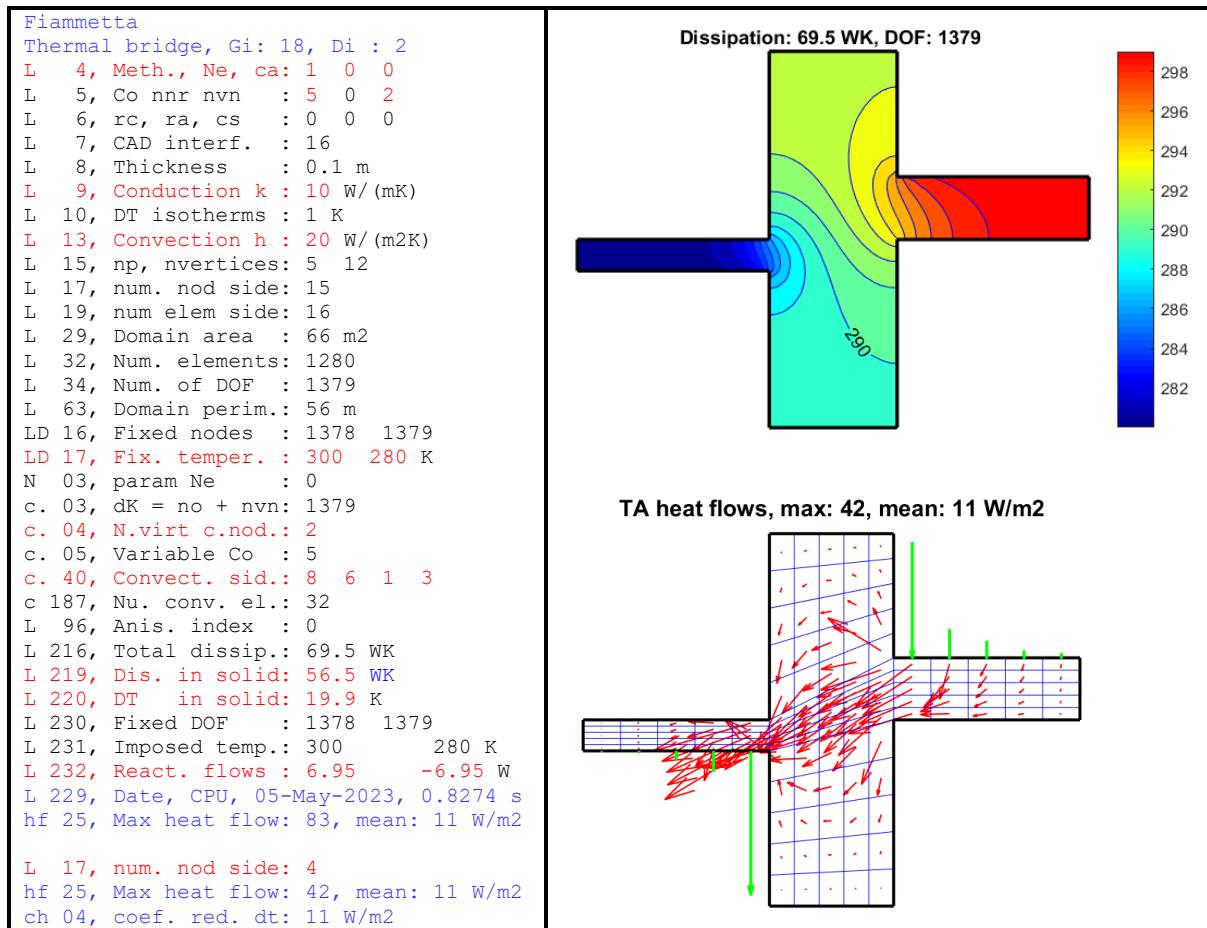
## 5.5 Thermal bridge

### 5.5.1 Stationary heat flow - 2 convective virtual nodes



*Figure 78: Thermal bridge, topological configuration*

The next example relates to a domain involving convective boundaries on sides 1-3 and 6-8 of the main body (*Figure 78*). The convection condition are imposed only on two horizontal sides: 8 – 6 on the right and 1 – 3 on the left (*c. 40, Convect. sid.:*). The localization matrix *lc* of the convective elements is computed in the function *cad\_con.m* (*Fiammetta, line 78*).



*Figure 79: Thermal bridge with 2 convective virtual nodes, steady state*

- 1 radiative virtual node & 1 convective virtual node

In the example of *Figure 80*, the horizontal edges of the right side of the domain are submitted to convection, and the full left side to radiation. In this example, we use an adaptation of the convective elements to generate radiative ones. Radiation is acting from the radiative boundary to a reference virtual node at imposed temperature of 270 K.

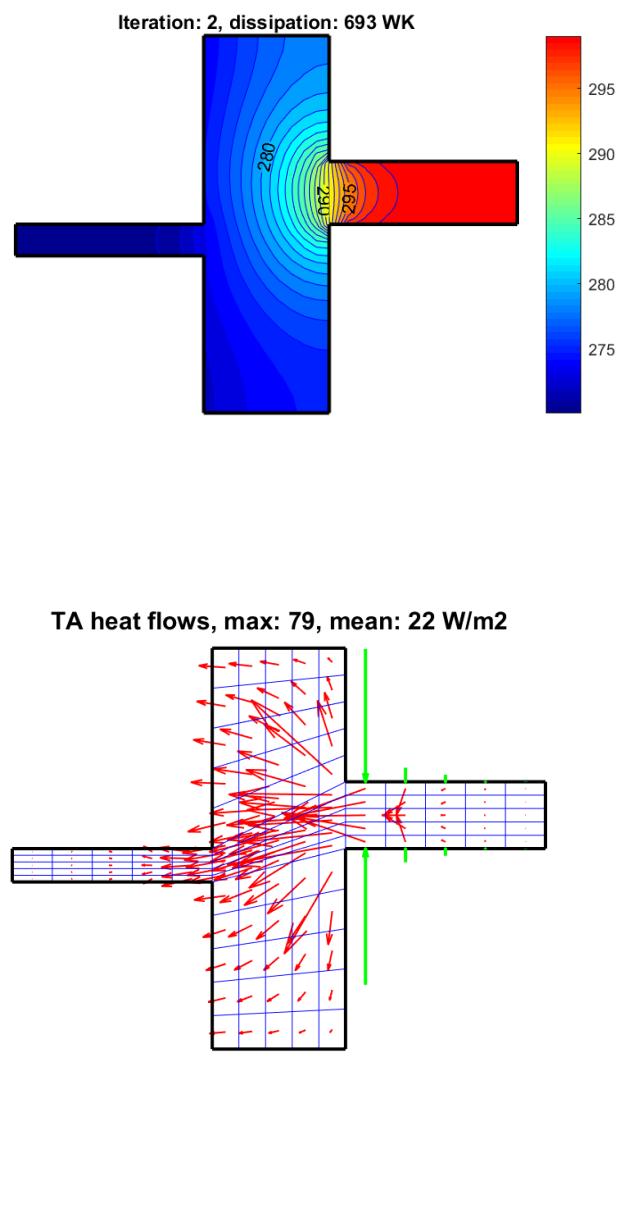
The product of the reaction flow: 23.103 W (*sd 40, Figure 80*) by the difference of temperature: 30 K (*sd 55, Figure 80*) is equal to the dissipation: 693.1 WK (*sd 39, Figure 80*). The use of “conductive-radiative” elements (*Table 46*) equivalent to “conductive-convective” elements (*Table 45*) is acceptable but not very effective because the heat exchange is concentrated on a single virtual node. To take correctly into account the interactions between radiative elements, it is necessary to link them through the view factors.

```

Fiammetta_20211111
Thermal bridge, Gi: 38, Di : 22
L 4, Meth., Ne, ca: 2 0 0
L 5, Co nnr nvn : 8 1 1
L 6, rc, ra, cs : 1 0 3
L 7, CAD interf. : 16
L 8, Thickness : 0.1 m
L 9, Conduction k : 10 W/ (mK)
L 10, DT isotherms : 1 K
L 11, Refl. coeff. : 0.5
L 13, Convection h : 20 W/ (m2K)
L 15, np, nvertices: 5 12
L 17, num. nod side: 9
L 19, num elem side: 10
L 29, Domain area : 66 m2
L 32, Num. elements: 500
L 35, Num. of nodes: 561
L 36, Num. of DOF : 563
L 63, Domain perim.: 56 m
L 68, St. Boltzmann: 5.67e-08 W/ (m2K4)
LD 21, Fixed nodes : 562 563
LD 22, Fix. temper. : 270 300 K
LD177, N. fix. nodes: 2
N 03, param Ne : 0
c. 03, dK = no + nvn: 563
c. 04, N.virt c.nod.: 1
c. 05, Variable Co : 8
c. 70, Convect. sid.: 8 6 5 7
c 179, Nu. conv. el.: 20
L 96, Anis. index : 0
L 125, Sky temper. : 280 K
L 130, Rad. edg. len: 6 6 1 6 5 m
L 131, Radiativ area: 2.4 m2
sd 5, N. fixed nod.: 2
sd 10, N. rad elem. : 50
sd 24, Iteration N. : 1 / 2
sd 39, Dissipation : 617.2 WK
sd 40, React. flows : 20.574 -20.574 W
sd 55, max - min tca: 30 K
sd 24, Iteration N. : 2 / 2
sd 39, Dissipation : 693.1 WK
sd 40, React. flows : 23.103 -23.103 W
sd 55, max - min tca: 30 K
L 237, T. vir. nod. : 300 K
L 241, Date, CPU, 05-May-2023, 1.5 s
hf 25, Max heat flow: 110, mean: 21 W/m2
g 18, Max temp grad: 11, mean: 2.1 K/m

With L 17, num. nod side: 4
hf 25, Max heat flow: 79, mean: 22 W/m2
g 18, Max temp grad: 7.9, mean: 2.2 K/m

```



*Figure 80: Thermal bridge, mixed b. c., 1 convective virtual node, 1 radiative v. node ( $\rho = .5$ )*

When the number of nodes per patch side is equal to one, the vector *lcont* defining the set of radiative nodes is: *lcont* = [11 28 4 15 2 16 1 13 3 25 9]. The patch vertices present in this sequence are given in the vector *lv* (*lv'* = [11 4 2 1 3 9]). The lengths involved in this set are given in the column vector *Lel* (output: *L 130, Rad. edg. len:* 6 6 1 6 5 *m*). Because the flow is stationary and virtual radiative node is used, the solution is computed with method 2: “Nonlinear stat. rad. heat transfer - conv. like virtual node” (*L 233*).

## 6.5.2 Transient heat exchanges

### a. Two convective virtual nodes

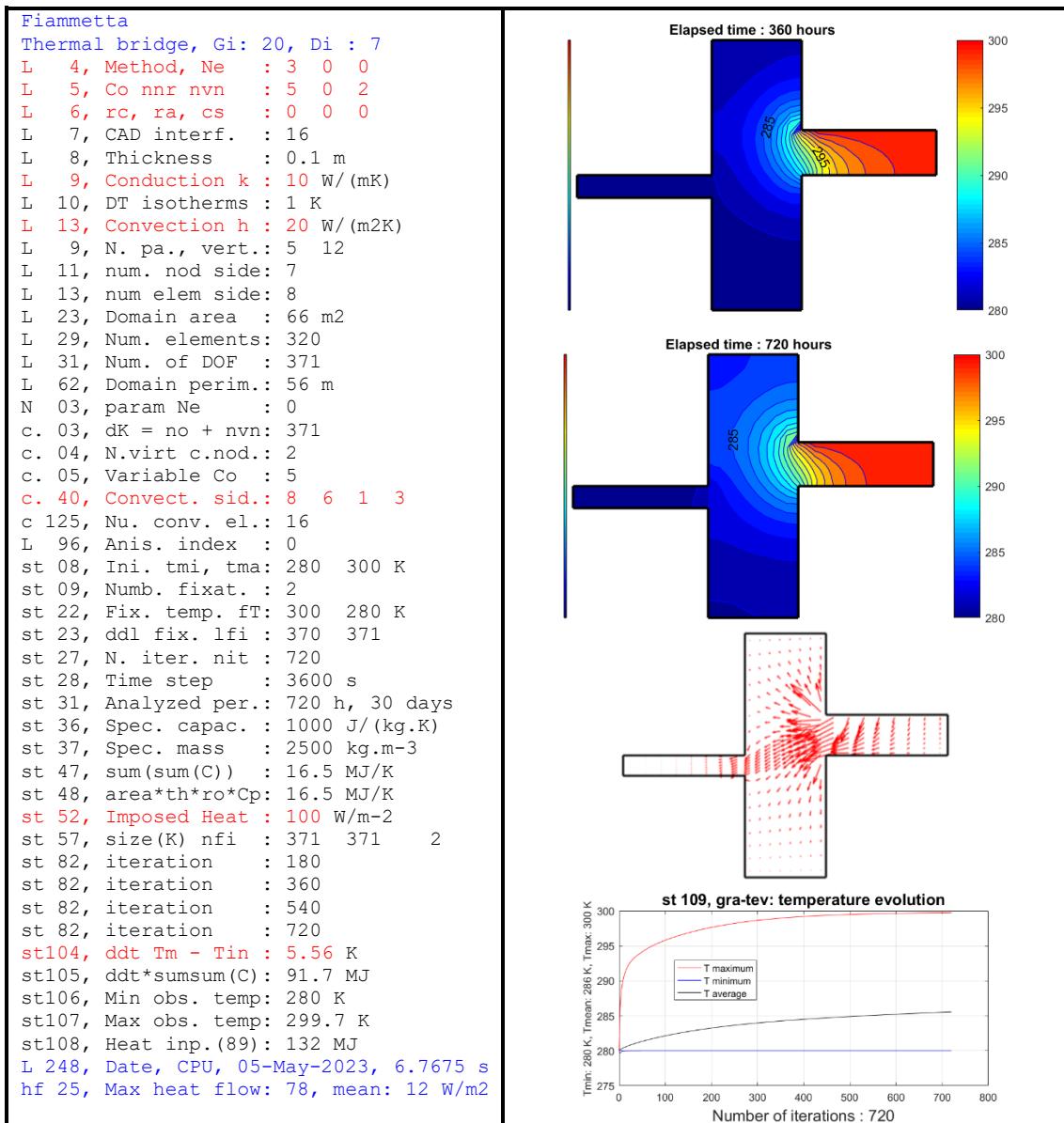
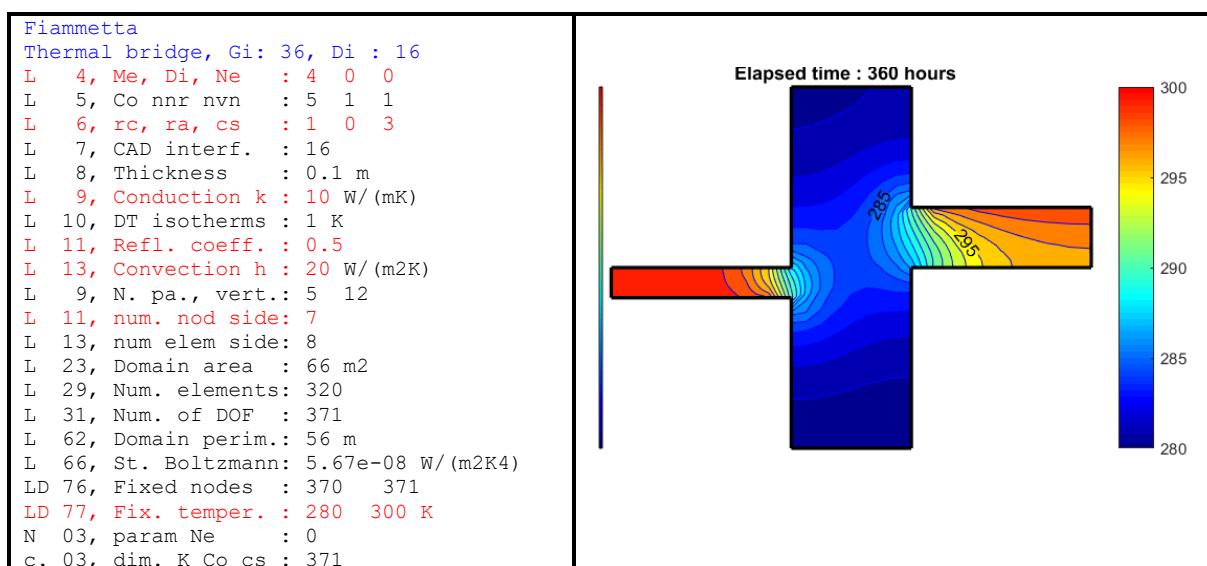


Figure 81: Thermal bridge, transient analysis, 2 convective virtual nodes

### b. One convective and one radiative virtual node



```

c. 04, N.virt c.nod.: 1
c. 05, Variable Co : 5
c. 40, Convect. sid.: 8 6 1 3
c 125, Nu. conv. el.: 16
L 89, numb ra.nodes: 41
L 96, Anis. index : 0
L 125, Sky temper. : 280 K
L 130, Rad. edg. len: 6 6 1 6 5 m
L 131, Radiat. area : 2.4 m2
L 258, Num. fix. DOF: 370 371
L 259, T. fix. nod. : 280 300 K
sq 04, Initial temp.: 280 K
sq 05, Fixed DOF : 370 371
sq 09, Time step : 3600 s
sq 11, Analyzed per.: 720 h, 30 days
sq 17, Spec. capac. : 1000 J/(kg.K)
sq 18, Spec. mass : 2500 kg.m-3
sq 28, sum(sum(C)) : 16.5 MJ/K
sq 29, area*th*ro*Cp: 16.5 MJ/K
sq 32, Impos. Heat : 0 W/m-2
sq 38, mcr nv : 0 1
sq 44, N. rad. edges: 5
sq 45, rad seg leng.: 6 6 1 6 5 m
sq 46, N. rad. elem.: 40
sq122, Iteration : 360 gt: 280 1 300 K
sq122, Iteration : 720 gt: 280 1 300 K
sq143, Tmean - Tini : 11.1 K
sq144, Stored heat : 183 MJ
sq146, Min it+1 temp: 282.3 K
sq147, Max it+1 temp: 291 K
sq148, Max it+1 temp: 300 K
sq154, Ejected heat : 0.202 MJ
L 260, Diss in solid: 127 WK
L 261, Total dissip.: 165 WK
L 262, Date, CPU, 05-May-2023, 4.3016 s
hf 25, Max heat flow: 77, mean: 16 W/m2

```

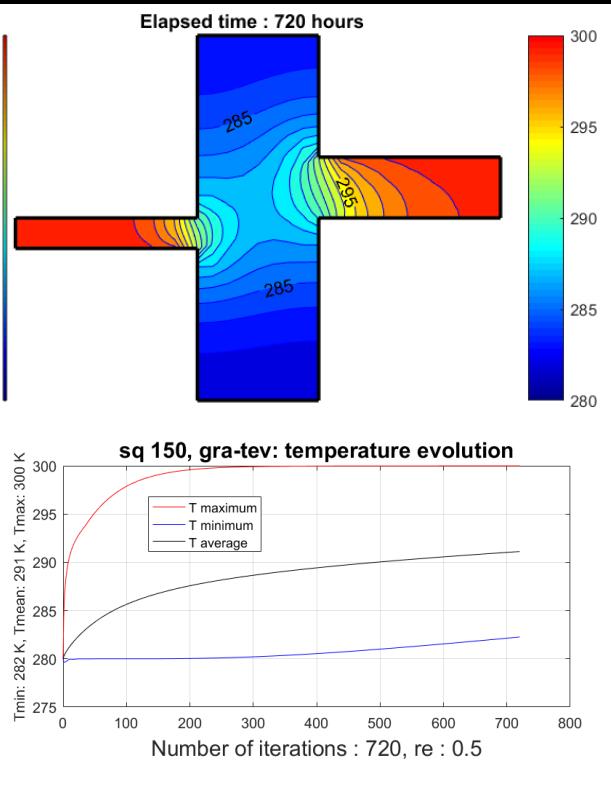


Figure 82: Thermal bridge, transient analysis, 1 convective & 1 radiative virt. node,  $\rho = .5$

### c. Radiative edge

Fiammetta	Fiammetta
Thermal bridge, Gi: 21, Di : 2	Thermal bridge, Gi: 22, Di : 12
L 4, Me, Di, Ne : 4 0 0	L 4, Me, Di, Ne : 4 0 0
L 5, Co nnr nvn : 6 1 1	L 5, Co nnr nvn : 6 0 1
L 6, rc, ra, cs : 1 0 3	L 6, rc, ra, cs : 1 1 3
L 7, CAD interf. : 16	L 7, CAD interf. : 16
L 8, Thickness : 0.1 m	L 8, Thickness : 0.1 m
L 9, Conduction k : 10 W/ (mK)	L 9, Conduction k : 10 W/ (mK)
L 10, DT isotherms : 1 K	L 10, DT isotherms : 1 K
L 11, Refl. coeff. : 1	L 11, Refl. coeff. : 1
L 13, Convection h : 20 W/ (m <sup>2</sup> K)	L 13, Convection h : 20 W/ (m <sup>2</sup> K)
L 9, N. pa., vert.: 5 12	L 15, np, nvertices: 5 12
L 11, num. nod side: 7	L 11, num. nod side: 7
L 13, num elem side: 8	L 13, num elem side: 8
L 30, Domain area : 66 m <sup>2</sup>	L 30, Domain area : 66 m <sup>2</sup>
L 33, Num. elements: 320	L 33, Num. elements: 320
L 35, Num. of nodes: 369	L 35, Num. of nodes: 369
L 36, Num. of DOF : 371	L 36, Num. of DOF : 370
L 62, Domain perim.: 56 m	L 59, Domain perim.: 56 m
L 68, St. Boltzmann: 5.6704e-08 W/ (m <sup>2</sup> K <sup>4</sup> )	L 66, St. Boltzmann: 5.6704e-08 W/ (m <sup>2</sup> K <sup>4</sup> )
LD 16, Fixed nodes : 370 371	LD110, Fixed nodes : 370
LD 17, Fix. temper. : 280 300 K	LD111, Fix. temper. : 300 K
N 03, param Ne : 0	N 03, param Ne : 0
c. 03, Numb. var. dK: 371	c. 03, Numb. var. dK: 370
c. 04, N.virt c.nod.: 1	c. 04, N.virt c.nod.: 1
c. 05, Variable Co : 6	c. 05, Variable Co : 6
c. 46, Convect. sid.: 8 6 5 7	c. 46, Convect. sid.: 8 6 5 7
c 125, Nu. conv. el.: 16	c 125, Nu. conv. el.: 16
L 96, Anis. index : 0	L 96, Anis. index : 0
L 127, Sky temper. : 280 K	L 125, Sky temper. : 280 K
L 131, Rad. edg. len: 6 6 1 6 5 m	L 130, Rad. edg. len: 6 6 1 6 5 m
L 132, Radiativ area: 2.4 m <sup>2</sup>	L 131, Radiativ area: 2.4 m <sup>2</sup>
	L 166, dim F = nv : 42
	L 175, determinant M: 0.6935
	L 179, Sum 2d member: -0.0836 W
I 239, Num. fix. DOF: 370 371	I 253, Num. fix. DOF: 370
L 240, T. fix. nod. : 280 300 K	L 254, T. fix. nod. : 300 K
sq 04, Initial temp.: 280 K	sq 04, Initial temp.: 280 K
sq 05, Fixed DOF : 370 371	sq 05, Fixed DOF : 370
sq 09, Time step : 3600 s	sq 09, Time step : 3600 s
sq 11, Analyzed per.: 720 h, 30 days	sq 11, Analyzed per.: 720 h, 30 days

sq 17, Spec. capac. : 1000 J/(kg.K) sq 18, Spec. mass : 2500 kg.m <sup>-3</sup> sq 28, sum(sum(C)) : 16.5 MJ/K sq 29, area*th*ro*Cp: 16.5 MJ/K sq 32, Impos. Heat : 0 W/m <sup>-2</sup> sq 38, mcr nv : 0 1 sq 44, N. rad. edges: 5 sq 45, rad seg leng.: 6 6 1 6 5 m sq 46, N. rad. elem.: 40  sq122, Iteration : 360 gt: 280 1 300 K sq122, Iteration : 720 gt: 280 1 300 K <b>sq143, Tmean - Tini : 6.89 K</b> sq144, Stored heat : 114 MJ sq146, Min it+1 temp: 280.3 K sq147, Max it+1 temp: 287 K sq148, Max it+1 temp: 300 K sq154, Ejected heat : 0.202 MJ <b>L 260, Diss in solid: 111 WK</b> L 261, Total dissip.: 140 WK <b>L 262, Date, CPU, 05-May-2023, 4.1104 s</b> <b>hf 25, Max heat flow: 75, mean: 13 W/m2</b>	sq 17, Spec. capac. : 1000 J/(kg.K) sq 18, Spec. mass : 2500 kg.m <sup>-3</sup> sq 28, sum(sum(C)) : 16.5 MJ/K sq 29, area*th*ro*Cp: 16.5 MJ/K sq 32, Impos. Heat : 0 W/m <sup>-2</sup> <b>sq 38, mcr nv : 0 42</b> sq 44, N. rad. edges: 5 sq 45, rad seg leng.: 6 6 1 6 5 m sq 46, N. rad. elem.: 40 sq 63, sum(gr) : 0 W sq 77, sum(Mn) : 0 W sq122, Iteration : 360 gt: 280. 1 300 K sq122, Iteration : 720 gt: 280.3 1 300 K <b>sq143, Tmean - Tini : 6.89 K</b> sq144, Stored heat : 114 MJ sq146, Min it+1 temp: 280.3 K sq147, Max it+1 temp: 287 K sq148, Max it+1 temp: 300 K sq154, Ejected heat : 0.202 MJ <b>L 260, Diss in solid: 111 WK</b> L 261, Total dissip.: 140 WK <b>L 262, Date, CPU, 05-May-2023, 4.4732 s</b> <b>hf 25, Max heat flow: 75, mean: 13 W/m2</b>
--	---

Figure 83: Thermal bridge comparison, radiative virtual node – radiative edge,  $\rho = 1$

With perfectly reflective boundaries, ( $\rho = 1$ ), the results are identical with either virtual radiative node or radiative edge. For radiative exchange handled with a virtual radiative node, the result is independent of the reflection coefficient  $\rho$  more or less in the range  $0 \leq \rho \leq .9$ .

<b>Fiammetta</b> Thermal bridge, Gi: 21, Di : 2 L 4, Me, Di, Ne : 4 0 0 L 5, Co nrr nvn : 6 1 1 L 6, rc, ra, cs : 1 0 3 L 7, CAD interf. : 16 L 8, Thickness : 0.1 m L 9, Conduction k : 10 W/(mK) L 10, DT isotherms : 1 K L 11, Refl. coeff. : 0.5 L 13, Convection h : 20 W/(m2K) L 9, N. pa., vert.: 5 12 L 11, num. nod side: 7 L 13, num elem side: 8 L 23, Domain area : 66 m <sup>2</sup> L 29, Num. elements: 320 L 31, Num. of DOF : 371 L 62, Domain perim.: 56 m L 68, St. Boltzmann: 5.6704e-08 W/(m2K4) LD 16, Fixed nodes : 370 371 <b>LD 17, Fix. temper. : 280 300 K</b> N 03, param Ne : 0 c. 03, Numb. var. dK: 371 c. 04, N.virt c.nod.: 1 c. 05, Variable Co : 5 c. 40, Convect. sid.: 8 6 5 7 c 125, Nu. conv. el.: 16 L 89, size ra.nodes: 41 L 96, Anis. index : 0 L 129, Sky temper. : 280 K L 134, Rad. edg. len: 6 6 1 6 5 m L 132, Radiativ area: 2.4 m <sup>2</sup>  <b>L 253, Num. fix. DOF: 370 371</b> L 254, T. fix. nod. : 280 300 K sq 04, Initial temp.: 280 K sq 05, Fixed DOF : 370 371 sq 09, Time step : 3600 s sq 11, Analyzed per.: 720 h, 30 days sq 17, Spec. capac. : 1000 J/(kg.K) sq 18, Spec. mass : 2500 kg.m <sup>-3</sup> sq 28, sum(sum(C)) : 16.5 MJ/K sq 29, area*th*ro*Cp: 16.5 MJ/K sq 32, Impos. Heat : 0 W/m <sup>-2</sup> sq 38, mcr nv : 0 1 sq 44, N. rad. edges: 5 sq 45, rad seg leng.: 6 6 1 6 5 m sq 46, N. rad. elem.: 40	<b>Fiammetta</b> Thermal bridge, Gi: 22, Di : 12 L 4, Me, Di, Ne : 4 0 0 L 5, Co nrr nvn : 6 0 1 L 6, rc, ra, cs : 1 1 3 L 7, CAD interf. : 16 L 8, Thickness : 0.1 m L 9, Conduction k : 10 W/(mK) L 10, DT isotherms : 1 K L 11, Refl. coeff. : .5 L 13, Convection h : 20 W/(m2K) L 15, np, nvertices: 5 12 L 11, num. nod side: 7 L 13, num elem side: 8 L 23, Domain area : 66 m <sup>2</sup> L 26, Num. elements: 320 L 28, Num. of DOF : 370 L 59, Domain perim.: 56 m L 66, St. Boltzmann: 5.6704e-08 W/(m2K4) LD110, Fixed nodes : 370 <b>LD111, Fix. temper. : 300 K</b> N 03, param Ne : 0 c. 03, Numb. var. dK: 370 c. 04, N.virt c.nod.: 1 c. 05, Variable Co : 6 c. 46, Convect. sid.: 8 6 5 7 c 125, Nu. conv. el.: 16 L 89, size ra.nodes: 41 L 96, Anis. index : 0 L 125, Sky temper. : 280 K L 130, Rad. edg. len: 6 6 1 6 5 m L 131, Radiativ area: 2.4 m <sup>2</sup> L 166, dim F = nv : 42 L 177, determinant M: 0.9161 L 181, Sum 2d member: -348 W <b>L 253, Num. fix. DOF: 370</b> L 254, T. fix. nod. : 300 K sq 04, Initial temp.: 280 K sq 05, Fixed DOF : 370 sq 09, Time step : 3600 s sq 11, Analyzed per.: 720 h, 30 days sq 17, Spec. capac. : 1000 J/(kg.K) sq 18, Spec. mass : 2500 kg.m <sup>-3</sup> sq 28, sum(sum(C)) : 16.5 MJ/K sq 29, area*th*ro*Cp: 16.5 MJ/K sq 32, Impos. Heat : 0 W/m <sup>-2</sup> sq 38, mcr nv : 0 42 sq 44, N. rad. edges: 5 sq 45, rad seg leng.: 6 6 1 6 5 m sq 46, N. rad. elem.: 40
--	--

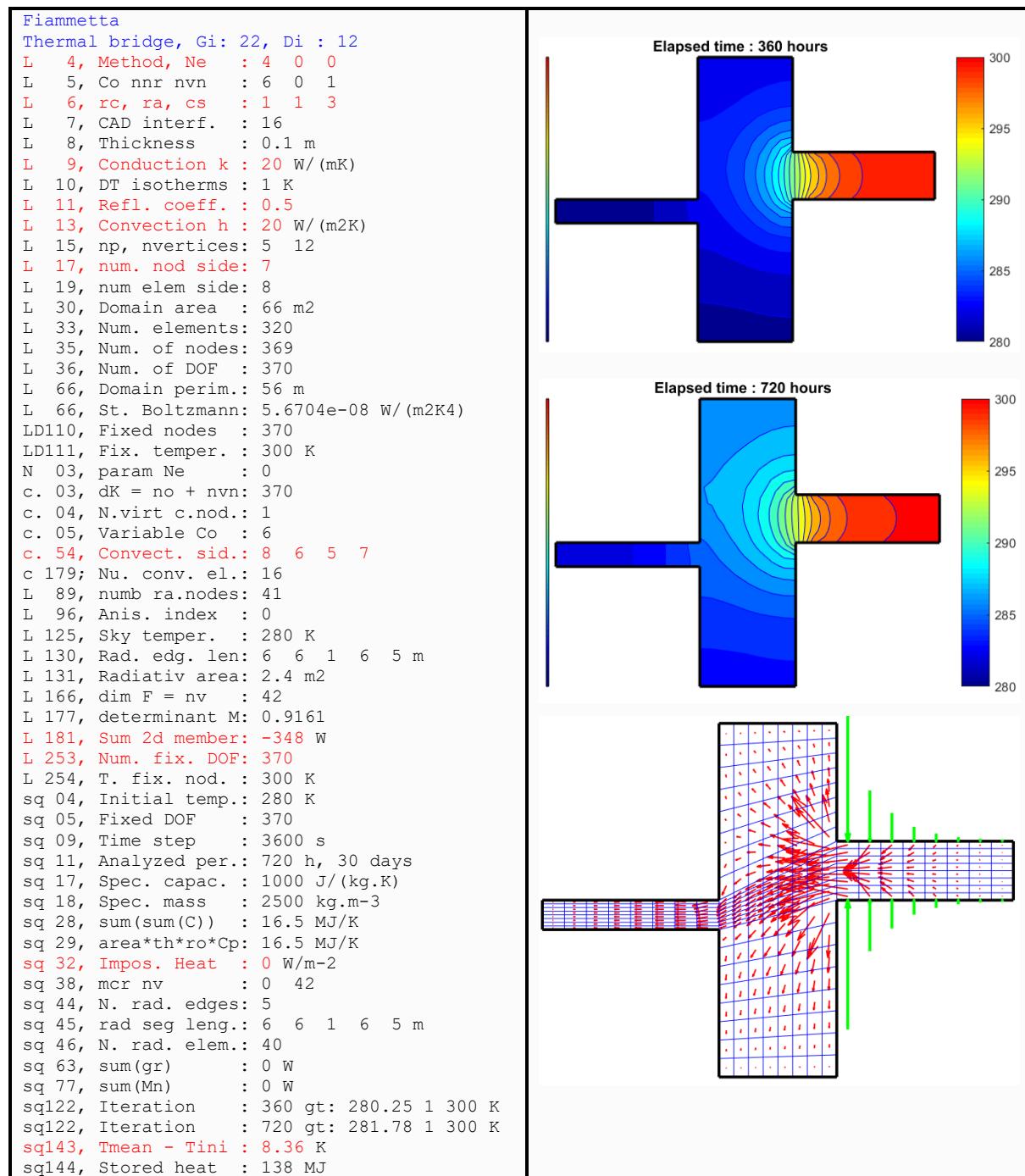
```

sq122, Iteration      : 360 gt: 280 1 300 K
sq122, Iteration      : 720 gt: 280 1 300 K
sql43, Tmean - Tini : 6.89 K
sq144, Stored heat   : 114 MJ
sq146, Min it+1 temp: 280.3 K
sq147, Max it+1 temp: 287 K
sq148, Max it+1 temp: 300 K
sq154, Ejected heat  : 0.202 MJ
L 260, Diss in solid: 111 WK
L 261, Total dissip.: 140 WK
L 262, Date, CPU, 05-May-2023, 4.1935 s
hf 25, Max heat flow: 75, mean: 13 W/m2
sql43, Tmean - Tini : 6.89 K
sq144, Stored heat   : 114 MJ
sq146, Min it+1 temp: 280.3 K
sq147, Max it+1 temp: 287 K
sq148, Max it+1 temp: 300 K
sq154, Ejected heat  : 0.202 MJ
L 260, Diss in solid: 111 WK
L 261, Total dissip.: 140 WK
L 262, Date, CPU, 05-May-2023, 4.2575 s
hf 25, Max heat flow: 75, mean: 13 W/m2

```

*Figure 84: Thermal bridge comparison, radiative virtual node – radiative edge,  $\rho = .5$*

Now, we take into account the inter element view factors in the treatment of the radiative part of the domain which is the left side, from node 11 to 9, in this example. The variable *ra* selects either a handling of the radiative exchanges with a virtual node similar to the virtual convective one (*ra* = 0) or a handling with inter element view factors (*ra* = 1). The convective boundary is on the right part of the domain, between nodes 6 – 8 and 5 – 7.



```

sq146, Min it+1 temp: 281.8 K
sq147, Max it+1 temp: 288 K
sq148, Max it+1 temp: 300 K
sq154, Ejected heat : 0.202 MJ
L 260, Diss in solid: 138 WK
L 261, Total dissip.: 181 WK
L 262, Date, CPU, 05-May-2023, 4.4825 s
hf 25, Max heat flow: 110, mean: 22 W/m2

```

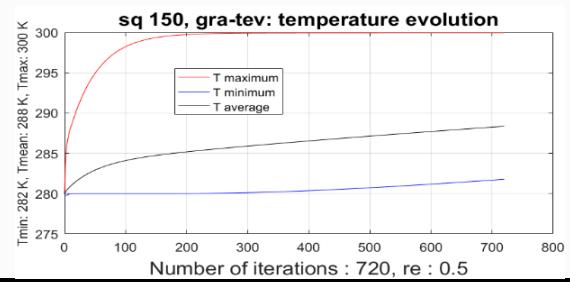


Figure 85: Thermal bridge, transient analysis, 1 convective node & 1 radiative side

## 6. Conclusion

In the transient problems, the non-linearity due to radiation does not seem to give any particular problem. In the cavity example, after the calculation of the thermal loading of the domain, we have successively examined four configurations: the cavity is the object of a convection phenomenon, the cavity has perfectly reflective walls, the walls of the cavity constitute a black body and finally the walls of the cavity constitute a gray body. We have completed with some gray body comparisons.

The tests carried out on the view factor matrix as well as on the solution of thermal problems including radiation are very encouraging. The visualizations of the results at different time steps by isotherms and by heat flow vectors for relatively coarse meshes are very effective and allow understanding the physics of heat exchanges by conduction, convection and radiation.

## 8. Additional information on functions and algorithms

### 8.1 Main procedure: *Fiammetta.m*

From *Table 28* to *Table 4*, we first, show the Matlab<sup>©</sup> procedure *Fiammetta.m* and, secondly, comment a scheme of the procedure, the meaning of each variable and a description of the output format.

Procedure <i>Fiammetta.m</i>	
1	CPU = tic; deb = 0;Ai = 0;fa=1000;F=0;lon=0.;
2	Gi = 22;[xyz_cao,car_cao,nbo,Me,Di,Ne,Co,nvn,nnr,fmd,ca] = cad_gin(Gi);
3	rc = 1;ra = 1;cs = 3; % rc = 4: rad. exch. computed, ra = 1: VF used
4	disp(['L 4, Meth., Ne, ca: ',num2str([Me Ne ca]))]
5	disp(['L 5, Co nnr nvn : ',num2str([Co nnr nvn]))]
6	disp(['L 6, rc, ra, cs : ',num2str([rc ra cs]))]
7	disp(['L 7, CAD interf. : ',num2str(nbo)])
8	th = .1 ; disp(['L 8, Thickness : ',num2str(th), ' m'])
9	k = 20; disp(['L 9, Conduction k : ',num2str(k), ' W/(mK)'])
10	pai = 1 ; disp(['L 10, DT isotherms : ',num2str(pai), ' K'])
11	re = .5;if rc>0;disp(['L 11, Refl. coeff. : ',num2str(re)]);end
12	if re>.99999;re=.9999;end
13	h = 20;if Co>0;disp(['L 13, Convection h : ',num2str(h), ' W/(m2K)']);end
14	np = size(car_cao,1);npv = size(xyz_cao,1); Ms=0;
15	disp(['L 15, np, nvertices: ',num2str([np npv])])
16	nni = 7;if nni<1;nni=1;end % Number nodes inserted on patches borders
17	disp(['L 17, num. nod side: ',num2str(nni)])
18	nci = nni+1;% mc=nci*4;if cs==2; mc = mc-nci; end%cs = 2 street 3 patches
19	disp(['L 19, num elem side: ',num2str(nci)])
20	nr = 4*nci;
21	area = 0;if nvn == 0;lc=zeros(1,3);end
22	for i = 1:np % Sum of patch areas = domain area: lines 13-21
23	area = area + norm(...
24	cross([xyz_cao(car_cao(i,3),:)-xyz_cao(car_cao(i,1),:) 0],...
25	[xyz_cao(car_cao(i,4),:)-xyz_cao(car_cao(i,1),:) 0]);
26	area = area + norm(...
27	cross([xyz_cao(car_cao(i,2),:)-xyz_cao(car_cao(i,1),:) 0],...
28	[xyz_cao(car_cao(i,3),:)-xyz_cao(car_cao(i,1),:) 0]));
29	end
30	area = area/2;disp(['L 30, Domain area : ',num2str(area), ' m2'])
31	if deb==1 ; disp('L 31, Call function: ..... cad_mes .....');end
32	[xvt,lK,bor,pbo] = cad_mes(xyz_cao,car_cao,nni,nbo); % Mesh generation
33	nel = size(lK,1) ; disp(['L 33, Num. elements: ',num2str(nel)])

```

34      no = size(xyt,1);dK = no + nvn + nnr;
35      disp(['L 35, Num. of nodes: ',num2str(no)])
36      disp(['L 36, Num. of DOF : ',num2str(dK)])
37  for i = 1:nel % Enforce anticlockwise ordering of element nodes
38      n = cross([xyt(lK(i,3),:)-xyt(lK(i,1),:) 0],...
39                  [xyt(lK(i,4),:)-xyt(lK(i,1),:) 0]);
40      if n(3) < 0;iq = lK(i,2);lK(i,2) = lK(i,4);lK(i,4) = iq;end
41  end
42  bov = zeros(nbo,npv+2); bov(:,1:2)=bor(:,1:2); pe = 0.;% Outward normals
43  if deb==1;disp('L 43, Call function: ..... gra_mnl .....');end
44  gra_mnl(xyz_cao,car_cao,[0 0 0],15);axis equal; % Drawing normals: + line 56
45  axis off;title(['Labels & normals of the ',num2str(np),' patch(es)'])
46  for i = 1 : nbo % ..... Loop on the nin CAD interfaces
47      if bor(i,4)==0% interface i must be a single one to be on the boundary
48          ne = cross([xyt(bor(i,2),:)-xyt(bor(i,1),:) 0],[0 0 1]);
49          nn = ne / norm(ne);
50          mi = [(xyt(bor(i,6),:) + xyt(bor(i,5),:))/2 0]; % Mid edge
51          li = norm(xyt(bor(i,2),:) - xyt(bor(i,1),:)); % Edge length
52          pe = pe + li; % Update the perimeter of the CAD domain
53          po = (mi + nn*li/5 ); % Normal vector: mi - po
54          plot([mi(1) po(1)], [mi(2) po(2)],'b','LineWidth',2);hold on
55          for j = 1:npv % Loop on npv vertices not in patch bor(i,3)
56              yes = 0;
57              for c = 1:4 % Check if vertex j is in patch bor(i,3)
58                  if j == car_cao(bor(i,3),c);yes = 1;end
59              end
60              if yes == 0 % Find vertices visible from interface bor(i,1:2)
61                  pn = [xyz_cao(j,:)-mi]; vis = dot(po-mi,pn);
62                  if vis > 0; bov(i,j+2)=j; end
63              end
64          end
65      end
66  end % ....End loop on the nbo CAD interfaces to draw the outwards normals
67  disp(['L 66, Domain perim.: ',num2str(pe),' m']) % End outward normals
68  xyz = zeros(dK,3); xyz(1:no,1:2) = xyt; % Coordinates expressed in 3D
69  SB = 5.6704e-8; % Stefan-Boltzmann constant Wm-2K-4
70  if rc > 0 ; disp(['L 69, St. Boltzmann: ',num2str(SB),' W/(m2K4)']);end
71  if deb == 1;disp('L 71, Call function: ..... cad_Dir .....');end
72  [lfi, fT] = cad_Dir(Di,no,nvn,car_cao,bor,pbo,nni); % Dirichlet b.c.
73  if lfi(1) == 0;nf = 0;else;nf=size(lfi,2);end
74  if deb == 1; disp('L 74, Call function: ..... cad_Neu .....');end
75  [gh,lg,bos]= cad_Neu(Ne,dK,car_cao,pbo,bor,th,xyz_cao); % Neumann b.c.
76  if deb == 1;disp('L 81, Call function: ..... gra_mnl .....');end
77  if Co > 0
78      if deb == 1;disp('L 78 , Call function: ..... cad_con .....');end
79      [lc,vc,he] = cad_con(car_cao,bor,pbo,h,dK,nci,deb,nvn,cs,Co,xyz);
80      xyz(not(1:nvn+nnr),1:2) = vc(1:nvn+nnr,:);% Virtual nodes coordinates
81  end % End option Co > 0. Some sides are linked to virtual convective nodes
82  if nel < 100;gra_mnl(xyz,lK,lc,10);axis equal,axis off;hold on;end
83  lccont = zeros(1,4*nci);lv=zeros(1,4); % DOF of the cavity
84  if cs == 0 % Compute the radiative nodes, vertices & number
85      mcr = 0; % mcr is the number of radiative nodes
86  else % Analysis of the 3 situations: cavity, street, balcony
87      if cs ~=4 % If cs = 5, both vertical sides are convective
88          if deb==1; disp('L 88, Call function: ..... cad_ban .....');end
89          [lccont,lv] = cad_ban(car_cao,bor,pbo,nni,cs);
90          % mcr = max(size(lcont));
91          mcr = size(lv,2)*(nni+1)-nni;%mcr = nni*4+4;% 
92          if size(lv,2) > 1;disp(['L 92, Rad. vert. lv: ',num2str(lv)]);end
93      end
94  % ..... Assembling the nel element conductivity matrices
95  Kk = zeros(dK,dK); % Conductivity matrix initialization
96  disp(['L 97, Anis. index : ',num2str(Ai)])
97  if Ai == 0;co = ones(1,nel)*k;else;co = mat_cok(Ai,nci,fa,xyz,lK,deb);end
98  if deb == 1; disp('L 99, Call function: ..... fem_Kco .....');end
99  for n = 1:nel % Compute global K matrix, loop on the elements
100     Kel = fem_Kco(xyz,lK(n,:))*co(n)*th; % Element conductivity matrix
101     for i = 1:4;il = lK(n,i);
102         for j = 1:4;jl = lK(n,j);Kk(il,jl) = Kk(il,jl) + Kel(i,j);end
103     end
104  end % End assembling the nel conductivity matrices
105  K = Kk;
106  if Co > 0 % Assembling the nco = size(lc,1) element convective matrices
107      if deb == 1;disp('L 109, Call function: ..... fem_Kcv .....');end
108      for n = 1:size(lc,1) % loop on elements
109          Kec = fem_Kcv(xyz,lc,he(n)*th); % Element convective matrices
110          for i = 1:3
111              for j=1:3;K(lc(n,i),lc(n,j))=K(lc(n,i),lc(n,j))+Kec(i,j);end
112          end
113      end
114  end % End assembling the nco convection matrices
115  if rc == 0 % Radiative exchanges are not present
116      Lel(1) = 0; ns = 0; Ms=zeros(1,1); % Init. edge elements lenghts

```

```

117
118     else % View factor and radiosity matrices for radiative transfers
119         if cs == 2;lcc = lv; ns = 3;% lcc & ns: street section
120         if cs == 3;lcc = lv; ns = 5;% lcc & ns: Thermal bridge
121         if cs == 4;lcc = lv; ns = 2;% lcc & ns: rectangle
122         if cs == 1;lcc = [lv' ;lv(1)];ns = 4;% lcc & ns: quadril. cavity
123         % if cs == 5;lcc = [lv' ;lv(1)];ns = 4;% lcc & ns: quadril. cavity
124         if cs == 6;lcc = [lv' ;lv(1)];ns = 4;% lcc & ns: quadril. cavity
125         if cs == 7;lcc = [lv' ;lv(1)];ns = 4;% lcc & ns: quadril. cavity
126         Lel = zeros(ns,1); % Compute the lengths of ns patch radiative edges
127         Tsky = 280; disp(['L 127, Sky temper. : ',num2str(Tsky),' K'])
128         for i = 1:ns
129             Lel(i)= sqrt((xyz_cao(lcc(i+1),1)-xyz_cao(lcc(i),1))^2 + ...
130                         (xyz_cao(lcc(i+1),2)-xyz_cao(lcc(i),2))^2);
131         end;
132         disp(['L 130, Rad. edg. len: ',num2str(Lel(1:ns)),' m'])
133         disp(['L 131, Radiativ area: ',num2str(sum(Lel)*th),' m^2'])
134         if cs == 1 % Quadrilateral cavity view factors
135             if deb==1;disp('L 134, Call function: ..... geo_vfc .....');end
136             Fs = geo_vfc(nci,Lel,ns);I=eye(nr);Ms=zeros(nr,1);
137             M = (I-re*Fs);disp(['L 135, Det. R. mat M: ',num2str(det(M))])
138         end
139         if cs == 6 % Quadrilateral cavity view factors
140             if deb==1;disp('L 139, Call function: ..... geo_vfc .....');end
141             Fs = geo_vfc(nci,Lel,ns);I=eye(nr);Ms=zeros(nr,1);
142             M = (I-re*Fs);disp(['L 140, Det. R. mat M: ',num2str(det(M))])
143         end
144         if cs == 7 % Quadrilateral cavity view factors
145             if deb==1;disp('L 144, Call function: ..... geo_vfr .....');end
146             Fs = geo_vfr(nci,Lel );I=eye(nr);Ms=zeros(nr,1);
147             M = (I-re*Fs);disp(['L 145, Det. R. mat M: ',num2str(det(M))])
148         end
149         if cs == 2 % Street section view factors 148 -- 168
150             if deb ==1;disp('L 151, Call function: .... . geo_stf .....');end
151             Fs = geo_stf(nci,Lel(2:3));nr = size(Fs,1);I = eye(nr);
152             M = (I-re*Fs); disp(['L 152, Det. R. mat M: ',num2str(det(M))])
153             Fsky = 1-sum(Fs,2); % Fsky is the line vector of sky view factors
154             disp(['L 154, SVF sum(Fsky): ',num2str(sum(Fsky),3)])
155             if nni ==1;disp(['L 155, Sky view fact: ',num2str(Fsky,3)]);end
156             if re == 1 % Sky & Ground contributions to radiative exchanges
157                 Ms = zeros(nr,1);
158             else
159                 lon = zeros(1,nci*3);
160                 for i = 1:nci % Element lengths on the 3 sides
161                     lon(i) = Lel(1)/(nci); lon(i+nci) = Lel(2)/(nci);
162                     lon(i+2*nci) = Lel(3)/(nci);
163                     lon(mcr-i) = Lel(3)/(nci);
164                 end
165                 Ms =(re*(I-Fs)*M^(-1)-I)*Fsky.*lon'*th*SB*Tsky^4; % Fiam. 119
166                 figure;bar(Ms');grid on
167                 title( ['L164, Sky loads, equation (119), sum(Ms) = ',...
168                         num2str(sum(Ms),2),' W ',num2str(Tsky),' K']);
169                 disp(['L 166, Sum sky loads: ',num2str(sum(Ms),4),' W'])
170             end
171         end
172         if cs == 3 % Thermal bridge view factors
173             if deb==1;disp('L 174, Call function: ..... geo_baf .....');end
174             nco = size(Lel,1);kn = 0;lon = zeros(1,nco*nci);%Ftot = 0 ;
175             for i = 1:nco;for j = 1:nci;kn=kn+1;lon(kn) = Lel(i)/nci;end;end;
176             if ra == 1;F = geo_baf(nci,xyz_cao);
177                 nv = size(F,2);disp(['L 166, dim F = nv : ',num2str(nv)])
178                 Fs = F(1:nv-2,1:nv-2);I = eye(size(Fs,1));M = (I-re*Fs);
179                 disp(['L 177, determinant M: ',num2str(det(M))])
180                 Fgr = F(1:nv-2,nv-1); Fsky=F(1:nv-2,nv);
181                 kn=0;%lon = zeros(1,nne)
182                 Ms =(re*(I-Fs)*M^(-1)-I)*SB*th*(Fsky+Fgr).*Tsky^4.*lon';
183                 disp(['L 181, Sum 2d member: ',num2str(sum(Ms),3),' W'])
184                 if nni == 1
185                     disp(['L 183, Ms Eq. 104 : ',num2str(Ms')])
186                     disp(['L 184, unicol Fgr : ',num2str(Fgr',3)]);
187                     disp(['L 185, unicol Fsky : ',num2str(Fsky',3)]);
188                     disp(['L 186, sum(F,2) : ',num2str(sum(F,2)',3)])
189                 end
190             end
191         end
192         % =====
193         if Me == 1 % .... Solution of linear steady state heat transfer problems
194             nf = size(lfi,2); N = zeros(nf,dK);% L. con. fix.%gh=zeros(nf,1);
195             for i = 1:nf;N(i,lfi(i)) = 1;gh(dK+i) = fT(i); end;
196             A = [K N';N zeros(nf,nf)];B = A\gh;tca = B(1:dK);
197             figure;gt =[min(tca) pai max(tca)];
198             if deb ==1;disp(['L 201, DT isoth gt : ',num2str(gt),' K']);end
199             nc3 = (nci)^2; % nc3 = numb elem / patch, same for all the patches

```

```

200 if deb ==1;disp( 'L 203, Call function: ..... gra_ipa .....');end
201 for i = 1 : np
202     gra_ipa(nci,nci,lK((i-1)*nc3+1:nel,:),tca,xyz,gt);hold on
203 end
204 axis equal;colorbar;axis off
205 title (['Dissipation: ',num2str(.5*tca'*K*tca,3),' WK, DOF: ',...
206         num2str(dK),'] )
207 for i = 1:nbo
208     % Drawing the border of the domain
209     if bor(i,4)==0
210         plot([xyz(bor(i,1),1) xyz(bor(i,2),1)], ...
211             [xyz(bor(i,1),2) xyz(bor(i,2),2)],'k','LineWidth',2)
212     end
213 end
214 % End drawing of the isotherms
215 disp(['L 216, Total dissip.: ',num2str(tca'*K*tca/2,3),' WK'])
216 % disp(['L 220, -dot(B,gh)/2 : ',num2str(-dot(B,gh)/2,3),' WK'])
217 dissol = .5*tca(1:no)'*Kk(1:no,1:no)*tca(1:no);
218 disp(['L 219, Dis. in solid: ',num2str(dissol,3),' WK'])
219 disp(['L 220, DT in solid: ',num2str(max(tca(1:no))-...
220         min(tca(1:no)),3), ' K'])
221 if deb==1;disp(['L 222, tca mi.ma.av.:',...
222         num2str([min(tca) max(tca) mean(tca)],3), ' K']);end
223 if nvn > 2;disp(['L 226, DT. convect :',...
224         num2str(max(tca(llic))-min(tca(llic)),3), ' K'])
225 end
226 % End standard linear heat transfer process
227 reac = K*tca;
228 if nf < 6
229     disp(['L 225, Fixed DOF : ',num2str(lfi)])
230     disp(['L 226, Imposed temp.: ',num2str( tca(lfi)',3), ' K'])
231     disp(['L 227, React. flows : ',num2str(reac(lfi)',3), ' W'])
232 end
233 disp(['L 229, Date, CPU, ',num2str(date),', ',num2str(toc(CPU)), ' s',])
234 if deb ==1;disp('L 192-232, End meth 1, stationary linear');end
235 end
236 % =====
237 if Me == 2 % Nonlinear stat. rad. heat transfer - conv. like virtual node
238     if nmr > 0 % nmr is the number of radiative virtual nodes
239         if deb ==1;disp('L 236, Call function: ..... fem_snl .....');end
240         tca=fem_smd(K,gh,lfi,fT,xyz,lK,lcont,nmr,nci,np,SB*th*(1-re),...
241             ca,nbo,bor);
242         disp(['L 239, T. vir. nod. : ',num2str(tca(dK),3), ' K'])
243     end
244     disp(['L 241, Date, CPU, ',num2str(date),', ',num2str(toc(CPU),2), ' s',])
245     if deb ==1;disp('L 233-243, End non lin virt rad node.');

```

```

283 if em < .001;em=0;end;xlabel (['Emissivity: ', num2str(em,2)])
284 title ( ['L 280, Final radiative loads Kk*tca, min max sum: ',...
285     num2str([min(gsm(lcont)) max(gsm(lcont)) sum(gsm(lcont))],3),...
286     ' W']);hold on
287 disp(['L 286, Ca fl. K*tca : ',num2str(sum(gsm(lcont)),3),' W'])
288 figure; % Drawing the temperature along the radiative border
289 if ca > 0
290     plot([tca(lcont); tca(lcont(1))], 'k*-')
291 else
292     plot(tca(lcont), 'k*-');
293 end
294 grid on;title('L 293, Temperature along radiative border');
295 xlabel([' Emissivity = ',num2str(em)])
296 end
297 disp(['L 296, Date, CPU, ',num2str(date),', ',num2str(toc(CPU)), ' s',])
298 if deb ==1;disp('L 266-298, End cavity, view factors, rad exch.');?>
299 end

```

Table 28: Matlab<sup>®</sup> procedure *Fiammetta.m*

Simplified scheme of the method used in the procedure <i>Fiammetta.m</i>			
<i>Line</i>	<i>1 → 20:</i>	Input data	<i>cad_gin.m</i> (Table 31)
<i>Line</i>	<i>21→ 30:</i>	Computation of the domain area	
<i>Line</i>	<i>32</i>	Mesh generation	<i>cad_mes.m</i> (Table 35)
<i>Line</i>	<i>33→ 66:</i>	Computing the outwards normal	<i>gra_mnl.m</i> (Table 53)
<i>Line</i>	<i>67→ 73:</i>	Dirichlet boundary conditions	<i>cad_Dir.m</i> (Table 32)
<i>Line</i>	<i>74 → 76:</i>	Neumann boundary conditions	<i>cad_Neu.m</i> (Table 33)
<i>Line</i>	<i>77→ 82:</i>	Topology of the convection elem.	<i>cad_con.m</i> (Table 34)
<i>Line</i>	<i>83→ 93:</i>	Identification of the radiating nodes	<i>cad_ban.m</i> (Table 37)
<i>Line</i>	<i>94→ 114:</i>	Assemble cond. & conv. el. matrices	<i>fem_Kco.m</i> (Table 44) <i>fem_Kcv.m</i> (Table 45)
<i>Line</i>	<i>115→ 191:</i>	Radiative heat exchanges cavity street balcony	<i>geo_vfs.m</i> (Table 67) <i>geo_stf.m</i> (Table 66) <i>geo_baf.m</i> (Table 65)
	<i>cs = 1</i>	<i>134 → 138</i>	
	<i>cs = 6</i>	<i>139 → 143</i>	
	<i>cs = 2</i>	<i>144 → 160</i>	
	<i>cs = 3</i>	<i>161 → 181</i>	
<i>Me</i>	= 1: linear permanent heat transfer	<i>195 → 230</i>	
<i>Me</i>	= 2: Nonlinear permanent heat transfer	<i>231 → 240</i>	<i>fem_smd.m</i> (Table 38)
<i>Me</i>	= 3: Transient linear heat transfer	<i>241 → 247</i>	<i>fem_smt.m</i> (Table 39)
<i>Me</i>	= 4: Nonlinear transient heat transfer	<i>248 → 260</i>	<i>fem_smq.m</i> (Table 40)
<i>Me</i>	= 5: Cavity, VF matrix radiat exchanges	<i>261 → 282</i>	<i>fem_smc.m</i> (Table 41)

Table 29: Scheme of the procedure *Fiammetta.m* (Table 28)

Line	Occ.	Name	Meaning of the variables used in <i>Fiammetta.m</i>
1	6	CPU	Initialization of time variable
1	31	deb	Flag to control the display of functions calls
1	4	Ai	Flag to control anisotropy of material
1	2	fa	Ratio between main material and strip material
1	8	F	Thermal bridge view factor matrix including mesh, sky, ground, computed in <i>geo_baf.m</i> (L 174),
1	10	lon	Initialization of the vector of elements lengths on patch sides for radiative exchanges
2	2	Gi	input data of <i>cad_gin.m</i> used for the identification of an application
2	20	xyz_cao	Coordinates of the patch vertices, output of <i>cad_gin.m</i> .
2	17	car_cao	Localization of patches

2	12	nbo	Number of CAD patches interfaces
2	7	Me	Method used for the solution <i>Me</i> = 1...5, see above
2	3	Di	Flag defining the type of Dirichlet boundary conditions
2	3	Ne	Flag indicating presence of von Neumann boundary conditions
2	6	Co	Flag for convection, variable used in <i>cad_con.m</i> ( <i>L 80</i> )
2	8	nvn	Number of virtual convection nodes, used in <i>cad_Dir.m</i> ( <i>L 71</i> ), <i>cad_con.m</i> ( <i>L 80</i> ), <i>fem_smd.m</i> ( <i>L 236</i> )
2	7	nvr	Number of virtual radiative nodes, used in <i>fem_smd.m</i> ( <i>L 226</i> )
2	2	fmd	Flag indicating the partition of the domain int 2 equal parts
2	5	ca	Flag indicating the presence of a cavity
3	7	rc	Flag indicating presence of radiative exchange (1 = yes, 0 = no), used in <i>fem_sm.m</i> ( <i>L 277</i> ), <i>fem_smq.m</i> ( <i>L 257</i> ),
3	4	ra	Flag indicating the use of conductive radiative elements, used in <i>fem_smq.m</i> ( <i>L 257</i> ),
3	19	cs	View factor matrix flag: 1 = cavity, 2 = street, 3 = thermal bridge, 5 = both vertical sides are convective, used in <i>cad_con.m</i> ( <i>L 80</i> ), <i>cad_ban.m</i> ( <i>L 88</i> ), <i>fem_smt.m</i> ( <i>L 246</i> ), <i>fem_smq.m</i> ( <i>L 257</i> ), <i>fem_sm.m</i> ( <i>L 277</i> ),
8	13	th	Thickness, <i>m</i> , used in <i>cad_Neu.m</i> , <i>fem_smd.m</i> ( <i>L 236</i> ), <i>fem_smt.m</i> ( <i>L 246</i> ), <i>fem_smq.m</i> ( <i>L 257</i> ), <i>fem_sm.m</i> ( <i>L 277</i> ).
9	3	k	Conductivity coefficient $Wm^{-1}K^{-1}$ .
10	6	pai	Temperature interval in isotherms drawing <i>K</i>
11	18	re	Coefficient of reflexion (adimensional), used in <i>fem_smd.m</i> ( <i>L 236</i> ), <i>fem_smq.m</i> ( <i>L 257</i> ), <i>fem_sm.m</i> ( <i>L 277</i> ).
13	3	h	Convection coefficient $Wm^{-2}K^{-1}$ , used in <i>cad_con.m</i> ( <i>L 80</i> )
14	9	np	Number of patches = size (car_cao,1);
14	4	npv	Number of patch vertices = size (xyz_cao,1);
16	10	nni	Number of nodes per side of CAD patch
18	29	nci	Number of elements per side of CAD patch
20	14	nr	Number of elements on a 4 sides border
21	11	area	Area of the solid domain $m^2$ , computed in <i>lines 22 - 30</i>
32	13	xyt	Nodal coordinates, computed in <i>cad_mes.m</i> , the mesh generation function
32	20	lK	Localization matrix of conductive elements (dimension: <i>nel</i> x 4), computed in <i>cad_mes.m</i>
32	23	bor	computed in <i>cad_edg.m</i>
32	5	pbo	computed in <i>cad_mes.m</i>
33	7	nel	Number of conductive elements = size ( <i>lK</i> , 1)
34	11	no	Number of nodes of the mesh = size ( <i>xyt</i> , 1)
34	15	dK	Number of <i>DOF</i> , <i>dK</i> = <i>no</i> + <i>nvn</i> + <i>nvr</i>
51	4	pe	Perimeter of the conductive domain, <i>m</i> . Initialized at <i>L 41</i> , and printed at <i>L 66</i> .

67	16	xyz	Matrix of the nodal coordinates expressed in 3D at the level $z = 0$ and expressed in $m$ .
68	7	SB	Stefan-Boltzmann constant: $5.6704 \cdot 10^{-8} \text{ W m}^{-2} \text{ K}^{-4}$ used in <a href="#">fem_smd.m</a> ( <i>L 236</i> ), <a href="#">fem_smt.m</a> ( <i>L 246</i> ), <a href="#">fem_sm.m</a> ( <i>L 277</i> ),
71	13	lfi	Uni-column matrix: list of fixed nodes (Dirichlet), computed in <a href="#">cad_Dir.m</a> ( <i>L 71</i> ), used in <a href="#">fem_smd.m</a> ( <i>L 236</i> ), <a href="#">fem_smt.m</a> ( <i>L 246</i> ), <a href="#">fem_smq.m</a> ( <i>L 257</i> ), <a href="#">fem_sm.m</a> ( <i>L 277</i> ).
71	7	fT	Uni-column matrix of fixed nodal temperatures, computed in <a href="#">cad_Dir.m</a> ( <i>L 71</i> ), used in <a href="#">fem_smd.m</a> ( <i>L 236</i> ), <a href="#">fem_smt.m</a> ( <i>L 246</i> ), <a href="#">fem_smq.m</a> ( <i>L 257</i> ), <a href="#">fem_sm.m</a> ( <i>L 277</i> ).
72	8	nf	Number of fixations = number of columns of lfi, it is also the dimension of <i>fT</i> & <i>lfi</i> , computed in <a href="#">cad_Dir.m</a> ( <i>L 71</i> ). It is used in the solution of linear steady state heat transfer problems ( $Me = 1$ )
74	7	gh	Weight functions of patch side loads computed in <a href="#">cad_Neu.m</a>
74	2	lg	Lengths of the patch sides, used in <a href="#">fem_sm.m</a> ( <i>L 277</i> ).
74	2	bos	Areas of the patch sides, used in <a href="#">fem_sm.m</a> ( <i>L 277</i> ).
80	2	he	Uni-line matrix of the element convective coefficients, computed in <a href="#">cad_con.m</a> ( <i>L 80</i> )
80	7	lc	Localizations of convective elements defined in <a href="#">cad_con.m</a>
88	14	lcont	Uni-column matrix of <i>mrr</i> radiative cavity, street and balcony nodes, computed in <a href="#">cad_ban.m</a> ( <i>L 88</i> ). It is initialized in <i>L 82</i> .
88	13	lv	Uni-column matrix of cavity, street, balcony vertices, computed in <a href="#">cad_ban.m</a> ( <i>L 88</i> ). It is initialized in <i>L 82</i> .
89	6	mcr	Number of radiative nodes, size (lcont,1)
95	6	Kk	Global conductivity matrix of meshed solid domain $WK^{-1}$
97	3	co	Uni-line matrix of the <i>nel</i> products of thickness by conductivity. coeff. <i>k</i> . It is also used in the procedure <a href="#">P_flg.m</a> and the function <a href="#">gra_ahf.m</a> to compute the heat flow vectors. It is computed in <a href="#">mat_cok.m</a>
100	2	Kel	Element conductive matrix $WK^{-1}$ , computed in <a href="#">fem_Kco.m</a>
109	2	Kec	Element “conductive – convective” matrix $WK^{-1}$ , computed in <a href="#">fem_Kcv.m</a> .
111	13	K	Global conductivity matrix $WK^{-1}$ (solid part + ...)
116	12	ns	Number of edges of the radiative part of the boundary
116	16	Lel	Uni-column matrix of the <i>ns</i> radiative edges’ lengths <i>m</i> , used in <a href="#">geo_vfc.m</a> ( <i>L 136</i> ), <a href="#">geo_stf.m</a> ( <i>L 146</i> ), <a href="#">fem_smq.m</a> ( <i>L 257</i> ), <a href="#">fem_sm.m</a> ( <i>L 277</i> ).
118	10	lcc	Sequence of radiative CAD vertices + first repeated if cavity
126	6	Tsky	Sky temperature, used in <a href="#">fem_smq.m</a> ( <i>L 257</i> ),
135	16	Fs	View factor matrix for radiative element from mesh border, computed in <a href="#">geo_vfc.m</a> ( <i>L 136</i> ), <a href="#">geo_vfr.m</a> ( <i>L 145</i> ), <a href="#">geo_stf.m</a> ( <i>L 150</i> ),
136	13	M	Radiosity matrix, $M = (I - re^*Fs)$
152	7	Fsky	Sky view factor

178	3	Fgr	Uni-column matrix of ground view factors
14	13	Ms	$Ms = -SB * Fsky * Tsky^4$
196	30	tca	Uni-column of the <i>dK DOF</i> or nodal temperatures
196	3	gt	Uni-line matrix for isolines definition in <i>gra_ipa.m</i>
223	4	reac	Second member, $reac = K * tca$
269	4	Mpr	$(I - Fs) M^T$ matrix - flow out of a segment (adim.)

Table 30: Matlab<sup>®</sup> variables used in the procedure *Fiammetta.m*

## 8.2 Input data functions

In this section, we show representative input data used in the examples. They are organized in sections including an identification number *Gi* and the name of the concerned problem. Input are defined in Matlab functions *cad\_gin.m*, *cad\_Dir.m*, *cad\_Neu.m*, *cad\_con.m*, *cad\_mes.m* and *cad\_edg.m* (*Table 31* to *Table 36*).

The first function called in *Fiammetta.m* is *cad\_gin.m*. It provides the *CAD* definition of the geometry selected with the parameter *Gi* (see for instance: *Figure 22*).

CAD data inserted at the beginning of <i>Fiammetta.m</i> with the function <i>cad_gin.m</i>	
1	<pre>function[xyz_cao,car_cao,nbo,Me,Di,Ne,Co,nvn,nnr,fmd,ca] = cad_gin(Gi) fmd = 0;ca=0; 3 4 if Gi == 8 % ..... 8. Standard rect. 1 rectangle ..... 5 ha = 2;xyz_cao = [0 0;1 0;1 ha;0 ha]; 6 car_cao = [1 2 3 4 ];nbo=4; 7 Me = 1;Di =13;Ne = 0;Co = 1;nvn = 2;nnr = 0; 8 disp(['Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)]) 10 end 11 if Gi == 9 % ..... 9. Standard rect. 1 rectangle ..... 12 ha = 2;xyz_cao = [0 0;1 0;1 ha;0 ha]; 13 car_cao = [1 2 3 4 ];nbo=4; 14 Me = 1;Di = 3;Ne = 0;Co = 0;nvn = 0;nnr = 0; 15 disp(['Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)]) 16 end 17 if Gi ==25 % ..... 8. Standard rect. 1 rectangle ..... 18 ha = 2;xyz_cao = [0 0;1 0;1 ha;0 ha]; 19 car_cao = [1 2 3 4 ];nbo=4; 20 Me = 1;Di =13;Ne = 0;Co = 1;nvn = 2;nnr = 0; 21 disp(['Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)]) 22 end 23 if Gi ==28 % ..... 8. Standard rect. 1 rectangle ..... 24 ha = 2;xyz_cao = [0 0;1 0;1 ha;0 ha]; 25 car_cao = [1 2 3 4 ];nbo=4; 26 Me = 1;Di =13;Ne = 0;Co = 1;nvn = 2;nnr = 0; 27 disp(['Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)]) 28 end 29 if Gi == 30 % ..... 9. Standard rect. 1 rectangle ..... 30 ha = 2;xyz_cao = [0 0;1 0;1 ha;0 ha]; 31 car_cao = [1 2 3 4 ];nbo=4;fmd=1; 32 Me = 3;Di = 3;Ne = 0;Co = 0;nvn = 0;nnr = 0; 33 disp(['Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)]) 34 end 35 if Gi == 7 % ..... 7. Standard rect. 2 squares ..... 36 xyz_cao = [0 0;1 0;0 1;1 1;0 2;1 2]; 37 car_cao = [1 2 4 3;3 4 6 5];nbo =7; 38 Me = 3;Di = 7;Ne = 0;Co = 3;nvn = 2;nnr = 0; 39 disp(['Rectan. 2 squares Gi: ',num2str(Gi),', Di : ',num2str(Di)]) 40 end 41 if Gi == 10 % ..... 7. Standard rect. 2 squares ..... 42 xyz_cao = [0 0;1 0;0 1;1 1;0 2;1 2]; 43</pre>

```

49      car_cao = [1 2 4 3;3 4 6 5];nbo =7;
50      Me = 4;Di = 7;Ne = 0;Co = 4;nvn = 1;nnr = 1;
51      disp(['g 72, 1 r., 2s. Gi : ',num2str(Gi),', Di : ',num2str(Di)])
52  end
53 if Gi == 24
54 % ..... 7. Standard rect. 2 squares .....
55     xyz_cao = [0 0;1 0;0 1;1 1;0 2;1 2];
56     car_cao = [1 2 4 3;3 4 6 5];nbo =7;
57     Me = 4;Di =16;Ne = 0;Co = 4;nvn = 1;nnr = 1;
58     disp(['Rectan. 2 squares Gi: ',num2str(Gi),', Di : ',num2str(Di)])
59 end
60 if Gi == 26
61 % ..... 7. Standard rect. 2 squares .....
62     xyz_cao = [0 0;1 0;0 1;1 1;0 2;1 2];
63     car_cao = [1 2 4 3;3 4 6 5];nbo =7;
64     Me = 1;Di =10;Ne = 0;Co = 7;nvn = 3;nnr = 0;
65     disp(['Rectan. 2 squares Gi: ',num2str(Gi),', Di : ',num2str(Di)])
66 end
67 if Gi == 27
68 % ..... 7. Standard rect. 2 squares .....
69     xyz_cao = [0 0;1 0;0 1;1 1;0 2;1 2];
70     car_cao = [1 2 4 3;3 4 6 5];nbo =7;
71     Me = 1;Di =19;Ne = 0;Co = 7;nvn = 3;nnr = 0;
72     disp(['Rectan. 2 squares Gi: ',num2str(Gi),', Di : ',num2str(Di)])
73 end
74 if Gi == 29
75 % ..... 7. Standard rect. 2 squares .....
76     xyz_cao = [0 0;1 0;0 1;1 1;0 2;1 2];
77     car_cao = [1 2 4 3;3 4 6 5];nbo =7;
78     Me = 1;Di = 9;Ne =11;Co = 0;nvn = 0;nnr = 0;
79     disp(['1 rect., 2 squar. Gi : ',num2str(Gi),', Di : ',num2str(Di)])
80 end
81 if Gi == 31
82 % ..... 7. Standard rect. 2 squares .....
83     xyz_cao = [0 0;1 0;0 1;1 1;0 2;1 2];
84     car_cao = [1 2 4 3;3 4 6 5];nbo =7;
85     Me = 3;Di =15;Ne = 0;Co = 7;nvn = 4;nnr = 0;
86     disp(['Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
87 end
88 if Gi == 32
89 % ..... 7. Standard rect. 2 squares .....
90     xyz_cao = [0 0;1 0;0 1;1 1;0 2;1 2];
91     car_cao = [1 2 4 3;3 4 6 5];nbo =7;
92     Me = 3;Di =14;Ne = 0;Co = 7;nvn = 4;nnr = 0;
93     disp(['Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
94 end
95 if Gi == 33
96 % ..... 7. Standard rect. 2 squares .....
97     xyz_cao = [0 0;1 0;0 1;1 1;0 2;1 2];
98     car_cao = [1 2 4 3;3 4 6 5];nbo =7;
99     Me = 3;Di =17;Ne = 0;Co = 7;nvn = 3;nnr = 0;
100    disp(['Rectan. 2 squares Gi: ',num2str(Gi),', Di : ',num2str(Di)])
101 end
102 if Gi == 23
103 % ..... 23. Vertical rectangle .....
104     ha = 1;xyz_cao = [0 0;1 0;1 ha;0 ha];
105         car_cao = [1 2 3 4 ];nbo=4;
106     Me = 3;Di = 0;Ne = 0;Co = 0;nvn = 0;nnr = 0;fmd=1;
107     disp(['Vertical. 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
108 end
109 if Gi == 17
110 % ..... 17. Horizontal rectangle .....
111     ha = 1;xyz_cao = [0 0;2 0;2 ha;0 ha];
112         car_cao = [1 2 3 4 ];nbo=4;
113     Me = 3;Di = 0;Ne = 0;Co = 0;nvn = 0;nnr = 0;fmd=1;
114     disp(['1 horizon. rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
115 end
116 if Gi == 34
117 %..... 9. Standard trapezoidal .....
118     ha = 1;xyz_cao = [0 0;1 0;.75 0.8;.25 ha];
119         car_cao = [1 2 3 4 ];nbo = 4;
120     Me = 1;Di = 7;Ne = 0;Co = 1;nvn = 2;nnr = 0;
121     disp(['Trapezoidal shape Gi: ',num2str(Gi),', Di : ',num2str(Di)])
122 end
123 if Gi == 6
124 %..... 9. Standard trapezoidal .....
125     ha = 1;xyz_cao = [0 0;1 0;.75 ha;.25 ha];
126         car_cao = [1 2 3 4 ];nbo = 4;
127     Me = 1;Di = 7;Ne = 0;Co = 1;nvn = 2;nnr = 0;
128     disp(['Trapezoidal shape Gi: ',num2str(Gi),', Di : ',num2str(Di)])
129 end
130 if Gi == 3
131 % ..... C shape three patches .....

```

```

132 xyz_cao = [2 2;2 3;1 2;0 3;0 0;1 1;3 0;3 1];% CAD
133 car_cao = [1 2 4 3;3 4 5 6;7 8 6 5]; nbo = 10;
134 Me = 1;Di = 6;Ne = 0;Co = 0;nvn = 0;nnr = 0;
135 disp(['C shape 3 patches Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
136 end
137 if Gi == 5
138 % ..... 5. C shape diag top left .....
139 xyz_cao = [3 2;3 3;0 3;1 2;0 1;1 1;2 1;0 0;1 0;2 0];
140 car_cao = [1 2 3 4;4 3 5 6;6 5 8 9; 6 9 10 7];nbo=13;
141 Me = 1;Di = 5;Ne = 0;Co = 0;nvn = 0;nnr = 0;
142 disp(['C shape 4 patches Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
143 end
144 if Gi == 1
145 % ..... 1. Standard cavity .....
146 xyz_cao = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
147 car_cao = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
148 Me = 3;Di = 1;Ne = 0;Co = 0;nvn = 0;nnr = 0;ca = 1;
149 disp(['Standard cavity, Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
150 end
151 if Gi == 2
152 % ..... 1. Standard cavity .....
153 xyz_cao = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
154 car_cao = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
155 Me = 3;Di = 1;Ne = 0;Co = 2;nvn = 1;nnr = 0;ca = 1;
156 disp(['Standard cavity, Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
157 end
158 if Gi == 4
159 % ..... 1. Standard cavity .....
160 xyz_cao = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
161 car_cao = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
162 Me = 1;Di = 4;Ne = 0;Co = 0;nvn = 0;nnr = 0;ca = 1;
163 disp(['Standard cavity, Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
164 end
165 if Gi == 11
166 % ..... 1. Standard cavity .....
167 xyz_cao = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
168 car_cao = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
169 Me = 3;Di = 1;Ne = 0;Co = 0;nvn = 0;nnr = 0;ca = 1;
170 disp(['Standard cavity, Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
171 end
172 if Gi == 39
173 % ..... 1. Standard cavity .....
174 xyz_cao = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
175 car_cao = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
176 Me = 2;Di = 4;Ne = 0;Co = 0;nvn = 0;nnr = 1;ca = 1;
177 disp(['Standard cavity, Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
178 end
179 if Gi == 40
180 % ..... 1. Standard cavity .....
181 xyz_cao=[0 0;1 0;2 0;3 0;0 1;1 1;2 1;3 1;0 2;1 2;2 2;3 2;0 3;1 3;2 3;3 3];
182 car_cao=[1 2 6 5;2 3 7 6;3 4 8 7;5 6 10 9;7 8 12 11;9 10 14 13;...
183 10 11 15 14;11 12 16 15];nbo=24;
184 Me = 2;Di =20;Ne = 0;Co = 0;nvn = 0;nnr = 1;ca = 1;
185 disp(['Standard cavity, Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
186 end
187 if Gi == 41
188 % ..... 1. Standard cavity .....
189 xyz_cao = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
190 car_cao = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
191 Me = 5;Di = 0;Ne = 1;Co = 0;nvn = 0;nnr = 0;ca = 1;
192 disp(['Standard cavity, Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
193 end
194 if Gi == 42
195 % ..... 1. Standard cavity .....
196 xyz_cao=[0 0;1 0;2 0;3 0;0 1;1 1;2 1;3 1;0 2;1 2;2 2;3 2;0 3;1 3;2 3;3 3];
197 car_cao=[1 2 6 5;2 3 7 6;3 4 8 7;5 6 10 9;7 8 12 11;9 10 14 13;...
198 10 11 15 14;11 12 16 15];nbo=24;
199 Me = 5;Di = 0;Ne = 2;Co = 0;nvn = 0;nnr = 0;ca = 1;
200 disp(['Standard cavity, Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
201 end
202 if Gi == 12
203 % ..... 12. Rectangular cavity ..radiative exchanges .....
204 xyz_cao = [0 0;3 0;3 6;0 6;1 1;2 1;2 5;1 5];
205 car_cao = [1 2 6 5;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
206 Me = 5;Di = 8;Ne = 0;Co = 7;nvn = 2;nnr = 0;ca = 1;
207 disp(['Rectang. cavity, Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
208 % cs : 1 = cavity: 2 = str.
209 end
210 if Gi == 13
211 % ..... 13. Rectangular cavity ..transient linear .....
212 xyz_cao = [0 0;3 0;3 6;0 6;1 1;2 1;2 5;1 5];
213 car_cao = [1 2 6 5;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
214 Me = 3;Di = 8;Ne = 0;Co = 7;nvn = 2;nnr = 0;ca = 1;

```

```

215 disp(['Rectang. cavity, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
216 end
217 if Gi == 14
218 % ..... 14. Street section .....
219 xyz_cao = [0 8;1 8;0 0;1 1;5 0;4 1;5 8;4 8];nbo = 10;
220 car_cao = [2 1 3 4;4 3 5 6; 6 5 7 8];
221 Me = 5;Di = 0;Ne = 0;Co = 0;nvn = 0;nnr = 0;
222 disp(['Street section, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
223 end
224 if Gi == 15
225 % ..... 15. Street section .....
226 xyz_cao = [0 8;1 8;0 0;1 1;5 0;4 1;5 8;4 8];nbo = 10;
227 car_cao = [2 1 3 4;4 3 5 6; 6 5 7 8];
228 Me = 3;Di = 0;Ne = 111;Co = 0;nvn = 0;nnr = 0;
229 disp(['Street section, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
230 end
231 if Gi == 35
232 % ..... 14. Street section .....
233 xyz_cao = [0 8;1 8;0 0;1 1;5 0;4 1;5 8;4 8];nbo = 10;
234 car_cao = [2 1 3 4;4 3 5 6; 6 5 7 8];
235 Me = 5;Di = 0;Ne = 111;Co = 0;nvn = 0;nnr = 0;
236 disp(['Street section, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
237 end
238 if Gi == 16
239 % ..... 16. Thermal bridge .....
240 xyz_cao = [-2 5; -2 6;4 5;4 6;8 6;8 8;14 6;14 8;4 0;8 0;4 12;8 12];
241 car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
242 Me = 2;Di = 2;Ne = 0;Co = 5;nvn = 1;nnr = 1;
243 disp(['Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
244 end
245 if Gi == 18
246 % ..... 18. Thermal bridge .....
247 xyz_cao = [-2 5; -2 6;4 5;4 6;8 6;8 8;14 6;14 8;4 0;8 0;4 12;8 12];
248 car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
249 Me = 1;Di = 2;Ne = 0;Co = 5;nvn = 2;nnr = 0;
250 disp(['Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
251 end
252 if Gi == 19
253 % ..... 19. Thermal bridge .....
254 xyz_cao = [-2 5; -2 6;4 5;4 6;8 6;8 8;14 6;14 8;4 0;8 0;4 12;8 12];
255 car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
256 Me = 1;Di = 11;Ne = 0;Co = 5;nvn = 2;nnr = 0;
257 disp(['Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
258 end
259 if Gi == 20
260 % ..... 19. Thermal bridge .....
261 xyz_cao = [-2 5; -2 6;4 5;4 6;8 6;8 8;14 6;14 8;4 0;8 0;4 12;8 12];
262 car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
263 Me = 3;Di = 7;Ne = 0;Co = 5;nvn = 2;nnr = 0;
264 disp(['Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
265 end
266 if Gi == 21
267 % ..... 16. Thermal bridge .....
268 xyz_cao = [-2 5; -2 6;4 5;4 6;8 6;8 8;14 6;14 8;4 0;8 0;4 12;8 12];
269 car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
270 Me = 4;Di = 2;Ne = 0;Co = 5;nvn = 1;nnr = 1;
271 disp(['Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
272 end
273 if Gi == 22
274 % ..... 16. Thermal bridge .....
275 xyz_cao = [-2 5; -2 6;4 5;4 6;8 6;8 8;14 6;14 8;4 0;8 0;4 12;8 12];
276 car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
277 Me = 4;Di = 12;Ne = 0;Co = 6;nvn = 1;nnr = 0;
278 disp(['Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
279 end
280 if Gi == 36
281 % ..... 16. Thermal bridge .....
282 xyz_cao = [-2 5; -2 6;4 5;4 6;8 6;8 8;14 6;14 8;4 0;8 0;4 12;8 12];
283 car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
284 Me = 4;Di = 16;Ne = 0;Co = 5;nvn = 1;nnr = 1;
285 disp(['Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
286 end
287 if Gi == 37
288 % ..... 16. Thermal bridge .....
289 xyz_cao = [-2 5; -2 6;4 5;4 6;8 6;8 8;14 6;14 8;4 0;8 0;4 12;8 12];
290 car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
291 Me = 4;Di = 16;Ne = 0;Co = 5;nvn = 1;nnr = 1;
292 disp(['Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
23 end
294 if Gi == 38
295 % ..... 16. Thermal bridge .....
296 xyz_cao = [-2 5; -2 6;4 5;4 6;8 6;8 8;14 6;14 8;4 0;8 0;4 12;8 12];
297 car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;

```

```

298     Me =2;Di = 22;Ne = 0;Co = 5;nvn = 1;nnr = 1;
299     disp(['Thermal bridge, Gi: ',num2str(Gi),' , Di : ',num2str(Di)])
300 end
301 end

```

Table 31: Matlab<sup>©</sup> function *cad\_gin.m* - input data - definition of domains

In the stationary problems, the Dirichlet boundary conditions are always present because at least one *DOF* must be specified.

In transient problems initial conditions must be specified.

The boundary conditions are specified with the parameter *Di*. When Dirichlet boundary conditions are not present, *Di* = 0 like in the test of *Figure 57*.

Dirichlet boundary conditions inserted with the function <i>cad_Dir.m</i>	
<pre> 1  function [lfi,fT] = cad_Dir(Di,no,nvn,car_cao,bor,pbo,nni) 2  % disp(['LD 2, num. nod side: ',num2str(nni)]) 3  % General data ..... . 4  % nnc = 5 ; % Number of fixed nodes on the horizontal sides 5  % disp(['L 25, N.fix h.-side: ',num2str(nnc)]) 6  if Di == 0 7      lfi(1)=0;fT(1)=0;nf=0;disp(['LD 7, Numb. fix nod: ',num2str(nf)]) 8  end 9  if Di == 1 % lfi = list of DOF on the top of the cavity 10    lfi = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)]; 11    nf = size(lfi,2);fT = ones(1,nf)*300; 12    disp(['LD 12, Fixed nodes : ',num2str(lfi)]) 13  end 14  if Di == 2 15    fT =[280 300];lfi=[no+1 no+2];% nf=2; % Fixation of two virtual nodes 16    disp(['LD 16, Fixed nodes : ',num2str(lfi)]) 17    disp(['LD 17, Fix. temper. : ',num2str(fT),' K']); 18  end 19  if Di == 22 20    fT =[270 300];lfi=[no+1 no+2];% nf=2; % Fixation of two virtual nodes 21    disp(['LD 21, Fixed nodes : ',num2str(lfi)]) 22    disp(['LD 22, Fix. temper. : ',num2str(fT),' K']); 23  end 24  if Di == 3 25    nnc = 5; % nnc = number of fixed nodes on the horizontal sides 26  if nni &lt; nnc;nnc = nni;end 27    a = [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)]; 28    b = [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)]; 29    nx = min(nnc,nni+3);lfi=[b(nni+3-nx:nni+2) a(nni+3-nx:nni+2)]; 30    nf = size(lfi,2);fT = [ones(1,nf/2)*270 ones(1,nf/2)*320]; 31    disp(['LD 23, N. fix. nodes: ',num2str(nf)]) 32  end 33  if Di == 4 34    lfi = [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4) ... 35    car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)]; 36    nf = size(lfi,2);fT = [ones(1,nf/2)*270 ones(1,nf/2)*300]; 37    disp(['LD 36, N. fix. nodes: ',num2str(nf)]) 38    if nf &lt; 20;disp(['LD 37, Fix. DOF, lfi: ',num2str(lfi)]);end 39  end 40  if Di == 20 41    lf = [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2) ... 42    car_cao(2,1) bor(pbo(2,1),5):bor(pbo(2,1),6) car_cao(2,2) ... 43    car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2) ... 44    car_cao(6,3) bor(pbo(6,3),5):bor(pbo(6,3),6) car_cao(6,4) ... 45    car_cao(7,3) bor(pbo(7,3),5):bor(pbo(7,3),6) car_cao(7,4) ... 46    car_cao(8,3) bor(pbo(8,3),5):bor(pbo(8,3),6) car_cao(8,4)]; 47    nt = size(lf,2);lfi = zeros(1,nt);k = 3;lfi(1:3)=lf(1:3); 48    for i = 4 : nt;n = 0 ; 49      for j = 1 : i-1;if lf(j) == lf(i);n = 1;end;end 50      if n &gt; 0;lf(i)=0;else;k=k+1;lfi(k)=lf(i);end 51    end;nf = k;fT = [ones(1,nf/2)*270 ones(1,nf/2)*300]; 52    disp(['LD 52, N. fix. nodes: ',num2str(nf)]) 53    if nf &lt; 20;disp(['LD 53, Fix. DOF, lfi: ',num2str(lfi(1:nf))]);end 54  end 55  if Di == 5 56    lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2) ... 57    car_cao(4,3) bor(pbo(4,3),5):bor(pbo(4,3),6) car_cao(4,4)]; 58    nf = size(lfi,2); 59    if nf &gt; 2 ;fT = [ones(1,nf/2)*300 ones(1,nf/2)*310 ];end 60  end 61  if Di == 6 62    lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2) ... 63    car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)]; </pre>	

```

64      lfi=[car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4) ...
65      car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)];
66      nf = size(lfi,2);
67      if nf > 2 ;fT = [ones(1,nf/2)*300 ones(1,nf/2)*310 ];end
68  end
69  if Di == 7
70      fT =[300 280];lfi=[no+1 no+2];% nf=2;           % Fix. of both virt. nodes
71  end
72  if Di == 8                                % Fixation of low horizontal cavity side
73      lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
74      nf = size(lfi,2);fT = ones(1,nf)*270; % Dirichlet boundary conditions
75      lfi(1,nf+1:nf+2)=[no+1 no+2];fT(1,nf+1:nf+2)=[300 300];nf=nf+2;
76      disp(['LD 61, N. fix. nodes: ',num2str(nf)])
77      disp(['LD 62, Fix. DOF, lfi: ',num2str(lfi)])
78      disp(['LD 63, Imposed temp.: ',num2str(fT), ' K'])
79  end
80  if Di ==18                                % Fixation of low horizontal cavity side
81      lfi=[car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
82      nf = size(lfi,2);fT = ones(1,nf)*270; % Dirichlet boundary conditions
83      lfi(1,nf+1:nf+2)=[no+1 no+2];fT(1,nf+1:nf+2)=[300 300];nf=nf+2;
84      disp(['LD 67, N. fix. nodes: ',num2str(nf)])
85      disp(['LD 68, Imposed temp.: ',num2str(fT), ' K'])
86      disp(['LD 69, Fix. DOF, lfi: ',num2str(lfi)])
87      disp(['LD 70, Av. imp. temp: ',num2str(mean(fT)), ' K'])
88  end
89  if Di == 16
90      fT =[280 300];lfi=[no+1 no+2];% nf=2;           % Fix. of 2 virt. nodes
91      disp(['LD 76, Fixed nodes : ',num2str(lfi)])
92      disp(['LD 77, Fix. temper. : ',num2str(fT), ' K']);
93  end
94  if Di == 9
95      lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
96      fT = ones(size(lfi,2),1)*273;
97      if(size(lfi,2)) < 10;disp(['LD 62, Fixed nodes : ',num2str(lfi)]);end
98      if(size(lfi,2)) < 10;disp(['LD 63, Fixed temp. : ',num2str(fT')]);end
99      disp(['LD 99, Numb. of fix.: ',num2str(size(lfi,2))])
100 end
101 if Di == 10
102     fT =[270 300 280 ];lfi=[no+1 no+2 no+3];% nf=3;% Fix. of 3 virt. nodes
103     disp(['LD 68, Fixed nodes : ',num2str(lfi)])
104     disp(['LD 69, Fix. temper. : ',num2str(fT), ' K']);
105 end
106 if Di == 17          % Dirichlet= fixation of 3 virt. nodes and the base
107     lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2) ...
108         no+1 no+2 no+3];
109     nf = size(lfi,2);fT = [ones(1,nf-3)*280 300 290 300] ;
110     if nf < 10
111         disp(['LD 76, Fixed nodes : ',num2str(lfi)])
112         disp(['LD 77, Fix. temper. : ',num2str(fT), ' K']);
113     else
114         disp(['LD 79, Fix. conv nod: ',num2str(lfi(nf-2:nf))])
115         disp(['LD 80, Fix. conv nod: ',num2str(fT (nf-2:nf))])
116     end
117 end
118 if Di == 11
119     fT =[300 280];lfi=[no+1 no+2];% nf=2; % Fixation of two virtual nodes
120     disp(['LD 73, Fixed nodes : ',num2str(lfi)])
121     disp(['LD 74, Fix. temper. : ',num2str(fT), ' K']);
122 end
123 if Di == 12
124     fT =300;lfi=no+1 ;% nf=1;           % Fixation of one virtual nodes
125     disp(['LD110, Fixed nodes : ',num2str(lfi)])
126     disp(['LD111, Fix. temper. : ',num2str(fT), ' K']);
127 end
128 if Di == 13
129     fT =[300 270];lfi=[no+1 no+2];% nf=2; % Fixation of two virtual nodes
130     disp(['LD 83, Fixed nodes : ',num2str(lfi)])
131     disp(['LD 84, Fix. temper. : ',num2str(fT), ' K']);
132 end
133 if Di == 14
134     fT=[300 300 300 300];lfi=[no+1 no+2 no+3 no+4];%nf=4;%Fix.4 virt. nod.
135     disp(['LD 88, Fix. nod. lfi: ',num2str(lfi)])
136     disp(['LD 89, Fix. temp. fT: ',num2str(fT), ' K']);
137 end
138 if Di == 15
139     fT=[280 280 280 280];lfi=[no+1 no+2 no+3 no+4];%nf=4;%Fix.4 virt. nod.
140     disp(['LD 93, Fix. nod. lfi: ',num2str(lfi)])
141     disp(['LD 94, Fix. temp. fT: ',num2str(fT), ' K']);
142 end
143 if Di == 19          % lfi = list of DOF on the top of the cavity
144     lfi = [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2) ...
145         no+1 no+2 no+3];
146     nf = size(lfi,2);fT = [ones(1,nf-3)*270 280 300 290];

```

```

147 end
148 % if nfi < 7;disp(['L 73, fix. top side: ',num2str(lfi)]);end
149 % bc = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
150 % [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
151 % [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
152 % [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)]];
153 % disp(['L 78, n.fix. cavity: ',num2str(size(bc))])
154 % fT = zeros(1,no); lfi = zeros(1,no);
155 if Di > 21
156 if nvn == 0 % lfi = uni-line matrix, fT = uni-line matrix
157     lfi = [[car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
158         [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)]];
159     nf = size(lfi,2)/2;fT=[ones(nf,1)*270 ;ones(nf,1)*300];
160     mfi = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
161 end
162 if nvn == 1
163     lfi = [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
164     nf=size(lfi,2);fT=ones(1,nfi)*280;
165 % % DOF of the 1. Standard cavity
166 % bc = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
167 % [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
168 % [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
169 % [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)]];
170 end
171 if nvn == 2 % Dirichlet boundary conditions for convection
172     fT =[270 300];lfi=[no+1 no+2]; % Fixation of 2 virtual nodes
173 % end
174 if nvn == 3
175     fT=[270 285 300];lfi=[no+1 no+2 no+3]; % Fixation of 3 virtual nodes
176 end
177 nfi = size(lfi,2);disp(['LD177, N. fix. nodes: ',num2str(nfi)])
178 if nfi > 0
179     if nfi < 15
180         disp(['LD109, Numb. fix. N.: ',num2str(nf)])
181         disp(['LD110, Fixed nodes : ',num2str(lfi)])
182         disp(['LD111, Fix. temper. : ',num2str(fT),' K']);
183     end
184 % end
185 end
186 end

```

Table 32: Matlab<sup>®</sup> function *cad\_Dir.m* - Dirichlet boundary conditions

The specification of explicit nodal heat flows corresponds to the von Neumann boundary conditions. Sun light is typically introduced in this way.

Neumann boundary conditions inserted with the function <i>cad_Neu.m</i>	
1	function [gh,lg,bos] = cad_Neu(Ne,dK,car_cao,pbo,bor,th,xyz_cao)
2	%..... Neumann boundary conditions
3	disp(['N 03, param Ne : ',num2str(Ne)]); bos=0;lg=zeros(1,1);
4	if Ne == 0
5	gh = zeros(dK,1); bos=0; % 2d member initialization always necessary
6	end
7	if Ne == 1
8	gh = zeros(dK,1); % second member initialization
9	lg = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
10	nh = size(lg,2);np=nh-1;g(1)=1/(2*np);g(nh)=g(1);g(2:nh-1)=1/np;
11	gh(lg(1:nh)) = g(1:nh)/sum(g);
12	bos = (xyz_cao(2,1)-xyz_cao(1,1))*th;
13	disp(['N 13, Loaded area : ',num2str(bos),' m2'])
14	end
15	if Ne == 2
16	gh = zeros(dK,1); % second member initialization
17	lg = [car_cao(8,3) bor(pbo(8,3),5):bor(pbo(8,3),6) car_cao(8,4)]...
18	[car_cao(7,3) bor(pbo(7,3),5):bor(pbo(7,3),6) car_cao(7,4)]...
19	[car_cao(6,3) bor(pbo(6,3),5):bor(pbo(6,3),6) car_cao(6,4)]];
20	ni=size(lg,2)/3;lg(ni)=0;lg(2*ni)=0;
21	k=0;for i = 1:size(lg,2);if lg(i) > 0;k=k+1;lg(k)=lg(i);end;end
22	nh = size(lg,2)-2;np=nh-1;g(1)=1/(2*np);g(nh)=g(1);g(2:nh-1)=1/np;
23	gh(lg(1:nh)) = g(1:nh)/sum(g);
24	bos = (xyz_cao(4,1)-xyz_cao(1,1))*th;
25	disp(['N 25, Loaded area : ',num2str(bos),' m2'])
26	end
27	if Ne == 11 % Used with Gi = 29 & Di= 9, Gi = 14 & Di= 0
28	gh = zeros(dK,1); % second member initialization
29	lg0=[[car_cao(2, 4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)]...
30	[car_cao(1, 4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
31	nh = size(lg0,2)/2; % Loaded nodes lg for CAD model 11
32	lg = [lg0(1:nh) lg0(nh+2:2*nh)];nh=2*nh-1;
33	for i = 1:nh; gh(lg(1,i),1) = 2; end % Weights right side

```

34      gh(lg(1,1),1) = 1 ; gh(lg(1,nh),1) = 1;
35      gh = gh*25/sum(gh);                                     % Total load = 25 W
36      disp(['N 36, N. load. nod.: ',num2str(size(lg,2))])
37      disp(['N 37, Total load : ',num2str(sum(gh)), ' W'])
38  end
39  if Ne == 111          % Used with Gi=15 Di= 0 or Gi=29 Di= 9 or Gi=35 Di= 0
40      gh = zeros(dK,1);                                     % second member initialization
41      lg = [car_cao(3, 3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)...
42           car_cao(1, 1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
43      nh = size(lg,2)/2;                                    % Loaded nodes lg for CAD model 11
44      for i = 1:nh;   gh(lg(1,i),1) = 2; end % Weights right side
45      gh(lg(1,1),1) = 1 ; gh(lg(1,nh),1) = 1;
46      gh(lg(1:nh)) = gh(lg(1:nh))/sum(gh(lg(1:nh)));
47      for i = nh+1:2*nh;gh(lg(i,1),1) = 2; end% Weights left side
48      gh(lg(1,nh+1),1) = 1 ; gh(lg(1,2*nh),1) = 1;
49      gh(lg(nh+1:2*nh))= gh(lg(nh+1:2*nh))/sum(gh(lg(nh+1:2*nh)));nh = 2*nh;
50      gh=gh/2;
51      disp(['N 51, Total load : ',num2str(sum(gh)), ' W'])
52      bos = (xyz_cao(2,1)-xyz_cao(1,1))*th*2;
53      disp(['N 53, Loaded area : ',num2str(bos), ' m2'])
54  end
55  if Ne == 0
56      gh = zeros(dK,1);%bos=0; % 2d member initialization always necessary
57  else
58      if nh < 10
59          disp(['N 59, Loaded nodes : ',num2str(lg(1:nh))])
60          disp(['N 60, Loads weights: ',num2str(gh(lg(1:nh)))])
61      else
62          disp(['N 62, N. load. nod.: ',num2str(size(lg,2))])
63      end
64  end
65 end

```

Table 33: Matlab<sup>®</sup> function *cad\_Neu.m* - Neumann boundary conditions, function

When convection is present along a part of the boundary of the domain, it is necessary to define specific finite elements (see chapter 2). These elements bear 3 nodes, two are on the domain boundary while the third one is a virtual node assumed to represent the fluid interacting with the solid. The output of the function *cad\_con.m* is twofold: the matrix of localization of the convective elements and a uni-line matrix containing their convection coefficients.

### Matlab<sup>®</sup> function *cad\_con.m* – convection boundary conditions

```

1  function [lc,vc,hv]=cad_con(car_cao,bor,pbo,h,dK,nes,deb,nvn,cs,Co,xyz)
2  npa = size(car_cao,1);% npa = number patches ; nes number of elem per side
3  disp(['c. 03, dK = no + nvn: ',num2str(dK)])           % nes = n. elem/side
4  disp(['c. 04, N.virt c.nod.: ',num2str(nvn)]);
5  disp(['c. 05, Variable Co : ',num2str(Co)]);
6  if Co == 1          % Convective elements on 2 sides of patch 1
7      bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
8           [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
9      cps = [car_cao(1,2) car_cao(1,3) car_cao(1,4) car_cao(1,1)];
10     vc(1,1)= max (xyz(:,1))*2 ; vc(2,1)=-max (xyz(:,1));
11     vc(1,2)= max (xyz(:,2))/2 ; vc(2,2)= vc(1,2);
12     disp(['c. 12, Convect. sid.: ',num2str(cps)]);
13 end
14 if Co == 2 % Convective elements on the four internal sides of the cavity
15     bt = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
16         [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
17         [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
18         [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)]];
19     cps = [car_cao(1,1) car_cao(1,2) car_cao(2,4) car_cao(2,1)...
20             car_cao(3,4) car_cao(3,1) car_cao(4,2) car_cao(4,3)];
21     vc(1,1)= mean (xyz(1:dK-nvn,1)) ; vc(1,2)= mean (xyz(1:dK-nvn,2)) ;
22     disp(['c. 22, Convect. sid.: ',num2str(cps)]);
23 end
24 if Co == 3
25     bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
26         [car_cao(2,2) bor(pbo(2,2),5):bor(pbo(2,2),6) car_cao(2,3)];
27         [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
28         [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
29     cps = [car_cao(1,2) car_cao(1,3) car_cao(2,2) car_cao(2,3)...
30             car_cao(2,4) car_cao(2,1) car_cao(1,4) car_cao(1,1)];
31     vc(1,1)= max (xyz(:,1))*2 ; vc(2,1)=-max (xyz(:,1));
32     vc(1,2)= max (xyz(:,2))/2 ; vc(2,2)= vc(1,2);
33     disp(['c. 33, Convect. sid.: ',num2str(cps)]);
34 end
35 if Co == 4
36     bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
```

```

37 [car_cao(2,2) bor(pbo(2,2),5):bor(pbo(2,2),6) car_cao(2,3)];;
38 cps = [car_cao(1,2) car_cao(1,3) car_cao(2,2) car_cao(2,3)];
39 vc(1,1)= max (xyz(:,1))*2 ;vc(1,2)= max (xyz(:,2))/2;
40 vc(2,1)= vc(1,1) ;vc(2,2)= vc(1,2);
41 disp(['c. 41, Convect. sid.: ',num2str(cps)]);
42 end
43 if Co == 5 % Two external vertical sides of a four patches cavity
44 bt = [[car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)];
45 [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)]];
46 cps = [car_cao(3,3) car_cao(3,4) car_cao(1,1) car_cao(1,2)];
47 vc(1,1)= (xyz(6,1)+xyz(8,1))/2; vc(1,2)=(xyz(6,2)+xyz(12,2))/2;
48 disp(['c. 48, Convect. sid.: ',num2str(cps)]);
49 end
50 if Co == 6 % Two horizontal sides on both sides of the thermal bridge
51 bt = [[car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)];
52 [car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)]];
53 cps = [car_cao(3,3) car_cao(3,4) car_cao(3,1) car_cao(3,2)];
54 vc(1,1)= (xyz(6,1)+xyz(8,1))/2; vc(1,2)=(xyz(6,2)+xyz(12,2))/2;
55 vc(2,1)= vc(1,1);vc(2,2)=vc(1,2);
56 % vc(2,1)= (xyz(1,1)+xyz(3,1))/2; vc(2,2)=(xyz(3,2)+xyz(9,2))/2;
57 disp(['c. 51, Convect. sid.: ',num2str(cps)]);
58 end
59 if Co == 7 % Two external vertical sides of a four patches cavity
60 bt = [[car_cao(2,2) bor(pbo(2,2),5):bor(pbo(2,2),6) car_cao(2,3)];
61 [car_cao(4,4) bor(pbo(4,4),5):bor(pbo(4,4),6) car_cao(4,1)]];
62 cps = [car_cao(2,2) car_cao(2,3) car_cao(4,4) car_cao(4,1)];
63 vc(1,1)= max (xyz(:,1))*2 ; vc(2,1)=-max (xyz(:,1));
64 vc(1,2)= max (xyz(:,2))/2 ; vc(2,2)= vc(1,2);
65 disp(['c. 65, Convect. sid.: ',num2str(cps)]);
66 end
67 if Co == 8 % Two external vertical sides of a four patches cavity
68 bt = [[car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)];
69 [car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)]];
70 cps = [car_cao(3,3) car_cao(3,4) car_cao(3,1) car_cao(3,2)];
71 vc(1,1)= (xyz(6,1)+xyz(8,1))/2; vc(1,2)=(xyz(6,2)+xyz(12,2))/2;
72 vc(2,1)= (xyz(5,1)+xyz(7,1))/2; vc(2,2)=(xyz(5,2)+xyz(10,2))/2;
73 disp(['c. 73, Convect. sid.: ',num2str(cps)]);
74 end
75 if Co > 8
76 if npa == 1 % The domain contains a single patch
77 if nvn == 2 % 2 convective sides
78 bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
79 [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
80 cps = [car_cao(1,2) car_cao(1,3) car_cao(1,4) car_cao(1,1)];
81 disp(['c. 81, Convect. sid.: ',num2str(cps)]);
82 end
83 if nvn == 3 % 3 convective sides
84 bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
85 [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
86 [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
87 cps = [car_cao(1,2) car_cao(1,3) car_cao(1,4) ...
88 car_cao(1,4) car_cao(1,1)];
89 disp(['c. 89, Convect. sid.: ',num2str(cps)]);
90 end
91 end
92 if npa == 2 % The domain contains two patches
93 if nvn == 3 % 3 convective sides
94 bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
95 [car_cao(2,2) bor(pbo(2,2),5):bor(pbo(2,2),6) car_cao(2,3)];
96 [car_cao(2,3) bor(pbo(2,3),5):bor(pbo(2,3),6) car_cao(2,4)];
97 [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
98 [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
99 cps = [car_cao(1,2) car_cao(1,3) car_cao(2,2) car_cao(2,3) ...
100 car_cao(2,3) car_cao(2,4) car_cao(2,4) car_cao(2,1) ...
101 car_cao(1,4) car_cao(1,1)];
102 disp(['c.102, Convect. sid.: ',num2str(cps)]);
103 end
104 if nvn == 4 % 4 convective sides on the rect. with 2 patches
105 bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
106 [car_cao(2,2) bor(pbo(2,2),5):bor(pbo(2,2),6) car_cao(2,3)];
107 [car_cao(2,3) bor(pbo(2,3),5):bor(pbo(2,3),6) car_cao(2,4)];
108 [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
109 [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)];
110 [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)]];
111 end
112 cps = [car_cao(1,2) car_cao(1,3) car_cao(2,2) car_cao(2,3) ...
113 car_cao(2,3) car_cao(2,4) car_cao(2,4) car_cao(2,1) ...
114 car_cao(1,4) car_cao(1,1) car_cao(1,1) car_cao(1,2)];
115 disp(['c.115, Convect. sid.: ',num2str(cps)]);
116 vc(1,1)= max (xyz(:,1))*1.5;vc(1,2)=max (xyz(:,2))/2 ;
117 vc(2,1)= max (xyz(:,1))/2;vc(2,2)=max (xyz(:,2))+max(xyz(:,1))/2;
118 vc(3,1)= -max (xyz(:,1))/2;vc(3,2)= max (xyz(:,2))/2;
119 vc(4,1)= max (xyz(:,1))/2 ;vc(4,2)= -max(xyz(:,1))/2;

```

```

120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202

    end
    if csw=5
        if npa > 2
            % The domain contains more than two patches
            if nvn == 1
                % 1 convective side
                bt = [[car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];
                       [car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)];
                       [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
                       [car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)];
                       [car_cao(5,2) bor(pbo(5,2),5):bor(pbo(5,2),6) car_cao(5,3)]];
                cps = [car_cao(4,2) car_cao(4,3) car_cao(3,1) car_cao(3,2) ...
                        car_cao(3,2) car_cao(3,3) car_cao(3,3) car_cao(3,4) ...
                        car_cao(5,2) car_cao(5,3)];
                disp(['c.132, Convect. sid.: ',num2str(cps)]);
                if deb == 1
                    a=1;hw = [a*0 a*0 a*0 a*h a*0];
                    disp(['c.135, Conv. coeff. : ',num2str(hw),' W/(m2K)']);
                end
            end
            if nvn == 2
                % 2 convective sides
                bt = [[car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];
                       [car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)];
                       [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
                       [car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)];
                       [car_cao(5,2) bor(pbo(5,2),5):bor(pbo(5,2),6) car_cao(5,3)];
                       [car_cao(5,4) bor(pbo(5,4),5):bor(pbo(5,4),6) car_cao(5,1)];
                       [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
                       [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)];
                       [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
                       [car_cao(4,4) bor(pbo(4,4),5):bor(pbo(4,4),6) car_cao(4,1)]];
                cps = [car_cao(4,2) car_cao(4,3) car_cao(3,1) car_cao(3,2) ...
                        car_cao(3,2) car_cao(3,3) car_cao(3,3) car_cao(3,4) ...
                        car_cao(5,2) car_cao(5,3) car_cao(5,4) car_cao(5,1) ...
                        car_cao(1,3) car_cao(1,4) car_cao(1,4) car_cao(1,1) ...
                        car_cao(1,1) car_cao(1,2) car_cao(4,4) car_cao(4,1)];
                disp(['c.154, Convect. sid.: ',num2str(cps)]);
                if deb == 1
                    a=1;hw = [a*0 a*0 a*h/2 a*h a*0 a*h a*h/2 a*0 a*0];
                    disp(['c.157, Conv. coeff. : ',num2str(hw),' W/(m2K)']);
                end
            end
            if nvn == 3
                % 3 convective sides - Exercice n° 4
                bt = [car_cao(2,1) bor(pbo(2,1),5):bor(pbo(2,1),6) car_cao(2,2);
                       car_cao(2,2) bor(pbo(2,2),5):bor(pbo(2,2),6) car_cao(2,3);
                       car_cao(2,3) bor(pbo(2,3),5):bor(pbo(2,3),6) car_cao(2,4);
                       car_cao(5,1) bor(pbo(5,1),5):bor(pbo(5,1),6) car_cao(5,2);
                       car_cao(5,2) bor(pbo(5,2),5):bor(pbo(5,2),6) car_cao(5,3);
                       car_cao(5,3) bor(pbo(5,3),5):bor(pbo(5,3),6) car_cao(5,4);
                       car_cao(8,1) bor(pbo(8,1),5):bor(pbo(8,1),6) car_cao(8,2);
                       car_cao(8,2) bor(pbo(8,2),5):bor(pbo(8,2),6) car_cao(8,3);
                       car_cao(8,3) bor(pbo(8,3),5):bor(pbo(8,3),6) car_cao(8,4)];
                cps = [car_cao(2,1) car_cao(2,2) car_cao(2,2) car_cao(2,3) ...
                        car_cao(2,3) car_cao(2,4) car_cao(5,1) car_cao(5,2) ...
                        car_cao(5,2) car_cao(5,3) car_cao(5,3) car_cao(5,4) ...
                        car_cao(8,1) car_cao(8,2) car_cao(8,2) car_cao(8,3) ...
                        car_cao(8,3) car_cao(8,4) ];
                disp(['c.175, Convect. sid.: ',num2str(cps)]);
                if deb == 1
                    a=1;hw = [a*h a*h a*h a*h a*h a*h a*h a*h];
                    disp(['c.178, Conv. coeff. : ',num2str(hw),' W/(m2K)']);
                end
            end
        end
    end
    nec = (nes)*size(bt,1); % There are nec convective elements
    hv = ones(1,nec)*h;
    disp(['c.186, Nu. conv. el.: ',num2str(nec)])
    if nec < 10;disp(['c.187, conv. coeff. : ',num2str(hv),' W/(m2K)']);end
    % if nec < 10;disp(bt);end
    lc = zeros(nec,3);nco = 0;% Localization matrix lc of convective elements
    for ic = 1:size(bt,1) % Loop on the convective edges
        for ie = 1:nes % Loop on the nes elements of a side
            nco = nco+1;lc(nco,1) = bt(ic,ie);lc(nco,2) = bt(ic,ie+1);
        end
    end
    if nvn == 1 % 1 convective side
        for i = 1:size(lc,1)
            lc(i,3) = dK; % Convective virtual node numb. = numb. of dof
        end
    else
        if nvn == 2 % 2 convective sides
            if Co == 5; npa =1;end
            if Co == 7; npa =1;end
        end
    end

```

```

203      k1 = [1      npa*nes+1  ];
204      k2 = [npa*nes (2*npa)*nes];
205  end
206  if nvn == 3
207      k1 = [1      2*nco/5+1 3*nco/5+1];
208      k2 = [2*nco/5 3*nco/5   nco      ];
209  end
210  if nvn == 4
211      k1 = [1      2*nco/6+1 3*nco/6+1 5*nco/6+1];
212      k2 = [2*nco/6 3*nco/6   5*nco/6   nco      ];
213  end
214  for i           = 1:nvn % if nvn > 1, generation of the conv V. nodes
215      for j       = k1(i):k2(i)
216          lc(j,3) = dK + i - nvn;
217      end
218  end
219 end
220 if deb==1
221     disp(['c.221, Co. virt. nod: ',num2str(dK-nvn+1:dK)])
222 end
223 end

```

Table 34: Matlab<sup>®</sup> function *cad\_con.m* - localization of convection elements

Matlab <sup>®</sup> function <i>cad_mes.m</i>	
1	<code>function [xyc,lK,bor,pbo] = cad_mes(xyc,lca,ni,nbo)</code>
2	<code>nec = size(lca,1);</code> % Number of CAD patches
3	<code>ndo = size(xyc,1);nd = ndo;</code> % Number of CAD nodes
4	<code>[bor, pbs] = cad_edg(lca,nbo);</code> % Compute the nbo patch sides in bor matrix
5	% =====
6	<code>if ni == 1 % ni = 1 : one node generated on the interfaces</code>
7	<code>    lai = zeros(1,nbo);</code> % List of border edges
8	<code>    for i = 1: nbo</code>
9	<code>        lai(i) = nd + i;</code>
10	<code>        bor(i,5) = lai(i);</code>
11	<code>        bor(i,6) = bor(i,5);</code>
12	<code>        xyc(nd+i,:) = (xyc(bor(i,1),:)+xyc(bor(i,2),:))/2;</code>
13	<code>    end</code>
14	<code>else % More than 1 node have to be generated on the interfaces</code>
15	<code>    k = nd;</code>
16	<code>    for i = 1 : nbo</code> % nbo is the number of interfaces
17	<code>        bor(i,5) = nd + (i-1)*ni+1;</code>
18	<code>        bor(i,6) = nd + i*ni;</code>
19	<code>        for j = 1 : ni</code>
20	<code>            A = xyc(bor(i,1),:);B = xyc(bor(i,2),:);</code>
21	<code>            k = k + 1;</code>
22	<code>            xyc(k,:) = A + (B-A)*j/(ni+1); % coord. of the interface nodes</code>
23	<code>        end</code>
24	<code>    end</code>
25	<code>end</code>
26	<code>pbo = sign(pbs(:,:,1)).*pbs;</code>
27	<code>nna = ndo + nbo*ni;</code> % Numb. of nodes after interfaces gen.
28	<code>lK = zeros(nec*(ni+1)^2,4);nu = 0;</code>
29	<code>for n = 1:nec % Loop on the CAD patches .....</code>
30	<code>    to = zeros(ni+2,ni+2);</code> % Gen. to = list of nodes matrix
31	<code>    to(1,1) = lca(n,1);</code> % Start with the 4 patch vertices
32	<code>    to(1,ni+2) = lca(n,2); to(ni+2,ni+2) = lca(n,3);</code>
33	<code>    to(ni+2,1) = lca(n,4);</code> % End patch vertices
34	<code>    if bor(pbo(n,1),3) == n % Line 1 of patch matrix to</code>
35	<code>        to(1,2:ni+1) = bor(pbo(n,1),5) :bor(pbo(n,1),6);</code>
36	<code>    else</code>
37	<code>        to(1,2:ni+1) = bor(pbo(n,1),6):-1:bor(pbo(n,1),5);</code>
38	<code>    end</code>
39	<code>    if bor(pbo(n,3),3) == n % Line ni+2 of patch matrix to</code>
40	<code>        to(ni+2,2:ni+1) = bor(pbo(n,3),6):-1:bor(pbo(n,3),5);</code>
41	<code>    else</code>
42	<code>        to(ni+2,2:ni+1) = bor(pbo(n,3),5) :bor(pbo(n,3),6);</code>
43	<code>    end</code>
44	<code>    if bor(pbo(n,4),3) == n % Side 4 = first column of matrix to</code>
45	<code>        to(2:ni+1,1) = bor(pbo(n,4),6):-1:bor(pbo(n,4),5);</code>
46	<code>    else</code>
47	<code>        to(2:ni+1,1) = bor(pbo(n,4),5) :bor(pbo(n,4),6);</code>
48	<code>    end</code>
49	<code>    if bor(pbo(n,2),3) == n % Side 2 = column ni+2 of matrix to</code>
50	<code>        to(2:ni+1,ni+2) = bor(pbo(n,2),5) :bor(pbo(n,2),6);</code>
51	<code>    else</code>
52	<code>        to(2:ni+1,ni+2) = bor(pbo(n,2),6):-1:bor(pbo(n,2),5);</code>
53	<code>    end</code>
54	<code>% Generation of internal nodes if ni &gt; 0 .....</code>
55	<code>x1 = xyc(to(1,1),:);x2 = xyc(to(1,ni+2),:);</code> % Patch vertices

```

56 x3 = xyc(to(ni+2,1),:);x4 = xyc(to(ni+2,ni+2),:);
57 for k = 1 : ni % ni x ni new nodes inside the patch
58   for j = 1: ni
59     s = k/(ni+1);t=j/(ni+1);% disp([s t])
60     nna = nna+1;
61     to(j+1,k+1) = nna; % Interior of the patch
62     xyc(nna,:) = x1*(1-s)*(1-t)+x2*s*(1-t)+x3*(1-s)*t+x4*s*t;
63   end
64 end
65 % End of generation of the internal nodes .....
66 for i = 1:ni+1 Mesh generation: (ni+1)(ni+1) elements per patch .....
67   for j = 1:ni+1
68     nu = nu+1;
69     lK(nu,:) = [to(i,j) to(i+1,j) to(i+1,j+1) to(i,j+1)];
70   end
71 end % End of mesh generation .....
72 end % End of loop on the CAD patches.....
73 end

```

Table 35: Matlab<sup>®</sup> function *cad\_mes.m* - construction of the CAD mesh topology

Matlab <sup>®</sup> function <i>cad_edg.m</i>	
1	<code>function [bor,pbo] = cad_edg(lca,nbo)</code>
2	<code>nec = size(lca,1);</code> % nec = number of elements
3	<code>bor = zeros(nbo,8);boe=zeros(nec,8);pbo = zeros(nec,4);% bor = patch sides</code>
4	<code>for ie = 1:nec % Loop ie on the nec patches</code>
5	<code>boe(1,1) = lca (ie,1);boe(1,2) = lca (ie,2);boe(1,3) = ie;</code>
6	<code>boe(2,1) = lca (ie,2);boe(2,2) = lca (ie,3);boe(2,3) = ie;</code>
7	<code>boe(3,1) = lca (ie,3);boe(3,2) = lca (ie,4);boe(3,3) = ie;</code>
8	<code>boe(4,1) = lca (ie,4);boe(4,2) = lca (ie,1);boe(4,3) = ie;</code>
9	<code>if ie == 1</code>
10	<code>    bor(1:4,:) = boe(1:4,:);</code>
11	<code>    pbo(1,1:4) = 1:4;</code>
12	<code>    nbo = 4;</code>
13	<code>else</code>
14	<code>    for i = 1 : 4 % Loop on the new sides</code>
15	<code>      flag = 0;</code>
16	<code>      for kl = 1:nbo % Loop on the yet detected sides</code>
17	<code>        if flag == 0</code>
18	<code>          if boe(i,2) == bor(kl,1)</code>
19	<code>            if boe(i,1) == bor(kl,2)</code>
20	<code>              bor(kl,4) = ie; % Side detected before</code>
21	<code>              pbo(ie,i) = -kl; % 2d occurence of a bor line</code>
22	<code>              flag = 1;</code>
23	<code>            end</code>
24	<code>          end</code>
25	<code>        end</code>
26	<code>      if flag == 0</code>
27	<code>        nbo = nbo +1;</code>
28	<code>        bor(nbo,:) = boe(i,:);</code>
29	<code>        pbo(ie,i) = nbo;</code>
30	<code>      end;</code>
31	<code>    end</code>
32	<code>  end</code>
33	<code>end</code>
34	<code>end</code>
35	<code>end</code>

Table 36: Matlab<sup>®</sup> function *cad\_edg.m* – definition of the CAD mesh interfaces

Matlab <sup>®</sup> function <i>cad_ban.m</i> – Radiative nodes of cavity, street or balcony	
1	<code>function[lic,lv] = cad_ban(car_cao,bor,pbo,nni,cs) % 20210929</code>
2	<code>% For the top of the cavity, see Di = 1 in cad_Dir.m</code>
3	<code>if cs ==1 % DOF on four sides</code>
4	<code>  lcont = [[car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];</code>
5	<code>           [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];</code>
6	<code>           [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];</code>
7	<code>           [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)]];</code>
8	<code>  lv =[lcont(1,1) lcont(1,nni+2) lcont(2,nni+2) lcont(3,nni+2)];</code>
9	<code>  lic=[lcont(1,1:nni+2) lcont(2,2:nni+2) lcont(3,2:nni+2) lcont(4,2:nni+1)];</code>
10	<code>end</code>
11	<code>if cs ==6 % DOF on the four sides of the quadrilateral cavity</code>
12	<code>  lcont = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];</code>
13	<code>           [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];</code>
14	<code>           [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];</code>
15	<code>           [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)]];</code>
16	<code>  lv =[lcont(1,1) lcont(1,nni+2) lcont(2,nni+2) lcont(3,nni+2)];</code>
17	<code>  lic=[lcont(1,1:nni+2) lcont(2,2:nni+2) lcont(3,2:nni+2) lcont(4,2:nni+1)];</code>

```

18 end
19 if cs ==7 % DOF on the four sides of the square cavity Gi = 40
20 lcont = [[car_cao(2,3) bor(pbo(2,3),5):bor(pbo(2,3),6) car_cao(2,4)];
21 [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];
22 [car_cao(7,1) bor(pbo(7,1),5):bor(pbo(7,1),6) car_cao(7,2)];
23 [car_cao(5,4) bor(pbo(5,4),5):bor(pbo(5,4),6) car_cao(5,1)]];
24 lv =[lcont(1,1) lcont(1,nni+2) lcont(2,nni+2) lcont(3,nni+2)];
25 lic=[lcont(1,1:nni+2) lcont(2,2:nni+2) lcont(3,2:nni+2) lcont(4,2:nni+1)];
26 end
27 if cs ==2 % ..... 11. Street section - 3 patches .....
28 c1 = [car_cao(1,1) bor(pbo(1,4),6):-1:bor(pbo(1,4),5) car_cao(1,4)];
29 c2 = [car_cao(2,1) bor(pbo(2,4),6):-1:bor(pbo(2,4),5) car_cao(2,4)];
30 c3 = [car_cao(3,1) bor(pbo(3,4),6):-1:bor(pbo(3,4),5) car_cao(3,4)];
31 lic = [c1 c2(2:size(c2,2)-1) c3]'; % List of street boundary nodes
32 lv = [lic(1) lic(nni+2) lic(2*nni+3) lic(3*nni+4)];% Vert.
33 end
34 if cs ==3 % List of balcony boundary nodes.....
35 bt = [[car_cao(5,4) bor(pbo(5,4),5):bor(pbo(5,4),6) car_cao(5,1)];
36 [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
37 [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)];
38 [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
39 [car_cao(4,4) bor(pbo(4,4),5):bor(pbo(4,4),6) car_cao(4,1)]];
40 lic = [bt(1,1:nni+2) bt(2,2:nni+2) bt(3,2:nni+2) bt(4,2:nni+2)...
41 bt(5,2:nni+2)]; % DOF concerned by radiative heat transfer
42 lv = [lic(1) lic(nni+2) lic(2*nni+3) lic(3*nni+4)...
43 lic(4*nni+5) lic(5*nni+6)]; % Vertices
44 end
45 if cs == 4 % List of rectangle right side nodes.....
46 bt = [[car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
47 [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
48 lic = [bt(1,1:nni+2) bt(2,1:nni+2)]';
49 lv = [lic(1) lic(nni+2) lic(2*nni+4)];
50 end
51 if cs ==5% Quadrilateral!!! previous version replaced by a new 20211021 !!
52 bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
53 [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
54 lic = [bt(1,1:nni+2) bt(2,1:nni+2)]';
55 lv = [lic(1) lic(nni+2) lic(nni+3) lic(2*nni+4)];
56 end
57 end

```

Table 37: Matlab<sup>®</sup> function *cad\_ban.m* – radiative nodes of cavity, street or balcony

### 8.3 Transient heat transfer

This section begins with the presentation of four solution methods of the system of equation. The fifth method is explicitly written in the principal procedure *Fiammetta.m* between lines 192 & 232.

Matlab <sup>®</sup> function <i>fem_smd.m</i> - solution of the nonlinear radiative system
<pre> 1 unction [tca] = fem_smd(Kk,gh,lfi,fT,xyz,lK,lcont,nnr,nnv,nci,np,SBte... 2 ,ca,nbo,bor) 3 nf = size(fT,2);% disp(['Ls 2, list f. nodes: ',num2str(lfi(1:nf))]) 4 % disp(['Ls 3, Fix. temper. : ',num2str(fT),' K']) 5 disp(['sd 5, N. fixed nod.: ',num2str(nf)]) 6 if ca == 1 7     ner = max(size(lcont));disp(['sd 7, N. rad elem. : ',num2str(ner)]) 8     lcont(ner+1)=lcont(1); 9 else 10    ner = max(size(lcont))-1;disp(['sd 10, N. rad elem. : ',num2str(ner)]) 11 end 12 if ner &lt; 11;disp(['sd 12, Radiat. nodes: ',num2str(lcont)]);end 13 dK = size(Kk,1); % Kk is the pure conductivity matrix 14 no = dK - nnr - nnv; % Number of unknowns of the system to solved 15 % lic = zeros(nnr,round(sqrt(no))); 16 % for ii=1:nnr % Loop on the sides involving radiation conditions 17 %     j = ii+2;if j==5;j=1;end 18 %     lic(ii,:)= [car_cao(1,ii+1) bor(pbo(1,ii+1),5):bor(pbo(1,ii+1),6)... 19 %                 car_cao(1,j)]; % List of the DOF of the irradiated patch side 20 % end 21 lc = zeros(ner,3); % Localization matrix of the radiative elements 22 K = Kk; % Pure conductivity matrix 23 tcant = [ones(1,no)*fT(1) fT]; % Initial temperature field 24 for iter = 1 : 2 ;disp(['sd 24, Iteration N. : ',num2str(iter),' / 2']) 25     for n = 1:nnr % Generation of cond. rad. element matrices 26         for i = 1:ner 27             lc(i,1) = lcont(1,i); lc(i,2) = lcont(1,i+1); lc(i,3) = no+n; 28             Ker =fem_Kcr(xyz,lc(i,:),SBte,tcant);% Elem cond rad matr. 29             for ii = 1:3 30                 for jj = 1:3 </pre>

```

31           K(lc(i,ii),lc(i,jj))=K(lc(i,ii),lc(i,jj))+Ker(ii,jj);
32       end
33   end % End assembling the conductive radiative element matrices
34   end
35 end
36 N = zeros(nf,no + nnr); % Linear constraint for fixations
37 for i = 1:nf ; N(i,lfi(i))=1 ; gh(dK+i)=fT(i); end;
38 A = [K N';N zeros(nf,nf)]; B = A\gh ; tca = B(1:dK);
39 disp(['sd 39, Dissipation : ',num2str(-B'*gh,4), ' WK'])
40 disp(['sd 40, React. flows : ',num2str(B(dK+1:dK+nf)',5), ' W'])
41 gt = [fT(1),1,fT(nf)];
42 nc2 = nci*nci;figure;
43 for i = 1 : np % .....
44     gra_ipa(nci,nci,lK((i-1)*nc2+1:i*nc2,:),tca(1:no),xyz,gt);
45     colorbar;hold on
46 end
47 title(['Iteration: ',num2str(iter),' dissipation:',...
48 num2str(tca'*K*tca,3), ' WK']);axis off
49 for i = 1:nbo % Drawing the border of the domain
50 if bor(i,4)==0
51 plot([xyz(bor(i,1),1) xyz(bor(i,2),1)], ...
52 [xyz(bor(i,1),2) xyz(bor(i,2),2)],'k','LineWidth',2)
53 end
54 end
55 disp(['sd 55, max - min tca: ',num2str(max(tca)-min(tca),3), ' K'])
56 tcant = [tca(1:no,1); fT']; % Store temp. of solid for next iteration
57 end
58 end % End of iterations

```

Table 38: Matlab<sup>®</sup> function *fem\_smd.m* - solution of the nonlinear radiative system

The *fem\_smd.m* function performs the required iterations to take into account the non-linearity of the “conduction” - “radiation” problem. All the iterations may provide an isotherm drawing.

Function <i>fem_smt.m</i> - solution of linear transient equations	
<pre> 1 function [tca]= ... 2 fem_smt(np,xyz,lK,dK,nci,deb,cs,area,th,pai,fT,lfi,gh,nbo,bor,Kk,fmd,Di) 3 tmi=280;tma=300;if Di==15;tmi=300;tma=280;end;if Di==8;tmi=270;tma=300;end 4 nel=size(lK,1);% gt=[270 pai 300];%gt = [min(tmi,tma) pai max(tmi,tma)]; 5 lfi(1)==0;nfi=0;else;nfi=size(lfi,2);end 6 if Di == 0;tmi=280;tma=300;end; 7 % if Di==17;gt=[min(min(tmi,min(fT)),tma) pai max(tmi,tma)];end; 8 disp(['st 08, Ini. tmi, tma: ',num2str([tmi tma]),' K']); 9 disp(['st 09, Numb. fixat. : ',num2str(nfi)]); 10 if fmd == 1 % In the example of figure 41 - 42, nni must be even 11     disp(['st 11, fmd : ',num2str(fmd)]); 12     tcan = [ones(round(dK/2),1)*tmi ;ones(dK-round(dK/2),1)*tma]; 13     nfi = 0; 14     for i = 1:nci*floor(nci/2) 15         for j = 1:4 16             tcan(lK(i,j)) = tmi; 17             tcan(lK(nel+1-i,j)) = tma; 18         end 19     end 20 else % Initial and fixed temperatures nfi = 0;tcan = ones(dK,1)*tmi; 21     tcan = ones(dK,1)*tmi;if nfi&gt;0;tcan(lfi)=fT;end % nfi = 0; 22     if nfi &lt; 10;disp(['st 22, Fix. temp. fT: ',num2str(fT),' K']);end 23     if nfi &lt; 10;disp(['st 23, ddl fix. lfi : ',num2str(lfi)]);end 24 end 25 if dK &lt; 5; disp(['st 25, Initial temp.: ',num2str(tcan),' K']);end 26 nit = 720;dth=1;dtd=nit/4; % Numb. iter. &amp; delta tau per it. (hours) 27 disp(['st 27, N. iter. nit : ',num2str(nit)]) 28 dti = dth*3600;disp(['st 28, Time step : ',num2str(dti),' s']) 29 tt = dti*nit; % Analyzed period in seconds 30 nd = tt/3600/24;f=zeros(1,tt/3600); % nd = number of days 31 disp(['st 31, Analyzed per.: ',num2str(tt/3600) , ' h,',... 32 num2str(nd) , ' days']) 33 for i = 1:nd % f is the imposed periodic function, f(h = 1:12) 34     f((i-1)*24+1:(i-1)*24+12) = sin((1:12)*pi/12); 35 end 36 Cp = 1000 ;disp(['st 36, Spec. capac. : ',num2str(Cp),' J/(kg.K)']); 37 ro = 2500 ;disp(['st 37, Spec. mass : ',num2str(ro),' kg.m-3']) 38 C = zeros(dK,dK); % Initialization of the global capacity matrix 39 if deb==1;disp('st 41, Call function: .... fem_Cae .....');end 40 for n = 1:nel % Glob. capacity matrix assembling, loop on nel elements 41     Cae = fem_Cae(xyz,lK(n,:))*th*Cp*ro; % Cae = element capacity matrix 42     for i = 1:4 43         for j=1:4;C(lK(n,i),lK(n,j)) = C(lK(n,i),lK(n,j)) + Cae(i,j);end 44     end 45 end % End of capacity matrices assembling </pre>	

```

46 cap = area*th*Cp*ro*1e-6; % Domain capacity computed from mat. data
47 disp(['st 47, sum(sum(C)) : ',num2str(sum(sum(C))*1e-6), ' MJ/K']);
48 disp(['st 48, area*th*ro*Cp : ',num2str(cap), ' MJ/K']);
49 tsmax = ones(1,nit+1)*tmi;tsmin=tmi;tmoy=tmi;tcav = tmi;
50 gou = zeros(1,nit+1);
51 ih = 100;% 4.04008;% Imposed heat load in Wm-2
52 disp(['st 52, Imposed Heat : ',num2str(ih), ' W/m-2']);
53 bos = th*(max(xyz(:,1))-min(xyz(:,1)));% Cross section area upper edge
54 g = zeros(dK,1);
55 K = Kk; ip=0; % Conduct. matrix related to solid and convective part
56 for it = 1:nit % ..... Loop on the time iterations
57 if it==1;disp(['st 57, size(K) nfi : ',num2str([size(K) nfi])]);end
58 ip = ip+1;
59 if cs < 3;g = gh*f(it)*ih*bos; end % Imposed generalized heat flows
60 if nfi == 0 % The problem does not include fixations
61 tca = (C + dti*K)\(C*tcan + dti*g); % Tutorial, pp 41, Equ. 67
62 else
63 if it == 1
64 if deb ==1
65 disp ('st 68, Call function: ..... fem_tra .....')
66 end;
67 end
68 tca = fem_tra(K,C,dti,g,lfi,tcan);
69 % if it==1;disp(['t. 69, diag K : ',num2str(diag(K))]);end
70 % if it==1;disp(['t. 70, diag C : ',num2str(diag(C))]);end
71 % if it==1;disp(['t. 71, tca... : ',num2str(tca')]);end
72 go = K * tca(1:size(K,1)); % Second member of the system
73 gou(it) = sum(go)*dti; % Reactions at iteration it
74 end
75 tcav (it+1) = tca (dK,1);
76 tsmax(it+1) = max (tca(1:dK-nfi));
77 tsmin(it+1) = min (tca(1:dK-nfi));
78 tmoy (it+1) = mean(tca(1:dK-nfi));
79 tcan = tca;
80 if ip == dtd % In this iteration, isotherms drawing is generated
81 ip = 0; gt=[min(tca) pai max(tca)];
82 disp(['st 82, iteration : ',num2str(it)])
83 if deb==1;disp ('st 86, Call function: ..... gra_ipa .....');
84 figure
85 ori = min(xyz(:,1));
86 a = ori-.2;b=ori-.1;ha = max(xyz(:,2)); %.... drawing a left bar
87 fill([a b b a],[0 0 ha ha],[tmi tmi tma tmaj]);hold on;
88 for i = 1 : np % ..... Loop on the np CAD patches
89 gra_ipa(nci,nci,1K((i-1)*nci^2+1:i*nci^2,:),tca',xyz,gt);
90 colorbar;axis off;hold on
91 end
92 xlabel(['st 77 : ',num2str([max(tca) min(tca)])]);hold on
93 axis equal;axis off
94 for i = 1:nbo % ..... Drawing the border of the domain
95 if bor(i,4)==0
96 plot([xyz(bor(i,1),1) xyz(bor(i,2),1)], ...
97 [xyz(bor(i,1),2) xyz(bor(i,2),2)],'k','LineWidth',2)
98 end
99 end % ..... End drawing the border of the domain
100 title(['Elapsed time : ',num2str(it*dti/3600),' hours']);hold on
101 end
102 end % ..... End of time iterations
103 ddt = tmoy(nit+1)-min(tmi,tma);ah=ddt*cap;% Stor.heat: ah=DT*area*th*ro*Cp
104 disp(['st104, ddt Tm - Tin : ',num2str(ddt,3),' K'])
105 disp(['st105, ddt*sumsum(C) : ',num2str(ah ,3),' MJ'])
106 disp(['st106, Min obs. temp: ',num2str(ceil(min(tsmin)),4),' K'])
107 disp(['st107, Max obs. temp: ',num2str(max(tsmax),4),' K'])
108 if deb==1;disp('t.100, Call function: ..... gra_tev .....');end
109 gra_tev(nit,tt,0,tsmax,tsmin,tmoy,3); % tcav(1) = -1;% Temper. evol.
110 eih = tt*ih/pi*bos*1e-6; % Equation (89) of tutorial
111 % disp(['t.103, S&G Heat in : ',num2str(ga*nit,3),' MJ'])
112 if ih ~= 0;disp(['st108, Heat inp.(89): ',num2str(eih,3),' MJ']);end
113 if nfi > 0 % Results shown & displayed only in presence of fixations
114 if sum(gou) > 1.e-6
115 disp(['st115, Tot. reaction: ',num2str(sum(gou)*1.e-6,3),' MJ'])
116 figure('Position',[100 100 700 300]);ylabel('MJ','fontsize',15);
117 hold on;plot(gou(1:it)/1.e6);grid on;hold on
118 title(['Reaction on fixed DOF, sum(gou): ',...
119 num2str(sum(gou)/1.e6,3),' MJ'],'fontsize',15);hold on
120 end
121 end
122 end

```

Table 39: Matlab<sup>®</sup> function **fem\_smt.m** – solution of linear transient equations

Function <b>fem_smq.m</b> – solution of nonlinear transient equations
---

1	<b>function[tca] = ...</b>
---	----------------------------

```

2      fem_smq(np,xyz,lK,dK,nci,deb,rc,ra,cs,area,th,xyz_cao,gh,lf1,fT, ...
3      Tsky,nbo,bor,Kk,Lel,re,SBt,lcont,Ms,pai,Ftot,lon) % SBT = SB*th*(1-re)
4      tmi = 280;disp(['sq 04, Initial temp.: ',num2str(tmi), ' K'])
5      disp(['sq 05, Fixed DOF : ',num2str(lfi)])
6      nfi = size(fT,2);ntca=dK-nfi;tcan = [ones(ntca,1)*tmi ; fT'];tca = tcan;
7      g = zeros(ntca,1);Mn = zeros(size(lcont,1),1); % 2d Memb. sid. to nod.
8      nit = 720;dth=1;dtd=nit/2; % Numb. iter. & delta tau per it. (hours)
9      dti = dth*3600;disp(['sq 09, Time step : ',num2str(dti), ' s'])
10     tt = dti*nit; % Analyzed period in seconds
11     disp(['sq 11, Analyzed per.: ',num2str(tt/3600), ' h ','...'])
12     num2str(tt/3600/24), ' days'])
13     nd = tt/3600/24;f=zeros(1,tt/3600); % nd = number of days
14     for i = 1:nd % f is the imposed periodic function, f(h = 1:12)
15       f((i-1)*24+1:(i-1)*24+12) = sin((1:12)*pi/12);
16     end
17     Cp = 1000 ;disp(['sq 17, Spec. capac. : ',num2str(Cp), ' J/(kg.K)']);
18     ro = 2500 ;disp(['sq 18, Spec. mass : ',num2str(ro), ' kg.m-3']);
19     C = zeros(dK-1,dK-1); % Initialization of the global capacity matrix
20     if deb==1;disp('sq 20, Call function: ..... fem_Cae .....');end
21     for n = 1:size(lK,1) % Glob. capacity mat. assemb., loop on nel elem.
22       Cae = fem_Cae(xyz,lK(n,:))*th*Cp*ro; % Cae = element capacity matrix
23       for i = 1:4
24         for j=1:4;C(lK(n,i),lK(n,j)) = C(lK(n,i),lK(n,j)) + Cae(i,j);end
25       end
26     end % End of capacity matrices assembling
27     cap = area*th*Cp*ro*1e-6; % Domain capacity computed from mat. data
28     disp(['sq 28, sum(sum(C)) : ',num2str(sum(sum(C))*1e-6), ' MJ/K']);
29     disp(['sq 29, area*th*ro*Cp: ',num2str(cap), ' MJ/K']);
30     tsmax=ones(1,nit+1)*tmi;tsmin=tsmax;t moy=tsmax;gou=t moy;
31     ih = 0;% 4.04008;% 0;% Imposed heat load in Wm-2
32     disp(['sq 32, Impos. Heat : ',num2str(ih), ' W/m-2']);
33     bos = th*(max(xyz(:,1))-min(xyz(:,1)));% Cross section area upper edge
34     if cs == 2 % Area of the top of the street canyon
35       bos = (xyz_cao(2,1)+xyz_cao(7,1)-xyz_cao(1,1)-xyz_cao(8,1))*th;
36       disp(['sq 35, Loaded area : ',num2str(bos), ' m2']);
37     end % lcont = list of radiative nodes
38     K = Kk;mcr = size(lcont,1)-1;lc = zeros(mcr,3);nv = size(Ftot,2);
39     disp(['sq 38, mcr nv : ',num2str([mcr nv])]);
40     % for i = 1:mcr;lc(i,1)=lcont(i,1);lc(i,2)=lcont(i+1,1);lc(i,3)=dK-1;end
41     if rc*cs>1 % Rad. present in street section (3 sides), on balc. (5 sides)
42       nu = size(Lel,1);k = 0; % nu = Number radiatives patch sides
43       for j = 1:nu;for i = 1:nci;k = k+1; lon(k) = Lel(j)/nci;end;end
44       disp(['sq 44, N. rad. edges: ',num2str(nu)]);
45       disp(['sq 45, rad seg leng.: ',num2str(Lel),' m']);% fT(2)=280;nfi=2;
46       disp(['sq 46, N. rad. elem.: ',num2str(size(lon,2))]);
47     if size(lon,2)<11;disp(['sq 48, rad sid leng.: ',num2str(lon),' m']);end
48     if ra==0 % Generation of conductive radiative element matrices
49       for i = 1:mcr % lc = loc. matrix of rad. elem.
50         lc(i,1) = lcont(i,1);lc(i,2) = lcont(i+1,1);lc(i,3) = dK-1;
51         Ker = fem_Kcr(xyz,lc(i,:),SBt,tcan); % Elem. rad. mat.
52         for ii = 1:3
53           for jj = 1:3
54             K(lc(i,ii),lc(i,jj))=K(lc(i,ii),lc(i,jj))+Ker(ii,jj);
55           end
56         end
57       end % End assembling the conductive radiative element matrices
58     end
59     if ra == 1 % Computation of K due to inter-element view factors
60       gr=zeros(1,dK-1);
61       [K,Qs,Qg] = fem_Kra(Kk,lcont,re,Ftot,SBt,tcan,Tsky,lon);
62       gr(lcont) = (Qs' + Qg'); % nodal loads issued by sky & ground
63       disp(['sq 63, sum(gr) : ',num2str(sum(gr),3), ' W']);
64       if deb==1
65         disp('sq 65, Call function: ..... fem_Kra .....');
66         if size(lon,2)<11
67           disp(['sq 67, gr : ',num2str(gr(lcont),3), ' W']);
68           disp(['sq 68, lcont : ',num2str(lcont)]);
69           disp(['sq 69, mean(tcan) : ',num2str(mean(tcan)), ' K']);
70           % tcant=tcan';disp(['t. 70, tcan : ',num2str(tcant), ' K']);
71         end
72       end
73       for i = 1:size(lcont,1)-1
74         Mn(i,1) = Mn(i,1)+Ms(i,1)/2;Mn(i+1,1) = Mn(i+1,1)+Ms(i,1)/2;
75       end % disp(['t. 81, Mn : ',num2str(Mn',3), ' W']);
76       disp(['sq 77, sum(Mn) : ',num2str(sum(Mn),3), ' W']);
77     end
78   end
79   ip = 0. ;
80   if deb ==1;disp('sq 97, Call function: ..... fem_Kra .....');end
81   for it = 1:nit % ..... Loop on the time iterations
82     ip = ip+1;
83     if ra == 0
84       if rc == 1 % Add radiative matrix Kr to conductive matrix K

```

```

85      for ir = 1:mcr
86          Kr = fem_Kcr(xyz,lc(ir,:),SBt,tcan);
87          for i=1:3
88              for j = 1:3
89                  K(lc(ir,i),lc(ir,j))=K(lc(ir,i),lc(ir,j))+Kr(i,j);
90              end
91          end
92      end
93  end
94
95  if ra == 1           % Computation of K due to inter element view factors
96      [K,Qs,Qg] = fem_Kra(Kk,lcont,re,Ftot,SBt,tcan,Tsky,lon);
97      g(lcont) = Qs' + Qg'*Mn;
98  end % Injected heat = weights * periodic function f(t)* load * area
99  if cs < 3; g = gh * f(it) * ih * bos; end % Injected heat
100 if nfi == 0           % The domain does not contain fixations
101     tca = (C + dti*K)\(C*tcan + dti*(g));    % Equation 68
102 else
103     gr = zeros(ntca,1);   % Sequence t 114 - t 115 equiv to fem_tra
104     for m = 1:size(lcont,1)
105         gr(lcont(m,1)) = gr(lcont(m,1))+Mn(m,1);
106     end
107     K11 = K(1:ntca,1:ntca); K12= K(1:ntca,ntca+1:dK);
108     CC = C(1:ntca,1:ntca); A = (CC + dti*K11);
109     tcb = A\((CC*tcan(1:ntca,1)-dti*(K12*fT'+ gr));
110     tca = [tcb ; fT' ];
111     go = K * tca;           % Second member of the system
112     gou(it) = sum(go(lfi))*dti;    % Outgoing heat at iteration it
113 end
114 tmax(it+1) = max (tca(1:dK-nfi)); % min(300,max (tca(1:dK-nfi))); %
115 tmin(it+1) = min (tca(1:dK-nfi)); % max(270,min (tca(1:dK-nfi))); %
116 tmoy (it+1) = mean(tca(1:dK-nfi)); % (tsmin(it+1)+tsmax(it+1))/2; %
117 tcan = tca;
118 if ip == dtd % In this iteration, isotherms drawing is generated
119     tmin = min(tca);tmax = max(tca);
120     gt = [tmin pai tmax];ip = 0;
121     disp(['sq122, Iteration : ',num2str(it),' gt: ',num2str(gt),' K']);
122     if deb==1;disp ('sq 129, Call function: .... gra_ipa .....');end;
123     figure
124     ori = min(xyz(:,1))-2;
125     a = ori-.2;b = ori-.1;ha=max(xyz(:,2)); %.... drawing a left bar
126     fill([a b b a],[0 0 ha ha],[280 280 300 300]);hold on;
127     for i = 1 : np % .... Loop on the np CAD patches
128         gra_ipa(nci,nci,1K((i-1)*nci^2+1:i*nci^2,:),tca',xyz,gt);
129         colorbar;hold on
130     end
131     xlabel(['sq139 : ',num2str([max(tca) min(tca)])]);hold on
132     axis equal;axis off
133     for i = 1:nbo % .... Drawing the border of the domain
134         if bor(i,4)==0
135             plot([xyz(bor(i,1),1) xyz(bor(i,2),1)], ...
136                 [xyz(bor(i,1),2) xyz(bor(i,2),2)],'k','LineWidth',2)
137         end
138     end % .... End drawing the border of the domain
139     title(['Elapsed time : ',num2str(it*dti/3600),' hours']);hold on
140 end
141 end % ..... End of time iterations
142 ddt = abs(tmoy(nit+1)-tmi);ah = ddt*cap; % DT *th*area *Cp*ro
143 disp(['sq143, Tmean - Tini : ',num2str(ddt,3),' K'])
144 disp(['sq144, Stored heat : ',num2str(ah ,3),' MJ'])
145 if ddt > .1
146     disp(['sq146, Min it+1 temp: ',num2str(tsmin(it+1),4),' K'])
147     disp(['sq147, Max it+1 temp: ',num2str(tmoy (it+1),3),' K'])
148     disp(['sq148, Max it+1 temp: ',num2str(tsmax(it+1),3),' K'])
149     if deb==1;disp ('r.150, Call function: .... gra_tev .....');end
150     gra_tev(nit,tt,re,tsmax,tsmin,tmoy,4); % Temperature evolution
151 end
152 if nfi > 0 % Results shown & displayed only in presence of fixations
153     if sum(gou) > 0
154         disp(['sq154, Ejected heat : ',num2str(sum(gou)*1.e-6,3),' MJ'])
155     end
156 end
157 end

```

Table 40: Matlab<sup>®</sup> function *fem\_smq.m* – solution of nonlinear transient equations

Function <i>fem_smc.m</i> - cavity, VF matrices, radiative transfers	
1	function[tca] = ...
2	fem_smc(np,xyz,1K,dK,nci,deb,rc,cs,area,gh,lg,...
3	fT,lfi,nbo,bor,Kk,mcr,Lel,SB,re,Ms,Mpr,lcont,pai,th,ca,bos)

```

4 tmi=300;nel=size(lk,1);disp(['sc 04, Initial temp.: ',num2str(tmi) , ' K'])
5 if lfi(1)== 0;nfi=0;else;nfi=size(lfi,2);end;
6 disp(['sc 06, Numb. fixat. : ',num2str(nfi)]);
7 % ng=size(lg,2);disp(['Lt 07, N. load. nod.: ',num2str(ng)]);
8 tcan = ones(dK,1)*tmi;if nfi>0;tcan(lfi)=ft;end % Initial conditions
9 nit = 720;dth=1;dtd=nit/4; % Numb. iter. & delta tau per it. (h)
10 dti = dth*3600;disp(['sc 10, Time step : ',num2str(dti), ' sec'])
11 tt = dti*nit; % Analyzed period in seconds
12 disp(['sc 12, Analyzed per.: ',num2str(tt/3600) , ' h , ...
13 num2str(tt/3600/24) , ' days'])
14 nd = tt/3600/24;f=zeros(1,tt/3600); % nd = number of days
15 for i = 1:nd % f is the imposed periodic function, f(h = 1:12)
16 f((i-1)*24+1:(i-1)*24+12) = sin((1:12)*pi/12);
17 end
18 Cp = 1000 ;disp(['sc 18, Spec. capac. : ',num2str(Cp), ' J/(kg.K)']);
19 ro = 2500 ;disp(['sc 19, Spec. mass : ',num2str(ro), ' kg.m-3']);
20 C = zeros(dK,dK); % Initialization of the global capacity matrix
21 % C(ndK,ndK) = 1; % Capacity of the fluid virtual node Jkg-1K-1
22 if deb==1;disp('sc 22, Call function: ..... fem_Cae .....');end
23 for n = 1:nel % Global capacity mat. assemb., loop on nel elements
24 Cae = fem_Cae(xyz,lK(n,:))*th*Cp*ro; % Element capacity matrix
25 for i = 1:4 % if n == 1;disp(Cae);end
26 for j=1:4;C(lK(n,i),lK(n,j)) = C(lK(n,i),lK(n,j)) + Cae(i,j);end
27 end
28 end % End of capacity matrices assembling
29 cap = area*th*Cp*ro*1e-6; % Domain capacity comp. from mat. data
30 disp(['sc 30, Solid heat C.: ',num2str(sum(sum(C))*1e-6), ' MJ/K'])
31 disp(['sc 31, area*th*ro*Cp: ',num2str(cap), ' MJ/K']);%disp(C)
32 ze = ones(1,nit+1)*tmi;tsmax=ze;tsmin=ze;tmoy=ze;% tcav=ze;
33 ga = zeros(1,nit+1);gou = zeros(1,nit+1);
34 % tca = zeros(ndK,1);% gtot = 0;
35 ih = 0;% 0;% 4.04008;% % Imposed heat load in Wm-2
36 disp(['sc 36, Imposed Heat : ',num2str(ih), ' Wm-2'])
37 % gt = [tmi pai tma]; % gt = [min(tcan) pai max(tcan)]
38 nc2 = nci*nci; % Number of elements in a CAD patch
39 % if cs< 2;bos=th*xext;end % Cross sect. area of upper edge, street canyon
40 %if cs==2;bos=(xyz_cao(2,1)+xyz_cao(7,1)-xyz_cao(1,1)-xyz_cao(8,1))*th;end
41 K = Kk; ip=0; % Conduct. matrix related to solid and convective part
42 if ca > 0 % Valid only for the rectangular cavity
43 lon = [ones(nci,1)*Lel(1)/nci;ones(nci,1)*Lel(2)/nci;...
44 ones(nci,1)*Lel(3)/nci;ones(nci,1)*Lel(4)/nci];
45 end
46 if rc*cs == 2 % Radiation in the street section (3 sides)
47 nne = (mcr-1)/3; % nne = numb segm per street side
48 for i = 1:nne % Computing the element lengths on the 3 sides
49 lon(i) = Lel(1)/(nne); lon(mcr-i) = Lel(3)/(nne);
50 lon(i+nne) = Lel(2)/(nne);
51 end
52 end % It is important to observe that: sum(esm) = sum(sn)
53 if rc*cs == 3 % Radiation around the thermal bridge - 5 segments
54 n = 0; % Segment lengths
55 for k = 1:5;for i= 1:nci;n = n+1;lon(n) = Lel(k)/(nci); end;end;
56 end % It is important to observe that: sum(esm) = sum(sn)
57 % ..... Loop on the time iterations
58 Mn = zeros(dK,1);% bos = max(xyz(:,1))-min(xyz(:,1));
59 disp(['sc 59, size(K) nfi : ',num2str([size(K) nfi] )])
60 if deb==1;disp('sc 65, Call function: ..... fem_rsm .....');end
61 % Loop on the nit iterations steps 62 -- 116 =====
62 for it = 1:nit % Loop on time iterations step
63 if rc == 1 % Add radiative matrix Kr to conductive matrix Kk
64 % disp(['sc 64, Le. rad. sid.: ',num2str(lon)])
65 [Kr,Mss] = fem_rsm(tcan,SB*th,re,Ms,Mpr,it,lcont,lon,ca,nit);
66 for i= 1:mcr
67 Mn(lcont(i)) = - Mss(i);
68 for j = 1:mcr
69 K(lcont(i),lcont(j)) = Kk(lcont(i),lcont(j)) + Kr(i,j);
70 end
71 end
72 end % Injected heat = weights * periodic function f(t)* load * area
73 g = gh * f(it) * ih * bos; % Injected heat
74 if lg(1)>0;ga(it+1)=sum(g(lg))*dti;end % Heat input at step it
75 ip = ip+1;
76 if nfi == 0 % The problem does not include fixations
77 tca = (C + dti*K)\(C*tcan + dti*(g + Mn)); % Equation 69
78 else % Domain including some fixations
79 tca = fem_tra(K,C,dti,(g + Mn),lfi,tcan);
80 if it == 1
81 if deb ==1
82 disp ('sc 82, Call function: ..... fem_tra .....')
83 end
84 end
85 go = K * tca; % Second member of the system
86 gou(it) = sum(go(lfi(1:nfi))*dti*1e-6);% Reactions at iteration it

```

```

87 % tcav (it+1) = tca (ndK,1)
88 tsmax(it+1) = max (tca(1:dK-nfi));
89 tsmin(it+1) = min (tca(1:dK-nfi));
90 tmoy (it+1) = mean(tca(1:dK-nfi)); % (tsmin(it+1)+tsmax(it+1))/2;
91 tcan = tca;
92 if ip == dtd % In this iteration, isotherms drawing is generated
93     pai = min((max(tca)-min(tca))/10,pai);
94     ip = 0;gt = [min(tca) pai max(tca)];%gt=[290 pai 300]; %
95     disp(['sc 95, iteration : ',num2str(it)])
96     if deb==1;disp(['sc 96, uniline gt : ',num2str(gt)]);end;
97     if deb==1;disp(['sc 97, size(tca) : ',num2str(size(tca))]);end;
98     if deb==1;disp ('sc 98, Call function: ..... gra_ipa .....');end;
99     figure
100    % a=-.2;b=-.1;ha=max(xyz(:,2));ti=281;ta=297;%...drawing a left bar
101    a=-.2;b=-.1;ha=max(xyz(:,2));ti=287;ta=308;%....drawing a left bar
102    fill([a b b a],[0 0 ha ha],[ti ti ta ta]);hold on;
103    for i = 1 : np % .....
104        gra_ipa(nci,nci,lK((i-1)*nc2+1:i*nc2,:),tca',xyz,gt);
105        colorbar;hold on
106    end
107    xlabel(['sc107 : ',num2str([max(tca) min(tca)])]);hold on
108    axis equal;axis off
109    for i = 1:nbo % .....
110        if bor(i,4)==0 % Drawing the border of the domain
111            plot([xyz(bor(i,1),1) xyz(bor(i,2),1)], ...
112                  [xyz(bor(i,1),2) xyz(bor(i,2),2)],'k','LineWidth',2)
113        end
114    end % .....
115    title(['Elapsed time : ',num2str(it*dti/3600),' hours']);hold on
116 end
117 end % .....
118 ddt = tmoy(nit+1)-tmi;ah = ddt*cap; % Stored heat: ah=DT*area*th*ro*Cp
119 disp(['sc119, DTm Tm - Tini: ',num2str(ddt,3),' K'])
120 disp(['sc120, Min obs. temp: ',num2str(min(tsmin),3),' K'])
121 disp(['sc121, Max obs. temp: ',num2str(max(tsmax),3),' K'])
122 disp(['sc122, Final temper.: ',num2str([tsmin(nit+1) tmoy(nit+1)...
123 tsmax(nit+1)],3)])
124 disp(['sc124, Capacity* DTm: ',num2str(ah ,3),' MJ'])
125 if deb==1;disp('Lc125, Call function: ..... gra_tev .....');end
126 gra_tev(nit,tt,re,tsmax,tsmin,tmoy,5) % Drawing temp. evolution
127 if deb == 1;disp('sc127, Call function: ..... gra_hie .....');end
128 if ih > 0
129     gra_hie(nit,tt,ga)
130     hdm = tt*ih/pi*bos*1e-6; % Explicit exact injected heat Equ. (80) pp 49
131     disp(['sc131, Injected heat: ',num2str(sum(ga)/1.e06,3),' MJ'])
132     disp(['sc132, Ex.heat input: ',num2str(hdm,3),' MJ'])
133 end % Drawing heat input evolution
134 if nfi > 0 % Results shown & displayed only in presence of fixations
135     disp(['sc135, Ejected heat : ',num2str(sum(gou),3),' MJ'])
136     figure('Position',[100 100 700 300]);
137     plot(gou(1:it));grid on;hold on
138     title(['sc138 - Ejected heat: ',num2str(sum(gou),4),' MJ'],...
139           'fontsize',15);hold on
140     ylabel('Ejected heat (J)', 'fontsize',15);hold on
141 end
142 end

```

Table 41: Matlab<sup>®</sup> function *fem\_sm.m* – Cavity, VF matrices, radiative transfers

			Name	Meaning of the variables used in the function <i>fem_sm.m</i>
input	2	2	Line	Occ.
input	2	8	xyz	Matrix of the 3D nodal coordinates expressed in, <i>m</i>
input	2	8	lK	Localization matrix of conductive elem. (dimension: <i>nel</i> x 4)
input	2	9	dK	Number of <i>DOF</i>
input	2	12	nci	Number of elements per patch edge
input	2	8	deb	Flag enabling the display of function call (debugging)
input	2	4	rc	Flag indicating for radiative exchanges (1 = yes, 0 = no)
input	2	5	cs	Flag for view factor matrix: 1 = cavity, 2 = street, 3 = balcony
input	2	3	area	Area of the solid domain <i>m</i> <sup>2</sup>

<b>input</b>	2	2	gh	Uni-column matrix of flow input distribution ( <i>cad_Neu.m</i> )
<b>input</b>	3	3	fT	If $nfi > 0$ , uni-column matrix of fixed nodal temperatures <i>K</i>
<b>input</b>	3	5	lfi	List of fixed nodes on a side (Dirichlet), ( <i>cad_Dir.m</i> )
<b>input</b>	3	2	nbo	Number of CAD patches interfaces
<b>input</b>	3	6	bor	matrices <i>bor</i> and <i>pbo</i> computed in <i>cad_mes.m</i> ( <i>Table 35</i> ).
<b>Input</b>	3	3	Kk	Global conductivity matrix of meshed solid domain <i>WK<sup>-1</sup></i>
<b>input</b>	3	3	mcr	Number of radiative nodes, <i>size(lcont,1)</i>
<b>input</b>	3	6	Lel	Uni-column matrix of *cavity or street section edges lengths <i>m</i>
<b>input</b>	3	3	SB	Stefan-Boltzmann constant: $5.6704 \cdot 10^{-8} \text{ Wm}^{-2}\text{K}^{-4}$
<b>input</b>	3	5	re	Coefficient of reflexion (adimensional)
<b>input</b>	3	3	Mpr	$(I - F) M^T$ matrix for flow outcoming from a segment (adim.)
<b>input</b>	3	3	Ms	Matrix related to radiative exchanges (see equation 104)
<b>input</b>	3	9	lcont	List of radiative boundary nodes ( <i>cad_ban.m</i> )
<b>input</b>	3	2	pai	Temperature interval in isotherms drawing <i>K</i>
<b>input</b>	3	6	th	Thickness, <i>m</i>
<b>input</b>	3	2	ca	Flag for presence of cavity
<b>1</b>	4	5	tmi	Initial temperature (scalar expressed in <i>K</i> )
<b>2</b>	5	10	nfi	Number of fixed nodes
<b>3</b>	4	2	nel	Number of conductive elements
<b>4</b>	7	9	tcan	Nodal temperatures of the solid before starting the iterations
<b>5</b>	8	13	nit	Number of iterations for transient analysis
<b>4</b>	6	2	dtd	Iteration leading to a drawing
<b>5</b>	7	8	dti	Time step in seconds <i>s</i>
<b>6</b>	8	7	tt	Analyzed period <i>s</i>
<b>7</b>	11	2	nd	Analyzed period in days
<b>8</b>	11	3	f	Periodic function expressed in days
<b>9</b>	15	5	c <sub>p</sub>	Specific capacity <i>Jkg<sup>-1</sup>K<sup>-1</sup></i>
<b>10</b>	16	5	ro	Specific mass <i>kgm<sup>-3</sup></i>
<b>11</b>	17	7	C	Global capacity matrix in <i>JK<sup>-1</sup></i>
<b>12</b>	21	2	Cae	Element capacity matrix <i>JK<sup>-1</sup></i> ( <i>fem_Cae.m</i> )
<b>13</b>	26	2	cap	Domain heat storage capacity <i>JK<sup>-1</sup></i>
<b>14</b>	28	5	tmoy	Uni-line vector, mean temperature in solid at each time step <i>K</i>
<b>15</b>	28	4	tsmin	Uni-line matrix, lowest temp. in solid at each time step <i>K</i>
<b>16</b>	28	4	tsmax	Uni-line matrix, highest temp. in solid at each time step <i>K</i>
<b>17</b>	29	4	ga	Uni-line matrix, heat input at each step <i>sum(g)*dti</i> , in <i>J</i>

18	29	16	tca	Uni-column matrix, output of unknown nodal temperatures $K$
19	29	6	gou	Uni-line matrix, outgoing heat at each step
20	31	6	ih	Module of imposed periodic heat flow $Wm^{-2}$
21	33	5	bos	Loaded area, $m^2$
22	37	5	ip	Counter of the iterations in transient analysis
23	42	8	lon	Uni-line matrix of radiative border element lengths $m$
24	49	7	esm	Element second member linked to sky & ground radiations $W$
25	53	6	sn	Nodal second member linked to sky & ground radiations $W$
26	61	12	it	Time iterations index (end of loop at line 104)
27	66	2	Kr	Radiative matrix added to the conductive one
28	64	4	K	Global conductivity matrix

Table 42: Matlab<sup>®</sup> variables used in the function *fem\_smc.m*

Matlab <sup>®</sup> function <i>fem_Cae.m</i> – element capacity matrix	
1	function [C] = fem_Cae(xyz,lo) % Capacity matrix of a quadrilateral
2	Q = [xyz(lo(1),1:3); xyz(lo(2),1:3); xyz(lo(3),1:3); xyz(lo(4),1:3)];
3	s = [.5-sqrt(3)/6 .5+sqrt(3)/6 .5+sqrt(3)/6 .5-sqrt(3)/6]; % 4 Gauss pts
4	t = [.5-sqrt(3)/6 .5-sqrt(3)/6 .5+sqrt(3)/6 .5+sqrt(3)/6]; % 4 Gauss pts
5	C = zeros(4,4); % area = 0.;
6	for i=1:4 % Loop on the 4 Gauss points
7	f = [(1-s(i))*(1-t(i)) s(i)*(1-t(i)) s(i)*t(i) (1-s(i))*t(i)];
8	fs = [-(1-t(i)) (1-t(i)) t(i) -t(i)]; % Derivative s
9	ft = [-(1-s(i)) -s(i) s(i) (1-s(i))]; % Derivative t
10	ds = fs * Q;
11	dt = ft * Q;
12	C = C + f'*f* sqrt(dot(cross(ds,dt),cross(ds,dt)))/4;
13	end
14	end

Table 43: Matlab<sup>®</sup> function *fem\_Cae.m* – capacity matrix of a quadrilateral

Matlab <sup>®</sup> function <i>fem_Kco.m</i> – conductivity matrix of isoparametric element	
1	function [K] = fem_Kco(xyz,lo) % Conductivity matrix K, 3D surf. 20211001
2	Q = [xyz(lo(1),: ); xyz(lo(2),: ); xyz(lo(3),: ); xyz(lo(4),: )];
3	s = [.5-sqrt(3)/6 .5+sqrt(3)/6 .5+sqrt(3)/6 .5-sqrt(3)/6]; % s 4 Gauss pts
4	t = [.5-sqrt(3)/6 .5-sqrt(3)/6 .5+sqrt(3)/6 .5+sqrt(3)/6]; % t 4 Gauss pts
5	K = zeros(4,4); % area = 0.;
6	for i=1:4 % Loop on the 4 Gauss points
7	fs = [-(1-t(i)) (1-t(i)) t(i) -t(i)]; % Derivative s
8	ft = [-(1-s(i)) -s(i) s(i) (1-s(i))]; % Derivative t
9	gra = [fs;ft]; % Gradient of the scalar bilinear function
10	ds = fs * Q; % Differential in the s direction
11	dt = ft * Q; % Differential in the t direction
12	% area = area + sqrt(dot(cross(ds,dt),cross(ds,dt)))/4;
13	J = [fs*Q(:,1) fs*Q(:,2);ft*Q(:,1) ft*Q(:,2)]; % Jacobian matrix
14	K=K+((J^(-1)*gra) * J^(-1)*gra)*sqrt(dot(cross(ds,dt),cross(ds,dt)))/4;
15	end
16	% disp(['Patch area : ',num2str(area)])
17	end % Multiplied by k and the thickness, the K matrix is adimensional

Table 44: Matlab<sup>®</sup> function *fem\_Kco.m* – element conduction matrix

Matlab <sup>®</sup> function <i>fem_Kcv.m</i> – conduction-convection matrix	
1	function [K] = fem_Kcv(xyz,lc,hh) % Convection matrix on a line segment
2	Q = [xyz(lc(1),1:3); xyz(lc(2),1:3)];
3	L = norm(Q(2,: )-Q(1,: ));
4	K = [2 1 -3;1 2 -3;-3 -3 6]*hh*L/6;
5	end

Table 45: Matlab<sup>®</sup> function *fem\_Kcv.m* – 3 x 3 element convection matrix

As a convective element, a radiative element may have any orientation. Its length is  $L$ , its thickness  $e$ , and the Stefan-Boltzmann coefficient is  $\sigma$  ( $Wm^{-2}K^{-4}$ ). The node sequence of an

element starts with the two real nodes pertaining to the mesh and finishes with the virtual one. The function *fem\_Kcr.m* (*Table 46*) computes the radiative matrix of an element as pseudo convection matrices. The third argument *SBt* of the function is the product of the Stefan-Boltzmann constant, the thickness and the emissivity ( $\epsilon = 1 - \rho$ ). The temperatures are stored in the vector *tca* (argument 4). The *xyz* matrix contains the nodal coordinates and *lc* is the element localization matrix. Because radiative boundary conditions lead to a nonlinear system of equation, a new Matlab<sup>©</sup> function *fem\_smd.m* is needed to solve the system (*Table 38*).

Matlab <sup>©</sup> function <i>fem_Kcr.m</i> – radiative boundary element matrix	
1	function [K] = fem_Kcr(xyz,lc,SBt,tca)%K is integrated on the bound. Segm. 2 Q = [xyz(lc(1),1:3); xyz(lc(2),1:3)]; % Vector of element extremities 3 L = norm(Q(2,:)-Q(1,:)); % Length of the element 4 T1 = (tca(lc(1))+tca(lc(2)))/2; 5 tv = tca(lc(3)); 6 co = (T1^2+tv^2)*(T1+tv)*L*SBt; 7 K = [2 1 -3;1 2 -3;-3 -3 6]*co/6; 8 end

Table 46: Matlab<sup>©</sup> function *fem\_Kcr.m* – 3 x 3 element radiation matrix

Matlab <sup>©</sup> function <i>fem_Kra.m</i> – inter elements view factors	
1	function[K,Qs,Qg] = fem_Kra(Kk,lcont,re,Ftot,SBt,tcan,Tsky,lon) % 20211118 2 K = Kk; 3 nd = size(lcont,1)-1; % nd = number radiative nodes 4 M = eye(nd)-re*Ftot(1:nd,1:nd); % Radiosity matrix 5 % S = eye(nd)*SBt*(1-re).*lon(1:nd); 6 S = eye(nd)*SBt.*lon(1:nd); 7 tb = zeros(nd,1); 8 for I = 1:nd; tb(I) = (tcan(lcont(i))+tcan(lcont(i+1)))/2; end 9 for I = 1:nd; S(I,i)= S(I,i)*tb(i)^3 ; end 10 KI = (eye(nd)-Ftot(1:nd,1:nd))*M^(-1)*S; 11 N = zeros(nd,nd+1);for I = 1:nd;N(I,i)=.5;N(I,i+1)=.5; end% edg to nod 12 KJ = N'*KI*N; 13 for I = 1:nd 14 for j = 1:nd;K(lcont(i),lcont(j))=K(lcont(i),lcont(j))+KJ(I,j);end 15 end 16 Qs = (Ftot(1:nd,1:nd)*M^(-1)*Ftot(1:nd,nd+1)*SBt*Tsky^4)'*N; 17 Qg = (Ftot(1:nd,1:nd)*M^(-1)*Ftot(1:nd,nd+2)*SBt*Tsky^4)'*N; 18 end

Table 47: Matlab<sup>©</sup> function *fem\_Kra.m* – Inter elements view factors

Matlab <sup>©</sup> function <i>fem_rsm.m</i> second member in a radiative section	
1	function[Mss] = fem_rsm(tca,SBt,re,Ms,Mpr,it,lcont,lon,ca,nit,deb) 2 mcr = max(size(lcont)); % Number of radiative nodes 3 cat = tca(lcont); % Nodal temperatures of radiative nodes 4 SBte = SBt*(1-re);if re > .999;SBte = 0;end;% Kr = zeros(mcr,mcr); % re = 1, mirror 5 if SBte == 0 6 MSS = zeros(mcr,1); 7 else % Emissivity is positive on the concerned boundary 8 if ca == 0 % if ca == 0: Street or open zone 9 si = mcr-1;Lo = zeros(si,mcr); % si = number of radiative sides 10 for i = 1:si;for j = i:i;Lo(i,j)=.5;Lo(i,j+1)=.5;end;end; 11 cam = zeros(si,1); % Initialization 12 cam(1:si,1) = (cat(1:si,1)+cat(2:mcr,1))/2; % Mid-segm. temp. 13 MSS = Lo'*Ms; 14 else % if ca ne 0: cavity or closed zone 15 Lc = zeros(mcr,mcr);Lc(mcr,mcr) = .5;Lc(mcr,1) = .5; 16 for i = 1:mcr-1; Lc(i,i)=.5; Lc(i,i+1)=.5;end ; 17 cam = (Lc*cat); 18 MSS = (eye(mcr).*lon) '*SBte*Lc'*Mpr*eye(mcr)*cam.^4; 19 end 20 end 21 if it ==nit/nit*10 22 if mcr < 10;disp(['rs 22, iteration : ',num2str(it)]) ;end 23 if mcr < 10;disp(['rs 23, lcont : ',num2str(lcont)]) ;end 24 if mcr<10;disp(['rs 25, tca(lcont) : ',num2str(tca(lcont)',5),' K']);end 25 if mcr < 10;disp(['rs 25, T. cotés cam : ',num2str(cam',5),' K']);end 26 if mcr < 10;disp(['rs 26, 2m. noeul MSS : ',num2str(Mss',3),' W']);end 27 if mcr < 10;disp(['rs 27, cam.^4 : ',num2str((SBte*cam.^4)',... ' m']));end 28 if mcr < 10;disp(['rs 29, lon : ',num2str(lon)', ' m']);end

```

30     if deb==1;disp('rs 31, Call function: ..... gra_tra .....');end
31     gra_tra(tca,lcont,re,ca);
32   end
33 end

```

Table 48: Matlab<sup>©</sup> function *fem\_rsm.m* – second member in a radiative section

The next function contains the method presented in § 1.2 to solve the linear transient equation expressed in equation (68).

In § 1.2 Matlab <sup>©</sup> function <i>fem_tra.m</i> – linear transient problem
<pre> 1  function [tca] = fem_tra(K,C,dti,g,lfi,tcan) 2    dK = size(K,1);                                     % Size of matrices K and C 3    nfi = size(lfi,2);                                % Number of fixed DOF 4    N = zeros(nfi,dK);for i=1:nfi;N(i,lfi(i))=1;end 5    A = [C+dti*K N';N zeros(nfi,nfi)]; 6    G = [C*tcan+g ; tcan(lfi)];B = A\G; tca = B(1:dK); 7  end </pre>

Table 49: Matlab<sup>©</sup> function *fem\_tra.m* – solution of the linear transient equations

## 8.4 Postprocessing

After running *Fiammetta.m*, it is possible to process the results throughout specific sequences of instructions and *ad hoc* Matlab<sup>©</sup> functions.

### 1. CAD patches and labels (see Table 53)

```

gra_mnl(xyz_cao,car_cao,[0 0 0],15);axis equal;axis off % Drawing CAD
elem.
title({'CAD elements & labels',' '})                           % End CAD drawing

```

### 2. Temperature gradient element by element (see Table 56)

```

figure           % Drawing the temperature gradients in the meshed domain
gra_atg(xyz,lK,tca);gra_mel(xyz,lK,0,.8);axis equal;axis off
for i = 1:nbo               % Drawing the border of the domain
  if bor(i,4)==0
    plot([xyz(bor(i,1),1) xyz(bor(i,2),1)], ...
          [xyz(bor(i,1),2) xyz(bor(i,2),2)],'k','LineWidth',2)
  end
end                 % End drawing temperature gradient and domain border

```

In the function *gra\_atg.m* (Table 56), the temperature gradient is computed in the barycenter of the elements and drawn as a blue arrow oriented as the gradient, its length being proportional to the value of the gradient module. It is convenient to call this function only for meshes that do not involve too many elements. The function is also displaying the maximum and the average of the gradient modules.

### 3. Heat flow element by element (see Table 57)

```

figure           % Drawing the element heat flows in the meshed domain
gra_ahf(xyz,lK,tca,co); gra_mel(xyz,lK,0,0);axis equal; hold on
for i = 1:nbo               % Drawing the border of the domain
  if bor(i,4)==0
    plot([xyz(bor(i,1),1) xyz(bor(i,2),1)], ...
          [xyz(bor(i,1),2) xyz(bor(i,2),2)],'k','LineWidth',2);hold on
  end
ylabel(['re: ',num2str(re)]);axis off;hold on
end                 % End drawing element heat flows and domain border

```

### 4. Node and element labels (see Table 53)

```

gra_mnl(xyt,lK,[0 0 0],15);axis equal;axis off      % Drawing node & el. labels
title({'Finite element mesh & labels',' '})% End N. & el. labels drawing

```

## 5. Displaying nodal temperatures (see *Table 52*)

```
figure; % Displaying nodal temperatures
gra_mel(xyz,lK,1,.9);axis equal;axis off;
for i=1:no;text(xyz(i,1),xyz(i,2),num2str(tca(i),4));hold on;end%or tcan
title({'Nodal temperatures',' '}) % End temperatures display
```

## 6. Displaying nodal second members (see *Table 52*)

```
figure; % Displaying nodal heat loads
gra_mel(xyz,lK,1,.9);axis equal;axis off;gk=round(10*Kk*tca)/10;
for i=1:no
    if gk(i) ~= 0;text(xyz(i,1),xyz(i,2),num2str(gk(i),3));hold on;end
end
title({'Nodal heat loads: gk = Kk*tca (Watt)',' '}) % End heat draw
```

## 7. Drawing isotherms after running *Fiammetta.m* (see *Table 60*)

```
figure;gt =[min(tca(1:no)) pai max(tca(1:no))];
nec = (nni+1)^2; % Standard isotherms
for i = 1 : np % Loop on CAD patches
    gra_ipa(nni+1,nni+1,lK((i-1)*nec+1:nel,:),tca(1:no),xyz,gt);
    hold on
end;axis equal;colorbar;axis off
title (['Dissipation: ',num2str(.5*tca'*K*tca,3),' WK, DOF: ',...
        num2str(ndK), ' '])
```

## 8. Drawing the boundaries of the domain

```
for i = 1:nbo % Drawing the border of the domain
    if bor(i,4)== 0
        plot([xyz(bor(i,1),1) xyz(bor(i,2),1)], ...
            [xyz(bor(i,1),2) xyz(bor(i,2),2)],'k','LineWidth',2)
    end
end
```

## 9. Drawing a nodal scalar quantity element by element (see *Table 61*)

```
figure;gra_lin(xyz,nel,lK,tca) % nodal scalar quantity isotherms
```

## 10. Drawing isotherms after running *Fia\_20221023.m* (*Table 1*)

```
figure;colormap(gra_cob);gra_ipa(nx,ny,lK,tca,xyz,[270 2 320]);
colorbar;axis equal;axis off
```

## 11. Drawing boundaries of the domain after running *Fiam\_33\_20220822.m* (*Table 2*)

```
plot ([xyz(1,1) xyz(nx+1,1) xyz(nx+1,1) xyz(1,1) xyz(1,1)], [xyz(1,2) ...
xyz(nx+1,2) xyz((nx+1)*(ny+1),2) xyz((nx+1)*(ny+1),2) xyz(1,2)], ...
'k','LineWidth',3);hold on;axis off;axis equal
```

## 12. Drawing heat flows on the street section boundary

```
figure;hst=K*tca;bar(hst(lcont));hold on
title (['Street boundary nodal heat flows, total: ',...
        num2str(sum(hst),3) ' (W)'])
```

## 13. Drawing the Sky or Ground View Factors on street walls

```
figure;bar(SVF);grid on;hold on % max (SVF)
title (['Street section sky view factors, average:', num2str(mean(SVF),3)])
figure('Position',[100 100 900 400]);bar(SVF);grid on;hold on % max (SVF)
```

```

title(['Street side with balcony sky view factors, average:', ...
    num2str(mean(SVF), 3))]

figure('Position',[100 100 900 400]);bar(F(:,n3-1));grid on;hold on
title(['Street side with balcony ground view factors, average:', ...
    num2str(mean(F(:,n3-1)),3)])

```

#### 14. Drawing injected heat evolution (see *Table 61*)

```
gra_hie(nit,tt,ga) % Drawing heat input evolution
```

*Table 50: Postprocessing functions*

Matlab <sup>®</sup> function <i>gra_ist.m</i> - isotherm drawing in a <i>nx x ny</i> mesh	
1	<pre> 1 function [] = gra_ist(nx,ny,z,xyz,gt) 2 no = (nx+1)*(ny+1);ii = 0; % Number of points of the grid 3 xx = zeros(ny+1,nx+1);yy = xx;tn = ones(ny+1,nx+1)*z(1); % Initializations 4 for i = 1:ny 5     for j = 1:nx+1; ii = ii+1; tn(i,j) = z(ii); end; 6 end 7 tn(ny+1,:) = z(ii+1:no); jj = 0; 8 for i = 1:ny+1 9     for j = 1:nx+1;jj = jj+1;xx(i,j) = xyz(jj,1);yy(i,j) = xyz(jj,2);end 10 end 11 colormap(gra_cob); % Color map definition 12 [CS,H] = contourf(xx,yy,tn,(gt(1):gt(2):gt(3)), 'b');hold on;axis equal 13 clabel(CS,H,[ 280 285 290 295 300 305 310 315 320]); 14 plot ([0 nx nx 0 0],[0 0 ny ny 0], 'k', 'LineWidth',2);hold on;axis equal 15 end </pre>

*Table 51: Matlab<sup>®</sup> function *gra\_ist.m* - isotherm drawing in a *nx x ny* mesh*

The postprocessing functions referred in *Table 50* are listed in *Table 52*, *Table 53* to *Table 63*.

Matlab <sup>®</sup> function <i>gra_mel.m</i> - drawing a shrunk mesh	
1	<pre> 1 function [] = gra_mel(xyz,lK,dn,sh,fs) % Drawing the shrunk mesh 2 % sh is the shrinking coefficient 0 &lt; sh &lt;= 1 3 nel = size(lK,1);nn=size(lK,2);X = zeros(nn+1,1);Y = zeros(nn+1,1); 4 for j = 1:nel % Nodes are numbered left - right, top - bottom 5     ce = zeros(2,1); 6     for i = 1:nn % Loop on the nn vertices of the elements 7         ce(1) = ce(1)+xyz(lK(j,i),1)/nn; 8         ce(2) = ce(2)+xyz(lK(j,i),2)/nn; 9         X(i) = xyz(lK(j,i),1); 10        Y(i) = xyz(lK(j,i),2); 11    end 12    X(nn+1) = X(1);Y(nn+1)=Y(1); % First node repeated at the end of list 13    if dn == 1 14        plot((1-sh)*ce(1)+sh*X,(1-sh)*ce(2)+sh*Y, 'k');hold on 15        text(ce(1),ce(2),num2str(j), 'Color', 'r', 'FontSize', fs);hold on 16    end 17    if dn == 2 18        plot((1-sh)*ce(1)+sh*X,(1-sh)*ce(2)+sh*Y, 'k');hold on 19        text(ce(1),ce(2),num2str(j), 'Color', 'b', 'FontSize', fs);hold on 20    end 21    if dn == 3;plot((1-sh)*ce(1)+sh*X,(1-sh)*ce(2)+sh*Y, '-b');hold on;end 22 end 23 end </pre>

*Table 52: Matlab<sup>®</sup> function *gra\_mel.m* - drawing a shrunk mesh*

Matlab <sup>®</sup> function <i>gra_mnl.m</i> - displaying nodes and element labels	
1	<pre> 1 function [] = gra_mnl(xyz,lK,lc,fs) % Display node and elements labels 2 figure;gra_mel(xyz,lK,1,1,fs) % Draw conductive elements and nodes labels 3 if lc(1,3) &gt; 0; gra_mel(xyz,lc,2,.8); axis equal; axis off;end 4 for i=1:size(xyz,1) 5     text(xyz(i,1),xyz(i,2),num2str(i), 'FontSize', fs);hold on 6 end 7 end </pre>

*Table 53: Matlab<sup>®</sup> function *gra\_mnl.m* - displaying node & element labels*

Matlab <sup>®</sup> function <i>gra_tra.m</i> – Temperature along radiative border	
1	<pre> 1 function [] = gra_tra(tca,lcont,re,ca) </pre>

```

2 figure; % Drawing the temperature along the radiative border
3 if ca > 0
4 plot([tca(lcont); tca(lcont(1))], 'k*-')
5 else
6 plot(tca(lcont), 'k*-');
7 end
8 if re > .999;re = 1;end
9 title(['tra 9, Temp. along radiative border, reflect.: ', num2str(re)]);
10 xlabel('From bottom right to left, to top left & top right')
11 ylabel('Temperature (K)');grid on;
12 end

```

Table 54: Matlab<sup>®</sup> function *gra\_tram* - Temperature along radiative border

Matlab <sup>®</sup> function <i>gra_2dm.m</i> – Second member along radiative border
<pre> 1 function []=gra_2dm(K,tca,re,lcont) 2 gsm = K*tca; % second members on the cavity border 3 figure ( 'Position', [100 100 800 300]);bar(gsm(lcont));grid on 4 xlabel (['Reflectivity: ', num2str(re,2)]) 5 title ( ['L 2dm, Final radiative loads K*tca, min max sum: ',... 6 num2str([min(gsm(lcont)) max(gsm(lcont)) sum(gsm(lcont))],3),... 7 ' W']);hold on 8 disp(['L 2dm, He fl. K*tca : ',num2str(sum(gsm(lcont)),3), ' W']) 9 end </pre>

Table 55: Matlab<sup>®</sup> function *gra\_2dm* – Second member along radiative border

Matlab <sup>®</sup> function <i>gra_atg.m</i> – temperature gradients
<pre> 1 function [] = gra_atg (xyz,lK,tca) % Element temperatures gradient arrows 2 nel = size(lK,1);% nel = 1;%  3 u = zeros(nel,1);v=zeros(nel,1);xx=zeros(nel,1);yy=zeros(nel,1); 4 area = zeros(nel,1); 5 for ii = 1:nel % Loop on the nel elements 6 Q = [xyz(lK(ii,1),1:2); xyz(lK(ii,2),1:2); xyz(lK(ii,3),1:2); ... 7 xyz(lK(ii,4),1:2)]; 8 X = Q(:,1);Y = Q(:,2);xx(ii) = sum(Q(:,1))/4;yy(ii) = sum(Q(:,2))/4; 9 area(ii) = (X(2)-X(1)+X(3)-X(4))/2*(Y(3)-Y(1)); 10 te = [tca(lK(ii,1)) tca(lK(ii,2)) tca(lK(ii,3)) tca(lK(ii,4))]; 11 J = [[-1 1 1 -1]*X [-1 1 1 -1]*Y;...% Jacob. matrix barycenter 12 [-1 -1 1 1]*X [-1 -1 1 1]*Y]/2; 13 gr = [-1 1 1 -1;-1 -1 1 1]*te'/2; % Parametric grad. barycenter 14 g = J^(-1)*gr; 15 u(ii) = g(1); 16 v(ii) = g(2); 17 end 18 scale = 2;quiver(xx,yy,u,v,scale,'b','LineWidth',1);hold on; 19 % ad = sum(area); % Maximum &amp; mean heat flow 20 gm = [max(sqrt(u.*u+v.*v)) mean(sqrt(u.*u+v.*v))];% grad: max &amp; average 21 % fm = mean(sqrt(u.*u+v.*v)); % W/m2 22 title(['Temperature gradient, max: ',num2str(gm(1),2),', mean: ',... 23 num2str(gm(2),2), ' K/m'], 'fontSize',15);axis off;hold on 24 disp(['g 18, Max temp grad: ', num2str(gm(1),2), ', mean: ',... 25 num2str(gm(2),2), ' K/m']) 26 end </pre>

Table 56: Matlab<sup>®</sup> function *gra\_atg.m* - visualization of temperature gradient arrows

Matlab <sup>®</sup> function <i>gra_ahf.m</i> - visualization of heat flow arrows in a mesh
<pre> 1 function [fm] = gra_ahf (xyz,lK,tca,co) % Element heat flows arrows 2 nel = size(lK,1); 3 u = zeros(nel,1);v=zeros(nel,1);xx=zeros(nel,1);yy=zeros(nel,1); 4 area = zeros(nel,1); 5 for ii = 1:nel % Loop on the nel elements 6 Q = [xyz(lK(ii,1),1:2);xyz(lK(ii,2),1:2);xyz(lK(ii,3),1:2);... 7 xyz(lK(ii,4),1:2)]; 8 X = Q(:,1);Y = Q(:,2);xx(ii) = sum(Q(:,1))/4;yy(ii) = sum(Q(:,2))/4; 9 area(ii) = (X(2)-X(1)+X(3)-X(4))/2*(Y(3)-Y(1)); 10 te = [tca(lK(ii,1)) tca(lK(ii,2)) tca(lK(ii,3)) tca(lK(ii,4))]; 11 J = 1/2*[[-1 1 1 -1]*X [-1 1 1 -1]*Y;... 12 [-1 -1 1 1]*X [-1 -1 1 1]*Y]; 13 gr = [-1 1 1 -1; -1 -1 1 1]*te'/2; % Parametric grad. barycenter 14 g = -co(ii)*J^(-1)*gr; 15 u(ii) = g(1); 16 v(ii) = g(2); 17 end 18 scale = 2;quiver(xx,yy,u,v,scale,'r','LineWidth',1);hold on; 19 ad = sum(area); % Maximum &amp; mean heat flow 20 gm = [max(sqrt(u.*u+v.*v)) sqrt(u.*u+v.*v)*area/ad]; % disp(gm(2)^2) 21 fm = mean(sqrt(u.*u+v.*v)); % W/m2 22 title(['TA heat flows, max: ',num2str(gm(1),2),', mean: ',... </pre>

```

23     num2str(mean(sqrt(u.*u+v.*v)),2), ' W/m2'], 'fontsize',15);hold on
24     disp(['hf 25, Max heat flow: ', num2str(gm(1),2), ', mean: ',...
25         num2str(mean(sqrt(u.*u+v.*v)),2), ' W/m2'])
26 end

```

Table 57: Matlab<sup>®</sup> function *gra\_ahf.m* - visualization of heat flow arrows

Matlab<sup>®</sup> function *gra\_chf.m* - visualization temperature arrows for convection

```

1 function [] = gra_chf(xyz,lc,tca,h,fm)% Temperatures arrows for convection
2 nel = size(lc,1); qn = zeros(nel,3);
3 %xm=zeros(nel,1);ym=zeros(nel,1);nor=zeros(nel,2);
4 disp(['ch 04, coef. red. dt: ',num2str(fm,2),' W/m2'])
5 for ii = 1:nel
6     % Loop on the nel elements
7     tgt = [xyz(lc(ii,2),1)-xyz(lc(ii,1),1);...
8             xyz(lc(ii,2),2)-xyz(lc(ii,1),2); 0]; % Vector tangent to edge
9     xm = xyz(lc(ii,1),1)+(xyz(lc(ii,2),1)-xyz(lc(ii,1),1))/2;
10    ym = xyz(lc(ii,1),2)+(xyz(lc(ii,2),2)-xyz(lc(ii,1),2))/2;
11    dt = ((tca(lc(ii,1))+tca(lc(ii,2)))/2-tca(lc(ii,3)))/fm;% Temp. diff.
12    nor = cross(tgt,[0 0 1])/norm(tgt)*h*norm(dt);
13    qn(ii,:)= h*dt/norm(tgt)*cross(tgt,[0 0 1]);% h*dt x unit normal
14    disp(['c 13, temp. grad. : ',num2str(dt),' K'])
15    if dt > 0
16        quiver(xm(1),ym(1),nor(1),nor(2),1,'g','LineWidth',2);hold on;
17    else
18        quiver(xm(1)+nor(1),ym(1)+nor(2),-nor(1),-nor(2),1,'g',...
19            'LineWidth',2);hold on;
20    end
21 end
22 disp(['ch 21, Ave conv flow: ',num2str(mean(qn,1)), ' W/m2'])

```

Table 58: Matlab<sup>®</sup> function *gra\_chf.m* - visualization of heat flow arrows

Matlab<sup>®</sup> function *gra\_cob.m*

1	<b>function</b> [bar] = gra_cob
2	bar=[ 0 0 0.5625
3	0 0 0.6250
4	0 0 0.6875
5	0 0 0.7500
6	0 0 0.8125
7	0 0 0.8750
8	0 0 0.9375
9	0 0 1.0000
10	0 0.0625 1.0000
11	0 0.1250 1.0000
12	0 0.1875 1.0000
13	0 0.2500 1.0000
14	0 0.3125 1.0000
15	0 0.3750 1.0000
16	0 0.4375 1.0000
17	0 0.5000 1.0000
18	0 0.5625 1.0000
19	0 0.6250 1.0000
20	0 0.6875 1.0000
21	0 0.7500 1.0000
22	0 0.8125 1.0000
23	0 0.8750 1.0000
24	0 0.9375 1.0000
25	0 1.0000 1.0000
26	0.0625 1.0000 0.9375
27	0.1250 1.0000 0.8750
28	0.1875 1.0000 0.8125
29	0.2500 1.0000 0.7500
30	0.3125 1.0000 0.6875
31	0.3750 1.0000 0.6250
32	0.4375 1.0000 0.5625
33	0.5000 1.0000 0.5000
34	0.5625 1.0000 0.4375
35	0.6250 1.0000 0.3750
36	0.6875 1.0000 0.3125
37	0.7500 1.0000 0.2500
38	0.8125 1.0000 0.1875
39	0.8750 1.0000 0.1250
40	0.9375 1.0000 0.0625
41	1.0000 1.0000 0
42	1.0000 0.9375 0
43	1.0000 0.8750 0
44	1.0000 0.8125 0
45	1.0000 0.7500 0
46	1.0000 0.6875 0

47	1.0000	0.6250	0
48	1.0000	0.5625	0
49	1.0000	0.5000	0
50	1.0000	0.4375	0
51	1.0000	0.3750	0
52	1.0000	0.3125	0
53	1.0000	0.2500	0
54	1.0000	0.1875	0
55	1.0000	0.1250	0
56	1.0000	0.0625	0
57	1.0000	0	0];
58	<b>end</b>		

Table 59: Matlab<sup>©</sup> function **gra\_cob.m** - color bar

Matlab <sup>©</sup> function <b>gra_ipa.m</b> - drawing isotherm lines in a Coons patch			
1	<b>function</b> [] = <b>gra_ipa</b> (nx,ny,el,z,xyz,gt) % Isotherms lines in a patch		
2	xx = zeros(ny+1,nx+1);yy = xx;mp = xx ;tn = ones(ny+1,nx+1)*z(1);ii = 0;		
3	<b>for</b> j = 1:ny		
4	<b>for</b> i = 1:nx		
5	ii = ii+1;		
6	mp(j , i) = el(ii,1); mp(j , i+1) = el(ii,2);		
7	mp(j+1, i+1) = el(ii,3); mp(j+1, i) = el(ii,4);		
8	<b>end</b>		
9	<b>end</b>		
10	<b>for</b> j = 1 : nx+1		
11	<b>for</b> i = 1 : ny+1		
12	xx(i,j) = xyz(mp(i,j),1);yy(i,j) = xyz(mp(i,j),2);		
13	tn(i,j) = z(mp(i,j));		
14	<b>end</b> ;		
15	<b>end</b>		
16	colormap( <b>gra_cob</b> ); % Color map definition		
17	[CS,H] = contourf(xx,yy,tn,(gt(1):gt(2):gt(3)), <b>'b'</b> );hold <b>on</b> ;axis <b>equal</b>		
18	clabel(CS,H,[ 280 285 290 295 300 305 310 315 320]);		
19	<b>end</b>		

Table 60: Matlab<sup>©</sup> function **gra\_ipa.m** - drawing isotherms in a Coons patch

Matlab <sup>©</sup> function <b>gra_lin.m</b> - visualization of the levels of a function			
1	<b>function</b> [] = <b>gra_lin</b> (xyz,nel,lK,tca)% Visualization element by element		
2	colormap( <b>gra_cob</b> )		
3	<b>for</b> i = 1:nel		
4	fill(xyz(lK(i,:),1)',xyz(lK(i,:),2)',tca(lK(i,:)));hold <b>on</b> ;		
5	<b>end</b> ;		
6	% fill([-1 -.9 -.9 -1],[0 0 1 1],[300 300 320 320]);hold <b>on</b> ;		
7	colorbar;axis <b>equal</b> ;axis <b>off</b>		
8	<b>end</b>		

Table 61: Matlab<sup>©</sup> function **gra\_lin.m** - visualization of the levels of a scalar function

Matlab <sup>©</sup> function <b>gra_hie.m</b> – visualization of a periodic heat load			
1	<b>function</b> [] = <b>gra_hie</b> (ni,dt,ga) % Evolution of incoming heat		
2	tem = (0:ni)*dt/3600/ni; % Time steps for the time graphics		
3	figure('Position',[100 100 700 300]);		
4	plot (tem,ga*1.e-3,'b');hold <b>on</b> ;grid <b>on</b>		
5	xlabel('Elapsed time (hours) ','fontsize',15)		
6	ylabel('Injected heat flow (kW) ','fontsize',15)		
7	title (['gra-hie: Total injected heat: ',num2str(sum(ga)*1.e-6,3),...		
8	' MJ'],'fontsize',15)		
9	<b>end</b>		

Table 62: Matlab<sup>©</sup> function **gra\_hie.m** – visualization of a periodic scalar field

Function Matlab <sup>©</sup> <b>gra_tev.m</b> – temperature evolution in transient applications			
1	<b>function</b> [] = <b>gra_tev</b> (ni,dt,re,tsmax,tsmin,tmoy,met) % Temp. evolution		
2	tem = (0:ni)*dt/3600/ni; % Time steps for the graphics		
3	st = size(tsmin,2);		
4	figure('Position',[100 100 600 300])		
5	plot (tem,tsmax,'r');hold <b>on</b> ; % Plotting the 3 evolutive functions		
6	plot (tem,tsmin,'b');hold <b>on</b> ;		
7	plot (tem,tmoy , 'k');hold <b>on</b> ;		
8	ree=re;if re>.999;ree=1;end		
9	if met == 3		
10	xlabel(['Number of iterations : ',num2str(ni),], 'fontsize',15);grid <b>on</b>		
11	title ( ' st 109, gra-tev: temperature evolution','fontsize',15)		
12	elseif met == 4		
13	xlabel (['Number of iterations : ',num2str(ni),', re : ','...		

```

14         num2str(ree)],'fontsize',15);grid on
15         title ('sq 150, gra-tev: temperature evolution','fontsize',15)
16 elseif met == 5
17     xlabel(['Number of iterations : ',num2str(ni),' re : ','...
18             num2str(ree)],'fontsize',15);grid on
19     title ('sc 139, gra-tev: temperature evolution','fontsize',15)
20 end
21 legend ('T maximum ','T minimum ','T average','Location','northwest')
22 ylabel(['Tmin: ',num2str(tsmin(st)*10/10,3),' K, Tmean: ',...
23         num2str(tmoy(st)*10/10,3),' K, Tmax: ',num2str(tsmax(st)*10/10,3),...
24         ' K'],'fontsize',10);hold on;grid on
25 end

```

*Table 63: Matlab<sup>®</sup> function *gra\_tev.m* – temperature evolution*

To perform the visualizations of the element heat flows and temperature gradients, it is possible to run the procedure *P\_flgr.m* (*Table 64*) after running *Fiammetta.m*.

### Matlab<sup>®</sup> procedure *P\_flgr.m* - drawing heat flows & temperature gradients

```

1 figure % Drawing the element heat flows in the meshed domain
2 [fm]=gra_ahf(xyz,lK,tca,co);gra_mel(xyz,lK,3,1,10);axis equal;hold on
3 for i = 1:nbo % Drawing the border of the domain
4     if bor(i,4)==0
5         plot([xyz(bor(i,1),1) xyz(bor(i,2),1)], ...
6               [xyz(bor(i,1),2) xyz(bor(i,2),2)],'k','LineWidth',2);hold on
7     end;axis off;hold on
8 end % End drawing element heat flows and domain border
9 if lc(1,3)> 0
10    gra_chf(xyz,lc,tca,h,fm); % fm is the averzge of element heat flows
11 end
12 figure % Drawing the temperature gradients in the meshed domain
13 gra_atg(xyz,lK,tca);gra_mel(xyz,lK,2,.9,10);axis equal;axis off
14 for i = 1:nbo % Drawing the border of the domain
15     if bor(i,4)==0
16         plot([xyz(bor(i,1),1) xyz(bor(i,2),1)], ...
17               [xyz(bor(i,1),2) xyz(bor(i,2),2)],'k','LineWidth',2)
18     end
19 end % End drawing temperature gradient and domain border
20 % disp(['re.....: ',num2str(re)]);

```

*Table 64: Matlab<sup>®</sup> procedure *P\_flgr.m* - heat flows & temperature gradients*

## 8.5 Additional procedures

In pure conduction problems, the solution of the heat transfer problems is independent of the geometric scale. However, in radiation as well as in convection, the size of the domain has to be given because the convective and radiative conduction matrices (*Table 45*) and (*Table 46*) depend on the size *L* of these elements.

### Matlab<sup>®</sup> function *geo\_baf.m* – view factor matrix of a wall with balcony

```

1 function[F] = geo_baf(nci,xyz_cao) % Balcony view factor matrix % 20210929
2 n3 = nci*5+2;% Size of view factors matrix including sky and ground
3 F = zeros(n3,n3); f=zeros(n3,1);
4 xc = zeros(n3,1) ;yc = xc;Le = yc; % Edges definition
5 xn = zeros(n3,1);yn = xn; % vertices definition
6 t = 0:1/nci:1;b = zeros(1,n3-2);
7 for i = 1:nci
8     xn(i,1) = xyz_cao(11,1)*(1-t(i))+xyz_cao(4,1)*t(i);
9     xn(i+nci,1) = xyz_cao(4,1)*(1-t(i))+xyz_cao(2,1)*t(i);
10    xn(i+2*nci,1) = xyz_cao(2,1)*(1-t(i))+xyz_cao(1,1)*t(i);
11    xn(i+3*nci,1) = xyz_cao(1,1)*(1-t(i))+xyz_cao(3,1)*t(i);
12    xn(i+4*nci,1) = xyz_cao(3,1)*(1-t(i))+xyz_cao(9,1)*t(i);
13    end;xn(n3-2,1) = xyz_cao(9,1);
14    xn(n3-1,1) = xyz_cao(9,1);
15    for i=1:nci
16        yn(i,1) = xyz_cao(11,2)*(1-t(i))+xyz_cao(4,2)*t(i);
17        yn(i+nci,1) = xyz_cao(4,2)*(1-t(i))+xyz_cao(2,2)*t(i);
18        yn(i+2*nci,1) = xyz_cao(2,2)*(1-t(i))+xyz_cao(1,2)*t(i);
19        yn(i+3*nci,1) = xyz_cao(1,2)*(1-t(i))+xyz_cao(3,2)*t(i);
20        yn(i+4*nci,1) = xyz_cao(3,2)*(1-t(i))+xyz_cao(9,2)*t(i);
21    end;yn(n3-2,1) = xyz_cao(3,2)*(1-t(i))+xyz_cao(9,2)*t(i);
22    yn(n3-1,1) = xyz_cao(9,2);
23    for i = 1: n3-2;Le(i) = sqrt((xn(i+1)-xn(i))^2+(yn(i+1)-yn(i))^2);end
24    for i = 1: n3-2;xc(i) = (xn(i)+xn(i+1))/2;yc(i)=(yn(i)+yn(i+1))/2;end

```

```

25 tsb = [0 -1 0 1 0;-1 0 -1 0 -1];k=0;for i=1:5;for j=1:nci;k=k+1;...
26 b(k)=i;end;end
27 ts(1,:) = tsb(1,b);ts(2,:) = tsb(2,b);
28 F(1:3*nci ,n3)=.5;F(2*nci+1:5*nci,n3-1)=.5;
29 for np = 1 : nci % Form fact. matrix for nci (upper L) vert. elem.
30 for i = nci+1 : 2*nci % Loop on the nci points
31 r0 = sqrt((xn(i)-xc(np))^2+(yn(i)-yc(np))^2);
32 r1 = sqrt((xn(i+1)-xc(np))^2+(yn(i+1)-yc(np))^2);
33 f(i) = (((xn(i)-xc(np))/r0 -(xn(i+1)-xc(np))/r1 ) *ts(1,np)-...
34 ((yn(i)-yc(np))/r0 -(yn(i+1)-yc(np))/r1 ) *ts(2,np))/2;
35 end
36 f(n3-1) = -(((xn(i)-xc(np))/r1-1) *ts(1,np)-...
37 ((yn(i)-yc(np))/r1 ) *ts(2,np))/2; f(n3) = 0.5;
38 F(np,:) = f';
39 end
40 f = zeros(n3,1);
41 for np = nci+1:2*nci % Form fact. for nci (upper L) horiz. elem.
42 for i = 1 : nci % Loop on the nci points
43 r0 = sqrt((xn(i)-xc(np))^2+(yn(i)-yc(np))^2);
44 r1 = sqrt((xn(i+1)-xc(np))^2+(yn(i+1)-yc(np))^2);
45 f(i) = (((xn(i)-xc(np))/r0 -(xn(i+1)-xc(np))/r1 ) *ts(1,np)-...
46 ((yn(i)-yc(np))/r0 -(yn(i+1)-yc(np))/r1 ) *ts(2,np))/2;
47 end
48 f(n3-1) = 0.;f(n3) = 1-sum(f(1:n3-2));
49 F(np,:) = f';
50 end
51 f = zeros(n3,1);
52 for np = 3*nci+1:4*nci % Form fact. for nci (lower L) horiz. elem.
53 for i = 4*nci+1:5*nci % Loop on the nci points
54 r0 = sqrt((xn(i)-xc(np))^2+(yn(i)-yc(np))^2);
55 r1 = sqrt((xn(i+1)-xc(np))^2+(yn(i+1)-yc(np))^2);
56 f(i) = (((xn(i)-xc(np))/r0 -(xn(i+1)-xc(np))/r1 ) *ts(1,np)-...
57 ((yn(i)-yc(np))/r0 -(yn(i+1)-yc(np))/r1 ) *ts(2,np))/2;
58 end
59 f(n3-1) = 1-sum(f(1:n3-2)); f(n3) = 0.;
60 F(np,:) = f';
61 end
62 f = zeros(n3,1);
63 for np = 4*nci+1:5*nci % Form fact. for nci (lower L) vert. elem.
64 for i = 3*nci+1:4*nci % Loop on the nci points
65 r0 = sqrt((xn(i)-xc(np))^2+(yn(i)-yc(np))^2);
66 r1 = sqrt((xn(i+1)-xc(np))^2+(yn(i+1)-yc(np))^2);
67 f(i) = -(((xn(i)-xc(np))/r0 -(xn(i+1)-xc(np))/r1 ) *ts(1,np)-...
68 ((yn(i)-yc(np))/r0 -(yn(i+1)-yc(np))/r1 ) *ts(2,np))/2;
69 end
70 f(n3-1) = .5 ;f(n3) = 1-sum(f(1:n3-1)); F(np,:) = f';
71 end
72 end

```

Table 65: Matlab<sup>®</sup> function *geo\_baf.m* – view factor matrix – wall with balcony

Matlab <sup>®</sup> function <i>geo_stf.m</i> – view factor matrix of a street section	
1	<b>function</b> [F] = geo_stf(n,Lel)
2	xs = Lel(1); ys = Lel(2); % Lel = vector of the side lengths
3	no = n-1;% n is the number of el. on a side, no the number of nodes
4	n3 = n*3;% * % View factors matrix: 3 sides and the sky
5	F = zeros(n3,n3); r0=zeros(1,n3); r1=zeros(1,n3); f = zeros(1,n3);
6	xc = zeros(1,n3) ;yc = xc; Le = yc; % Edges definition
7	xn = zeros(1,n3+1);yn = xn; % vertices definition
8	xn(1:no) = 0; k = 0; <b>for</b> i = n+2:2*n+1; k=k+1; xn(i)=xs/(n)*k; <b>end</b> ;
9	xn(2*n+2:3*n+1) = xs;% disp(['xn : ',num2str(xn)])
10	<b>for</b> i=1:n;yn(i) = ys-(i-1)*ys/(n); <b>end</b> ;yn(n+2:2*n) =0;k=-1;
11	<b>for</b> i=2*n+1:3*n+1;k=k+1;yn(i)=k*ys/(n); <b>end</b>
12	<b>for</b> i = 1: n3;Le(i) = sqrt((xn(i+1)-xn(i))^2+(yn(i+1)-yn(i))^2); <b>end</b>
13	<b>for</b> i = 1: n3;xc(i) = (xn(i)+xn(i+1))/2;yc(i)=(yn(i)+yn(i+1))/2; <b>end</b>
14	ts = [0 -1 -1;1 0 1;0 1 -1]; % Tangents of edges & their sign
15	<b>for</b> np = 1 : n3 % Form factor matrix for the n3 elements
16	i0 = 0; il = 1;
17	<b>for</b> i = 1 : n3 % Loop on the n4 points
18	il0 = i0+1;
19	il1 = il+1;
20	nuc = ceil(np/n);
21	r0(i) = sqrt((xn(i0)-xc(np))^2+(yn(i0)-yc(np))^2);
22	r1(i) = sqrt((xn(il1)-xc(np))^2+(yn(il1)-yc(np))^2);
23	f(i) = (((xn(i0)-xc(np))/r0(i)-(xn(il1)-xc(np))/r1(i))*...
24	ts(nuc,1)-((yn(i0)-yc(np))/r0(i)-(yn(il1)-yc(np))/r1(i))*...
25	ts(nuc,2))*ts(nuc,3)/2;
26	<b>end</b>
27	<b>for</b> i = 1:n;f((nuc-1)*n+i) = 0.; <b>end</b> % View factor of anal. face
28	F (np,: ) = f;
29	<b>end</b>

30 end

Table 66: Matlab<sup>®</sup> function geo\_stf.m – view factor matrix of a street section

Matlab <sup>®</sup> function geo_vfc.m	
<pre> 1  function[F] = geo_vfs(n,Lel,ns) % n is the numb. of elements on the 4 sides 2  xs      = Lel(1); ys = Lel(2);           % Lel = vector of the side lengths 3  no      = n+1;                         % no is the number of nodes on each side 4  if ns == 4;n4=n*ns;else;n4=n*ns+2;end % VF mat.: ns sides or ns + sky 5  F       = zeros(n4,n4); r0=zeros(1,n4); r1=zeros(1,n4); f = zeros(1,n4); 6  xc     = zeros(1,n4); yc   = zeros(1,n4); Le = yc; % Edges definition 7  xn     = zeros(1,n4+1);yn   = zeros(1,n4+1);          % vertices definition 8  for i = 1:n    % Ordered nodes in the street: from bottom-left, area left 9    xn(i+1)    = xs/n*i; xn(no+i)    = xs; xn(no+n+i) = xs-xs/n*i; 10   yn(i+no)   = ys/n*i; yn(no+n+i) = ys; yn(3*n+1+i) = ys-ys/n*i; 11 end 12 for i      = 1: n4;Le(i)=sqrt((xn(i+1)-xn(i))^2+(yn(i+1)-yn(i))^2);end 13 for i      = 1: n4;xc(i)=(xn(i)+xn(i+1))/2; yc(i)=(yn(i)+yn(i+1))/2;end 14 ts        = [1 0 1;0 1 -1;-1 0 1;0 -1 -1];% Tangents of edges &amp; their sign 15 for np      = 1 : n4% Compute the form factor matrix for the np elements 16   i0      = 0; i1 = 1;                      % Loop on the n4 points 17   for i      = 1 : n4 18     i0      = i0+1; 19     i1      = i1+1; 20     nuc    = ceil(np/n); 21     r0(i)  = sqrt((xn(i0)-xc(np))^2+(yn(i0)-yc(np))^2); 22     r1(i)  = sqrt((xn(i1)-xc(np))^2+(yn(i1)-yc(np))^2); 23     f(i)   = (((xn(i0)-xc(np))/r0(i)-(xn(i1)-xc(np))/r1(i))*... 24                 ts(nuc,1)-((yn(i0)-yc(np))/r0(i)-(yn(i1)-yc(np))/r1(i))*... 25                 ts(nuc,2))*ts(nuc,3)/2; 26   end 27   for i      = 1:n;f((nuc-1)*n+i) = 0.;end % View factor of anal. face 28   F(:,np)   = f; 29 end 30 end </pre>	

Table 67: Matlab<sup>®</sup> function geo\_vfc.m – view factor matrix – ns sides domain

Matlab <sup>®</sup> function geo_vfr.m	
<pre> 1  function[F] = geo_vfr(n,Lel)           % Rectangular cavity view factor matrix 2  xs      = Lel(1); ys = Lel(2);           % Lel = vector of the side lengths 3  no      = n+1;% n: number of elem. &amp; no : number of nodes per patch side 4  nf      = n*4;                          % Size of the view factors matrix 5  F       = zeros(nf,nf);r0 = zeros(1,nf);r1 = r0;f = r0;xc = r0;yc = r0; 6  xn     = zeros(1,nf+1);yn = xn;          % vertices definition 7  for I = 1:n    % Ordered nodes from bottom-left, area left 8    xn(I+1)    = xs/n*I; xn(no+i)    = xs; xn(no+n+i) = xs-xs/n*I; 9    yn(I+no)   = ys/n*I; yn(no+n+i) = ys; yn(3*n+1+i) = ys-ys/n*I; 10 end 11 for I      = 1: nf;xc(i)=(xn(i)+xn(i+1))/2; yc(i)=(yn(i)+yn(i+1))/2;end 12 ts        = [1 0 1;0 1 -1;-1 0 1;0 -1 -1];% Tangents of edges &amp; their sign 13 for np      = 1 : nf% Compute the form factor matrix for the nf elements 14   i0      = 0; 15   xcl=(.5-sqrt(3)/6)*(xn(np+1)-xn(np))+xn(np); 16   xc2=(.5+sqrt(3)/6)*(xn(np+1)-xn(np))+xn(np); 17   ycl=(.5-sqrt(3)/6)*(yn(np+1)-yn(np))+yn(np); 18   yc2=(.5+sqrt(3)/6)*(yn(np+1)-yn(np))+yn(np); 19   for I      = 1 : nf                      % Loop on the nf visible segments 20     i0      = i0+1;i1 = i0+1; 21     nuc    = ceil(np/n);                  % nuc is the patch side number 22     r0(i)  = sqrt((xn(i0)-xcl)^2+(yn(i0)-ycl)^2); 23     r1(i)  = sqrt((xn(i1)-xcl)^2+(yn(i1)-ycl)^2); 24     f1    = (((xn(i0)-xcl)/r0(i)-(xn(i1)-xcl)/r1(i))*... 25                 ts(nuc,1)-((yn(i0)-ycl)/r0(i)-(yn(i1)-ycl)/r1(i))*... 26                 ts(nuc,2))*ts(nuc,3)/2; 27     r0(i)  = sqrt((xn(i0)-xc2)^2+(yn(i0)-yc2)^2); 28     r1(i)  = sqrt((xn(i1)-xc2)^2+(yn(i1)-yc2)^2); 29     f2    = (((xn(i0)-xc2)/r0(i)-(xn(i1)-xc2)/r1(i))*... 30                 ts(nuc,1)-((yn(i0)-yc2)/r0(i)-(yn(i1)-yc2)/r1(i))*... 31                 ts(nuc,2))*ts(nuc,3)/2; 32   end 33   for I      = 1:n;f((nuc-1)*n+i) = 0.;end % View factor of anal. face 34   F(:,np)   = f; 35 end 36 end </pre>	

Table 68: Matlab<sup>®</sup> function geo\_vfr.m – view factor matrix – 2 Gauss points

A particular function has been developed to take into account non homogeneous as well as anisotropic materials. This option is managed with the variables *Ai* and *fa*.

Matlab <sup>©</sup> function <i>mat_cok.m</i> – non homogeneous conductivity	
1	<code>function [co] = mat_cok (Ai,nci,fa,xy,lK,deb) % Non uniform conductivity</code>
2	<code>k = 1; % W/(m K)</code>
3	<code>nel = nci * nci; % Number of elements per patch</code>
4	<code>co = ones(nel,1)*k; % By default, the system is isotropic, k constant</code>
5	<code>if Ai == 1 % nci is odd</code>
6	<code>    if floor(nci/2) &lt; ceil(nci/2)</code>
7	<code>        tr=1/nci;</code>
8	<code>        for i =floor(nci/2)*nci+1:floor(nci/2)*nci+nci;co(i)=k*fa;end% (n)</code>
9	<code>    else % nci is even</code>
10	<code>        tr=2/nci;</code>
11	<code>        for i =(nci/2-1)*nci+1:(nci/2*nci)+nci;co(i)=k*fa;end % (W)</code>
12	<code>    end</code>
13	<code>end</code>
14	<code>if Ai == 3 % nci is odd</code>
15	<code>    if floor(nci/2) &lt; ceil(nci/2)</code>
16	<code>        m = -nci; tr=1/nci;</code>
17	<code>        for j = 1:nci % Definition of 1 element wide vertical strip</code>
18	<code>            m = m+nci;for i =((nci+1)/2):((nci+1)/2);co(m+i)=k*fa;end% (W)</code>
19	<code>        end</code>
20	<code>    else % nci is even</code>
21	<code>        m = -nci; tr=2/nci;</code>
22	<code>        for j = 1:nci % Definition of 2 elements wide vertical strip</code>
23	<code>            m = m+nci;for i =(nci/2):(nci/2+1);co(m+i)=k*fa;end% (W)</code>
24	<code>        end</code>
25	<code>    end</code>
26	<code>end</code>
27	<code>if Ai == 4 % nci is odd</code>
28	<code>    if floor(nci/2) &lt; ceil(nci/2)</code>
29	<code>        m = -nci; tr=3/nci;</code>
30	<code>        for j = 1:nci % Definition of 3 elements wide vertical strip</code>
31	<code>            m=m+nci;for i =((nci+1)/2-1):((nci+1)/2+1);co(m+i)=k*fa;end% (W)</code>
32	<code>        end</code>
33	<code>    else % nci is even</code>
34	<code>        m = -nci; tr=4/nci;</code>
35	<code>        for j = 1:nci % Definition of 4 elements wide vertical strip</code>
36	<code>            m = m+nci;for i =(nci/2-1):(nci/2+2);co(m+i)=k*fa;end% (W)</code>
37	<code>        end</code>
38	<code>    end</code>
39	<code>end</code>
40	<code>if deb == 1 % Isotherms</code>
41	<code>    figure;nu=0;% colormap(gra_cob)</code>
42	<code>    for i=1:nel</code>
43	<code>        nu=nu+1;fill(xy(lK(i,:),1)',xy(lK(i,:),2)',co(nu));hold on</code>
44	<code>    end</code>
45	<code>    colorbar;axis equal;axis off</code>
46	<code>end</code>
47	<code>disp(['Lm 47, Anis. index : ',num2str(Ai)])</code>
48	<code>disp(['Lm 48, k &amp; coef*k : ',num2str([k k*fa]),' W/(m K)'])</code>
49	<code>disp(['Lm 49, Thickn. ratio: ',num2str(tr)])</code>
50	<code>if nel &lt; 50;disp(['Lm 26, Anis. vector : ',num2str(co')]);end</code>
51	<code>end</code>

Table 69: Matlab<sup>©</sup> function *mat\_cok.m* - non homogeneous conductivity

## 8.6 List of Matlab<sup>©</sup> procedures and functions

<i>Fia_20221023.m</i>	<i>Table 1</i>
<i>Fiam_20221023.m</i>	<i>Table 2</i>
<i>P_flg.m</i>	<i>Table 64</i>
<i>Fiammetta.m</i>	<i>Table 28</i>
<i>cad_ban.m</i>	<i>Table 37</i>
<i>cad_Dir.m</i>	<i>Table 32</i>
<i>cad_Neu.m</i>	<i>Table 33</i>
<i>cad_con.m</i>	<i>Table 34</i>
<i>cad_edg.m</i>	<i>Table 36</i>
<i>cad_gin.m</i>	<i>Table 31</i>

<i>cad_mes.m</i>	<i>Table 35</i>
<i>fem_Cae.m</i>	<i>Table 43</i>
<i>fem_Kco.m</i>	<i>Table 44</i>
<i>fem_Kcr.m</i>	<i>Table 46</i>
<i>fem_Kcv.m</i>	<i>Table 45</i>
<i>fem_Kra.m</i>	<i>Table 47</i>
<i>fem_rsm.m</i>	<i>Table 48</i>
<i>fem_smd.m</i>	<i>Table 38</i>
<i>fem_smt.m</i>	<i>Table 39</i>
<i>fem_smq.m</i>	<i>Table 40</i>
<i>fem_smc.m</i>	<i>Table 41</i>
<i>fem_tra.m</i>	<i>Table 49</i>
<i>geo_baf.m</i>	<i>Table 65</i>
<i>geo_stf.m</i>	<i>Table 66</i>
<i>geo_vfc.m</i>	<i>Table 67</i>
<i>geo_yfr.m</i>	<i>Table 68</i>
<i>gra_atg.m</i>	<i>Table 56</i>
<i>gra_ahf.m</i>	<i>Table 57</i>
<i>gra_chf.m</i>	<i>Table 58</i>
<i>gra_cob.m</i>	<i>Table 59</i>
<i>gra_hie.m</i>	<i>Table 62</i>
<i>gra_ipa.m</i>	<i>Table 60</i>
<i>gra_ist.m</i>	<i>Table 51</i>
<i>gra_lin.m</i>	<i>Table 61</i>
<i>gra_mel.m</i>	<i>Table 52</i>
<i>gra_mnl.m</i>	<i>Table 53</i>
<i>gra_tev.m</i>	<i>Table 63</i>
<i>gra_tra.m</i>	<i>Table 54</i>
<i>gra_2dm.m</i>	<i>Table 55</i>
<i>mat_cok.m</i>	<i>Table 69</i>

*Table 70: Matlab<sup>®</sup> procedures and functions used in Fiammetta*

## 8.7 Exercises proposed in 2020

Steady State Heat Transfer

Exercise 2: Building thermal bridges

Define a vertical strip of insulating material and a horizontal strip of highly conductive material, the intersection of which is at the center of the domain. This intersection will first be considered as insulating, then as very conductive. The boundary conditions are: a temperature of 273 *K* on the left and 298 *K* on the right, the upper and lower sides being considered adiabatic. Examine gradients and fluxes in the domain.

Coons Patch Based Structured Mesh

Exercise 3: Free convection node in a cavity

We consider a square cavity two meters side surrounded by a wall 0.2 *m* thick. The outside air temperature is 273 *K*. The indoor air temperature is free. Play on the convection coefficients and on the conductivity of the wall. The lower side (the ground) is considered at temperature of 293 *K*; you can also apply a constant flux (heated floor). Note that the radiative aspects are not taken into account in this exercise.

## Transient Heat Transfer

### Exercise 4: Balconies and cooling fins

A concrete slab crosses an exterior wall to form a balcony. The outside temperature varies according to a sinusoidal function (273  $K$  at midnight, 288  $K$  at noon). The interior temperature is left free. The initial temperature is equal to 273  $K$ . Show how it varies, with a delay which depends, in particular, on thermal capacities. Note that the radiative aspects are not taken into account in this exercise.

## Gray Body Radiation

### Exercise 5: Radiation through a cavity

A flow of heat passes through a concrete block in the center of which is a cavity. Show how the emissivity of the interior walls of the cavity affects the flows and temperatures in the concrete block. What happens for a zero or unitary emissivity?

## 8.8 Exercises proposed in 2021

### Steady state heat transfer including conduction and convection

Exercise 2: Same problem as exercise 1, but convection elements are present on a part of the boundary and the Dirichlet boundary conditions are applied only on the virtual convective nodes so that all the nodes of the domain are free.

### Coons Patch Based Structured Mesh

### Exercise 3: Utilization of the Matlab<sup>®</sup> procedure Fiammetta

Using the same domain shape as in the first exercise, it is asked to reproduce the same boundary conditions now applied on the patches sides and to study the convergence when the number of variables is increasing. Plot the convergence curve in logarithmic coordinates. A comment about the shape of this curve is welcome.

## Transient Heat Transfer

### Exercise 4: Heating and cooling fins

A domain has the same shape as the capital letter E, with thick vertical part and very thin horizontal ones. These parts are immersed in three fluids with low temperature on the top and bottom parts. The middle part is immersed in a high temperature fluid. All the temperatures of the mesh are free and the boundaries of the vertical part are adiabatic. Both isotherm and heat flows graphics have to be computed and displayed, the first from a very fine mesh and the second with a relatively coarse mesh.

## Gray Body Radiation

### Exercise 5: Radiation through a cavity

A heat flow is crossing a concrete block in the center of which is a cavity. Show how the emissivity of the interior walls of the cavity affects the heat flows and temperatures in the concrete block. What happens in extreme situations where the emissivity is zero or equal to one?

## 9. References

[Barlow 1976] Barlow John, “Optimal stress locations in finite element models”, International Journal for Numerical Methods in Engineering, Vol 10, **1976**, pages. 243-251.

[Beckers *et al*, 2009] Beckers Benoit. Masset Luc. & Beckers Pierre, “Commentaires sur l’analogie de Nusselt”, Rapport Heli 004 fr, **2009**,  
<http://www.heliodon.net/heliodon/documents.html>.

[Beckers 2011] Beckers Benoit, “Urban outlines 2D abstraction for flexible and comprehensive analysis of thermal exchanges”, Conférence Internationale Scientifique pour le Bâtiment CISBAT 2011, EPFL, September **2011**, Lausanne, Suisse,  
<http://heliodon.net/heliodon/references.html>.

[Beckers & Beckers 2012] Beckers Benoit, Beckers Pierre, “Radiative Simulation Methods”, in Solar Energy at Urban Scale, chap. 10, Ed. B. Beckers, John Wiley and Sons, Inc., pp 205-236, **2012**.

[Beckers 2013] Beckers Benoit, “Taking Advantage of Low Radiative Coupling in 3D Urban Models”, Eurographics Workshop on Urban Data Modelling and Visualization, May 6 -10, **2013**, Girona, Spain.

[Beckers & Beckers 2014] Beckers Benoit, Beckers Pierre, “Reconciliation of Geometry and Perception in Radiation Physics”, Focus Series in Numerical Methods in Engineering, Wiley-ISTE, 192 pages, July **2014**.

[Beckers & Beckers 2015] Beckers Pierre, Beckers Benoit, “A 66-line heat transfer finite element code to highlight the dual approach”, *Computers & Mathematics with Applications*, Volume 70 Issue 10, November **2015**, pages 2401 - 2413.

[Beckers & Beckers 2016] Beckers Pierre, Beckers Benoit, “A 33-line heat transfer finite element code”, *Report Helio\_010\_en*, **2016**. [www.heliodon.net/heliodon/documents.html](http://www.heliodon.net/heliodon/documents.html)

[Beckers 2016] Beckers Benoit, “Multiscale Analysis as a Central Component of Urban Physics Modeling”, In *Computational Methods for Solids and Fluids* (pp. pp 1-27), Volume 41 of the series Computational Methods in Applied Sciences, Springer International Publishing; Ed. Adnan Ibrahimbegovic, February **2016**, DOI:[10.1007/978-3-319-27996-1\\_1](https://doi.org/10.1007/978-3-319-27996-1_1)

[Beckers 2017] Beckers Benoit, “Géométrie assistée par ordinateur”, *Architecture et Physique Urbaine - ISA BTP Université de Pau et des Pays de l'Adour*, **2017**.

<http://heliodon.net/geometry/references.html>

[Beckers 2019] Beckers Benoit, “Five Lectures on Finite Element Method Applied to Heat Transfer”, **2019**, <http://heliodon.net/heliodon/documents>

[Coons 1967] Coons Steven A., “Surfaces for Computer-Aided Design of Space Forms”, Project MAC-TR-41, Massachusetts Institute of Technology, **1967**.

[Coulon 2006] Coulon, N., “Nouvel algorithme pour traiter le rayonnement thermique en milieu transparent dans Cast3m”, Rapport Technique, Commissariat à l'énergie atomique, **2006**.

[Courant 1943] Courant Richard, “Variational methods for solution of problems of equilibrium and vibrations”, Bull. Amer. Math. Soc. 49 (**1943**), no. 1, 1-23.

[Courant & Hilbert 1953] Courant Richard, Hilbert David, “Methods of Mathematical Physics”, Volume 1, Library of Congress Catalog Card Number 53-7164, ISBN 0 470 17952 X, **1953**.

[Debongnie, Zhong & Beckers 1995] Debongnie Jean-François, Zhong Hai Guang. & Beckers Pierre, “Dual Analysis with General boundary conditions”, Comput. Methods Appl. Mech. Engrg. 122 (**1995**) 183-192.

[Debongnie & Beckers 2001] Debongnie Jean-François, Beckers Pierre, “On a general decomposition of the error of an approximate stress field in elasticity”, Computer Assisted Mechanics and Engineering Sciences, 8; 261-270, **2001**.

[Debongnie] Debongnie Jean-François, “Fundamentals of finite elements”, Les Editions de l’Université de Liège, **2003**.

[Ergatoudis, Irons & Zienkiewicz 1968] Ergatoudis I., Irons Bruce M., Zienkiewicz Oleg C., “Curved, Isoparametric, “Quadrilateral” elements for finite element analysis”, Int. J. Solids Structures. **1968**, Vol. 4, pp. 31 to 42.

[Fish, Belytschko 2007] Fish Jacob, Belytschko Ted, “A First Course in Finite Elements”, (Wiley, **2007**).

[Fraeijs de Veubeke et al 1972] Fraeijs de Veubeke Baudouin, Sander Guy, Beckers Pierre, “Dual analysis by finite elements linear and nonlinear applications”, AFFDL TR 72\_93, **1972**.

[Fraeijs de Veubeke & Hogge 1972] Fraeijs de Veubeke Baudouin, Hogge Michel, “Dual Analysis for Heat Conduction Problems by Finite Elements”, International Journal for Numerical Methods in Engineering, vol. 5, 65-82 (**1972**).

[Fraeijs de Veubeke et al 1977] Fraeijs de Veubeke Baudouin, Beckers Pierre, Canales Edgardo, Galaz Sergio, “Principios variacionales en conducción de calor”, Informe del departamento de Ingeniería Mecánica de la Escuela de Ingeniería, Universidad de Concepción, Chile, **1977**.

[Goral et al 1984] Goral Cindy M., Torrance Kenneth E., Greenberg Donald P., Battaile Bennett, “Modeling the Interaction of Light Between Diffuse Surfaces”, Computer Graphics 18(3): 213-222, July **1984**.

[Irons 1966] Irons Bruce, “Numerical integration applied to finite element methods”, *Int. Symposium on the Use of Digital Computers in Structural Engineering*, University of Newcastle upon Tyne, July **1966**. (“This was a complete résumé of my ideas on isoparametric elements. Unfortunately, the organizers required that the length be halved, thus excluding the section on large deflections, etc.”).

- [Lee & Jackson 1976] Lee Hwa-Ping, Jackson Clifton C., “Finite Element Solution for Radiative-Conductive Analyses with mixed diffuse specular radiation”, AIAA, **1976**
- [Lee 1977] Lee Hwa-Ping, “Nastran Thermal Analyzer – Theory and Application Including a Guide to Modeling Engineering Problems”, Report NASA TM X-3503, **1977**
- [Lee & Mason 2008] Lee Hwa-Ping, Mason James B., “Nastran Thermal Analyzer A general purpose finite-element heat transfer computer program”, **2008**
- [Lewis et al 2004] Lewis Roland W., Nithiarasu Perumal, Seetharamu Kankanhally N., “Fundamentals of the Finite Element Method for Heat and Fluid Flow”, John Wiley & Sons Ltd, **2004**, p. 356.
- [Lobo & Emery 1995] Lobo M., Emery A. F, “Use of the discrete maximum principle in Finite-Element analysis of combined conduction and radiation in nonparticipating media”, Numerical Heat Transfer, Part B: Fundamentals: An International Journal of Computation and Methodology, 27:4, 447 - 465, **1995**.
- [Nusselt 1928] Nusselt Wilhelm, Graphische Bestimmung des Winkel Verhältnisses bei der Wärmestrahlung, Zeitschrift des Vereines Deutscher Ingenieure, 72(20):673 **1928**, see [Beckers et al, 2009]
- [Rupp & Péniguel 1999] Rupp I., Péniguel C., (1999), "Coupling heat conduction, radiation and convection in complex geometries", International Journal of Numerical Methods for Heat & Fluid Flow, Vol. 9 Iss: 3 pp. 240 - 264
- [Sander & Beckers 1977] Sander Guy, Beckers Pierre, “The influence of the choice of connectors in the finite element method”, International Journal for Numerical Methods in Engineering, vol. 11, 1491 - 1505 (**1977**).
- [Siemens 2017] Siemens Thermal Analysis User's guide, © **2017** Siemens Product Lifecycle Management Software Inc. All Rights Reserved.
- [Sillion & Puech 1994] Sillion François, Puech Claude, “Radiosity and Global Illumination”, Morgan Kauffman Publishers Inc. **1994**.
- [van Eekelen 2012] van Eekelen Tom, “Radiation Modeling Using the Finite Element Method”, in Solar Energy at Urban Scale, ed. Beckers Benoit, ISTE, **2012**
- [Szabó & Babuska 1991] Szabó Barna, Babuska Ivo, “Finite element analysis”, John Wiley & sons, **1991**.
- [Zienkiewicz 1971] Zienkiewicz Oleg C., “The Finite Element Method in Engineering Science”, McGraw-Hill, London, **1971**.

## 10. List of tables and figures

<i>Table 1: Compact Matlab<sup>®</sup> procedure Fia_20230130.m .....</i>	5
<i>Table 2: Matlab<sup>®</sup> procedure Fiammetta_33_20230130.m - Conduction problems .....</i>	7
<i>Table 3: Instructions substituted to lines 13 to 16 in Fiammetta_33_20230130.m .....</i>	9
<i>Table 4: Format of displayed output .....</i>	14
<i>Table 5: Temperatures and heat flows as functions of Biot number .....</i>	16
<i>Table 6: Coons patch definition in Cartesian and parametric spaces .....</i>	22
<i>Table 7: Numerically integrated conductivity matrix .....</i>	24
<i>Table 8: Instructions used to display input data of Figure 30 .....</i>	24
<i>Table 9: Instructions to identify nodes along patch sides .....</i>	25
<i>Table 10: Numerical integration of the capacity matrix .....</i>	34
<i>Table 11: Matrix F &amp; reciprocity in a square cavity – 4 segments, 1 Gauss point .....</i>	48
<i>Table 12: Matrix F &amp; reciprocity in a square cavity – 4 segments, 2 Gauss points .....</i>	48
<i>Table 13: Matrix F &amp; reciprocity in a square cavity – 8 segments, 1 Gauss point .....</i>	48
<i>Table 14: Matrix F &amp; reciprocity in a square cavity – 8 segments, 2 Gauss points .....</i>	49
<i>Table 15: Rectangular cavity: F and reciprocity test– 4 segments, 1 Gauss point .....</i>	49
<i>Table 16: Rectangular cavity: F and reciprocity test– 4 segments, 2 Gauss points .....</i>	49
<i>Table 17: Rectangular cavity: F and reciprocity test - 8 segments, 1 Gauss point .....</i>	50
<i>Table 18: Rectangular cavity: F and reciprocity test - 8 segments, 2 Gauss points .....</i>	50
<i>Table 19: View factor matrices for street section (top) and rectangular cavity (bottom) .....</i>	51
<i>Table 20: View factor matrix and sky view factor vector - street section (6 segments) .....</i>	52
<i>Table 21: View factor matrix and sky view factor vector - street section (9 segments) .....</i>	52
<i>Table 22: View factor matrix F around a balcony – 10 segments +ground + sky .....</i>	53
<i>Table 23: Radiosity matrix of an adiabatic (<math>\rho = 1</math>) square cavity – 8 segments,.....</i>	54
<i>Table 24: View factor and radiosity matrices of a street section, <math>\rho = 0</math> .....</i>	55
<i>Table 25: Radiosity matrices of a street section, <math>\rho = 1</math> .....</i>	55
<i>Table 26: Sky view factor – uni-column matrix Fsky of a street section .....</i>	55
<i>Table 27: Influence of emissivity when using one virtual radiative node, .....</i>	59
<i>Table 28: Matlab<sup>®</sup> procedure Fiammetta.m .....</i>	86
<i>Table 29: Scheme of the procedure Fiammetta.m (Table 28) .....</i>	86
<i>Table 30: Matlab<sup>®</sup> variables used in the procedure Fiammetta.m .....</i>	89
<i>Table 31: Matlab<sup>®</sup> function cad_gin.m - input data - definition of domains .....</i>	93
<i>Table 32: Matlab<sup>®</sup> function cad_Dir.m - Dirichlet boundary conditions .....</i>	95
<i>Table 33: Matlab<sup>®</sup> function cad_Neu.m - Neumann boundary conditions, function .....</i>	96
<i>Table 34: Matlab<sup>®</sup> function cad_con.m - localization of convection elements .....</i>	99
<i>Table 35: Matlab<sup>®</sup> function cad_mes.m - construction of the CAD mesh topology .....</i>	100
<i>Table 36: Matlab<sup>®</sup> function cad_edg.m – definition of the CAD mesh interfaces .....</i>	100
<i>Table 37: Matlab<sup>®</sup> function cad_ban.m – radiative nodes of cavity, street or balcony .....</i>	101
<i>Table 38: Matlab<sup>®</sup> function fem_smd.m - solution of the nonlinear radiative system .....</i>	102
<i>Table 39: Matlab<sup>®</sup> function fem_smt.m – solution of linear transient equations .....</i>	103
<i>Table 40: Matlab<sup>®</sup> function fem_smq.m – solution of nonlinear transient equations .....</i>	105
<i>Table 41: Matlab<sup>®</sup> function fem_smc.m – Cavity, VF matrices, radiative transfers .....</i>	107
<i>Table 42: Matlab<sup>®</sup> variables used in the function fem_smc.m .....</i>	109
<i>Table 43: Matlab<sup>®</sup> function fem_Cae.m – capacity matrix of a quadrilateral .....</i>	109
<i>Table 44: Matlab<sup>®</sup> function fem_Kco.m – element conduction matrix .....</i>	109
<i>Table 45: Matlab<sup>®</sup> function fem_Kcv.m – 3 x 3 element convection matrix .....</i>	109
<i>Table 46: Matlab<sup>®</sup> function fem_Kcr.m – 3 x 3 element radiation matrix .....</i>	110
<i>Table 47: Matlab<sup>®</sup> function fem_Kra.m – Inter elements view factors .....</i>	110
<i>Table 48: Matlab<sup>®</sup> function fem_rsm.m – second member in a radiative section .....</i>	111
<i>Table 49: Matlab<sup>®</sup> function fem_tra.m – solution of the linear transient equations .....</i>	111
<i>Table 50: Postprocessing functions .....</i>	113
<i>Table 51: Matlab<sup>®</sup> function gra_ist.m - isotherm drawing in a nx x ny mesh .....</i>	113
<i>Table 52: Matlab<sup>®</sup> function gra_mel.m - drawing a shrunk mesh .....</i>	113
<i>Table 53: Matlab<sup>®</sup> function gra_mnl.m - displaying node &amp; element labels .....</i>	113

Table 54: Matlab <sup>®</sup> function <i>gra_trm.m</i> - Temperature along radiative border.....	114
Table 55: Matlab <sup>®</sup> function <i>gra_2dm</i> – Second member along radiative border.....	114
Table 56: Matlab <sup>®</sup> function <i>gra_atg.m</i> - visualization of temperature gradient arrows .....	114
Table 57: Matlab <sup>®</sup> function <i>gra_ahf.m</i> - visualization of heat flow arrows.....	115
Table 58: Matlab <sup>®</sup> function <i>gra_chf.m</i> - visualization of heat flow arrows.....	115
Table 59: Matlab <sup>®</sup> function <i>gra_cob.m</i> - color bar.....	116
Table 60: Matlab <sup>®</sup> function <i>gra_ipa.m</i> - drawing isotherms in a Coons patch .....	116
Table 61: Matlab <sup>®</sup> function <i>gra_lin.m</i> - visualization of the levels of a scalar function.....	116
Table 62: Matlab <sup>®</sup> function <i>gra_hie.m</i> – visualization of a periodic scalar field .....	116
Table 63: Matlab <sup>®</sup> function <i>gra_tev.m</i> – temperature evolution.....	117
Table 64: Matlab <sup>®</sup> procedure <i>P_flg.m</i> - heat flows & temperature gradients .....	117
Table 65: Matlab <sup>®</sup> function <i>geo_baf.m</i> – view factor matrix – wall with balcony.....	118
Table 66: Matlab <sup>®</sup> function <i>geo_stf.m</i> – view factor matrix of a street section .....	119
Table 67: Matlab <sup>®</sup> function <i>geo_vfc.m</i> – view factor matrix – ns sides domain .....	119
Table 68: Matlab <sup>®</sup> function <i>geo_vfr.m</i> – view factor matrix – 2 Gauss points .....	119
Table 69: Matlab <sup>®</sup> function <i>mat_cok.m</i> - non homogeneous conductivity.....	120
Table 70: Matlab <sup>®</sup> procedures and functions used in Fiammetta .....	121

Figure 1: A rectangular element and its conductivity matrix .....	2
Figure 2: A square element and its conductivity matrix .....	3
Figure 3: Mesh with nodes & elements labels .....	3
Figure 4: Result of the compact procedures 1 & 2 with different color maps .....	5
Figure 5: Finite element mesh with nodes and elements labels .....	5
Figure 6: Example with imposed temperatures producing a vertical gradient.....	7
Figure 7: Imposed temp. on fragments of the horizontal faces ( $r = 6$ , $nx = 5$ & $r = 2$ , $nx = 50$ ) .....	8
Figure 8: Isocurves for the problem with prescribed heat flows .....	9
Figure 9: Exercise 1: mesh & isotherms .....	11
Figure 10: Isotherms orthogonal to both adiabatic boundaries. Biot number = 1 .....	14
Figure 11: Nodes and elements numbering: heat flows with convective boundary conditions .....	15
Figure 12: Example of heat transfer through a wall with a high Biot number $\beta = 18$ .....	17
Figure 13: Isocurves in a rectangle with fixed DOF on horizontal edges, isotropic material .....	17
Figure 14: Isocurves for material with vertical strip, 0.1 k (variable $fa = .1$ ) .....	18
Figure 15: Isocurves for the vertical strip 2 elements wide, 16 x 16 mesh .....	18
Figure 16: Heat flows and temperature gradients for a vertical thermal bridge .....	19
Figure 17: Isocurves in presence of a vertical small conductivity strip (0.1 k) .....	19
Figure 18: Heat flows and temperature gradients (vertical strip with small conductivity) .....	19
Figure 19: Visualizations of scalars defined element by element .....	20
Figure 20: Isocurves - horizontal thermal bridge with high conductivity (1000 k) .....	20
Figure 21: Heat flows and temperature gradients (horizontal strip with high conductivity) .....	21
Figure 22: CAD data defining a domain surrounding a cavity .....	25
Figure 23: Finite element mesh corresponding to the CAD definition of Figure 22 .....	25
Figure 24: Heat flow around an adiabatic square cavity .....	26
Figure 25: Heat flow around a perfectly reflective cavity .....	27
Figure 26: Cad model with nodes and patches labels .....	27
Figure 27: F.E. mesh with nodes and elements labels .....	27
Figure 28: CAD model with diagonal on the long horizontal side .....	28
Figure 29: CAD model fully based on rectangular patches.....	29
Figure 30: CAD model based on 3 trapezoidal patches, 75 elements .....	29
Figure 31: 2 imposed temperatures, 2 adiabatic faces in a trapezoidal domain .....	30
Figure 32: Non homogeneous trapezoidal domain .....	31
Figure 33: Horizontal strip with high conductivity in a trapezoidal domain .....	31
Figure 34: Smoothing process convergence in temperature homogenization .....	36
Figure 35: Evolution of the temperature field in a smoothing process .....	36
Figure 36: Evolution of the temperature field in a heating process .....	37
Figure 37: Evolution of specific temperatures in a heating operation .....	37

Figure 38: Evolution of isotherms in a heating operation: 100 h.....	38
Figure 39: Adiabatic cavity, 1 month, $T_{ini} = 280 \text{ K}$ , $T_{top} = 300 \text{ K}$ .....	39
Figure 40: Adiabatic cavity, evolution of the temperature field .....	39
Figure 41: Isotherms, convection in the cavity, 1 month .....	40
Figure 42: Thermal loading over a period of 10 days (240 hours) .....	41
Figure 43: 2D “point – segment” view factor [Beckers 2011].....	43
Figure 44: Rectangular cavity with the display of four matrices describing the geometry .....	47
Figure 45: Data – $Gi = 14$ , $Gi = 15$ : Street section, aspect ratio $dx/dy = 3/7$ .....	51
Figure 46: Vertical wall above horizontal edge.....	52
Figure 47: Vertical wall below horizontal edge.....	52
Figure 48: Thermal bridge data: $Gi = 16, 18, 19, 20, 21, 22$ .....	53
Figure 49: Fsky in a street with balcony – 16 segments / patch side.....	53
Figure 50: Fgr in a street with balcony – 16 segments / patch side .....	54
Figure 51: Sky View Factors in the street section – 200 radiative segments per side .....	55
Figure 52: Transient heat transfer in rectangular domain with two convective walls. ....	56
Figure 53: A simple convex domain, CAD definition and finite element mesh.....	57
Figure 54: Rectangle with one radiative wall (left) and one convective wall (right) .....	58
Figure 55: Isotherms and heat flows through a cavity, 1 virtual radiative node, black body.....	58
Figure 56: Isotherms and heat flows, gray body & mirror, 30 days, 1 virtual radiative node .....	59
Figure 57: Isotherms of radiative exchanges, 256 elements, 1-month, black body, $\rho = 0$ .....	60
Figure 58 : Flux de chaleur nodaux le long de la cavité (gra_2dm.m, Table 55) .....	60
Figure 59: Grad., H.F. & T. after 30 days, left: black body, $\rho = 0$ ; right: mirror, $\rho = 1$ .....	61
Figure 60: Cavity, $\rho = 0$ (Figure 57), but 1224 DOF, nni = 16, (gra_tra.m, Table 54) .....	61
Figure 61: Isotherms, radiation, 4 patches, 800 elements, 1 month, $\rho = 0.5$ .....	63
Figure 62: Isotherms, radiation, 8 patches, 200 elements, 1 month, $\rho = 0.5$ .....	63
Figure 63: Isotherms, gray body, evolution after 60, 120, ..., 660 hours, $\rho = 0.5$ .....	64
Figure 64: Isotherms and heat flows, black, gray body & mirror, 30 days .....	65
Figure 65: Adiabatic rectangular cavity, method 3, 2 virt. conv. nodes .....	66
Figure 66: Adiabatic rectangular cavity, isotherms & heat flows after 180 hours .....	66
Figure 67: Cavity with adiabatic boundary $\varepsilon = 0$ , $\rho = 1$ .....	67
Figure 68: Street section – adiabatic walls - 225 DOF .....	68
Figure 69: Adiabatic Street Section .....	69
Figure 70: Temperature evolution, adiabatic walls, 3 days, 32 elements per side .....	69
Figure 71: Identity between adiabatic and perfectly reflective walls ( $\rho = 1$ ).....	70
Figure 72: Street section, comparison of two low emissivities: 0 (left), 0.05 (right) .....	71
Figure 73: Street section, comparison of two low emissivities: 0.1 (left), 0.15 (right) .....	71
Figure 74: Street section, $\rho = 0.5$ , sky loads and wall temperatures after 72 hours .....	72
Figure 75: Street section, $\rho = 0.5$ , injected heat: 16.4 MJ, sky temperature = 280 K .....	73
Figure 76: Street section with black body walls and injected heat of 100 Wm <sup>-2</sup> , 16.4 MJ.....	74
Figure 77: Street section with black body walls and without injected heat .....	75
Figure 78: Thermal bridge, topological configuration .....	75
Figure 79: Thermal bridge with 2 convective virtual nodes, steady state.....	76
Figure 80: Thermal bridge, mixed b. c., 1 convective virtual node, 1 radiative v. node ( $\rho = .5$ ).....	77
Figure 81: Thermal bridge, transient analysis, 2 convective virtual nodes .....	78
Figure 82: Thermal bridge, transient analysis, 1 convective & 1 radiative virt. node, $\rho = .5$ .....	79
Figure 83: Thermal bridge comparison, radiative virtual node – radiative edge, $\rho = 1$ .....	80
Figure 84: Thermal bridge comparison, radiative virtual node – radiative edge, $\rho = .5$ .....	81
Figure 85: Thermal bridge, transient analysis, 1 convective node & 1 radiative side .....	82

## 11. Content

1.	<i>Tutorial I: Basic problem of thermal conduction .....</i>	2
1.1	<i>Basic finite element model.....</i>	2
1.2	<i>Innovative handling of the Dirichlet boundary conditions.....</i>	6
1.3	<i>Dirichlet boundary conditions.....</i>	8
1.4	<i>Neumann boundary conditions.....</i>	8
1.5	<i>Explanation of the Matlab<sup>®</sup> procedure Fiammetta_33_20230130.m .....</i>	10
1.6	<i>Exercises.....</i>	10
	<i>Tutorial II: Convection .....</i>	11
2.1	<i>Formulation of the convection.....</i>	11
2.2	<i>Two convective and two adiabatic faces .....</i>	14
2.3	<i>Material anisotropy.....</i>	17
2.4	<i>Thermal horizontal bridge.....</i>	20
2.	<i>Tutorial III: Structured mesh based on Coons' patch.....</i>	21
3.1	<i>Numerical evaluation of the temperature gradient in a Coons patch .....</i>	21
3.2	<i>CAD model of the domain to be analyzed .....</i>	24
3.3	<i>Identification of the DOF pertaining to a patch side .....</i>	24
3.4	<i>Cavity with adiabatic hole.....</i>	25
3.5	<i>C shaped domain .....</i>	27
3.6	<i>Boundary conditions.....</i>	29
3.7	<i>Basic isotherm drawing.....</i>	30
3.8	<i>Thermal bridge in a trapezoidal domain.....</i>	30
3.	<i>Tutorial IV: Transient heat transfer.....</i>	32
4.1	<i>Solution of the transient problem .....</i>	32
4.2	<i>Capacity matrix .....</i>	33
4.3	<i>Temperature evolution.....</i>	35
4.4	<i>Temperature homogenization in an insulated solid .....</i>	35
4.5	<i>Heating of a solid at initial uniform temperature.....</i>	37
4.6	<i>Adiabatic cavity with imposed temperatures on the top of the domain.....</i>	38
4.7	<i>Square cavity with convection .....</i>	39
4.8	<i>Periodic imposed heat flow .....</i>	41
4.	<i>Tutorial V: Radiosity .....</i>	42
5.1	<i>Theoretical background.....</i>	42
5.2	<i>View factor matrix .....</i>	47
→ a)	<i>Quadrilateral cavities .....</i>	47
→ b)	<i>Street section .....</i>	50
→ c)	<i>L shape configurations .....</i>	52
→ d)	<i>Thermal bridge.....</i>	52

<i>5.3 Radiosity matrix</i> .....	54
→ a) <i>Rectangular cavity</i> .....	54
→ b) <i>Street section</i> .....	54
<i>5. Tutorial VI: Radiative heat exchanges</i> .....	56
<i>6.1 A simple convex domain</i> .....	56
<i>6.2 Square cavity</i> .....	58
a) <i>Virtual radiative node concept</i> .....	58
b) <i>Radiosity method, black body square cavity <math>\rho = 0</math></i> .....	60
c) <i>Radiosity method, gray body square cavity</i> .....	62
d) <i>Radiosity method, comparison of gray cavities</i> .....	64
<i>6.3 Rectangular cavity</i> .....	66
<i>6.4 Street section</i> .....	68
<i>6.4.1 Periodic heat loading</i> .....	68
<i>6.4.2 Adiabatic or pure reflective walls, <math>\rho = 1</math></i> .....	68
<i>6.4.3 Variation of the street walls emissivity</i> .....	70
<i>6.4.4 Black body walls, <math>\rho = 0</math></i> .....	73
<i>5.5 Thermal bridge</i> .....	75
<i>5.5.1 Stationary heat flow - 2 convective virtual nodes</i> .....	75
<i>6.5.2 Transient heat exchanges</i> .....	78
a. <i>Two convective virtual nodes</i> .....	78
b. <i>One convective and one radiative virtual node</i> .....	78
c. <i>Radiative edge</i> .....	79
<i>6. Conclusion</i> .....	82
<i>8. Additional information on functions and algorithms</i> .....	82
<i>8.1 Main procedure: Fiammetta.m</i> .....	82
<i>8.2 Input data functions</i> .....	89
<i>8.3 Transient heat transfer</i> .....	101
<i>8.4 Postprocessing</i> .....	111
<i>8.5 Additional procedures</i> .....	117
<i>8.6 List of Matlab<sup>®</sup> procedures and functions</i> .....	120
<i>8.7 Exercises proposed in 2020</i> .....	121
<i>8.8 Exercises proposed in 2021</i> .....	122
<i>9. References</i> .....	122
<i>10. List of tables and figures</i> .....	125
<i>11. Content</i> .....	128
<i>12. End</i> .....	129

## 12. End