Fiammetta, Finite Element Method Applied to Heat Transfer

– Tutorials –

Benoit Beckers
August 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©, 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 Partial differential equations of heat transfer in a solid

The Fourier equation links the temperature field $\tau\,(K)$ to the heat flow vector $\vec{q}$ expressed in Wm^{-2}. This flow is proportional to the temperature gradient and the thermal conductivity k expressed in $Wm^{-1}K^{-1}$.

$$\vec{q} = -k\,\vec{\nabla}\tau \tag{1}$$

If there is no heat sink or source, the heat flow verifies the continuity equation (or equilibrium equation).

$$\vec{\nabla}.\vec{q} = 0 \tag{2}$$

When the conductivity coefficient is constant, the continuity equation expressed in terms of temperature becomes the Laplacian.

$$\Delta\tau = 0 \tag{3}$$

The basic boundary conditions concern the temperatures fixed on the part S_1 of the boundary and the heat fluxes imposed on its complement S_2. Their union represents the boundary S of the domain.

$$\begin{array}{lcl} \tau & = & \bar{\tau} \quad on \quad S_1 \\ \vec{n}\cdot\vec{q} & = & \bar{q}_n \quad on \quad S_2 \\ S_1 \cup S_2 = S & & \end{array} \tag{4}$$

To compute the temperature field in a domain Ω, subjected to boundary conditions on the temperature and/or the heat flow, the following system of partial differential equations as well as the associated boundary conditions must be solved:

$$\begin{array}{l} \vec{\nabla}.\left(k\,\vec{\nabla}\tau\right) = 0 \quad in \quad \Omega \\ \tau \quad = \quad \bar{\tau} \quad on \quad S_1 \\ \vec{n}\cdot\vec{q} \quad = \quad \bar{q}_n \quad on \quad S_2 \end{array} \tag{5}$$

The boundary of the domain is divided into 2 parts:

S_1 where the temperature is imposed (Dirichlet boundary conditions),

S_2 where the normal heat flow is imposed (von Neumann boundary conditions).

A bar indicates that the concerned quantity is imposed.

1.2 Variational method

The variational method shows that solving the system of partial differential equations is equivalent to find the stationarity conditions of the functional (expressed in WK).

$$< I(\tau) = \int_{\Omega} \frac{1}{2} k\,\vec{\nabla}\tau\,.\,\vec{\nabla}\tau\,d\Omega + \int_{S_2} \bar{q}_n\,\tau\,dS > \qquad stationary \tag{6}$$

To express the stationarity condition, with the boundary conditions

$$\tau = \bar{\tau}, \qquad \delta\tau = 0 \; on \; S_1 \tag{7}$$

we write that the difference between the functional calculated with a small arbitrary variation $\delta\tau$ of the temperature and the present functional is equal to zero.

$$I(\tau + \delta\tau) - I(\tau) = 0 \tag{8}$$

After some developments, because the variation $\delta\tau$ is arbitrary, its coefficient must be zero, which allows writing the equations satisfied by the thermal field.

$$\begin{aligned} \vec{\nabla}.(k\,\vec{\nabla}(\tau)) &= 0 && in\ \Omega \\ -\vec{n}\cdot(k\,\vec{\nabla}\tau) &= \bar{q}_n && on\ S_2 \end{aligned} \tag{9}$$

Obviously, these equations are the same as in (5).

1.3 Finite element model

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

$$\tau_i = \sum_{j=1}^{4} T_{ij} f_{ij}(x,y) \tag{10}$$

Explicitly, for a rectangle of dimensions a x b, we have:

$$\tau_i = T_{i1}\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right) + T_{i2}\frac{x}{a}\left(1-\frac{y}{b}\right) + T_{i3}\frac{x}{a}\frac{y}{b} + T_{i4}\left(1-\frac{x}{a}\right)\frac{y}{b} \tag{11}$$

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

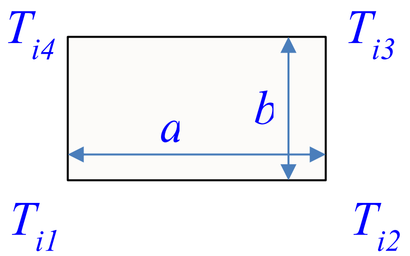
$$\begin{bmatrix} \frac{\partial\tau_i}{\partial x} \\ \frac{\partial\tau_i}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{1}{a}\left(\frac{y}{b}-1\right) & \frac{1}{a}\left(1-\frac{y}{b}\right) & \frac{y}{ab} & -\frac{y}{ab} \\ \frac{1}{b}\left(\frac{x}{a}-1\right) & -\frac{x}{ab} & \frac{x}{ab}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

$$\begin{bmatrix} \frac{\partial\tau_i}{\partial x} \\ \frac{\partial\tau_i}{\partial y} \end{bmatrix} = \frac{1}{ab}\begin{bmatrix} y-b & b-y & y & -y \\ x-a & -x & x & a-x \end{bmatrix} \begin{bmatrix} T_{i1} \\ T_{i2} \\ T_{i3} \\ T_{i4} \end{bmatrix} \tag{12}$$

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

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

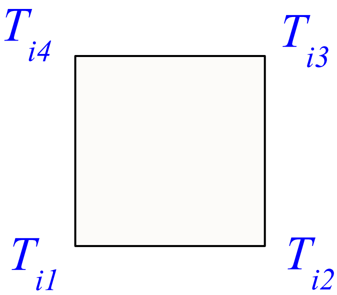
(13)



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

Figure 1: A rectangular element and its conductivity matrix

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



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

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 $I\,(T)$:

$$< I(T) = \sum_{i=1}^{nel} \Big(\int_{\Omega j} \frac{1}{2} k_i \; \vec{\nabla} \sum_{j=1}^{4} T_{ij} f_{ij} \,.\, \vec{\nabla} \sum_{j=1}^{4} T_{ij} f_{ij} \; d\Omega_i + \int_{S_{2i}} \bar{q}_n \; \sum_{j=1}^{4} T_{ij} f_{ij} \; dS_i \Big) > \tag{14}$$

After introducing the polynomial trial functions given in (10), we can write (14) in matrix form:

$$< I(T) = \sum_{i=1}^{nel} [T_i]^T [K_i][T_i] + [T_i]^T [F_i] > \tag{15}$$

The last term $[F_i]$ of (15) is the vector (uni-column matrix) 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*).

```
k=0; for j =5:-1:1;for i = 1:3
k=k+1;xy(k,1)=i-1;xy(k,2)=j-1;end;end

lK = [13    14    11    10
      14    15    12    11
      10    11     8     7
      11    12     9     8
       7     8     5     4
       8     9     6     5
       4     5     2     1
       5     6     3     2];

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

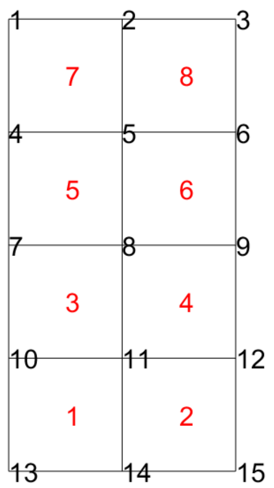


Figure 3: Mesh with nodes & elements labels

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

$$\frac{\partial \tau_i}{\partial x} = \frac{1}{2a}\left(\left(T_{i2}+T_{i3}\right)-\left(T_{i1}+T_{i4}\right)\right)$$
$$\frac{\partial \tau_i}{\partial y} = \frac{1}{2b}\left(\left(T_{i3}+T_{i4}\right)-\left(T_{i1}+T_{i2}\right)\right) \tag{16}$$

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 (1).

$$\begin{bmatrix} q_{xi} \\ q_{yi} \end{bmatrix} = -k_i \begin{bmatrix} \dfrac{\partial \tau_i}{\partial x} \\ \dfrac{\partial \tau_i}{\partial y} \end{bmatrix} \tag{17}$$

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 (or free) 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:

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} F_1 \\ R_2 \end{bmatrix} \tag{18}$$

The solution is:

$$[T_1] = [K_{11}]^{-1}\left([F_1]-[K_{12}][T_2]\right) \tag{19}$$

In the next example, the imposed second member [F_1] corresponding to von Neumann boundary conditions is equal to zero. The solution (19) without [F_1] is written explicitly in *line 14* of the compact *procedure 1* (*Table 1*).

Compact Matlab© procedure *Fia_20230130.m*

Compact procedure 1 (*Figure 4*, left)

```
1  k = 0;for i=1:3;for j=1:4;k=k+1;xy(k,:)=[i-1,j-1];end;end;
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  for 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 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;
15 figure                                               % Isotherms of figure 4, left
16 for i=1:ne;fill(xy(lK(i,:),1)',xy(lK(i,:),2)',T(lK(i,:)));hold on;end
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 (*Figure 4*, left)

```
1  k = 0;for i=1:3;for j=1:4;k=k+1;xy(k,:)=[i-1,j-1];end;end;
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  for 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© 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 in *line 16*; the function *gra_cob.m* (*Table 59*). With this color bar, the visual quality of the output is significantly improved.

The conclusion of this test is twofold:

1. The only needed input are the matrix of node coordinates (*line 1, line 2*), the localization matrix (*line 3*) and the list of fixed temperatures (*line 5*).
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.

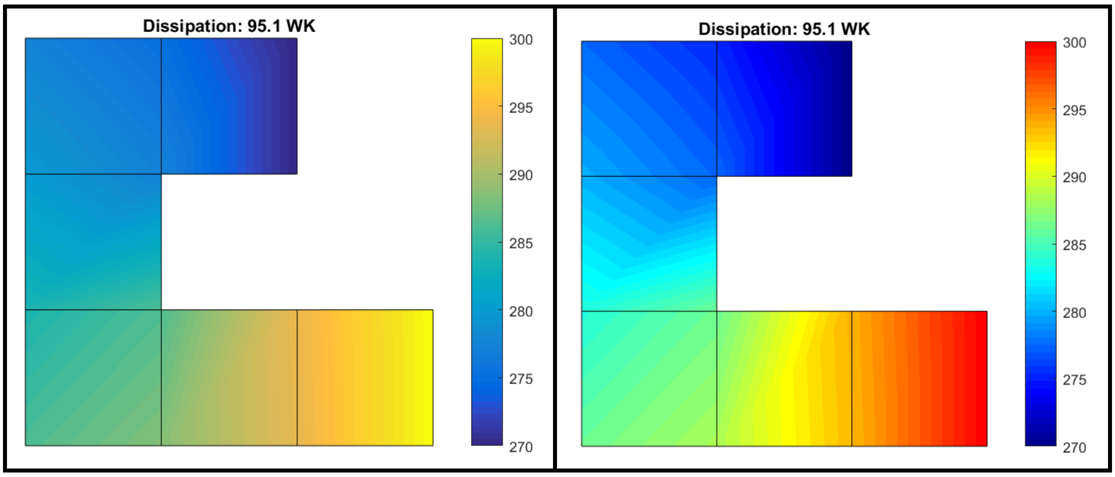


Figure 4: Result of the compact procedures 1 & 2 with different color maps

To obtain a visualization of the mesh, including node and element labels, it is sufficient to use the *Matlab©* instruction "`gra_mnl(xy, lK,[0 0 0],15),axis equal;grid on;axis off`", after running the procedure. 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.

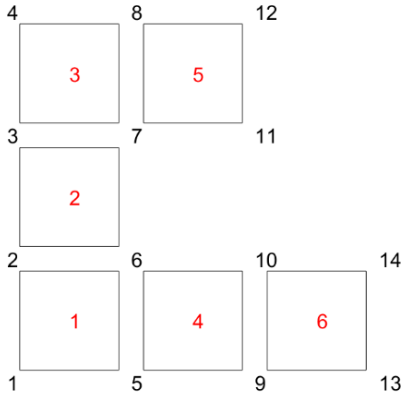


Figure 5: Finite element mesh with nodes and elements labels

1.4 Innovative handling of the Dirichlet boundary conditions

The Matlab© 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 x h x e = 1 m x 2 m x 1 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:

$$[N]\,[\,T\,] = [\,\bar{T}\,] \tag{20}$$

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 constrains we define a new term (blue) in the functional thanks to the Lagrange multipliers $[\Lambda]$:

$$< I(T) = [T]^T [K][T] + [\Lambda]^T \left([N][T] - [\bar{T}]\right) \tag{21}$$

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 $[T]$:

$$[K][T] + [N]^T [\Lambda] = [0] \tag{22}$$

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:

$$\begin{bmatrix} K & N^T \\ N & 0 \end{bmatrix} \begin{bmatrix} T \\ \Lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{T} \end{bmatrix} \quad \rightarrow \quad [A][B] = [G] \tag{23}$$

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© procedure *Fiammetta_33_20230130.m*

```
r=1 ;nx=1*r;ny=2*nx;ii=0;nn=(nx+1)*(ny+1);xy =zeros(nn,2);CPU=tic;
for j = ny + 1:-1:1                  % Geometry: definition of nodal coordinates
    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 % gra_ist(nx,ny,T,xy,[min(T) 2 max(T)])
30 colorbar;axis equal;axis off
31 % disp(['Dissipation ....... : ',num2str(T'*K*T/2,3),' WK'])
32 % disp(['CPU ............... : ',num2str(toc(CPU ),3),' sec. '])
33 % clear all; clc % To reset input and data
```

Table 2: Matlab© 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*). Disabling *line 28* and enabling *line 35*, the use of the function *gra_ist.m* (*Table 51*) instead of the Matlab© function *fill*, provides graphics with isotherm lines (*Figure 6*, *right*).

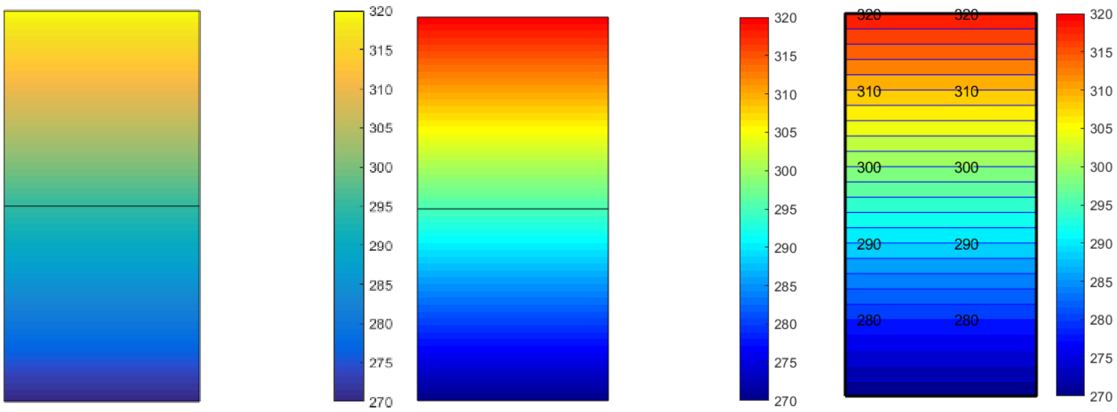


Figure 6: Example with imposed temperatures producing a vertical gradient

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:

$$le\,k\frac{\Delta T}{\Delta y} = k\frac{50}{2} = 25\ W \tag{24}$$

Inserted 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 [*Λ*], *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© notation, the reactive generalized heat flows of the top horizontal side are given by:

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

With different definitions of variables *r* and *nx* in *line 1* of *Fiammetta_33_20230130.m*, we obtain the solutions shown in *Figure 7*.

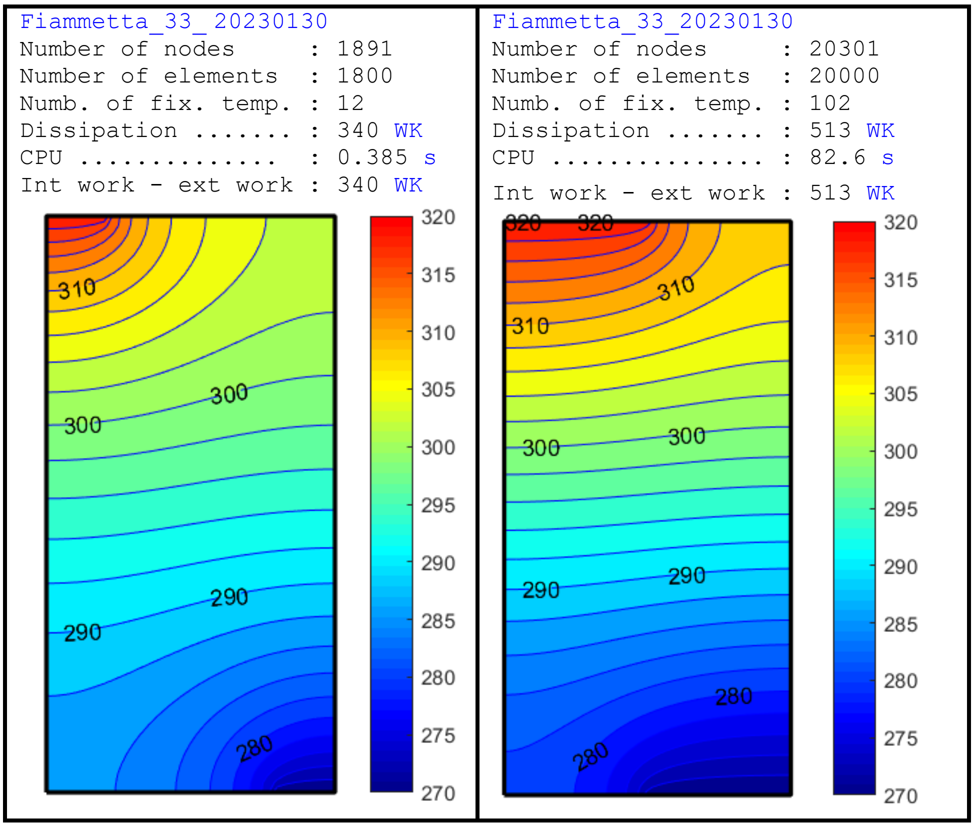


Figure 7: Imposed temp. on fragments of the horizontal faces (r = 6, nx = 5 & r = 2, nx = 50)

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 Matlab© procedure of *Table 2* is providing one single output: the visualization of the domain with colored temperature levels (`r=3;  nx=2*r;`). The temperature is always expressed in Kelvin (*K*). 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, *lfi* and *fT* are uni-line matrices. The drawings of *Figure 7* are obtained with a command calling the function *gra_ist.m* (*Table 51*), which is more efficient to represent isotherms than the original Matlab© command *fill*.

1.5 Neumann boundary conditions

The variational formulation of heat transfers was formulated in (6).

$$< I(\tau) = \int_{\Omega} \frac{1}{2} k \, \vec{\nabla}\tau \, . \, \vec{\nabla}\tau \, d\Omega + \int_{S_2} \bar{q}_n \; \tau \, dS > \qquad stationary \tag{25}$$

The von Neumann boundary conditions were deduced from the variational principle in (9).

$$-\vec{n} \cdot (k \, \vec{\nabla}\tau) = \bar{q}_n \quad on \; S_2 \tag{26}$$

Limiting the demonstration to one element edge, we can write that the second term of equation (25) corresponds to the sum of products of generalized nodal heat flows g_i (*W*) by temperatures T_i (*K*) and we can write it as follows:

We express the edge temperature in term of edge weight functions

$$\tau_{edge} = T_1 \left(1 - \frac{x}{l}\right) + T_2 \frac{x}{l} \tag{27}$$

Then:

$$\int_{S_{edge}} \bar{q}_n \tau_{edge} \; dS_{el} = g_1 T_1 + g_2 T_2 \tag{28}$$

We can write the discretized functional:

$$\int_{S_{edge}} \bar{q}_n \tau dS_{el} = \int_{S_{edge}} \bar{q}_n \left(T_1 \left(1 - \frac{x}{l}\right) + T_2 \frac{x}{l} \right) dS_{el} \tag{29}$$

In matrix form, we have:

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

(30)

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

$$\int_{S_{edge}} \bar{q}_n \tau dS_{el} = \int_{S_{edge}} \bar{q}_n \left[\left(1 - \frac{x}{l}\right) \quad \frac{x}{l} \right] dS_{el} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \tag{31}$$

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

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix}^T = \int_{S_{edge}} \bar{q}_n \left[\left(1 - \frac{x}{l}\right) \quad \frac{x}{l} \right] dS_{el} \tag{32}$$

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

Table 3: Instructions substituted to lines 13 to 16 in Fiammetta_33_20230130.m

```
Fiammetta_33_20230130
Number of nodes      : 231
Number of elements   : 200
Numb. of fix. temp. : 11
Heat loading         : 40 W
Dissipation          : 650 WK
CPU .............. : 0. 299 sec.
Int work - ext work : 650 WK
```

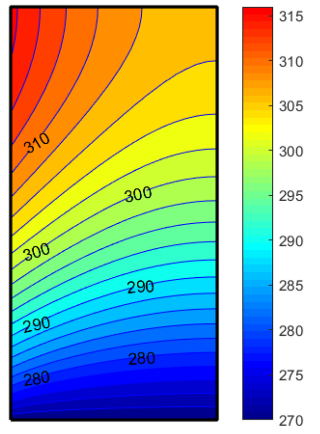


Figure 8: Isocurves for the problem with prescribed heat flows

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:

$$F_1 = \bar{q}\,\frac{el}{2} \quad ; \quad F_2 = \bar{q}\,\frac{el}{2} \qquad (33)$$

1.6 Matlab© procedure *Fiammetta_33_20230130.m*

The Matlab© 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* x *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 2d 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.

1.7 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© instruction used to produce this drawing is the last one of *Figure 9*.

```
xy = [0 3;1 3;2 3;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    % 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);                                         % Isotherms
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
```

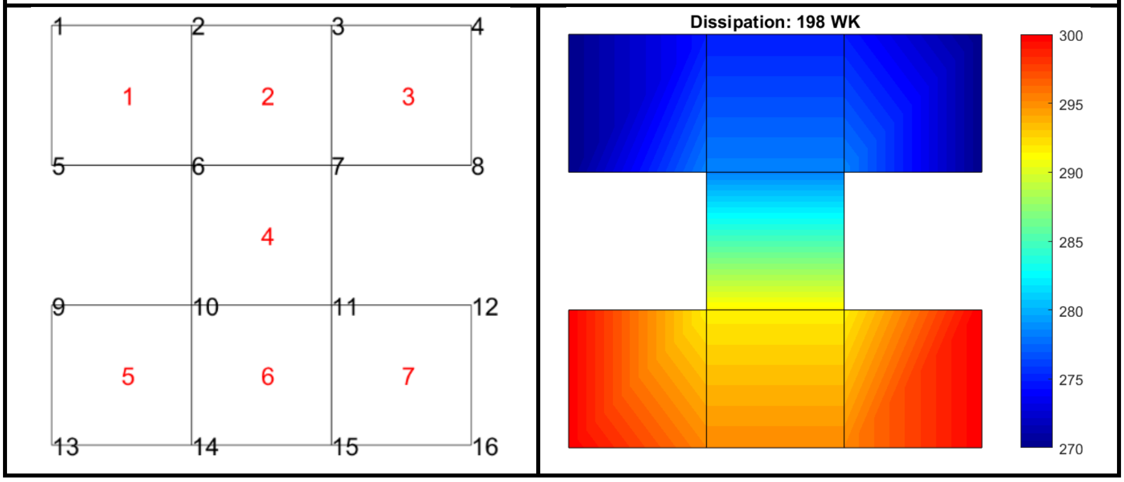


Figure 9: Exercise 1: mesh & isotherms

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:

$$< I(\tau) = \int_{\Omega} \frac{1}{2} k \, \vec{\nabla} \tau \, . \, \vec{\nabla} \tau \; d\Omega + \frac{1}{2} \int_{S_3} h \left(\tau - \tau_f \right)^2 dS + \int_{S_2} \overline{q}_n \; \tau dS \; > \quad minimum \tag{34}$$

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.

$$\text{Functional at element level: } I_{el} = \frac{1}{2} \int_{S_3} h \left(\tau - \tau_f \right)^2 dS \tag{35}$$

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

$$\tau = T_0 (1 - \frac{x}{L}) + T_1 \frac{x}{L} \tag{36}$$

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

$$\begin{aligned} I_{el} &= \frac{1}{2} \int_0^L h \left(\begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} - t_f \right)^2 dx \\ &= \frac{1}{2} h \int_0^L \left\{ [T]^T \begin{bmatrix} 1 - \frac{x}{L} \\ \frac{x}{L} \end{bmatrix} \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} [T] - 2 \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} [T] t_f + t_f^2 \right\} dx \end{aligned} \tag{37}$$

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[1]:

$$\left[T_{el} \right] = \left[T_0 \quad T_1 \quad T_f \right]^T \tag{38}$$

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

[1] 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.

$$I_{el}=\frac{1}{2}hT_{el}^{T}(\int_{0}^{L}\begin{bmatrix}1-\frac{x}{L}\\ \frac{x}{L}\\ -1\end{bmatrix}\begin{bmatrix}1-\frac{x}{L} & \frac{x}{L} & -1\end{bmatrix}dx)\ T_{el} \tag{39}$$

Developing the expression:

$$I_{el}=\frac{1}{2}hT_{el}^{T}\int_{0}^{L}\begin{bmatrix}\left(1-\frac{x}{L}\right)^{2} & \left(1-\frac{x}{L}\right)\frac{x}{L} & -\left(1-\frac{x}{L}\right)\\ \left(1-\frac{x}{L}\right)\frac{x}{L} & \left(\frac{x}{L}\right)^{2} & -\frac{x}{L}\\ -\left(1-\frac{x}{L}\right) & -\frac{x}{L} & 1\end{bmatrix}dx\ T_{el} \tag{40}$$

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:

$$I_{el}=\frac{1}{2}h\ e\ T_{el}^{T}\begin{bmatrix}\int_{0}^{L}\left(1-\frac{x}{L}\right)^{2}dx & \int_{0}^{L}\left(1-\frac{x}{L}\right)\frac{x}{L}dx & -\int_{0}^{L}\left(1-\frac{x}{L}\right)dx\\ \int_{0}^{L}\left(1-\frac{x}{L}\right)\frac{x}{L}dx & \int_{0}^{L}\left(\frac{x}{L}\right)^{2}dx & -\int_{0}^{L}\frac{x}{L}dx\\ -\int_{0}^{L}\left(1-\frac{x}{L}\right)dx & -\int_{0}^{L}\frac{x}{L}dx & \int_{0}^{L}dx\end{bmatrix}T_{el} \tag{41}$$

The integrals present in the discretized functional are easily computed:

$$\int_{0}^{L}\left(1-\frac{x}{L}\right)^{2}dx=\int_{0}^{L}\left(1-2\frac{x}{L}+\frac{x^{2}}{L^{2}}\right)dx=L-L+\frac{L}{3}=\frac{L}{3} \tag{42}$$

$$\int_{0}^{L}\left(1-\frac{x}{L}\right)\frac{x}{L}dx=\int_{0}^{L}\left(\frac{x}{L}-\frac{x^{2}}{L^{2}}\right)dx=\frac{L}{2}-\frac{L}{3}=\frac{L}{6} \tag{43}$$

$$-\int_{0}^{L}\left(1-\frac{x}{L}\right)dx=-L+\frac{L}{2}=-\frac{L}{2} \tag{44}$$

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

$$I_{el}=\frac{1}{2}h\ eL\ T_{el}^{T}\begin{bmatrix}\frac{1}{3} & \frac{1}{6} & -\frac{1}{2}\\ \frac{1}{6} & \frac{1}{3} & -\frac{1}{2}\\ -\frac{1}{2} & -\frac{1}{2} & 1\end{bmatrix}T_{el}=\frac{1}{2}T_{el}^{T}\ K_{h}\ T_{el} \tag{45}$$

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

$$K_h = \frac{ehL}{6}\begin{bmatrix} 2 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & 6 \end{bmatrix} \tag{46}$$

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.

$$K = K_k + K_h \tag{47}$$

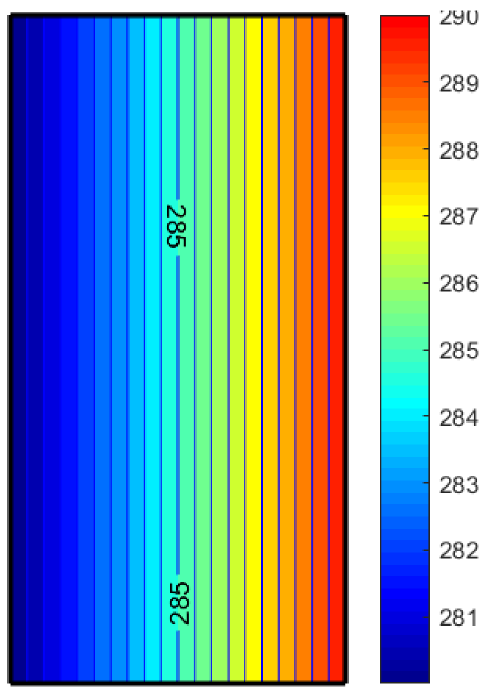
The convection virtual nodes may be free or fixed. The convection matrix defined in (34) is computed in the function *fem_Kcv.m* (*Table 45*). The input is the matrix *xyz* of node coordinates, the localization matrix *lc* of the convective element and the coefficient *hh*, product of the thickness by 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 ($Wm^{-2}K^{-1}$). 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*.

```
Fiammetta
Standard 1 rect.,    25, Di : 13
L    4, Method, Ne    : 1   0   0
L    5, Co nnr nvn    : 1   0   2
L    6, rc, ra, cs    : 0   0   0
L    7, CAD interf.   : 4
L    8, Thickness     : 0.1 m
L    9, Conduction k  : 1  W/(mK)
L   10, DT isotherms  : 0.5 K
L   13, Convection h  : 1  W/(m2K)
L   15, np, nvertices: 1   4
L   17, num. nod side: 1
L   19, num elem side: 2
L   20, Domain area   : 2  m2
L   32, Num. elements: 4
L   34, Num. of DOF   : 11
L   63, Domain perim.: 6  m
LD  83, Fixed nodes   : 10   11
LD  84, Fix. temper. : 300   270 K
N   03, param Ne      : 0
c.  03, dK = no + nvn: 11
c.  04, N.virt c.nod.: 2
c.  05, Variable Co   : 1
c.  12, Convect. sid.: 2   3   4   1
c  125, Nu. conv. el.: 4
c  126, conv. coeff. : 1   1   1   1  W/(m2K)
L   96, Anis. index   : 0
L  239, Total dissip.: 30  WK
L  241, Dis. in solid: 10  WK
L  242, DT   in solid: 10  K
L  248, Fixed DOF     : 10   11
L  249, Imposed temp.: 300    270 K
L  250, React. flows : 2        -2 W
L  252, Date, CPU, 01-Jul-2023, 0.56435 s
g   18, Max temp grad: 10, mean: 10 K/m
```



```
ch 03, coef. red. dt: 25 W/m2
ch 19, temp. grad.   : 10 K
ch 20, Mean conv. fl: -10    0    0 W/m2
hf 25, Max heat flow: 10, mean: 10 W/m2
```

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

Matlab© function or procedure	Displayed output format, *n* is the line where the command is issued
Fiammetta.m	['L n,
	.
	.
Cad_ban.m	-
cad_con.m	['c. n,
cad_Dir.m	['LD n,
cad_Neu.m	['N n,
cad_mes.m	-
	.
fem_caK.m	['cK n,
fem_Kcv.m	
fem_Kco.m	
fem_rsm.m	['rs n,
fem_snl.m	['Ls n,
fem_smd.m	['sd n,
fem_smt.m	['st n,
fem_smq.m	['sq n,
fem_smc.m	['sc n,
	.
geo_baf.m	['Lb n,
geo_stf.m	-
geo_vfr.m	-
geo_vfc.m	-
	.
gra_ahf.m	['hf n,
gra_2dm.m	['L 2dm,
gra_chf.m	['ch n,
gra_atg.m	['g n,
gra_tra.m	-
gra_ipa.m	-
gra_mnl.m	-
	.
mat_cok.m	['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 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.

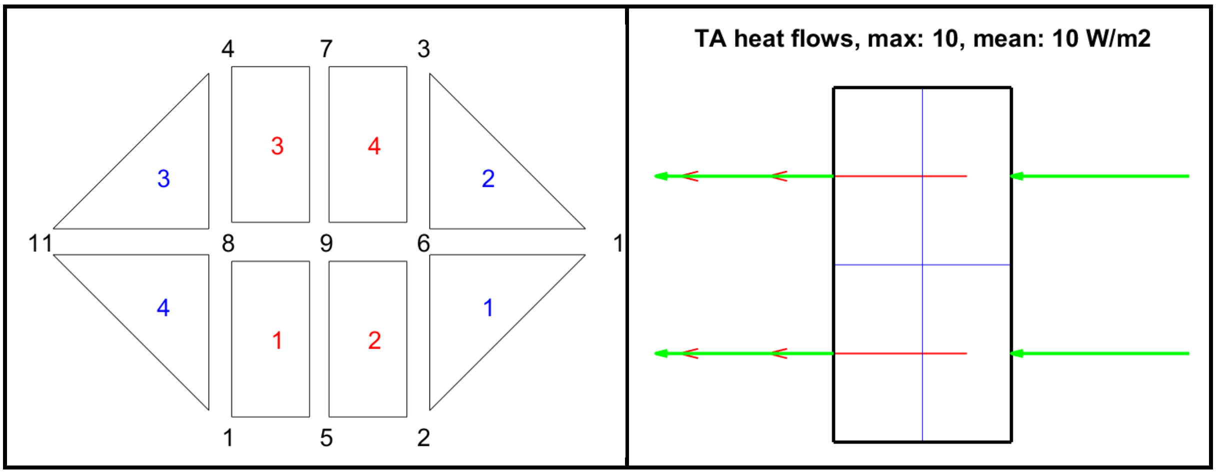


Figure 11: Nodes and elements numbering: heat flows with convective boundary conditions

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:

$t_0 < t_1$	*Conductive zone*	$t_2 < t_3$

$$q_x = h(t_0 - t_1) = k\frac{(t_1 - t_2)}{w} = h(t_2 - t_3) \tag{48}$$

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

$$\beta = \frac{h\,w}{k}, \qquad \beta\,(t_0 - t_1) = t_1 - t_2 = \beta\,(t_2 - t_3) \tag{49}$$

From the first relation, we deduce:

$$t_1 = \frac{\beta\,t_0 + t_2}{1+\beta} \tag{50}$$

Now, we develop the second one:

$$\frac{\beta\,t_0 + t_2}{1+\beta} - t_2 = \beta\,t_2 - \beta\,t_3 \tag{51}$$

We obtain:

$$t_2 = \frac{t_0 + (1+\beta)t_3}{2+\beta} \tag{52}$$

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

$$t_1 = \frac{\beta\,t_0}{1+\beta} + \frac{t_0 + (1+\beta)\,t_3}{(2+\beta)\,(1+\beta)} = \frac{t_0}{1+\beta}\left(\beta + \frac{1}{(2+\beta)}\right) + \frac{t_3}{(2+\beta)} = \frac{(1+\beta)\,t_0 + t_3}{(2+\beta)} \tag{53}$$

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

$$(t_2 - t_1) = (t_3 - t_0)\frac{\beta}{2+\beta} \tag{54}$$

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 β	$-q_x$ (Wm^{-2})	t_1 (K)	t_2 (K)	(t_1-t_0) (K)	(t_2-t_1) (K)	(t_3-t_2) (K)	Dissipation
.5	6	282	288	12	6	12	3.6 *WK*
1	10	280	290	10	10	10	10 *WK*

2	15	277.5	292.5	7.5	15	7.5	22.5 *WK*
18	27	271.5	298.5	1.5	27	1.5	72.9 *WK*
20	27.3	271.36	298.64	1.36	27.3	1.36	74.4 *WK*

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^{-2}. The quantity of heat crossing the virtual nodes is the product of the heat flux by the section of the vertical side: 27 Wm^{-2} x 0.2 m^2 = 5.4 *W*. The ratio between the temperature gap in the solid and the total gap is equal to 27/30*100 = 90 %.

```
Fiammetta
Standard 1 rect., Gi: 25, Di : 13
L   4, Meth., Ne, ca: 1  0  0
L   5, Co nnr nvn   : 1  0  2
L   6, rc, ra, cs   : 0  0  0
L   7, CAD interf.  : 4
L   8, Thickness    : 0.1 m
L   9, Conduction k : 1   W/(mK)
L  10, DT isotherms : 2   K
L  13, Convection h : 18  W/(m2K)
L  15, np, nvertices: 1  4
L  17, num. nod side: 1
L  19, num elem side: 2
L  29, Domain area  : 2 m2
L  32, Num. elements: 4
L  35, Num. of nodes: 9
L  36, Num. of DOF  : 11
L  63, Domain perim.: 6 m
LD 83, Fixed nodes  : 10  11
LD 84, Fix. temper. : 300  270 K
N  03, param Ne     : 0
c. 03, dK = no + nvn: 11
c. 04, N.virt c.nod.: 2
c. 05, Variable Co  : 1
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 252, Date, CPU, 01-Jul-2023, 0.469 s
g  18, Max temp grad: 27, mean: 27 K/m
ch 03, coef. red. dt: 25 W/m2
ch 19, temp. grad.  : 1.5 K
ch 20, Mean conv. fl: -1.5 0 0 W/m2
hf 25, Max heat flow: 27, mean: 27 W/m2
```

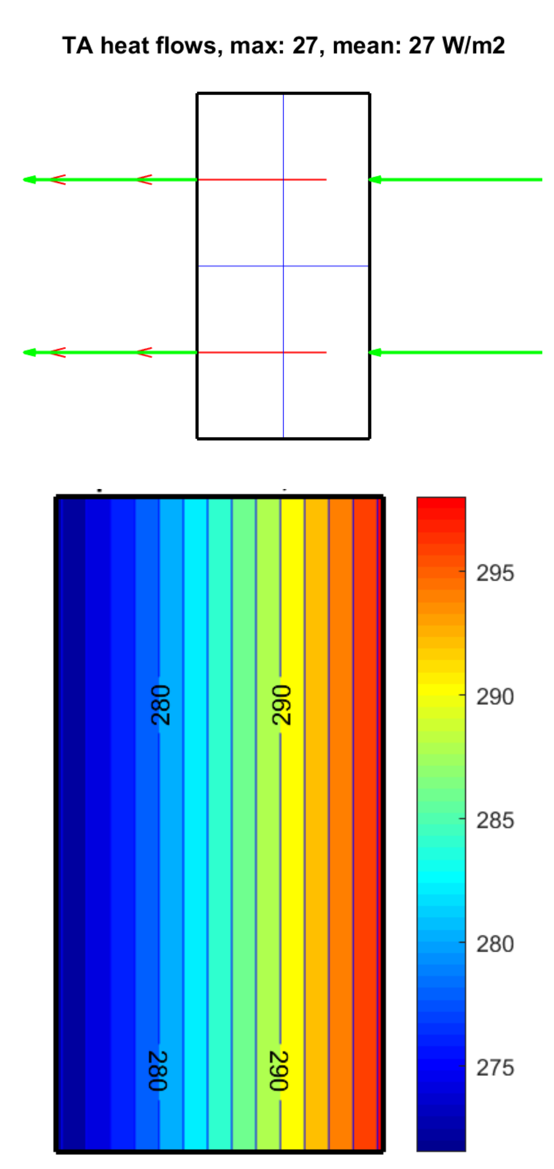


Figure 12: Example of heat transfer through a wall with a high Biot number $\beta = 18$

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 nvn    : 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 239, Total dissip.: 390 WK
L 241, Dis. in solid: 390 WK
L 242, DT   in solid: 50 K
L 252, Date, CPU, 01-Jul-2023, 0.36019 s
hf 25, Max heat flow: 66, mean: 18 W/m2
g  18, Max temp grad: 66, mean: 18 K/m
```

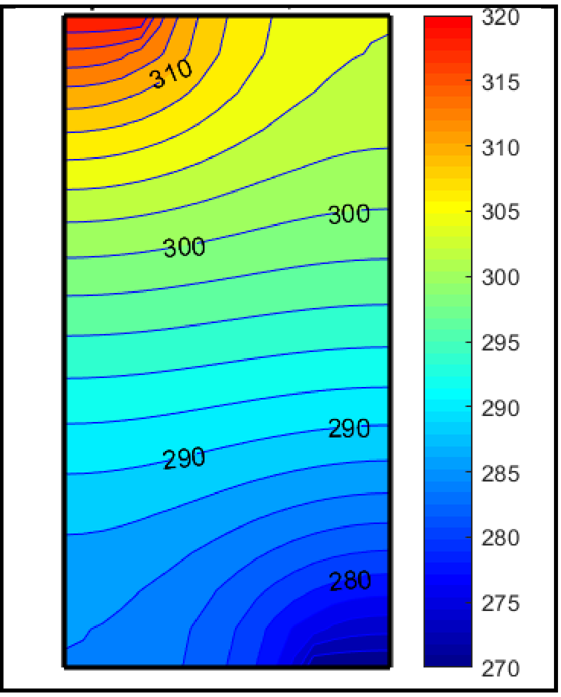
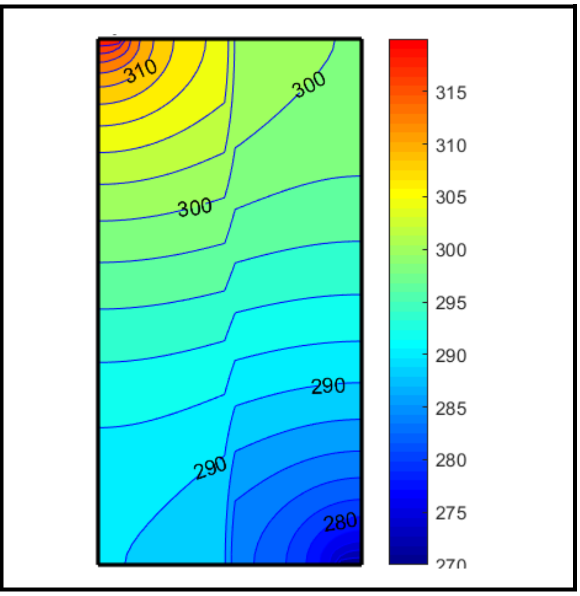


Figure 13: Isocurves in a rectangle with fixed DOF on horizontal edges, isotropic material

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 nvn    : 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   94, Anis. index   : 3
Lm 48, k & coef*k    : 1 0.1 W/(m K)
Lm 49, Thickn. ratio: 0.04
L 239, Total dissip.: 259 WK
L 241, Dis. in solid: 259 WK
L 242, DT   in solid: 50 K
```



```
L 252, Date, CPU, 01-Jul-2023, 1.3933 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© 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 nvn    : 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  239, Total dissip.: 741 WK
L  241, Dis. in solid: 741 WK
L  242, DT   in solid: 50 K
L  252, Date, CPU, 01-Jul-2023, 0.35784 s
h   23, Max heat flow: 220, mean: 35 W/m2
g   18, Max temp grad: 140, mean: 13 K/m
```

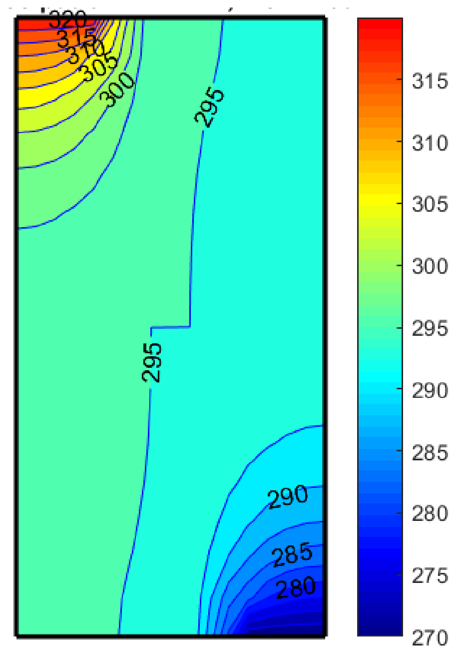


Figure 15: Isocurves for the vertical strip 2 elements wide, 16 x 16 mesh

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

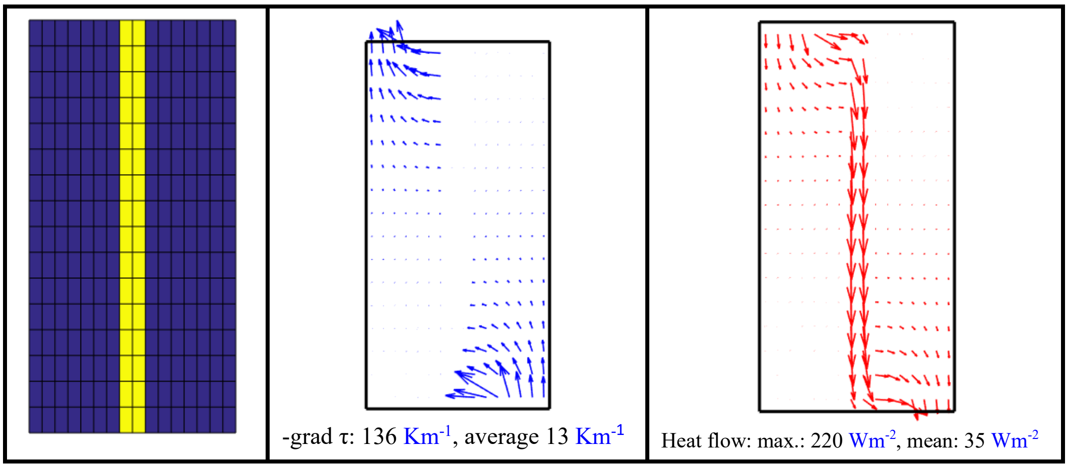


Figure 16: Heat flows and temperature gradients for a vertical thermal bridge

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.

```
Fiammetta
Standard 1 rect., Gi: 9, Di : 3
L   4, Meth., Ne, ca: 1  0  0
L   5, Co nnr nvn   : 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: 31
L  19, num elem side: 32
L  29, Domain area  : 2 m2
L  32, Num. elements: 1024
L  34, Num. of DOF  : 1089
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  0.1 W/(m K)
Lm 49, Thickn. ratio: 0.0625
L 239, Total dissip.: 283 WK
L 241, Dis. in solid: 283 WK
L 242, DT   in solid: 50 K
L 252, Date, CPU, 01-Jul-2023, 0.64607 s
h  23, Max heat flow: 92, mean: 13 W/m2
g  18, Max temp grad: 92, mean: 16 K/m
```

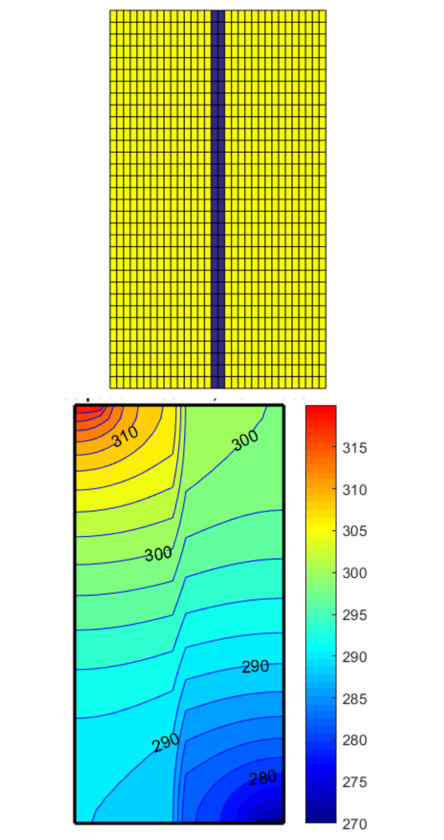


Figure 17: Isocurves in presence of a vertical small conductivity strip (0.1 k)

With16 elements per patch side we obtain the pictures of the heat flows and the temperature gradients shown in *Figure 16*.

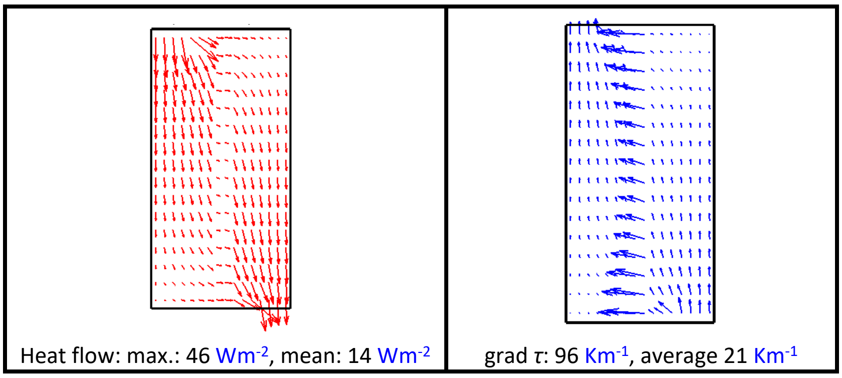


Figure 18: Heat flows and temperature gradients (vertical strip with small conductivity)

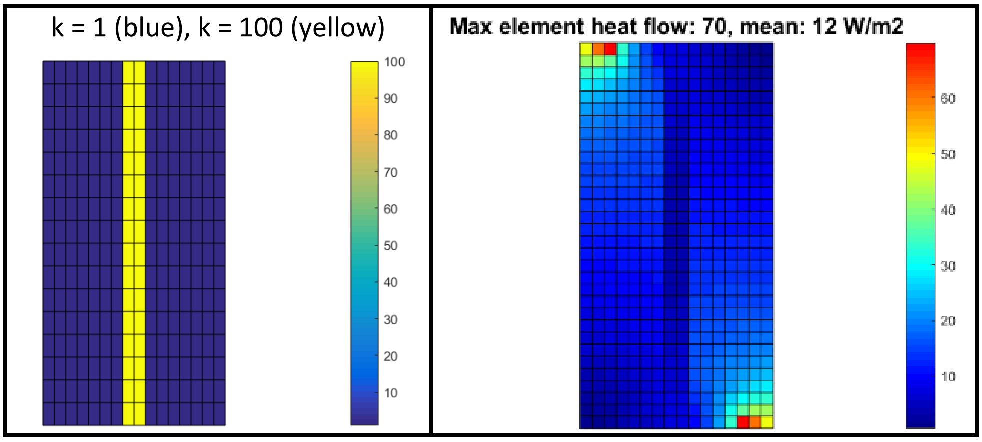


Figure 19: Visualizations of scalars defined element by element

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 Horizontal thermal bridge

```
Fiammetta
Standard 1 rect., Gi: 8, Di : 13
L   4, Meth., Ne, ca: 1  0  0
L   5, Co nnr nvn   : 1  0  2
L   6, rc, ra, cs   : 0  0  0
L   7, CAD interf.  : 4
L   8, Thickness    : 0.1 m
L   9, Conduction k : 1 W/(mK)
L  10, DT isotherms : 0.5 K
L  13, Convection h : 1 W/(m2K)
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  : 291
L  63, Domain perim.: 6 m
LD 83, Fixed nodes  : 290  291
LD 84, Fix. temper. : 300  270 K
N  03, param Ne     : 0
c. 03, dK = no + nvn: 291
c. 04, N.virt c.nod.: 2
c. 05, Variable Co  : 1
c. 10, Convect. sid.: 2  3  4  1
c 125, Nu. conv. el.: 32
L  96, Anis. index  : 1
Lm 48, k & coef*k   : 1  1000 W/(m K)
Lm 49, Thickn. ratio: 0.125
L 242, Total dissip.: 35.2 WK
L 244, Dis. in solid: 7.14 WK
L 245, DT   in solid: 9.44 K
L 251, Fixed DOF    : 290  291
L 252, Imposed temp.: 300       270 K
L 253, React. flows : 2.34     -2.34 W
L 255, Date, CPU, 05-Jul-2023, 0.40212s
g  18, Max temp grad: 18, mean: 7.5 K/m
ch 03, coef. red. dt: 25 W/m2
ch 19, temp. grad.  : 10.2972 K
ch 20, Mean conv. fl: -11.7249 0 0 W/m2
hf 25, Max heat flow: 50, mean: 13 W/m2
```

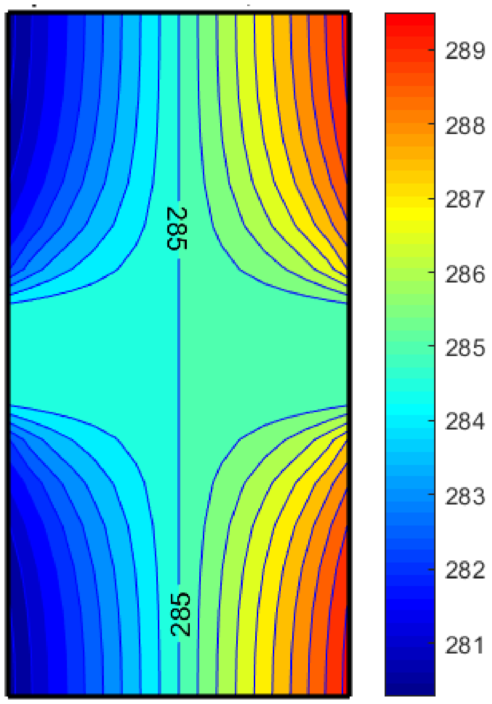


Figure 20: Isocurves - horizontal thermal bridge with high conductivity (1000 k)

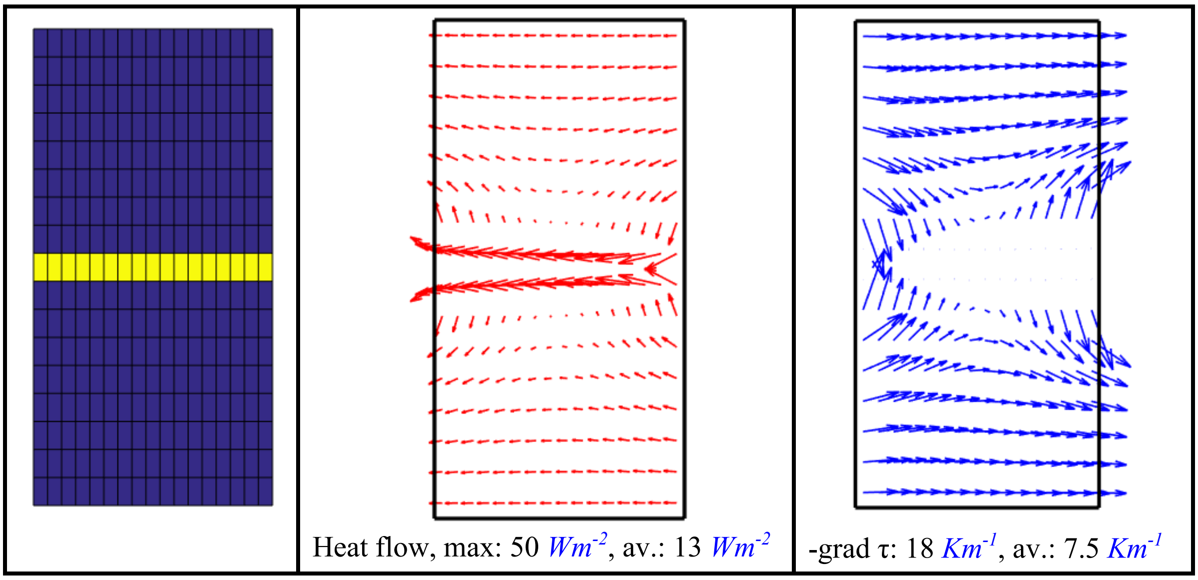


Figure 21: Heat flows and temperature gradients (horizontal strip with high conductivity)

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.

3. 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 [Irons 1966] is based on the Coons patch developed in the frame of Computed Aided Design (CAD) [Coons 1967].

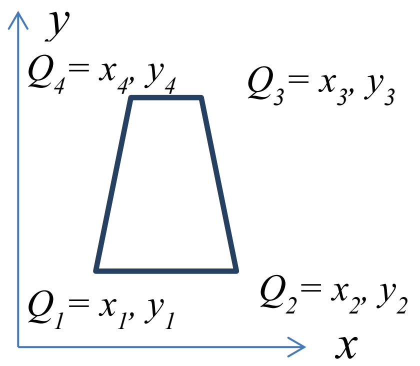
3.1 Numerical evaluation of the temperature gradient in a Coons patch

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*].

$$[Q]=\begin{bmatrix}Q_1\\Q_2\\Q_3\\Q_4\end{bmatrix}=\begin{bmatrix}x_1 & y_1\\x_2 & y_2\\x_3 & y_3\\x_4 & y_4\end{bmatrix}=\begin{bmatrix}X & Y\end{bmatrix} \quad (55)$$

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*]:

$$\begin{aligned}x(s,t)&=[F][X]=\begin{bmatrix}(1-s)(1-t) & s(1-t) & st & (1-s)t\end{bmatrix}[X]\\y(s,t)&=[F][Y]=\begin{bmatrix}(1-s)(1-t) & s(1-t) & st & (1-s)t\end{bmatrix}[Y]\end{aligned} \quad (56)$$

<table>
<tr><th>Cartesian space
x, y
$dS = dx\,dy$</th><th>Parametric space
s, t
$dS = J(s, t)\,ds\,dt$</th></tr>
<tr><td>

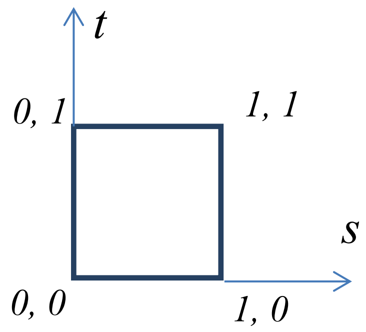
$[X]=\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}$ $[Y]=\begin{bmatrix}y_1\\y_2\\y_3\\y_4\end{bmatrix}$</td><td>
</td></tr>
</table>

Table 6: Coons patch definition in Cartesian and parametric spaces

The barycenter of the four vertices is the point situated at $s = ½$, $t = ½$.

$$\begin{aligned}x(s=1/2,t=1/2)&=\begin{bmatrix}1/4 & 1/4 & 1/4 & 1/4\end{bmatrix}[X]=(x_1+x_2+x_3+x_4)/4\\y(s=1/2,t=1/2)&=\begin{bmatrix}1/4 & 1/4 & 1/4 & 1/4\end{bmatrix}[Y]=(y_1+y_2+y_3+y_4)/4\end{aligned} \quad (57)$$

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

$$\begin{aligned}\frac{\partial x(s,t)}{\partial s}&=\frac{\partial[F]}{\partial s}[X]=\begin{bmatrix}-(1-t) & (1-t) & t & -t\end{bmatrix}[X]\\ \frac{\partial x(s,t)}{\partial t}&=\frac{\partial[F]}{\partial t}[X]=\begin{bmatrix}-(1-s) & -s & s & (1-s)\end{bmatrix}[X]\\ \frac{\partial y(s,t)}{\partial s}&=\frac{\partial[F]}{\partial s}[Y]=\begin{bmatrix}-(1-t) & (1-t) & t & -t\end{bmatrix}[Y]\\ \frac{\partial y(s,t)}{\partial t}&=\frac{\partial[F]}{\partial t}[Y]=\begin{bmatrix}-(1-s) & -s & s & (1-s)\end{bmatrix}[Y]\end{aligned} \tag{58}$$

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

$$[J]=\begin{bmatrix}\frac{\partial x}{\partial s} & \frac{\partial y}{\partial s}\\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t}\end{bmatrix}=\begin{bmatrix}\begin{bmatrix}-(1-t) & (1-t) & t & -t\end{bmatrix}[X] & \begin{bmatrix}-(1-t) & (1-t) & t & -t\end{bmatrix}[Y]\\ \begin{bmatrix}-(1-s) & -s & s & (1-s)\end{bmatrix}[X] & \begin{bmatrix}-(1-s) & -s & s & (1-s)\end{bmatrix}[Y]\end{bmatrix} \tag{59}$$

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:

$$[J]_{s=t=\frac{1}{2}}=\frac{1}{2}\begin{bmatrix}\begin{bmatrix}-1 & 1 & 1 & -1\end{bmatrix}[X] & \begin{bmatrix}-1 & 1 & 1 & -1\end{bmatrix}[Y]\\ \begin{bmatrix}-1 & -1 & 1 & 1\end{bmatrix}[X] & \begin{bmatrix}-1 & -1 & 1 & 1\end{bmatrix}[Y]\end{bmatrix} \tag{60}$$

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

$$[J]_{s=t=\frac{1}{2}}=\frac{1}{2}\begin{bmatrix}x_2+x_3-x_1-x_4 & y_2+y_3-y_1-y_4\\ x_4+x_3-x_1-x_2 & y_4+y_3-y_1-y_2\end{bmatrix} \tag{61}$$

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:

$$J_{s=t=\frac{1}{2}}=\frac{1}{2}\left((x_2+x_3-x_1-x_4)(y_4+y_3-y_1-y_2)-(x_4+x_3-x_1-x_2)(y_2+y_3-y_1-y_4)\right) \tag{62}$$

Finally:

$$J_{s=t=\frac{1}{2}} \quad = \quad \frac{1}{2}\left((x_2-x_4)(y_3-y_1)+(x_3-x_1)(y_4-y_2)\right) \tag{63}$$

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).

$$\begin{bmatrix}\frac{\partial\tau}{\partial s}\\ \frac{\partial\tau}{\partial t}\end{bmatrix}=\begin{bmatrix}\frac{\partial x}{\partial s} & \frac{\partial y}{\partial s}\\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t}\end{bmatrix}\begin{bmatrix}\frac{\partial\tau}{\partial x}\\ \frac{\partial\tau}{\partial y}\end{bmatrix}=[J]\begin{bmatrix}\frac{\partial\tau}{\partial x}\\ \frac{\partial\tau}{\partial y}\end{bmatrix}=[J]\vec{\nabla}\tau \tag{64}$$

After inverting (52), we obtain:

$$\vec{\nabla}\tau=\begin{bmatrix}\frac{\partial\tau}{\partial x}\\ \frac{\partial\tau}{\partial y}\end{bmatrix}=[J]^{-1}\begin{bmatrix}\frac{\partial\tau}{\partial s}\\ \frac{\partial\tau}{\partial t}\end{bmatrix} \tag{65}$$

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:

$$\tau = \left[(1-s)(1-t) \quad s(1-t) \quad st \quad (1-s)t \right][T] \tag{66}$$

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

$$\begin{bmatrix} \dfrac{\partial \tau}{\partial s} \\ \dfrac{\partial \tau}{\partial t} \end{bmatrix} = \begin{bmatrix} -(1-t) & (1-t) & t & -t \\ -(1-s) & -s & s & (1-s) \end{bmatrix} [T] \tag{67}$$

In the barycenter:

$$\begin{bmatrix} \dfrac{\partial \tau}{\partial s} \\ \dfrac{\partial \tau}{\partial t} \end{bmatrix}_{s=t=\frac{1}{2}} = \frac{1}{2} \begin{bmatrix} -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix} [T] \tag{68}$$

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

$$\vec{\nabla} \tau = \begin{bmatrix} \dfrac{\partial \tau}{\partial x} \\ \dfrac{\partial \tau}{\partial y} \end{bmatrix} = [J]^{-1} \frac{1}{2} \begin{bmatrix} -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix} [T] \tag{69}$$

Being able to calculate the temperature gradients, it is now possible to compute the conductivity matrices. The Matlab© function *fem_Kco.m* (*Table 44*) allows computing the conductivity matrix [*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	`xyz = [0 0 0;1 0 0;1 1 0;0 1 0];lo=[1 2 3 4];` `[K] = fem_Kco (xyz,lo)*6`

Matlab Output	`K =   4    -1    -2    -1`
	`     -1     4    -1    -2`
	`     -2    -1     4    -1`
	`     -1    -2    -1     4`

Table 7: Numerically integrated conductivity matrix

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

3.2 *CAD* model of the domain

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 28*, 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.

```
npv=size(xyz_cao,1);[(1:npv)'  xyz_cao]
   1    2    2
   2    2    3
   3    1    2
   4    0    3
   5    0    0
   6    1    1
   7    3    0
   8    3    1
```

```
np=size(xyz_cao,1);[(1:np)' car_cao]
1    1    2    4    3
2    3    4    5    6
3    7    8    6    5
```

Table 8: Instructions used to display input data of Figure 28

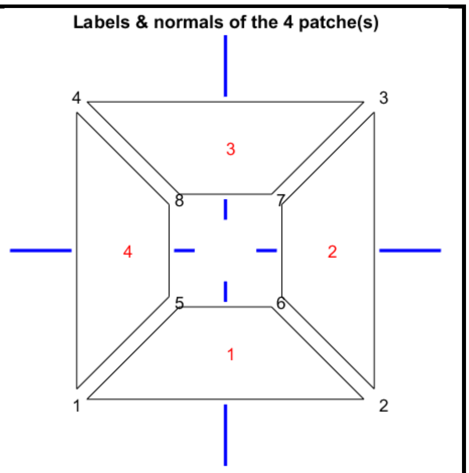
Lines 16 - 19 of *Table 31* are generating the Coons patches displayed in *Table 8* and drawn in *Figure 28*.

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*.

```
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
```




```
[(1:np)' car_cao] →
   1    6    5    1    2
   2    6    2    3    7
   3    7    3    4    8
   4    1    5    8    4
```

```
[(1:np)' pbo] →
1    1    2    3    4
2    4    5    6    7
3    6    8    9   10
4    2   11    9   12
```

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
```

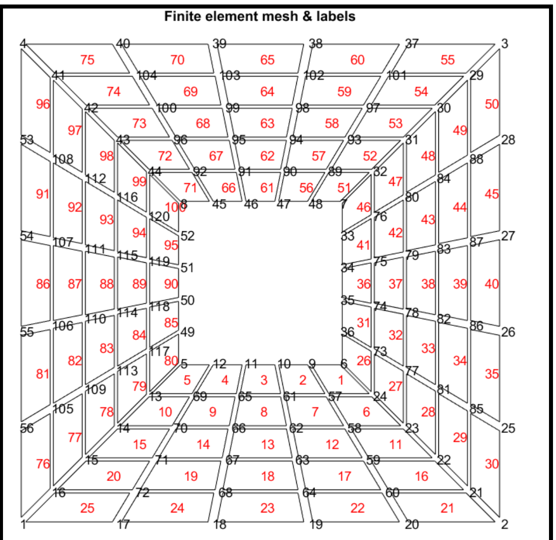
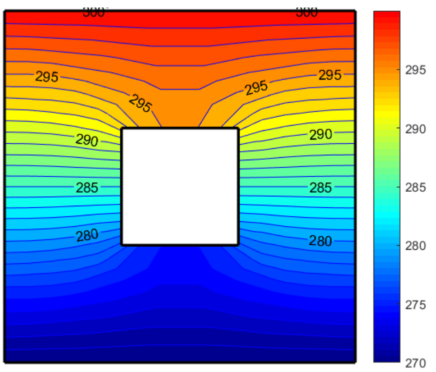


Figure 23: Finite element mesh corresponding to the CAD definition of Figure 22

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*).

```
Fiammetta
Standard cavity,  Gi: 4, Di : 4
L   4, Meth., Ne, ca: 1  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    : 1 m
L   9, Conduction k : 1 W/(mK)
L  10, DT isotherms : 1 K
L  15, np, nvertices: 4  8
L  17, num. nod side: 4
L  19, num elem side: 5
L  29, Domain area  : 8 m2
L  32, Num. elements: 100
L  36, Num. of DOF  : 120
L  63, Domain perim.: 16 m
LD 36, N. fix. nodes: 12
LD 37, Fix. DOF, lfi: 1 … 2   3  … 4
N  03, param Ne     : 0
L  96, Anis. index  : 0
L 242, Total dissip.: 357 WK
L 244, Dis. in solid: 357 WK
L 245, DT   in solid: 30 K
L 255, Date, CPU, 13-Jul-2023, 1.4601 s
g  18, Max temp grad: 14, mean: 9.2 K/m
hf 25, Max heat flow: 14, mean: 9.2 W/m2
```



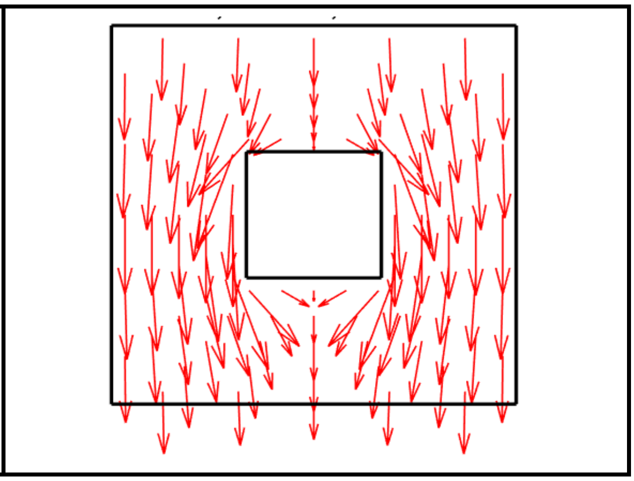


Figure 24: Heat flow around an adiabatic square cavity

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

```
Fiammetta
Standard cavity,  Gi: 4, Di : 4
L    4, Meth., Ne, ca: 1  0  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    : 1 m
L    9, Conduction k : 1 W/(mK)
L   10, DT isotherms : 1 K
L   11, Refl. coeff. : 1
L   15, np, nvertices: 4  8
L   17, num. nod side: 4
L   19, num elem side: 5
L   29, Domain area  : 8 m2
L   32, Num. elements: 100
L   34, Num. of DOF  : 120
L   63, Domain perim.: 16 m
L   66, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD  36, N. fix. nodes: 12
LD  37, Fix. DOF, lfi: 1 17 … 2 3 … 40 4
N   03, param Ne     : 0
L   89, numb ra.nodes: 20
L   92, R. vertic. lv: 6  5  8  7
L   96, Anis. index  : 0
L  127, Sky temper.  : 280 K
L  131, Rad. edg. len: 1  1  1  1 m
L  132, Radiativ area: 4 m2
L  142, Det. R. mat M: 0.
L  216, Total dissip.: 357 WK
L  219, Dis. in solid: 357 WK
L  220, DT   in solid: 30  K
L  255, Date, CPU, 13-Jul-2023, 0.61806 s
g   18, Max temp grad: 14, mean: 9.2 K/m
hf  25, Max heat flow: 14, mean: 9.2 W/m2
```

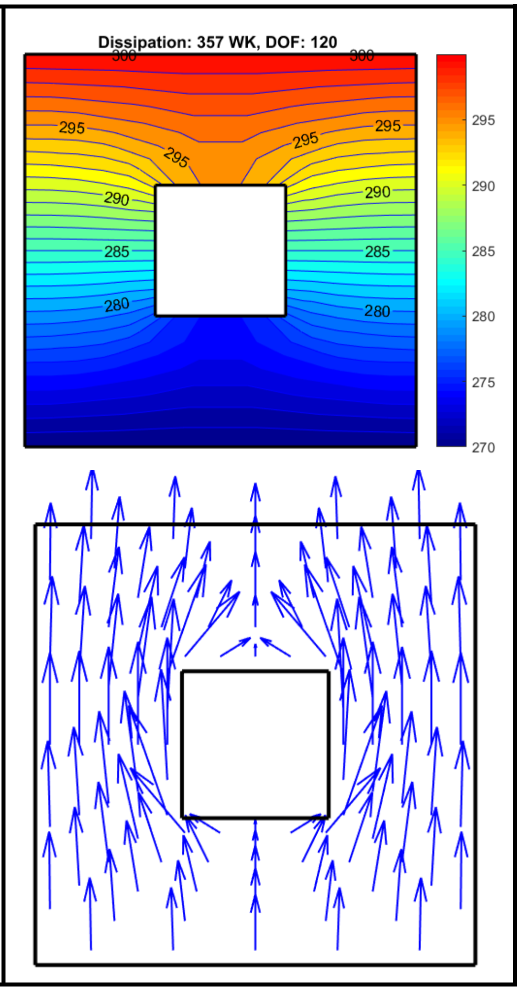


Figure 25: Heat flow around a perfectly reflective cavity

With the radiative boundary conditions taken into account in the procedure: *rc* = 1, *cs* = 6, the function *geo_vfc.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

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*). In *cad_Dir.m*, instead of *line 60*, *line 59* is activated. The domain is only composed of rectangular patches in *Figure 29*. 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 (*Figure* ***29***), it converges to 0.979 *WK* when we have 8241 *DOF*. The Matlab© functions *gra_mel.m* (*Table 52*) and *gra_mnl.m* (*Table 53*) enable the visualization of the finite element meshes (*Figure 29*). 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  30, Domain area   : 6 m2
L  33, 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 242, Total dissip.: 0.993 WK
L 244, Dis. in solid: 0.993 WK
L 245, DT   in solid: 10 K
L 255, Date, CPU, 18-Jul-2023, 0.41841 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
```

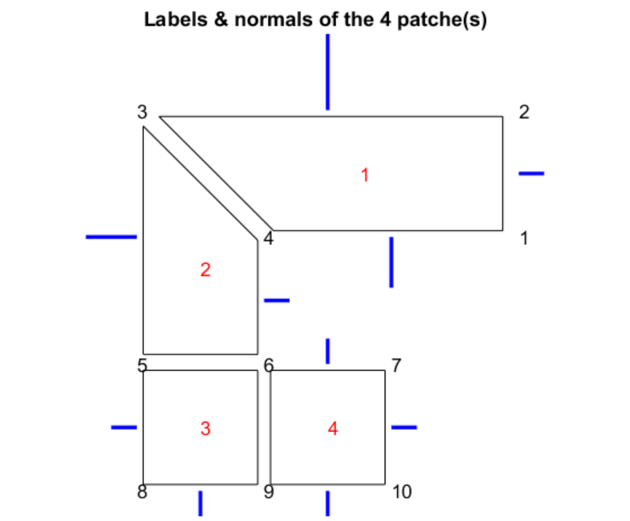


Figure 26: Cad model with patches labels

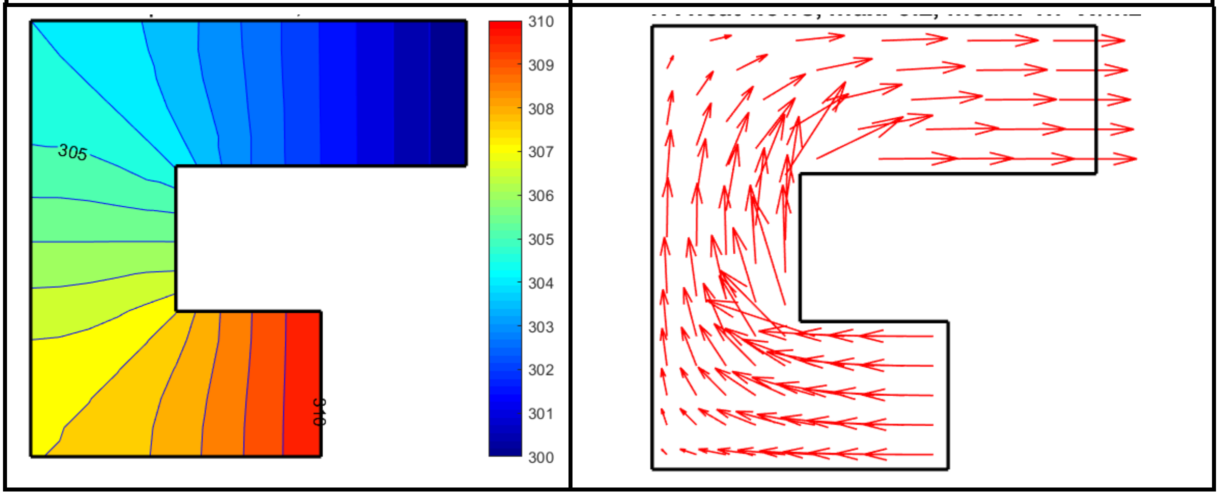


Figure 27: CAD model with diagonal on the long horizontal side

In the next test, we reproduce the same conditions as in *Figure 4*. In *cad_Dir.m*, instead of *line 70*, *line 68* is activated.

```
Fiammetta
C shape 3 patches Gi: 3, Di : 6
L   4, Meth., Ne, ca: 1  0  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  : 1 W/(mK)
L  10, DT isotherms  : 0.5 K
L  15, np, nvertices: 3  8
L  17, num. nod side: 4
L  19, num elem side: 5
L  29, Domain area   : 6 m2
L  33, Num. elements: 75
L  34, Num. of DOF   : 96
L  66, Domain perim.: 14 m
```

```
LD 69, Fixed nodes  : 1 … 2   7  … 8
N  03, param Ne     : 0
L  96, Anis. index  : 0
L 242, Total dissip.: 0.997 WK
L 244, Dis. in solid: 0.997 WK
L 245, DT   in solid: 10 K
L 255, Date, CPU, 13-Jul-2023, 0.65762 s
g  18, Max temp grad: 3, mean: 1.8 K/m
hf 25, Max heat flow: 3, mean: 1.8 W/m2
```

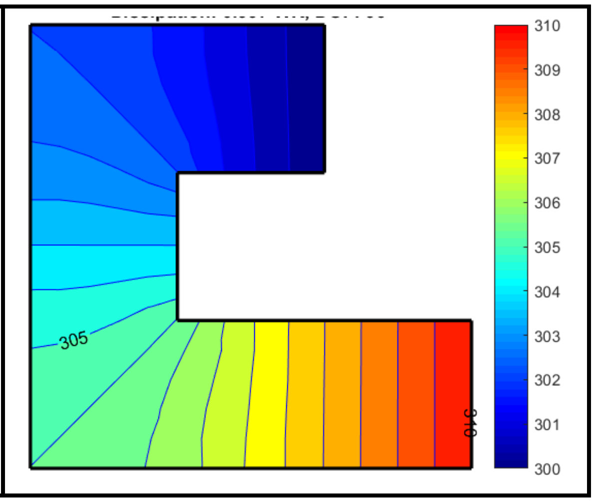
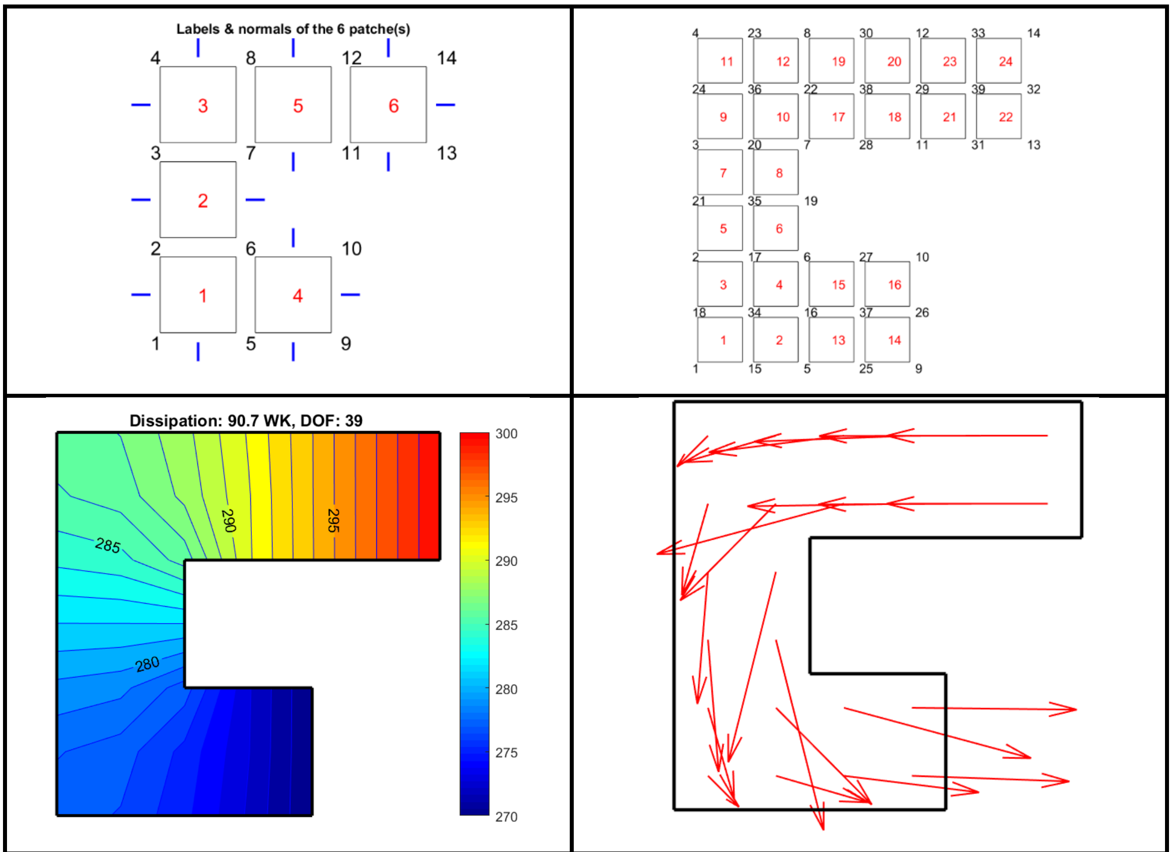


Figure 28: CAD model based on 3 trapezoidal patches, 75 elements

In the next test, we avoid the distorted shapes by using only squares.




```
Fiammetta
C shape 6 patches Gi: 43, Di : 23
L   4, Meth., Ne, ca: 1  0  0
L   5, Co nnr nvn   : 0  0  0
L   6, rc, ra, cs   : 0  0  0
L   7, CAD interf.  : 19
L   8, Thickness    : 1 m
L   9, Conduction k : 1 W/(mK)
L  10, DT isotherms : 1 K
L  11, Refl. coeff. : 0
L  15, np, nvertices: 6  14
L  17, num. nod side: 1
L  19, num elem side: 2
```

```
L  30, Domain area   : 6 m2
L  33, Num. elements: 24
L  35, Num. of nodes: 39
L  36, Num. of DOF  : 39
L  66, Domain perim.: 14 m
N  03, param Ne     : 0
L  94, Anis. index  : 0
L 215, Total dissip.: 90.7 WK
L 217, Dis. in solid: 90.7 WK
L 218, DT   in solid: 30 K
L 255, Date, CPU, 13-Jul-2023, 0.55447 s
g  18, Max temp grad: 7.4, mean: 5.2 K/m
hf 25, Max heat flow: 7.4, mean: 5.2 W/m2
```

Figure 29: CAD model fully based on square patches

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 x 3 convective matrices (34) and for each element the convection coefficient (uni-line matrix *he*).

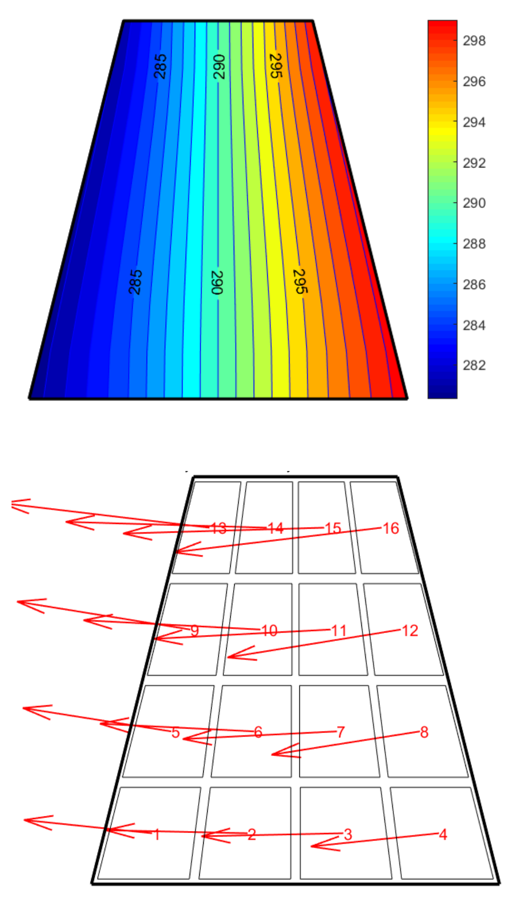
3.7 Basic isotherm drawing

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* x *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© 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.

```
Fiammetta
Trapezoidal shape Gi:  6, Di : 7
L   4, Meth., Ne, ca: 1  0  0
L   5, Co nnr nvn    : 1  0  2
L   6, rc, ra, cs    : 0  0  0
L   7, CAD interf.   : 4
L   8, Thickness     : 0.1 m
L   9, Conduction k  : 1 W/(mK)
L  10, DT isotherms  : 1 K
L  13, Convection h  : 18 W/(m2K)
L  15, np, nvertices: 1  4
L  17, num. nod side: 3
L  19, num elem side: 4
L  30, Domain area   : 0.75 m2
L  33, Num. elements: 16
L  35, Num. of DOF   : 27
L  66, Domain perim.: 3.5616 m
N  03, param Ne      : 0
c. 03, dK = no + nvn: 27
c. 04, N.virt c.nod.: 2
c. 05, Variable Co   : 1
c. 12, Convect. sid.: 2  3  4  1
c 125, Nu. conv. el.: 8
c 126, conv. coeff. : 18 … 18 W/(m2K)
L  96, Anis. index   : 0
L 216, Total dissip.: 24.3 WK
L 219, Dis. in solid: 20.9 WK
L 220, DT   in solid: 18.5 K
L 225, Fixed DOF     : 26  27
L 226, Imposed temp.: 300    280 K
L 227, React. flows : 2.43 -2.43 W
L 255, Date, CPU, 13-Jul-2023, 0.5411 s
g  18, Max temp grad: 30, mean: 24 K/m
```



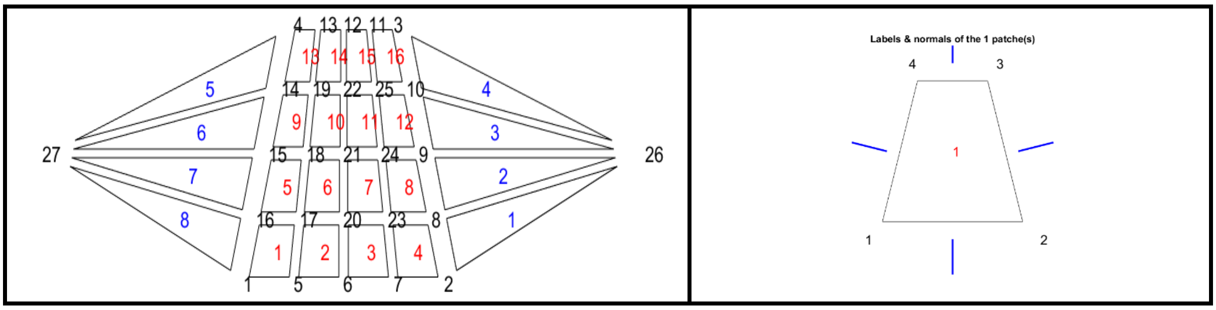


Figure 30: Two imposed temperatures, two adiabatic faces in a trapezoidal domain

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.

```
Fiammetta
Trapezoidal shape Gi:  6, Di : 7
L   4, Meth., Ne, ca: 1  0  0
L   5, Co nnr nvn   : 1  0  2
L   6, rc, ra, cs   : 0  0  0
L   7, CAD interf.  : 4
L   8, Thickness    : 0.1 m
L   9, Conduction k : 1 W/(mK)
L  10, DT isotherms : 1 K
L  13, Convection h : 18 W/(m2K)
L  15, np, nvertices: 1  4
L  17, num. nod side: 31
L  19, num elem side: 32
L  29, Domain area  : 0.75 m2
L  32, Num. elements: 1024
L  34, Num. of DOF  : 1091
L  63, Domain perim.: 3.5616 m
N  03, param Ne     : 0
c. 03, dK = no + nvn: 1091
c. 04, N.virt c.nod.: 2
c. 05, Variable Co  : 1
c. 10, Convect. sid.: 2  3  4  1
c 125, Nu. conv. el.: 64
L  96, Anis. index  : 1
Lm 48, k & coef*k   : 1  1000 W/(m K)
Lm 49, Thickn. ratio: 0.0625
L 242, Total dissip.: 53.6 WK
L 219, Dis. in solid: 27.7 WK
L 220, DT   in solid: 18.3 K
L 225, Fixed DOF    : 1090  1091
L 252, Imposed temp.: 300       280 K
L 253, React. flows : 5.36      -5.36 W
L 255, Date, CPU, 13-Jul-2023, 0.57642 s
g  18, Max temp grad: 130, mean: 23 K/m
ch 03, coef. red. dt: 25 W/m2
ch 19, temp. grad.  : 0.9164 K
ch 20, Mean conv. fl: -2.8007 0  0 W/m2
```

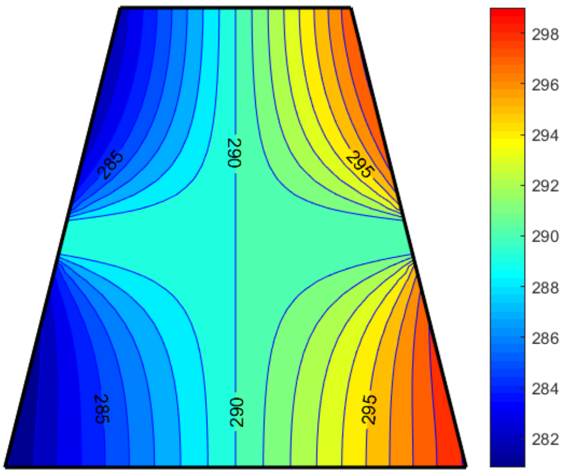


Figure 31: Non homogeneous trapezoidal domain

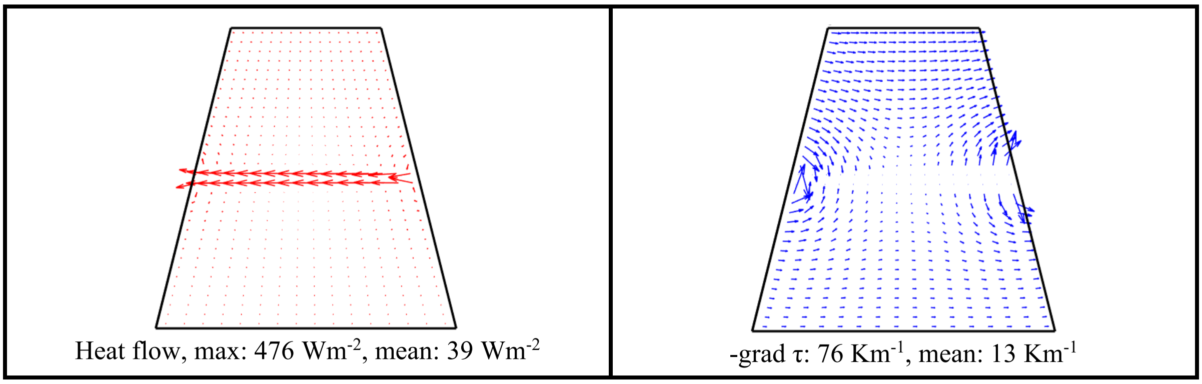


Figure 32: Horizontal strip with high conductivity in a trapezoidal domain

4. 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 C (JK^{-1}) is an extensive scalar quantity.

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

$$\alpha = \frac{k}{\rho\, c_p} \tag{70}$$

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**.**

$$\left([C]+\theta\,\Delta t\,[K]\right)\left[T^{n+1}\right]=\left([C]-(1-\theta)\,\Delta t\,[K]\right)\left[T^{n}\right]+\Delta t\,\left(\theta\left[f^{n+1}\right]+(1-\theta)\left[f^{n}\right]\right) \tag{71}$$

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

$$\left([C]+\Delta t\,[K]\right)\left[T^{n+1}\right]=[C]\left[T^{n}\right]+\Delta t\,\left[f^{n+1}\right] \tag{72}$$

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

1. the unknown nodal temperatures T_1 and
2. the fixed and therefore, constant temperatures T_f

$$[T]=\begin{bmatrix} T_1 \\ T_f \end{bmatrix} \tag{73}$$

Equation (62) becomes:

$$\left(\begin{bmatrix} C_{11} & C_{1f} \\ C_{f1} & C_{ff} \end{bmatrix}+\Delta t\begin{bmatrix} K_{11} & K_{1f} \\ K_{f1} & K_{ff} \end{bmatrix}\right)\begin{bmatrix} T_1^{n+1} \\ T_f^{n+1} \end{bmatrix}=\begin{bmatrix} C_{11} & C_{1f} \\ C_{f1} & C_{ff} \end{bmatrix}\begin{bmatrix} T_1^{n} \\ T_f^{n} \end{bmatrix}+\Delta t\begin{bmatrix} f^{n+1} \\ r^{n+1} \end{bmatrix} \tag{74}$$

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:

$$\left(\begin{bmatrix} C_{11} & C_{1f} \end{bmatrix} + \Delta t \begin{bmatrix} K_{11} & K_{1f} \end{bmatrix}\right) \begin{bmatrix} T_1^{n+1} \\ T_f^{n+1} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{1f} \end{bmatrix} \begin{bmatrix} T_1^n \\ T_f^n \end{bmatrix} + \Delta t \left[f^{n+1} \right] \tag{75}$$

Developing this relation gives:

$$\left(\left[C_{11}\right] + \Delta t\ \left[K_{11}\right]\right)\left[T_1^{n+1}\right] = \left[C_{11}\right]\left[T_1^n\right] + \Delta t\left(\left[f^{n+1}\right] - \left[K_{1f}\right]\left[T_f\right]\right) \tag{76}$$

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

$$\left[T_1^{n+1}\right] = \left(\left[C_{11}\right] + \Delta t\ \left[K_{11}\right]\right)^{-1} \left\{\left[C_{11}\right]\left[T_1^n\right] + \Delta t\left(\left[f^{n+1}\right] - \left[K_{1f}\right]\left[T_f\right]\right)\right\} \tag{77}$$

Without fixation:

$$\left[T^{n+1}\right] = \left(\left[C\right] + \Delta t\ \left[K\right]\right)^{-1} \left\{\left[C\right]\left[T^n\right] + \Delta t\ \left[f^{n+1}\right]\right\} \tag{78}$$

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

$$\left[T^{n+1}\right] = \left(\left[C\right] + \Delta t\ \left[K\right]\right)^{-1} \left[C\right]\left[T^n\right] \tag{79}$$

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

$$\left(\begin{bmatrix} C_{f1} & C_{ff} \end{bmatrix} + \Delta t \begin{bmatrix} K_{f1} & K_{ff} \end{bmatrix}\right) \begin{bmatrix} T_1^{n+1} \\ T_f \end{bmatrix} = \begin{bmatrix} C_{f1} & C_{ff} \end{bmatrix} \begin{bmatrix} T_1^n \\ T_f \end{bmatrix} + \Delta t \left[r^{n+1} \right]$$

$$\left[r^{n+1}\right] = \frac{1}{\Delta t}\left[C_{f1}\right]\left(\left[T_1^{n+1}\right] - \left[T_1^n\right]\right) + \left[K_{f1}\right]\left[T_1^{n+1}\right] + \left[K_{ff}\right]\left[T_f\right] \tag{80}$$

4.2 Capacity matrix

4.2.1 Quadrilateral

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

$$\tau = T_1\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right) + T_2\frac{x}{a}\left(1-\frac{y}{b}\right) + T_3\frac{x}{a}\frac{y}{b} + T_4\left(1-\frac{x}{a}\right)\frac{y}{b} \tag{81}$$

$$\begin{aligned} \tau &= [F][T] \\ [F] &= \left[\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right) \quad \frac{x}{a}\left(1-\frac{y}{b}\right) \quad \frac{x}{a}\frac{y}{b} \quad \left(1-\frac{x}{a}\right)\frac{y}{b}\right] \\ [T]^T &= \left[T_1 \quad T_2 \quad T_3 \quad T_4\right] \end{aligned} \tag{82}$$

$$[C] = \int_V \rho c_p [F]^T [F] dV \tag{83}$$

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.

$$C_{33} = e\int_0^a \left(\int_0^b \rho c_p \ \frac{x^2 y^2}{a^2 b^2} \ dy \right) \ dx$$

$$= \rho e c_p \ \int_0^a \frac{b^3 x^2}{3a^2 b^2} \ dx = \rho e c_p \ \frac{b}{3}\int_0^a \frac{x^2}{a^2} dx = \rho e c_p \ \frac{ab}{9} = \frac{\rho V c_p}{9} \quad (84)$$

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^{-1}.

$$[C] = \frac{\rho\ V\ c_p}{36} \begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix} \quad (85)$$

The capacity matrix of an element given by its localization vector *lo* and the set of nodal coordinates *xyz* is computed in the Matlab© function *fem_Cae.m* (*Table 43*).
Table 10 shows Matlab© 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 ($Jkg^{-1}K^{-1}$) and the specific mass ro (kgm^{-3}). 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^{-1} for the four first cases and 10^6 JK^{-1} for the last one.

Input	`xyz =[0 0 0;1 0 0;1 1 0;0 1 0];lo=[1 2 3 4]; [C] = fem_Cae(xyz,lo)*36`

Matlab Output	`C = 4 2 1 2`
	`2 4 2 1`
	`1 2 4 2`
	`2 1 2 4`
	`sum(sum(C)) = 36.`
	`total capacity: 250000` JK^{-1}

Input	`xyz=[0 0 0;2 0 0;2 .5 0;0 .5 0];lo=[1 2 3 4];[C] = fem_Cae(xyz,lo)*36`

Matlab Output	`C = 4 2 1 2`
	`2 4 2 1`
	`1 2 4 2`
	`2 1 2 4`
	`sum(sum(C)) = 36.`
	`total capacity: 250000` JK^{-1}

Input	`xyz=[0 0 0;.5 0 0;.5 2 0;0 2 0];lo=[1 2 3 4];[C] = fem_Cae(xyz,lo)*36`

Matlab Output	`C = 4 2 1 2`
	`2 4 2 1`
	`1 2 4 2`
	`2 1 2 4`
	`sum(sum(C)) = 36.`
	`total capacity: 250000` JK^{-1}

Input	`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`

Matlab Output	`C = 5 2.5 1 2`
	`2.5 5 2 1`
	`1 2 3 1.5`
	`2 1 1.5 3`
	`sum(sum(C)) = 36.`
	`total capacity: 250000` JK^{-1}

Input	`xyz =[0 0 0;2 0 0;2 2 0;0 2 0];lo=[1 2 3 4];[C] = fem_Cae(xyz,lo)*36`

Matlab Output	`C = 16 8 4 8`
	`8 16 8 4`
	`4 8 16 8`
	`8 4 8 16`
	`sum(sum(C)) = 144`
	`total capacity: 1000000` JK^{-1}

Table 10: Numerical integration of the capacity matrix

4.2.2 Triangle

The capacity matrix [C] is a function of the density ρ of the material, its heat capacity c_p and its volume V. The temperature field in the triangle S_1 - S_2 - S_3 is a first-degree polynomial:

$$\tau = \alpha_0 + \alpha_1 x + \alpha_2 y \tag{86}$$

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

$$T' = \begin{bmatrix} T_1 & T_2 & T_3 \end{bmatrix} \tag{87}$$

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

$$[R] = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} \tag{88}$$

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

$$A = \frac{1}{2}\left(x_1 y_2 + x_2 y_3 + x_3 y_1 - x_2 y_1 - x_3 y_2 - x_1 y_3\right) \tag{89}$$

The relation is now:

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = [R] \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \tag{90}$$

And, by inverting this relation:

$$[R]^{-1} = \frac{1}{2A} \begin{bmatrix} x_2 y_3 - x_3 y_2 & x_3 y_1 - x_1 y_3 & x_1 y_2 - x_2 y_1 \\ y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{bmatrix} = [F] \tag{91}$$

We can now express the quantities in term of nodal temperatures

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = [F] \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{92}$$

$$\tau = \begin{bmatrix} 1 & x & y \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 1 & x & y \end{bmatrix} [F] \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{93}$$

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

$$[C] = e\rho c_p [F]^T \int_S \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \begin{bmatrix} 1 & x & y \end{bmatrix} dS [F] \qquad (94)$$

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 33*.

4.4 Temperature homogenization in an insulated solid

A uniform temperature test is carried out on a domain of dimensions (2 *m* x 1*m* 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© instruction: *sum (sum (C))*. For this example, it is equal to 2.5 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 33*). 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*.

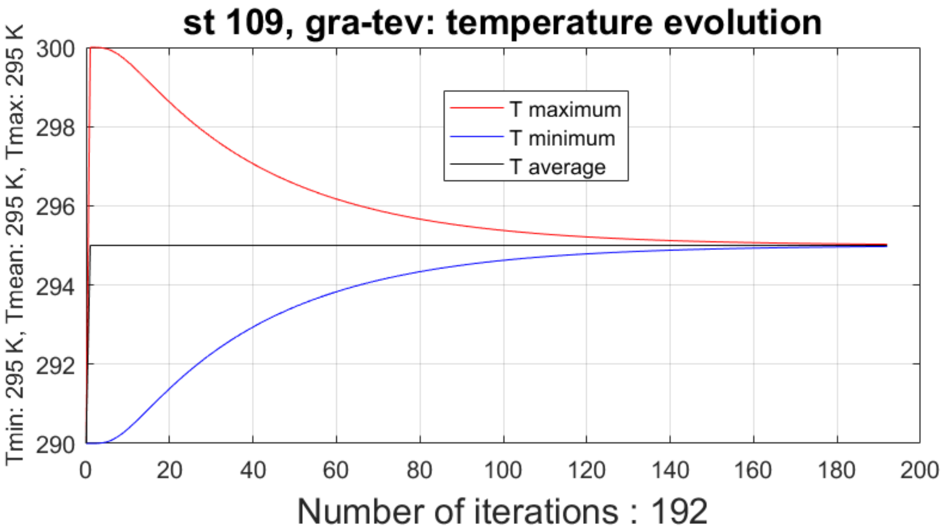
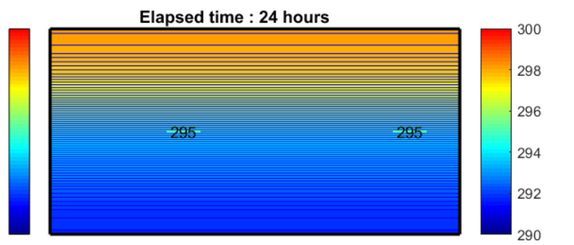


Figure 33: Smoothing process convergence in temperature homogenization

```
Fiammetta
Horizon. 1 rect., Gi: 17, Di : 0
L   4, Meth., Ne, ca: 3  0  0
L   5, Co nnr nvn    : 0  0  0
L   6, rc, ra, cs    : 0  0  5
L   7, CAD interf.   : 4
L   8, Thickness     : 0.5 m
L   9, Conduction k  : 2 W/(mK)
L  10, DT isotherms  : 0.1 K
L  11, Refl. coeff.  : 0
L  15, np, nvertices: 1  4
L  17, num. nod side: 8
L  19, num elem side: 9
L  30, Domain area   : 2 m2
```



```
L  33, Num. elements: 81
L  35, Num. of nodes: 100
L  36, Num. of DOF  : 100
L  66, Domain perim.: 6 m
L  70, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD  2, num. nod side: 8
LD  3, Dirich. label: 0
LD  4, N. virt. nod.: 0
LD  7, Numb. fix nod: 0
N  03, param Ne     : 0
L  89, Numb ra nodes: 20
L  90, Rad. vert. lv: 2  3  4  1
L  94, Anis. index  : 0
st 08, Ini. tmi, tma: 290  300 K
st 09, Numb. fixat. : 0
st 11, fmd          : 1
st 27, N. iter. nit : 96
st 28, Time step    : 3600 s
st 31, Analyzed per.: 96 h, 4 days
st 36, Spec. capac. : 1000 J/(kg.K)
st 37, Spec. mass   : 2500 kg.m-3
st 47, sum(sum(C))  : 2.5  MJ/K
st 48, area*th*ro*Cp: 2.5  MJ/K
st 52, Imposed Heat : 100  W/m-2
st 57, size(K) nfi  : 100  100  0
st 82, iteration    : 24
st 82, iteration    : 48
st 82, iteration    : 72
st 82, iteration    : 96
st104, ddt Tm - Tin : 5    K
st105, ddt*sumsum(C): 12.5 MJ
st106, Min obs. temp: 290  K
st107, Max obs. temp: 300  K
st108, Heat inp.eih : 11   MJ
L 245, Date, CPU, 18-Jul-2023, 1.2298 s
g  18, Max temp grad: 1.3, mean: 0.84 K/m
hf 24, Max heat flow: 2.6, mean: 1.7 W/m2
```

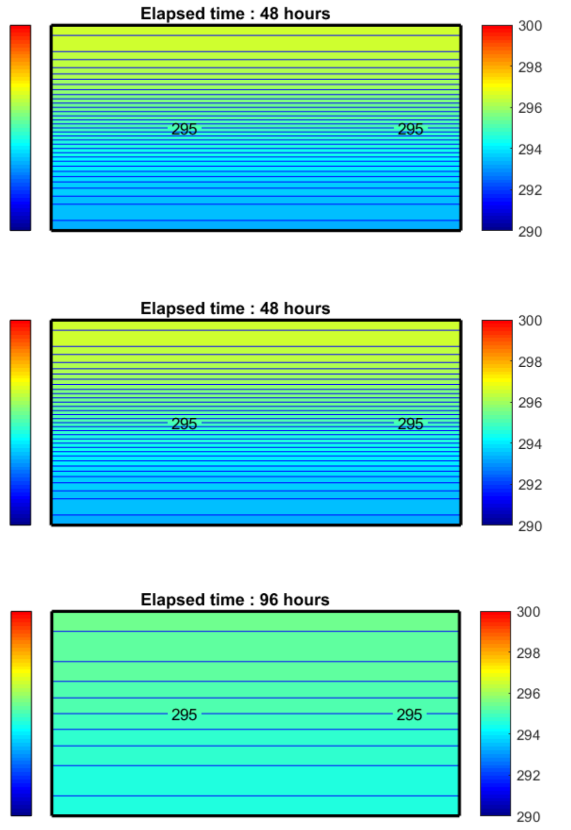
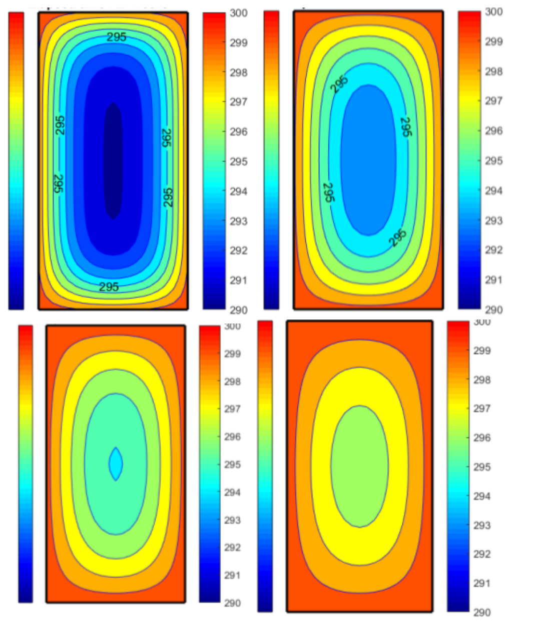


Figure 34: Evolution of the temperature field in a smoothing process

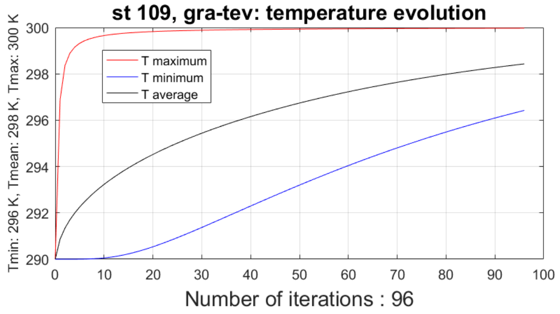
4.5 Heating of a solid at initial uniform temperature

The next example (*Figure 35*, *Figure 36*) 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 290 *K*. After 96 hours, the mean temperature is 300 *K* (*Figure 35e*). 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: 4.21 *MJ* (output *st105*).

```
Fiammetta
Standard 1 rect., Gi: 32, Di : 14
L   4, Meth., Ne, ca: 3  0  0
L   5, Co nnr nvn   : 9  0  4
L   6, rc, ra, cs   : 0  0  0
L   7, CAD interf.  : 7
L   8, Thickness    : 0.1 m
L   9, Conduction k : 1   W/(mK)
L  10, DT isotherms : 1   K
L  13, Convection h : 25  W/(m2K)
L  17, num. nod side: 19
L  19, num elem side: 20
L  30, Domain area  : 2   m2
L  33, Num. elements: 800
L  35, Num. of nodes: 861
L  36, Num. of DOF  : 865
L  66, Domain perim.: 6 m
LD  4, N. virt. nod.: 4
LD 88, Fix. nod. lfi: 862  863  864  865
LD 89, Fix. temp. fT: 300  300  300  300 K
N  03, param Ne     : 0
c. 03, dK = no + nvn: 865
c. 04, N.virt c.nod.: 4
c. 05, Variable Co  : 9
c.116, Convect. sid.: 2 4 4 6 6 5 5 3 3 1 1 2
c.122, cad_con cs   : 0
c.192, Nu. conv. el.: 120
```



```
L  94, Anis. index  : 0
st 08, Ini. tmi, tma: 290  300 K
st 09, Numb. fixat. : 4
st 22, Fix. temp. fT: 300  300  300  300 K
st 23, ddl fix. lfi : 862  863  864  865
st 27, N. iter. nit : 96
st 28, Time step    : 3600 s
st 31, Analyzed per.: 96 h, 4 days
st 36, Spec. capac. : 1000 J/(kg.K)
st 37, Spec. mass   : 2500 kg.m-3
st 47, sum(sum(C))  : 0.5  MJ/K
st 48, area*th*ro*Cp: 0.5  MJ/K
st 52, Imposed Heat : 100  W/m-2
st 57, size(K) nfi  : 865  865    4
st 82, iteration    : 24 - 48 - 72 - 96
st104, ddt Tm - Tin : 8.43 K
st105, ddt*sumsum(C): 4.21 MJ
st106, Min obs. temp: 290  K
st107, Max obs. temp: 300  K
st112, Heat inp.eih : 2.2  MJ
L 272, Date, CPU, 12-Aug-2023, 3.365 s
g  18, Max temp grad: 10, mean: 5.6 K/m
hf 24, Max heat flow: 10, mean: 5.6 W/m2
```



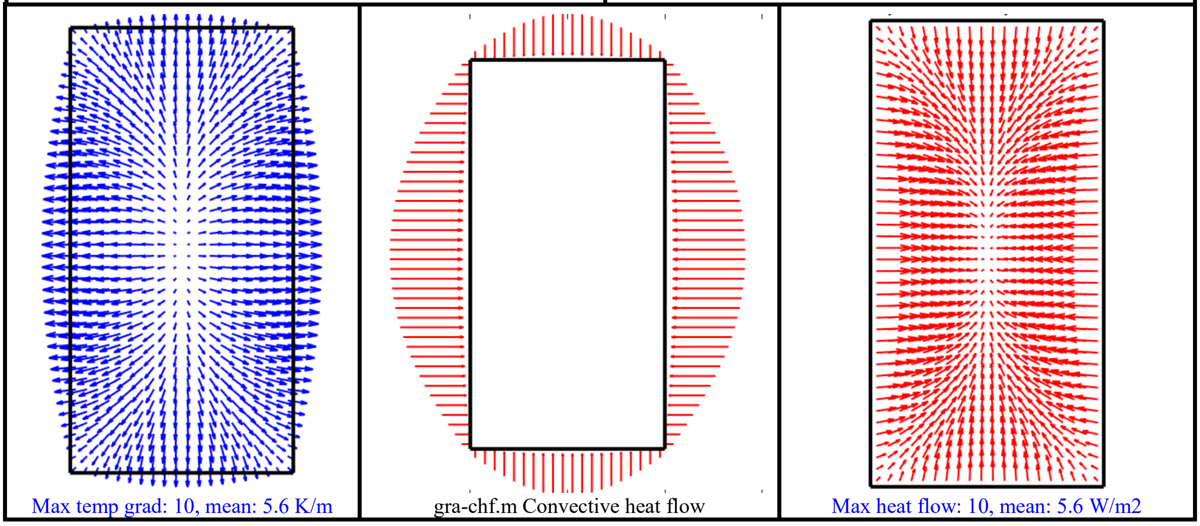


Figure 35: Evolution of the temperature field in a heating process

In *Figure 36*, the mesh involves 3200 conductive elements, 3325 *DOF*. The initial temperature is 280 K. Some output is 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
```

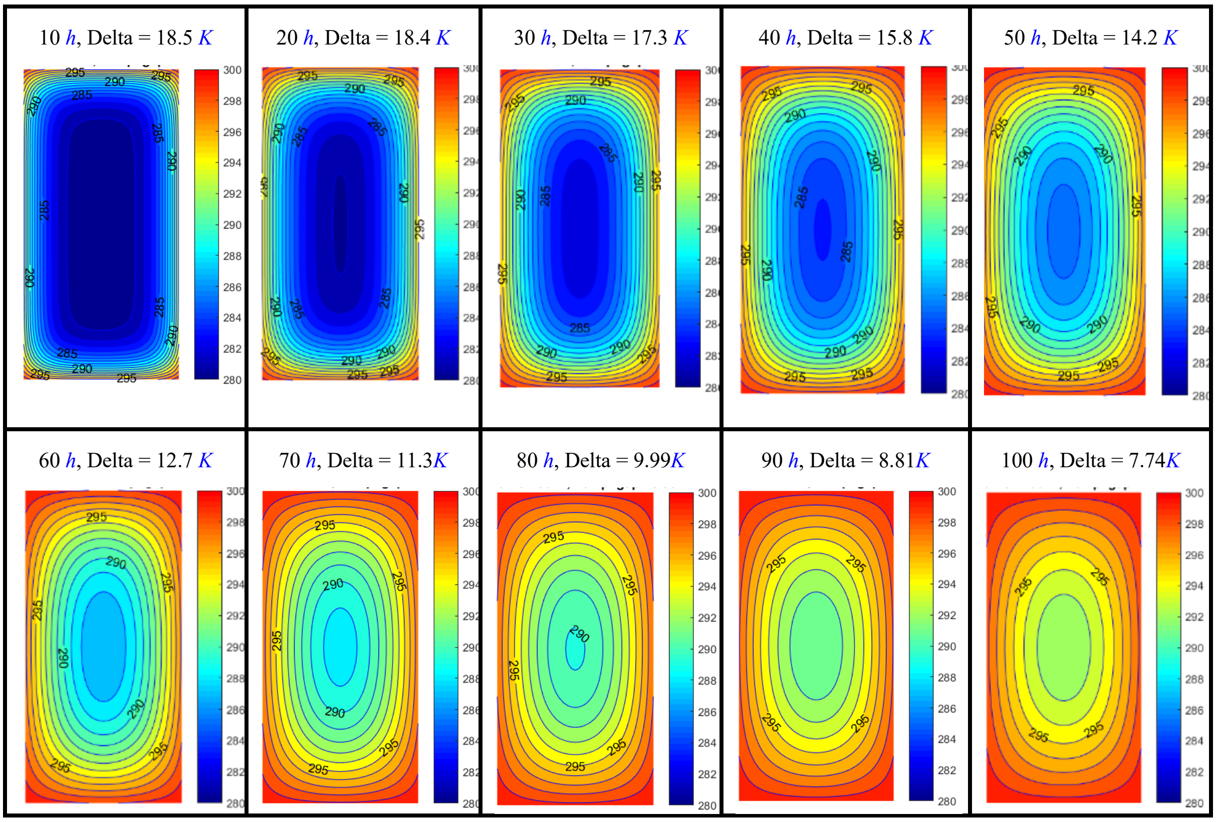


Figure 36: Evolution of isotherms in a heating operation: 100 h

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
LD 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/m-2
st 57, size(K) nfi   : 288  288    9
st 82, iteration     : 180 360 540 720
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
st112, Heat inp.eih : 12.4 MJ
L 272, Date, CPU, 21-Jul-2023, 3.2234 s
hf 24, Max heat flow: 20, mean: 9.3 W/m2
```

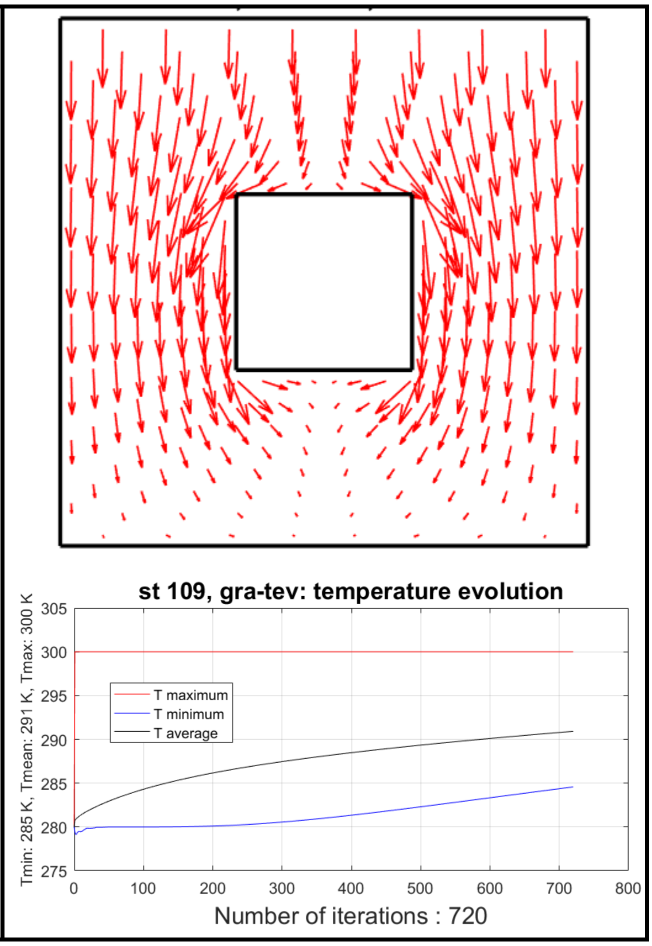
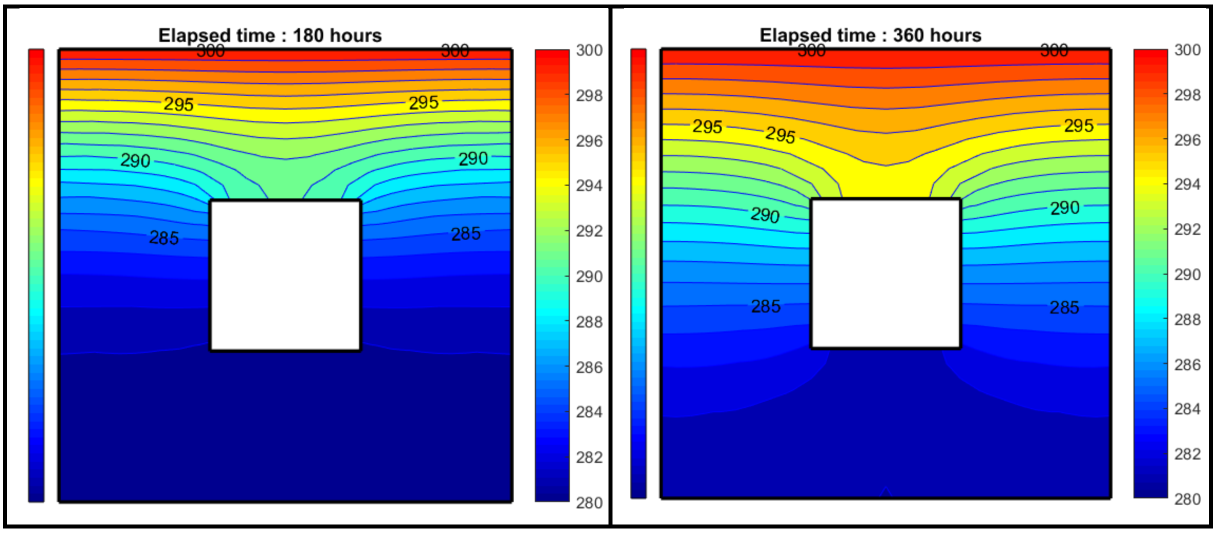


Figure 37: Adiabatic cavity, 1 month, T_{ini} = 280 K, T_{top} = 300 K

The results of *Figure 37* confirms that the stored heat (`t.101`) is equal to 21.8 *MJ* which is the product of the total capacity (`t.47`) of 2 *MJ/K* by the difference of temperature of 10.9 *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 38* , all the illustrations correspond to the temperature gap: 280 - 300 *K*.

To enforce the colorbar to act always between the initial temperature *tmi* and 300 *K*, the Matlab© instructions *80* & *83* are enabled in *fem_til.m*.



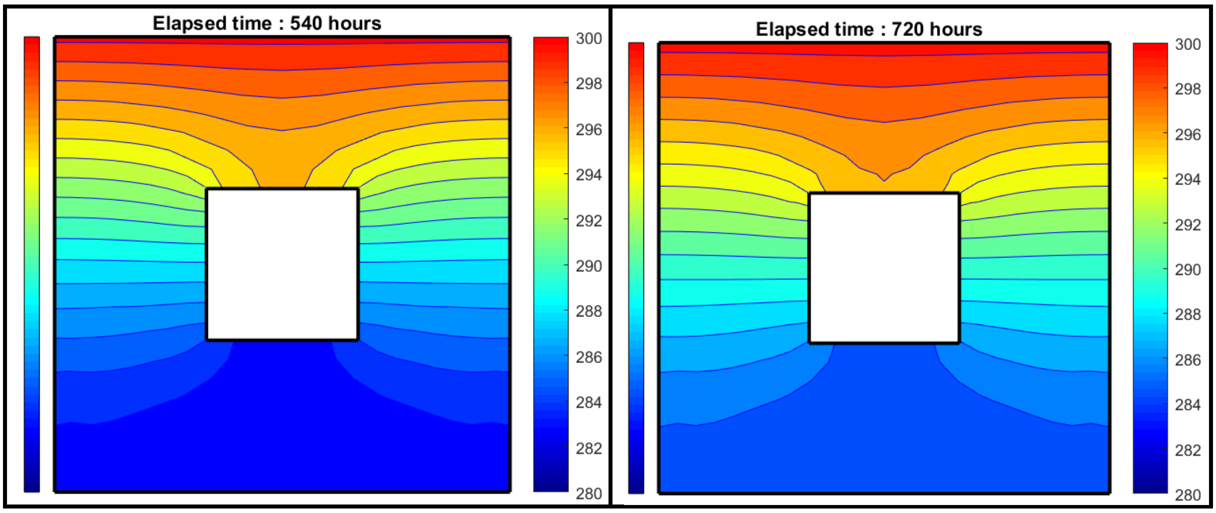


Figure 38: Adiabatic cavity, evolution of the temperature field

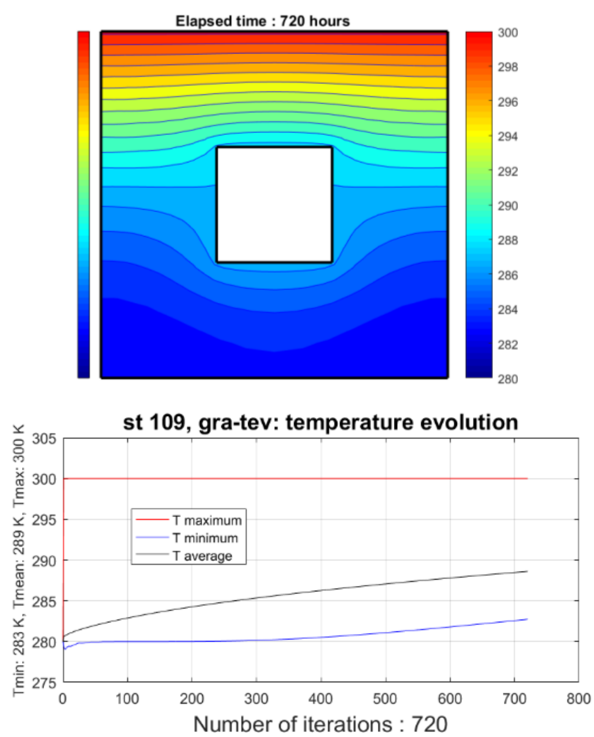
4.7 Square cavity with convection

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 *Figure 39*. This action is performed in the *gra_chf.m* function (*Table 58*).

```
Fiammetta
Standard cavity,  Gi: 2, Di : 1
L   4, Meth., Ne, ca: 3  0  1
L   5, Co nnr nvn   : 2  0  1
L   6, rc, ra, cs   : 0  0  5
L   8, Thickness    : 0.1 m
L   9, Conduction k : 1 W/(mK)
L  10, DT isotherms : 1 K
L  13, Convection h : 18 W/(m2K)
L  15, np, nvertices: 4  8
L  17, num. nod side: 9
L  19, num elem side: 10
L  30, Domain area  : 8 m2
L  33, Num. elements: 400
L  36, Num. of DOF  : 441
L  66, Domain perim.: 16 m
LD  2, num. nod side: 9
LD  4, N. virt. nod.: 1
LD 12, Fixed nodes  : 3  72 … 80   4
N  03, param Ne     : 0
c. 03, dK = no + nvn: 441
c. 04, N.virt c.nod.: 1
c. 05, Variable Co  : 2
c. 22, Convect. sid.: 6 5 7 6 8  7  5  8
c.192, Nu. conv. el.: 40
L  89, Numb ra nodes: 22
L  90, Rad. vert. lv: 5  1  2  6
L  94, Anis. index  : 0
st 08, Ini. tmi, tma: 280  300 K
st 09, Numb. fixat. : 11
st 27, N. iter. nit : 720
st 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/m-2
st 82, iteration    : 180 360 - 540 - 720
```



Heat flows, 4 nodes per side:

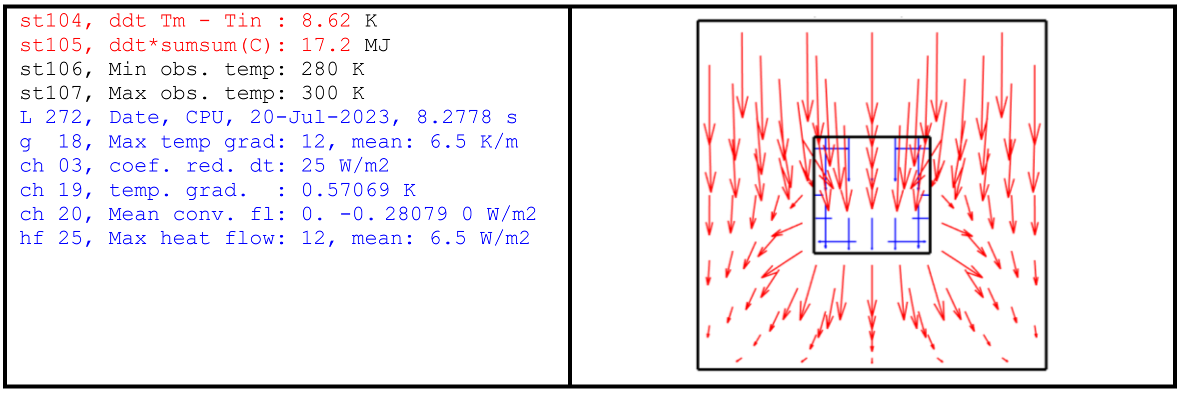


Figure 39: Isotherms, convection in the cavity, 1 month

5. 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 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 40*) according to the formula:

$$F_{dL-j} = \int_{y\in L_j} \frac{\cos\theta_{dL}\cos\theta_j}{2r}\, dy \tag{95}$$

The explicit expression of the view factor is computed via the Nusselt analogy (*Figure 40*) [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:

$$F_{dL-j} = \frac{1}{2}\left(\frac{x_1}{r_1} - \frac{x_0}{r_0}\right) \tag{96}$$

To obtain the view factors of the elements of a surrounding square from a differential element *dL* (*Figure 40*) 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*]:

$$\sum_{j=1}^{N} F_{ij} = 1 \qquad \textit{closure} \tag{97}$$

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

$$A_i F_{ij} = A_j F_{ji} \qquad \textit{reciprocity} \tag{98}$$

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 (96) to the side on which the element dL is located. Assuming that $\vec{r}_i / |r_i|$ 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:

$$F_{dL-j} = \frac{1}{2}\left(\frac{\vec{r}_1}{|\vec{r}_1|} - \frac{\vec{r}_0}{|\vec{r}_0|} \right) \cdot \vec{t} \quad (99)$$

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

$$F_{ij} = \frac{1}{A_i} \int_i F_{dL-j} dA_j \quad (100)$$

In this relation, dAj 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*© code. Capital letters represent vectors (seen as uni-column matrices) and capital letters between brackets represent matrices.

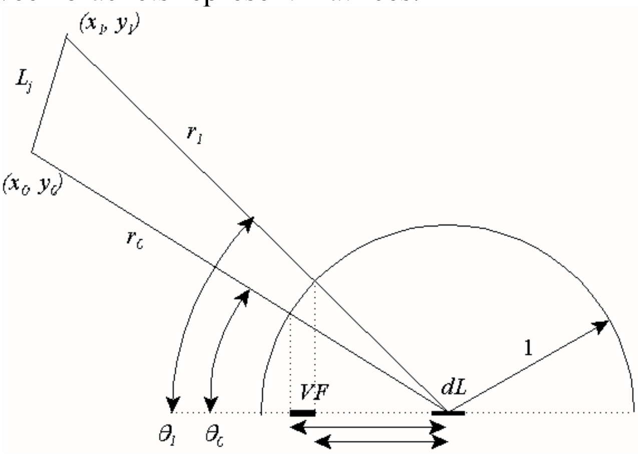


Figure 40: 2D "point – segment" view factor [Beckers 2011]

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}

$$B = E + [R]\ J$$

(101)

In (98), the adimensional diagonal matrix $[R]$ contains the reflection coefficients. They satisfy the Kirchhoff' law relating, in each element i, the emissivity ε_i, the absorptivity α_i, and the reflectivity ρ_i:

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

(102)

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 ρ_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):

$$J = [F]B + F_{sky}E_{sky} + F_{gr}E_{gr} \tag{103}$$

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

$$B = E + [R]\{ [F]B + F_{sky}E_{sky} + F_{gr}E_{gr}\} \tag{104}$$

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

$$[M] = [I] - [R][F] \tag{105}$$

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

$$\begin{aligned} E &= B - [R][F]B - [R]\{F_{sky}E_{sky} + F_{gr}E_{gr}\} \\ &= \{[I]-[R][F]\}B - [R]\{F_{sky}E_{sky} + F_{gr}E_{gr}\} \\ &= [M]B - [R]\{F_{sky}E_{sky} + F_{gr}E_{gr}\} \end{aligned} \tag{106}$$

From (102), we deduce:

$$B = [M]^{-1}\{E + [R]\{F_{sky}E_{sky} + F_{gr}E_{gr}\}\} \tag{107}$$

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

$$Q = B - J \tag{108}$$

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

$$\begin{aligned} Q &= B - [F]B - \{F_{sky}E_{sky} + F_{gr}E_{gr}\} \\ &= \{[I]-[F]\}B - \{F_{sky}E_{sky} + F_{gr}E_{gr}\} \end{aligned} \tag{109}$$

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

$$Q = \{[I]-[F]\}[M]^{-1}E + \{[I]-[F]\}[M]^{-1}[R]\{F_{sky}E_{sky} + F_{gr}E_{gr}\} - \{F_{sky}E_{sky} + F_{gr}E_{gr}\} \tag{110}$$

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

$$Q = \{[I]-[F]\}[M]^{-1}E + \left[\!\left[\{[I]-[F]\}[M]^{-1}[R] - \lfloor I \rfloor\right]\!\right]\{F_{sky}E_{sky} + F_{gr}E_{gr}\} \tag{111}$$

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 ε_i and the Stefan-Boltzmann constant σ ($Wm^{-2}K^{-4}$):

$$E \rightarrow E_i = \varepsilon_i \quad \sigma \quad T_i^4 \tag{112}$$

Mirror. If the surfaces are perfectly reflective (like mirrors): $[R] = [I]$, the emissivities ε_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 $\rho = .9999$

$$\left[\!\left[\{[I]-[F]\}[M]^{-1}[R]-\lfloor I \rfloor \right]\!\right] = \{[I]-[F]\}\{[I]-[F]\}^{-1} - \lfloor I \rfloor = \lfloor I \rfloor - \lfloor I \rfloor = 0 \qquad (113)$$

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 ε_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:

$$Q = E - \left\{ F_{sky} E_{sky} + F_{gr} E_{gr} \right\} \qquad (114)$$

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:

$$E = S^T T \text{ where } S_i T_i = \left(\varepsilon_i \ \sigma T_i^3 \right)^{it-1} \left(T_i \right)^{it} \qquad (115)$$

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

$$\begin{aligned} E_{sky} &\rightarrow E_{sky} = \varepsilon_{sky} \ \sigma \ T_{sky}^4 = \sigma \ T_{sky}^4 \text{(if sky emissivity = 1)} \\ E_{gr} &\rightarrow E_{gr} = \varepsilon_{gr} \ \sigma \ T_{gr}^4 = \sigma \ T_{gr}^4 \text{(if sky emissivity = 1)} \end{aligned} \qquad (116)$$

$$Q = \{[I]-[F]\}[M]^{-1} E + \left[\!\left[\{[I]-[F]\}[M]^{-1}[R]-\lfloor I \rfloor \right]\!\right] \left\{ F_{sky} E_{sky} + F_{gr} E_{gr} \right\} \qquad (117)$$

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.

$$Q = \{[I]-[F]\}[M]^{-1} S^T T + \left\{ \{[I]-[F]\}[M]^{-1}[R]-\lfloor I \rfloor \right\} \left\{ F_{sky} \varepsilon_{sky} \ \sigma \ T_{sky}^4 + F_{gr} \ \varepsilon_{gr} \ \sigma \ T_{gr}^4 \right\} \qquad (118)$$

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 3*D* and edges in 2*D*).

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 3*D* 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:

$$H = [W]\, Q \tag{119}$$

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*:

$$E_i = \varepsilon_i\ \sigma\ \left(\tau_i^4\right)_{it-1}\ \ \rightarrow\ \varepsilon_i\ \sigma\ \left(\tau_i^3\right)_{it-1}\left(\tau_i\right)_{it} \tag{120}$$

For the sky and the ground, the relations are:

$$e_{sky} = \sigma\tau_{sky}^4 \qquad \& \qquad e_{gr} = \sigma\tau_{gr}^4 \tag{121}$$

Equation (113) becomes:

$$\begin{aligned} Q &= \left[\{[I]-[F]\}[M]^{-1}\ \varepsilon_i\ \sigma\ \left(\tau_i^3\right)_{it-1}\right]\ \left(\tau_i\right)_{it} \\ &+ \ \left\{\{[I]-[F]\}[M]^{-1}[R]-\lfloor I\rfloor\right\}\ \left\{F_{sky}\sigma\tau_{sky}^4\ +\ F_{gr}\sigma\tau_{gr}^4\ \right\} \end{aligned} \tag{122}$$

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:

$$Q = F_{sid} = \{[I]-[F]\}\ [M]^{-1}\ S^T\ T_{sid} \tag{123}$$

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:

$$\begin{bmatrix} T_{sid}(1) \\ T_{sid}(2) \\ T_{sid}(3) \\ T_{sid}(4) \\ \\ \\ T_{sid}(n) \end{bmatrix} = 0.5 \begin{bmatrix} 1 & 1 & & & & & \\ & 1 & 1 & & & & \\ & & 1 & 1 & & & \\ & & & 1 & 1 & & \\ & & & & 1 & 1 & \\ & & & & & 1 & 1 \\ 1 & & & & & & 1 \end{bmatrix} \begin{bmatrix} T(1) \\ T(2) \\ T(3) \\ T(4) \\ \\ \\ T(n) \end{bmatrix} \quad \text{or} \quad [T_{sid}] = [L_c][T] \tag{124}$$

For the heat flows, we have the relation:

$$[F] = [L_c]^T\,[F_{sid}] \tag{125}$$

This relation is expanded according to (120).

$$[F] = [L_c]^T\ F_{sid} = [L_c]^T\ \{[I]-[F]\}\ [M]^{-1}\ S^T\ [L_c][T] \tag{126}$$

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

$$\left[K_{rad}\right]=\left[L_c\right]^T\left\{\left[I\right]-\left[F\right]\right\}\left[M\right]^{-1}S^T\left[L_c\right] \tag{127}$$

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:

$$\left[F\right]=\left[K_{rad}\right]T \tag{128}$$

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. When we only consider the sky, the equation becomes:

$$\left[K_{rad}\right]=\left[L_o\right]^T\left\{\left[I\right]-\left[F\right]\right\}\left[M\right]^{-1}S^T\left[L_o\right] \tag{129}$$

$$[F]=\left[K_{rad}\right][T]+L_{sky}^T\left[\left\{\left[I\right]-\left[F\right]\right\}\left[M\right]^{-1}\left[R\right]-\left\lfloor I\right\rfloor\right]F_{sky}\sigma\tau_{sky}^4 \tag{130}$$

$$\begin{bmatrix} T_{sid}(1) \\ T_{sid}(2) \\ T_{sid}(3) \\ \\ \\ T_{sid}(n) \end{bmatrix}=\frac{1}{2}\begin{bmatrix} 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & 1 & 1 & & \\ & & & & & \\ & & & & & \\ & & & & 1 & 1 \end{bmatrix}\begin{bmatrix} T(1) \\ T(2) \\ T(3) \\ T(4) \\ \\ \\ T(n) \end{bmatrix} \quad \text{or} \quad \left[T_{sid}\right]=\left[L_o\right]\left[T\right] \tag{131}$$

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 41*.

→ a) Quadrilateral cavities

```
[(1:npv)'  xyz_cao]              1     0     0
                                 2     3     0
                                 3     3     6
                                 4     0     6
                                 5     1     1
                                 6     2     1
                                 7     2     5
                                 8     1     5
[(1:np)'  car_cao]      1     6     5     1     2
                        2     6     2     3     7
                        3     7     3     4     8
                        4     1     5     8     4
[(1:nbo)'  bor(:,1:6)]
     1     6     5     1     0     9    10
     2     5     1     1     4    11    12
     3     1     2     1     0    13    14
     4     2     6     1     2    15    16
     5     2     3     2     0    17    18
     6     3     7     2     3    19    20
     7     7     6     2     0    21    22
     8     3     4     3     0    23    24
     9     4     8     3     4    25    26
    10     8     7     3     0    27    28
    11     5     8     4     0    29    30
    12     4     1     4     0    31    32
[(1:np)'  pbo]          1     1     2     3     4
                        2     4     5     6     7
                        3     6     8     9    10
                        4     2    11     9    12
```

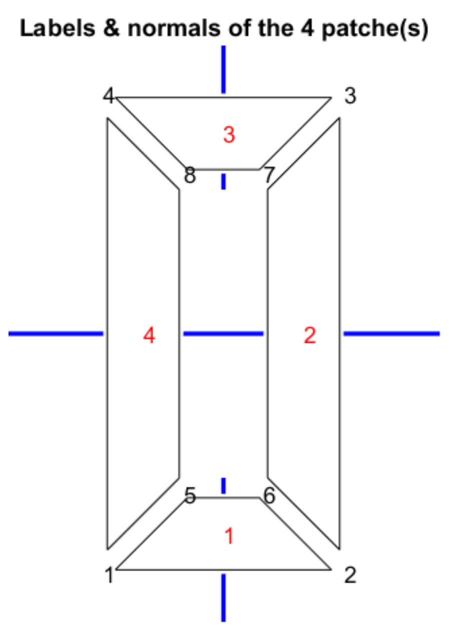


Figure 41: Rectangular cavity with the display of four matrices describing the geometry

The view factors are calculated using the "*point – segment*" method (96). In *Figure 41*, 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
   0.2764         0    0.2764    0.4472
   0.4472    0.2764         0    0.2764
   0.2764    0.4472    0.2764         0
```

```
Closure tests

 1     1     1     1
 1     1     1     1
```

```
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
   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

   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    0.1023    0.1160    0.1787    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)')    1     1     1     1     1     1     1     1
```

```
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     1
  disp(sum(F,2)')    1     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

```
disp(sum(F))    ; columns     1      1      1      1      1      1      1      1
disp(sum(F,2)'); lines    0.2954 0.2954 1.7046 1.7046 0.2954 0.2954 1.7046 1.7046
disp(sum([sum(F,2)'])/8): 1
```

```
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)')
```

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

```
disp(sum(Reci));       1.3150  1.3150 -0.2582 -0.2582 1.3150  1.3150 -0.2582 -0.2582
disp(sum(Reci,2)'); -0.2582 -0.2582 1.3150  1.3150 -0.2582 -0.2582 1.3150  1.3150
```

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

```
F = geo_vfr(2,[1 4]);disp(F); disp(sum(F)); disp(sum(F,2)'); Lel=[1 4 1 4];
```

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

```
disp(sum(F)      lines      1  1  1  1  1  1  1  1
disp(sum(F,2)) columns     0.3428 0.3428 1.6572 1.6572 0.3428 0.3428 1.6572 1.6572
```

```
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)');
```

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

disp(sum(Reci));     1.3153  1.315  -0.3295 -0.3295  1.3153  1.3153 -0.3295 -0.3295
disp(sum(Reci,2)');-0.3295 -0.3295  1.3153  1.3153 -0.3295 -0.3295  1.3153  1.3153
```

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 42*). 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© 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© 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© 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 42* with node (black) and patch (red) numbering. On the boundary of the domain, the outward normal vectors (blue) are also represented.

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 41*) using the Matlab© function *geo_vfc.m* of *Table 67* and the street section (*Figure 42*) using the Matlab© 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.

The street section analyses use the parameters *Gi* = 14 when radiation is present and Gi =15 for the linear transient analyses. (Matlab© function *cad_gin.m*, *Table 31*).

```
[(1:npv)'  xyz_cao]          1     0     8
                             2     1     8
                             3     0     0
                             4     1     1
                             5     5     0
                             6     4     1
                             7     5     8
                             8     4     8
[(1:np)'  car_cao]      1     2     1     3     4
                        2     4     3     5     6
                        3     6     5     7     8
[(1:nbo)'  bor]  1     2     1     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
[(1:np)'  pbo]          1     1     2     3     4
                        2     3     5     6     7
                        3     6     8     9    10
```

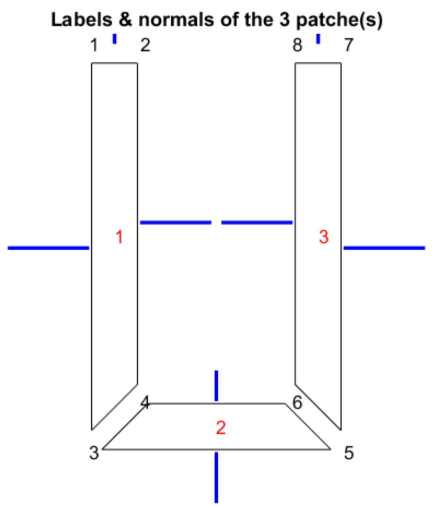


Figure 42: Data – Gi = 14, Gi = 15: Street section, aspect ratio dx/dy = 3/7

`F = geo_stf (2, [3 7]); disp(F); disp(sum(F,1)); disp(sum(F,2)')`

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

`F = geo_vfc (2, [3 7],4); disp(F); disp(sum(F,1)); disp(sum(F,2)')`

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	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)

`F = geo_stf (2, [3 7]); disp ([F 1-sum(F,2)]);disp (sum ([F 1-sum(F,2)],2)')`

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
1	1	1	1	1	1	

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	

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 43*), 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 44*), the vertical wall is always seeing half the ground, while the horizontal one is seeing more than half the ground.

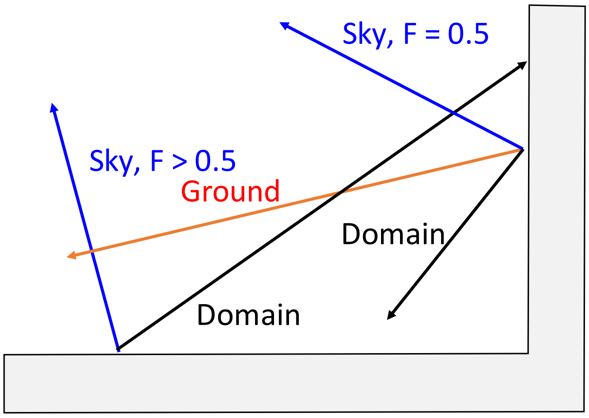


Figure 43: Vertical wall above horizontal edge

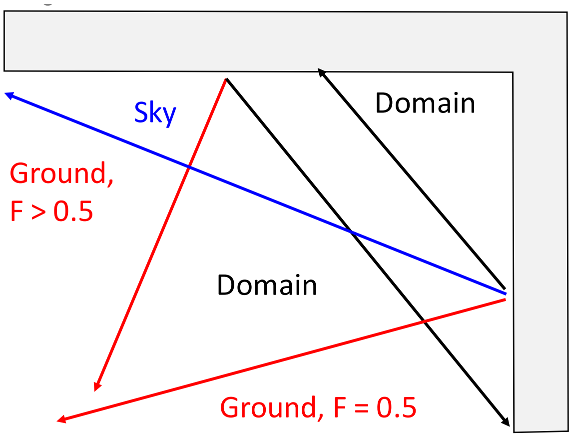


Figure 44: Vertical wall below horizontal edge

→ d) Thermal bridge

In this section we define the thermal bridge:

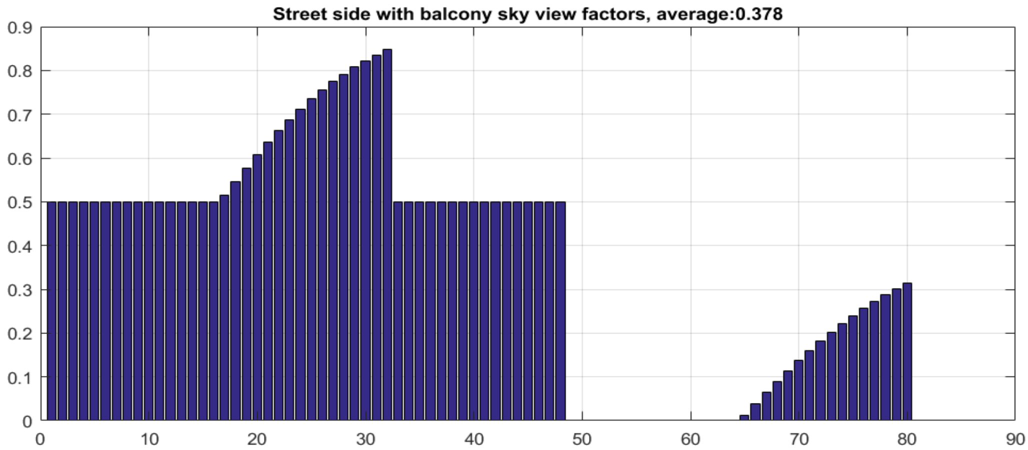
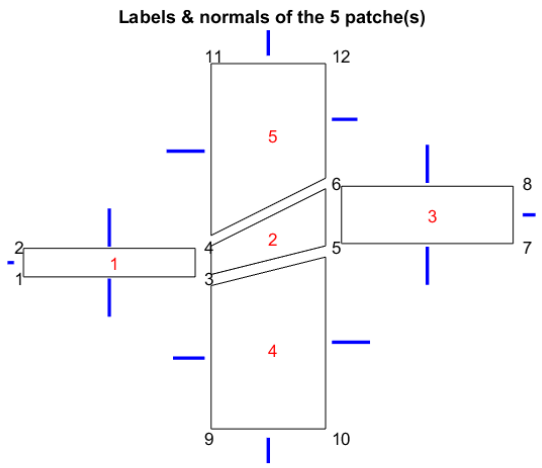


Figure 45: Fsky in a street with balcony – 16 segments / patch side

The data of the parameter *Gi* are defined in the Matlab© function *cad_gin.m*, *Table 31*

```
[(1:npv)'  xyz_cao]          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
[(1:np)'  car_cao]   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
[(1:nbo)'  bor]
      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]

    1     1     2     3     4
    2     5     6     7     2
    3     8     9    10     6
    4    11    12     5    13
    5     7    14    15    16
```

Figure 46: 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	0.0629	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	0.1254	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 45*), with the consequence that the corresponding lines and columns of matrix *F* are also equal to zero.

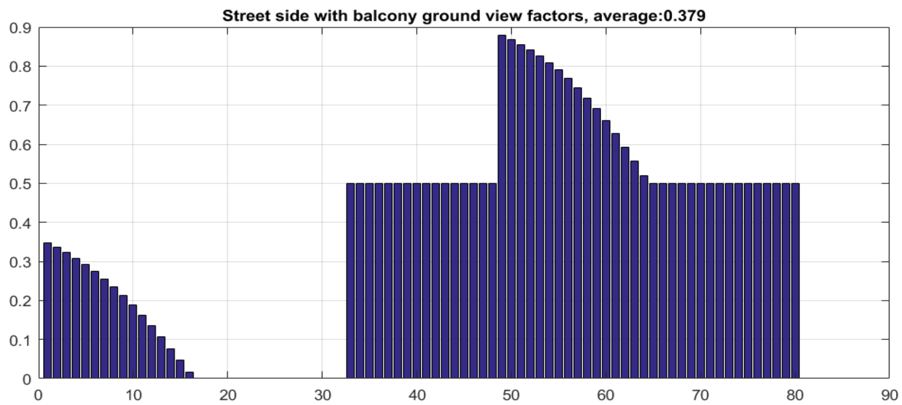


Figure 47: Fgr in a street with balcony – 16 segments / patch side

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© procedures presented in this document, ρ is noted *re*), the radiosity matrix [*M*] is computed according to the sequence of Matlab© 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 [*I*].
- 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
disp(det(M)); 8.7522e-17							
disp (sum(M)): 1.0e-15 *							
-0.1110	-0.0555	0	0	-0.0555	-0.0833	0	0
disp(sum(M,2)'): 1.0e-15 *							
-0.0555	-0.0833	-0.1110	-0.0555	0.0278	-0.0555	0	0

Table 23: Radiosity matrix of an adiabatic (ρ = 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© 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 ρ, noted *re* in Matlab© instructions, is equal to zero, it reduces to the identity matrix (*Table 24*). For ρ=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	0.1204	0.7593
0.1174	0.2296	0	0	0.3952	0.0515	0.3952	0	0.3952
0.1822	0.5039	0.1277	0.1204	0	0	0.7593	0.1204	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	1	0	0
0	0	1	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1
0	0	0	0	1	0			
0	0	0	0	0	1			

Table 24: View factor and radiosity matrices of a street section, ρ = 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         0   -0.0192   -0.0466   -0.1822   -0.5039
      0    1.0000   -0.1204   -0.1277   -0.5039   -0.1822
-0.0515   -0.3952    1.0000         0   -0.2296   -0.1174
-0.1174   -0.2296         0    1.0000   -0.3952   -0.0515
-0.1822   -0.5039   -0.1277   -0.1204    1.0000         0
-0.5039   -0.1822   -0.0466   -0.0192         0    1.0000
disp(det(M));   0.2570
```

```
 1.0000   -0.1204   -0.7593
-0.3952    1.0000   -0.3952
-0.7593   -0.1204    1.0000
disp(det(M));   0.2561
```

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

The sky view factor uni-column matrix *Fsky* 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 *Figure 48*

```
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 Fsky of a street section

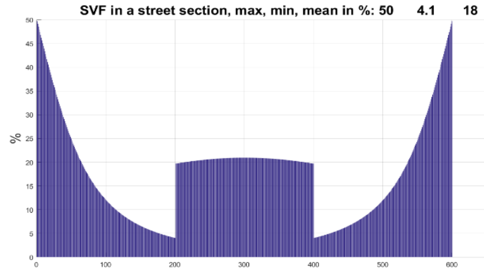


Figure 48: Sky View Factors in the street section – 200 radiative segments per side

The distribution of the sky view factor along the street walls (*Figure 48*) is computed with the Matlab© 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 %.

6. 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 50*). The display output of Fiammetta refers to the Cad definition of the problem and to the finite element mesh generated

```
Fiammetta
Rectan. 2 squares Gi: 7, Di : 7
L   4, Meth., Ne, ca: 3  0  0
L   5, Co nnr nvn   : 3  0  2
L   6, rc, ra, cs   : 0  0  0
L   7, CAD interf.  : 7
L   8, Thickness    : 0.1 m
L   9, Conduction k : 1 W/(mK)
L  10, DT isotherms : 1 K
L  13, Convection h : 18 W/(m2K)
L  15, np, nvertices: 2  6
L  17, num. nod side: 9
L  19, num elem side: 10
L  29, Domain area  : 2 m2
L  32, Num. elements: 200
L  34, Num. of DOF  : 233
L  63, Domain perim.: 6 m
N  03, param Ne     : 0
c. 03, dK = no + nvn: 233
c. 04, N.virt c.nod.: 2
c. 05, Variable Co  : 3
c. 28, Convect. sid.: 2 4 4 6 5 3 3 1
c 125, Nu. conv. el.: 40
L  96, Anis. index  : 0
st 08, Ini. tmi, tma: 280  300 K
st 09, Numb. fixat. : 2
st 22, Fix. temp. fT: 300  280 K
st 23, ddl fix. lfi : 232  233
st 27, N. iter. nit : 720
st 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))  : 0.5 MJ/K
st 48, area*th*ro*Cp: 0.5 MJ/K
st 52, Imposed Heat : 0 W/m-2
st 57, size(K) nfi  : 233  233 2
st 82, iteration    : 180 360 540 750
st104, ddt Tm - Tin : 10 K
st105, ddt*sumsum(C): 5 MJ
st106, Min obs. temp: 280 K
st107, Max obs. temp: 299 K
L 272, Date, CPU, 25-Jul-2023, 4.3465 s
g  18, Max temp grad: 18, mean: 18 K/m
ch 03, coef. red. dt: 25 W/m2
ch 19, temp. grad.  : 1 K
ch 20, Mean conv. fl: -1 0 0 W/m2
hf 25, Max heat flow: 18, mean: 18 W/m2
```

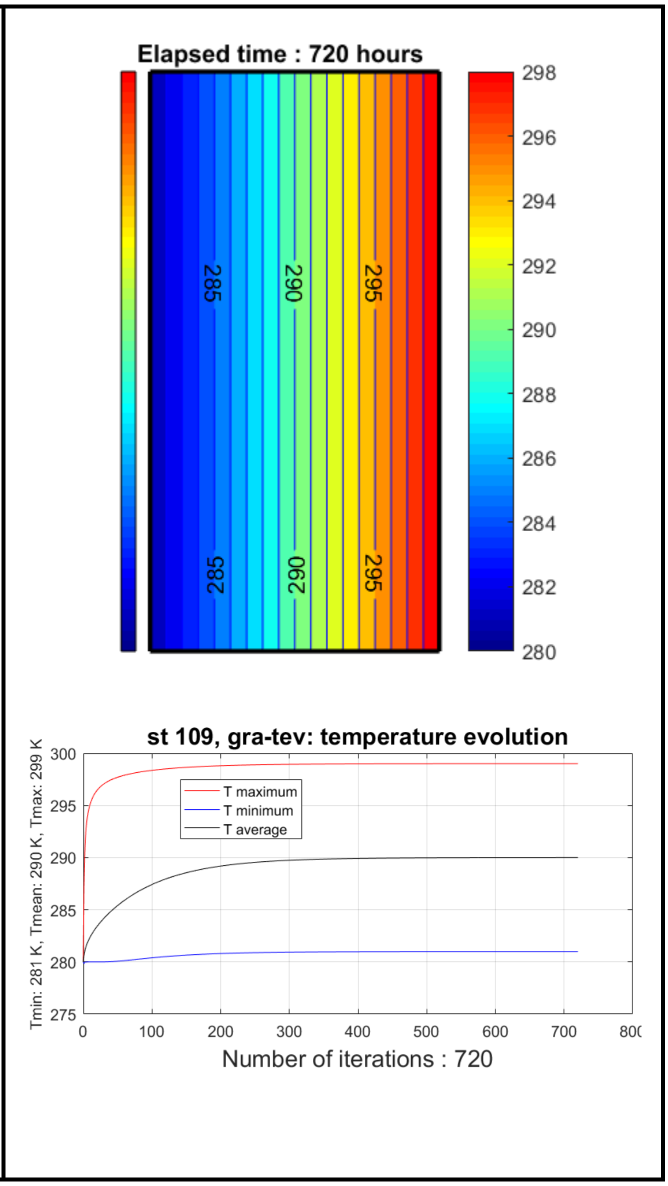


Figure 49: Transient heat transfer in rectangular domain with two convective walls.

The difference of temperature between both sides expressed in the Matlab© notations of the procedure *Fiammetta.m* (*Table 28*) is given by: max (tca(1:15)) - min (tca(1:15)) = 18 *K*, while the differences of temperature between the walls and the virtual nodes are equal to 1 *K* on both sides. Stable values of temperature components are obtained after about 300 iterations (*Figure 49*).

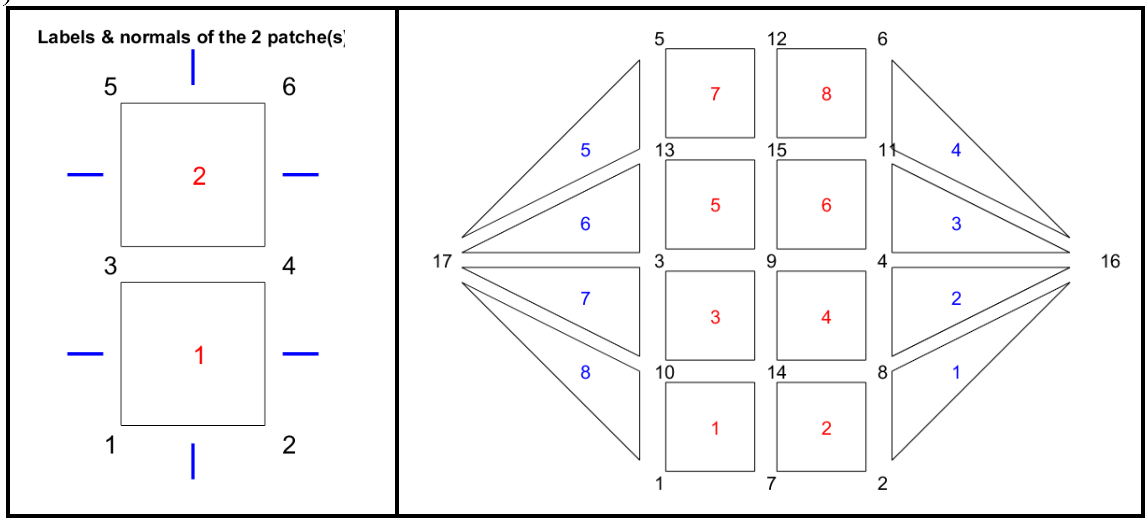


Figure 50: A simple convex domain, CAD definition and finite element mesh

We compare this problem to another one (*Figure 51*) 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* ≈ 19 *K*). In the solid, the temperature gradient is equal to -19 *K* and consequently, the heat flow = -(-19 *K* * 1 $Wm^{-1}K^{-1}$) = 19 Wm^{-1}. In the previous example (*Figure 49*), the difference was only equal to 18 *K*. As a conclusion, the heat flow is higher with radiative than with convective boundary.

```
Fiammetta
Rectan. 2 squares Gi: 24, Di : 16
L    4, Me,       Ne    : 4  0  0
L    5, Co nnr nvn      : 4  1  1
L    6, rc, ra, cs      : 1  0  4
L    7, CAD interf.     : 7
L    8, Thickness       : 0.1 m
L    9, Conduction k    : 1 W/(mK)
L   10, DT isotherms    : 1 K
L   12, Refl. coeff.    : 0
L   13, Convection h    : 18 W/(m2K)
L   15, np, nvertices:  2  6
L   17, num. nod side:  1
L   19, num elem side:  2
L   29, Domain area     : 2 m2
L   32, Num. elements:  8
L   34, Num. of DOF     : 17
L   63, Domain perim.:  6 m
L   66, St. Boltzmann:  5.6704e-08 W/(m2K4)
LD 76, Fixed nodes     : 16  17
LD 77, Fix. temper.    : 280  300 K
N  03, param Ne        : 0
c. 03, dK = no + nvn:  17
c. 04, N.virt c.nod.:  1
c. 05, Variable Co     : 4
c. 39, Vir. node vc    : 2  1
c. 41, Convect. sid.:  2  4  4  6
c 176, Nu. conv. el.:  4
c 177, conv. coeff. :  18 18 18 18 W/(m2K)
L  90, numb ra.nodes:  5
L  91, Rad. vertices:  5  3  1
L  96, Anis. index     : 0
L 126, Sky temper.     : 280 K
L 131, Rad. edg. len:  1  1 m
L 133, Radiativ area:  0.2 m2
L 258, Num. fix. DOF:  16  17
L 252, T. fix. nod. :  280  300 K
sq 04, Initial temp.:  280 K
sq 05, Fixed DOF       : 16  17
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))     : 0.5 MJ/K
sq 29, area*th*ro*Cp:  0.5 MJ/K
sq 32, Impos. Heat     : 0 W/m-2
sq 38, mcr nv          : 5  1
sq 44, N. rad. edges:  2
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 283, Diss in solid:  35.9 WK
L 284, Total dissip.:  37.9 WK
L 285, Date, CPU, 25-Jul-2023, 1.6456 s
g  18, Max temp grad:  19, mean: 19 K/m
ch 03, coef. 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
```

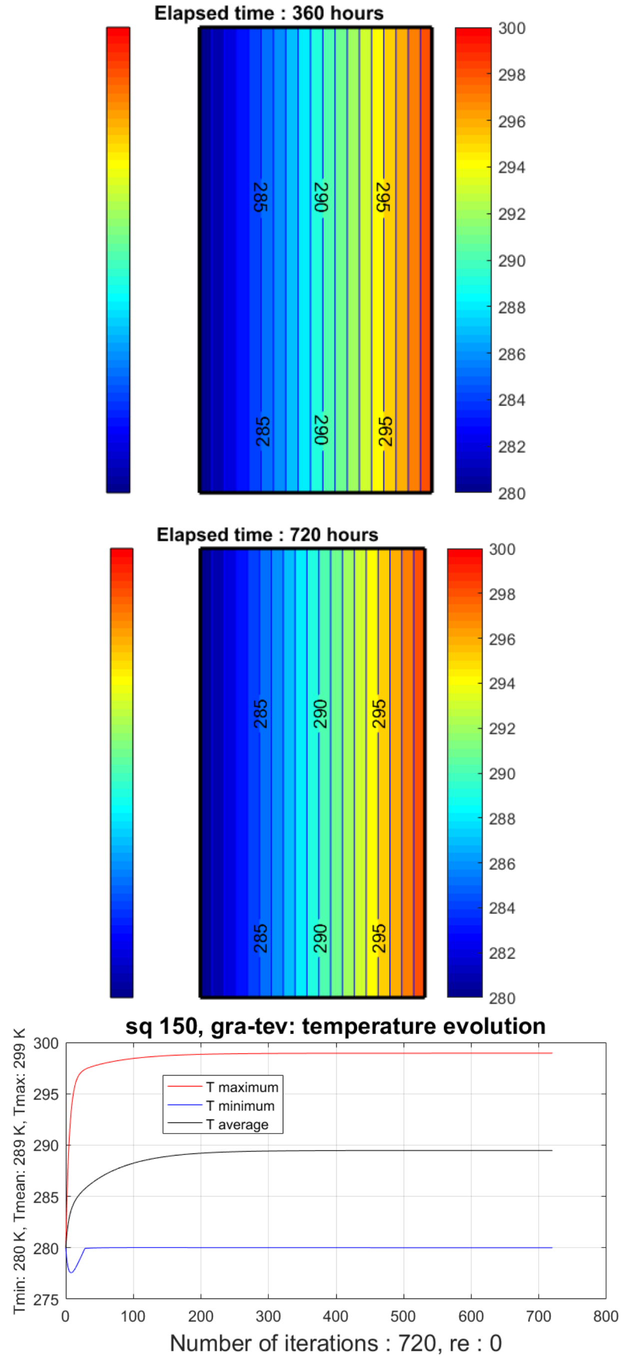


Figure 51: Rectangle with one radiative black body wall (left) and one convective wall (right)

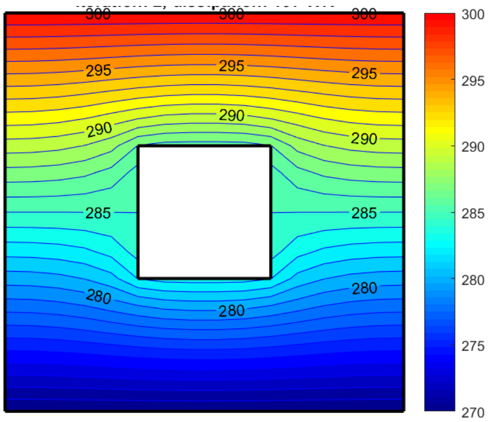
6.2 Square cavity

a) Virtual radiative node

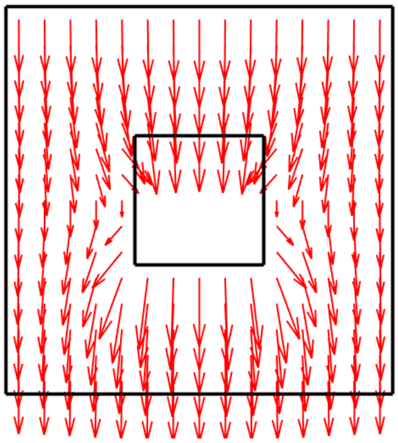
In the next test, radiative heat exchanges are analyzed in a cavity, using the concept of *virtual radiative node*. The domain is submitted to a vertical uniform temperature gradient. Without the cavity, the dissipation should be equal to 100 *WK* and the mean heat flow to 10 Wm^{-1}. The heat flow is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

Black body

```
Fiammetta
Standard cavity,  Gi: 40, Di : 20
L    4, Meth., Ne, ca: 2  0  1
L    5, Co nnr nvn   : 0  1  0
L    6, rc, ra, cs   : 4  1  7
L    7, CAD interf.  : 24
L    8, Thickness    : 0.1 m
L    9, Conduction k : 1 W/(mK)
L   10, DT isotherms : 1 K
L   12, Refl. coeff. : 0
L   15, np, nvertices: 8  16
L   17, num. nod side: 4
L   19, num elem side: 5
L   29, Domain area  : 8 m2
L   33, Num. elements: 200
L   35, Num. of nodes: 240
L   36, Num. of DOF  : 241
L   63, Domain perim.: 16 m
L   66, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD  52, N. fix. nodes: 32
N   03, param Ne     : 0
L   89, numb ra.nodes: 20
L   91, Rad. vertices: 7   6  10  11
L   96, Anis. index  : 0
L  126, Sky temper.  : 280 K
L  132, Rad. edg. len: 1  1  1  1 m
L  133, Radiativ area: 0.4 m2
L  146, Det. R. mat M: 1
sd   5, N. fixed nod.: 32
sd   7, N. rad elem. : 20
sd  24, Iteration N. : 1 / 2
sd  39, Dissipation  : 100.9 WK
sd  55, max - min tca: 30    K
sd  24, Iteration N. : 2 / 2
sd  39, Dissipation  : 106.8 WK
sd  55, max - min tca: 30    K
L  263, T. vir. nod. : 285   K
L  265, Date, CPU, 28-Aug-2023, 1.7 s
g   18, Max temp grad: 15, mean: 11 K/m
hf  25, Max heat flow: 15, mean: 11 W/m2
```



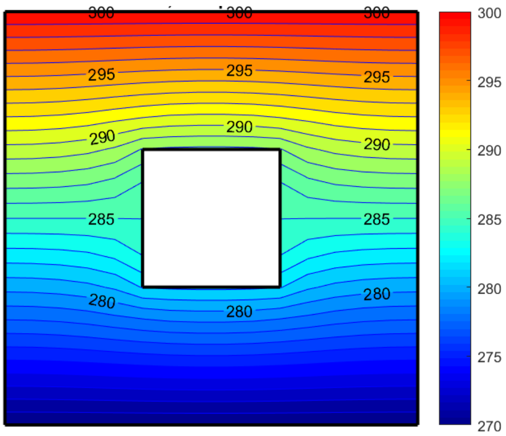
Dissipation 106.8 *WK*, $\rho = 0.$



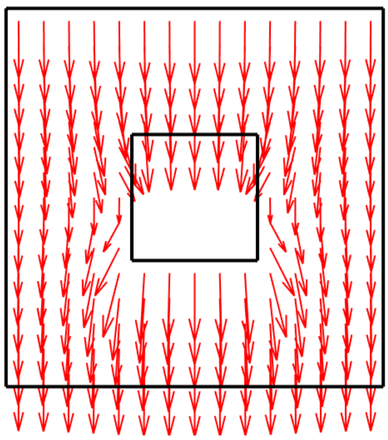
Max heat flow: 15, mean: 11 Wm^{-2}

Figure 52: Isotherms and heat flows through a cavity, 1 virtual radiative node, black body

Gray body

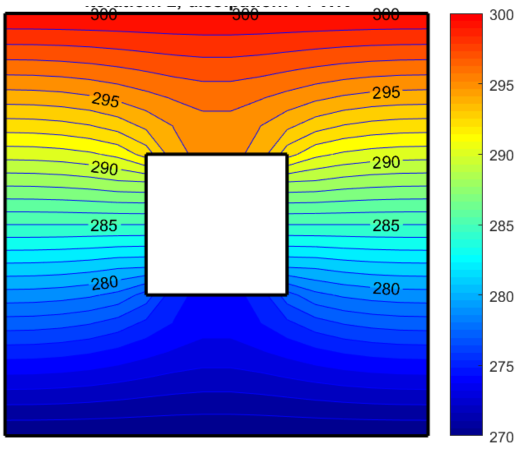


Dissipation 101.3 *WK*, $\rho = 0.5$

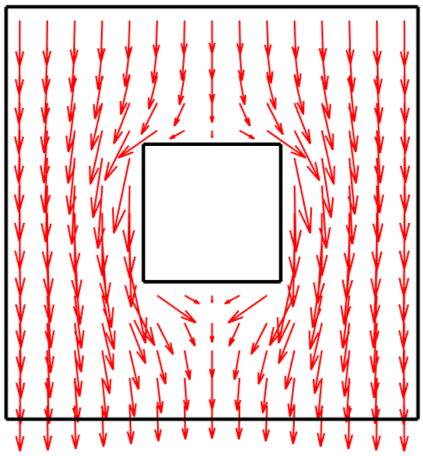


Max heat flow: 13, mean: 11 Wm^{-2}

Mirror



Dissipation 71.04 *WK*, $\rho = 0.9999$



Max heat flow: 15, mean: 9.1 Wm^{-2}

Figure 53: Isotherms and heat flows, gray body & mirror, 30 days, 1 virtual radiative node

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

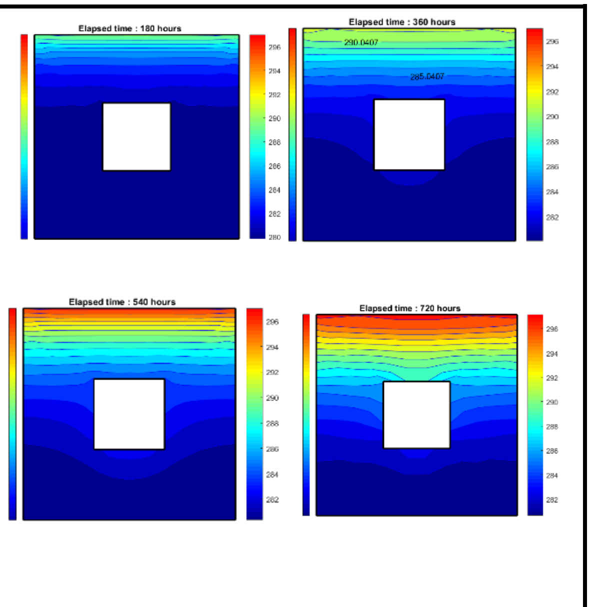
Emissivity $\varepsilon = (1 - \rho)$	ρ	Dissipation	Heat flow, max, mean
1.	0.	106.8 *WK*	15 Wm^{-2} 11 Wm^{-2}
.8	.2	105.2 *WK*	15 Wm^{-2} 11 Wm^{-2}
.5	.5	101.3 *WK*	13 Wm^{-2} 11 Wm^{-2}
.2	.8	91.77 *WK*	11 Wm^{-2} 10 Wm^{-2}
.1	.9	84.67 *WK*	12 Wm^{-2} 9.7 Wm^{-2}
0.0001	0.9999	71.04 *WK*	15 Wm^{-2} 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 Wm^{-2} is applied on the upper horizontal border of the domain (*Figure 54*, display output: `sc 36, Imposed Heat : 50 Wm-2`). The cavity is acting as a black body and some results are compared to a cavity acting as a mirror (*Figure 56*).

```
Fiammetta
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
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  35, Num. of nodes: 120
L  36, Num. of DOF  : 120
L  63, Domain perim.: 16 m
L  66, St. Boltzmann: 5.7e-08 W/(m2K4)
LD  7, Numb. fix nod: 0
N  03, param Ne     : 1
N  13, Loaded  area : 0.3 m2
N  59, Loaded nodes : 3  37 38 39 40 4
L  89, numb ra.nodes: 20
L  90, Rad. vert. lv: 6  5  8  7
```



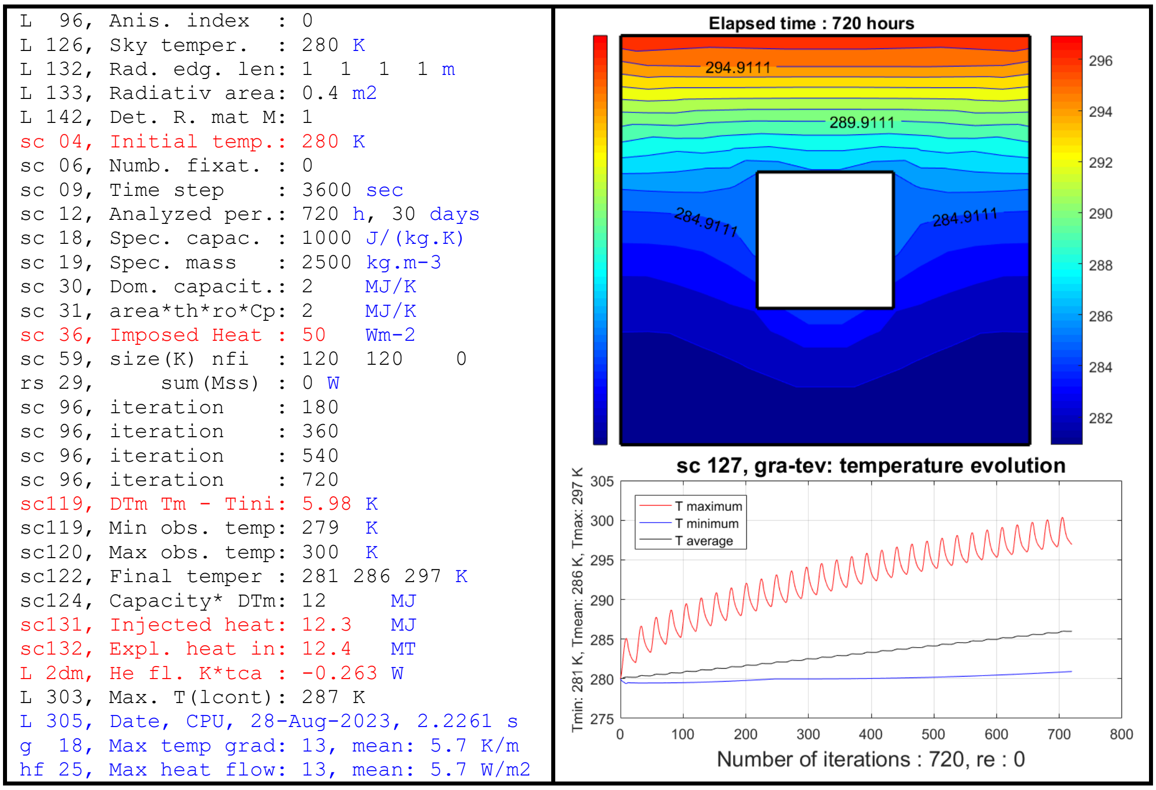


Figure 54: Isotherms of radiative exchanges, 256 elements, 1-month, black body, ρ = 0

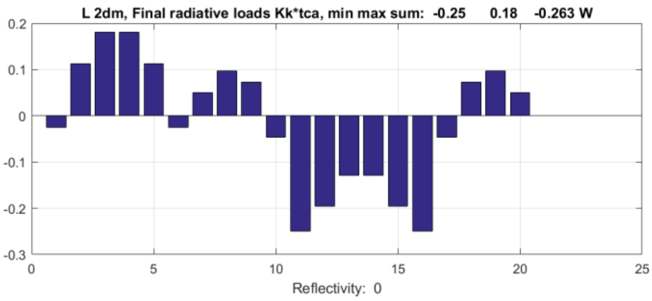
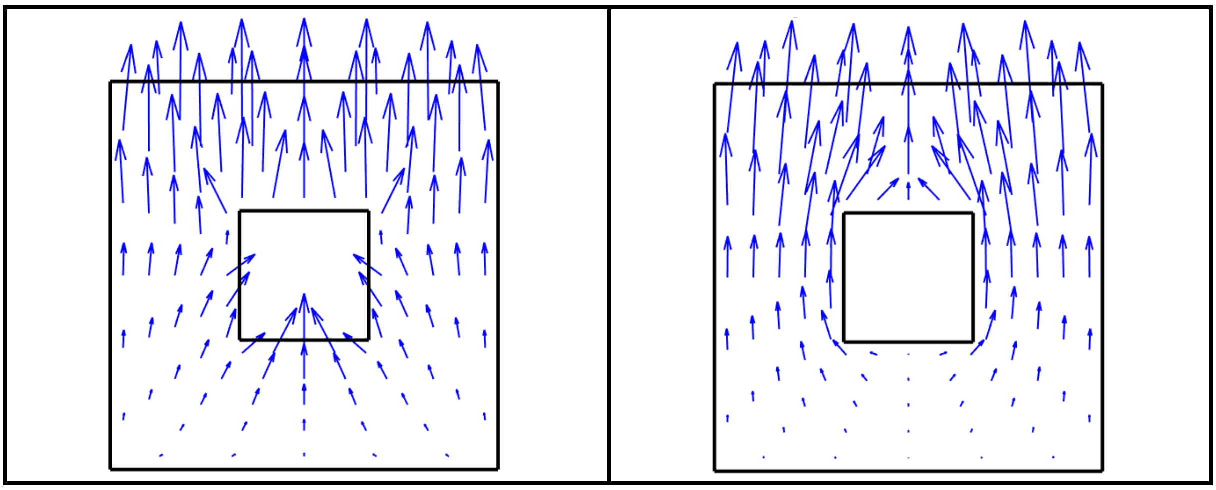


Figure 55 : Flux de chaleur nodaux le long de la cavité (gra_2dm.m, Table 55)



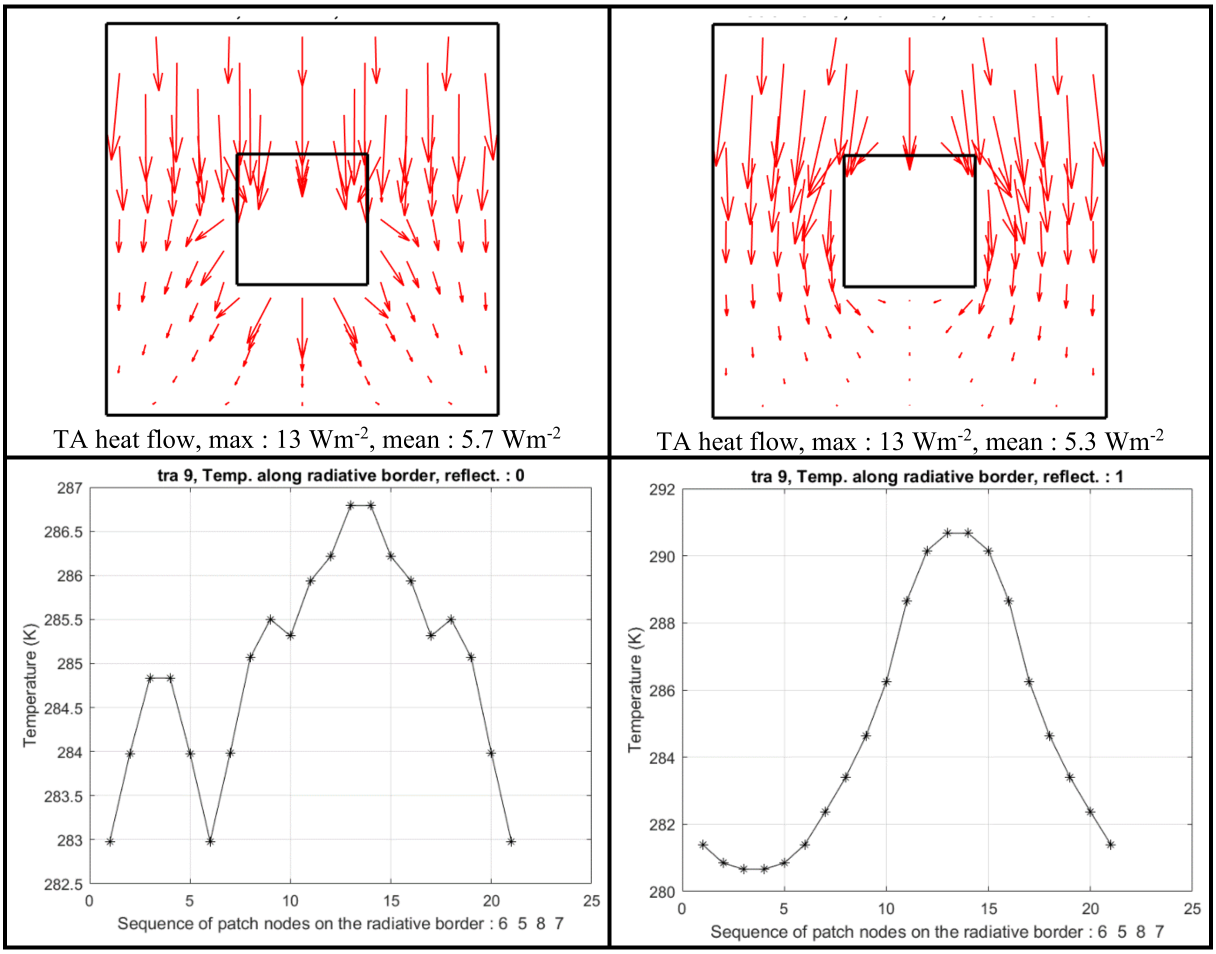


Figure 56: Grad., H.F. & T. after 30 days, left: black body, ρ = 0; right: mirror, ρ = 1

With 1088 *DOF*, the solution of *Figure 56* becomes that of *Figure 57*.

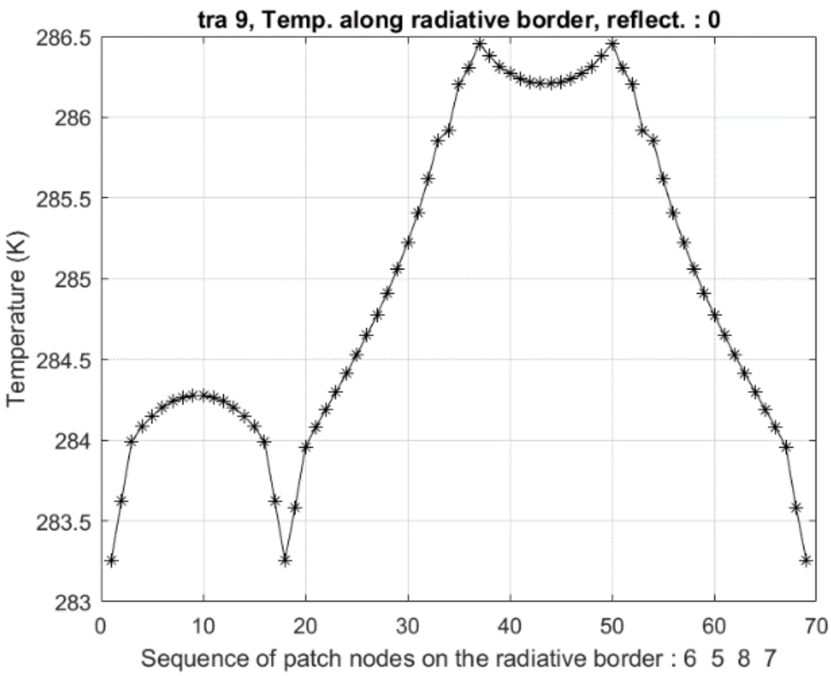


Figure 57: Cavity, black body, ρ = 0 (Figure 54), 1224 DOF, nni = 16, (gra_tra.m, Table 54)

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 *SBt* of the Stefan-Boltzmann by the thickness *th* of the radiative element,
- matrix $Mpr = (I - F)\, M^{-1}$ 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     : 180 360 540 720
sc 96, Line vect. gt: 280.79 1 297.0766 K
sc119, DTm Tm - Tini: 5.97 K
sc120, Min obs. temp: 279  K
sc121, Max obs. temp: 301  K
sc135, Final temper.: 281 286 297 K
sc124, Capacity* DTm: 11.9 MJ
sc131, Injected heat: 12.3 MJ
sc132, Ex.heat input: 12.4 MJ
L 2dm, He fl. K*tca : -0.263 W
L 303, Max. T(lcont): 288  K
L 305, Date, CPU, 28-Aug-2023, 2.7221 s
g  18, Max temp grad: 13, mean: 5.5 K/m
hf 25, Max heat flow: 13, mean: 5.5 W/m2
```

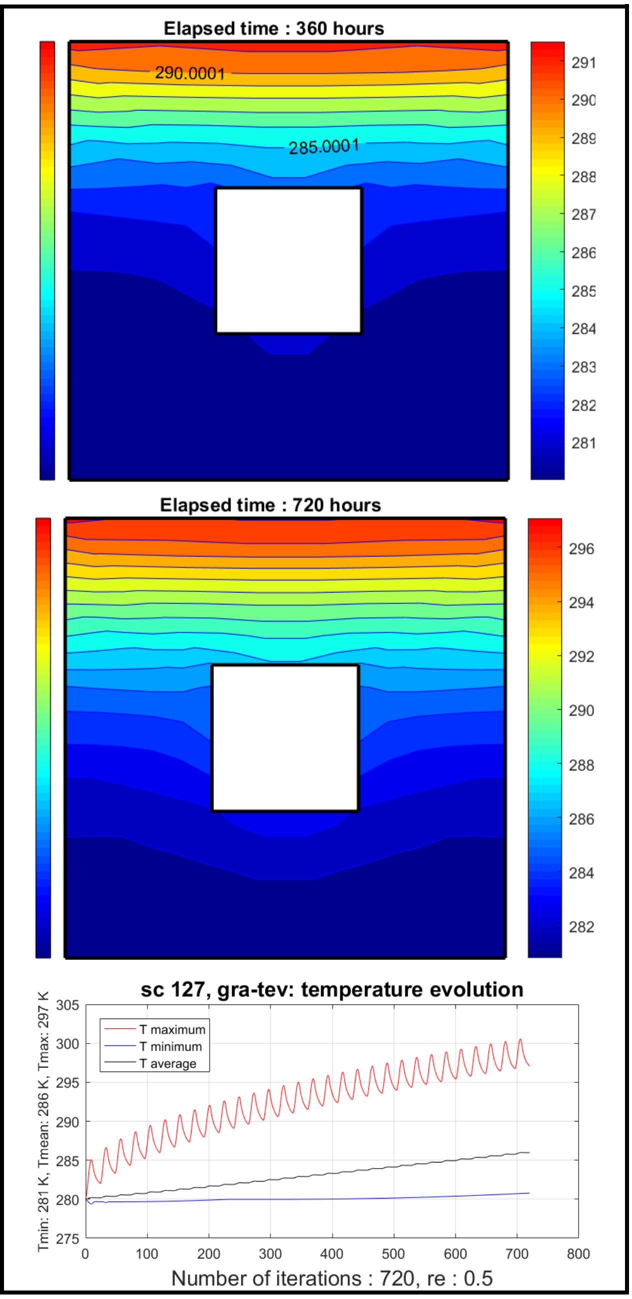


Figure 58: Isotherms, radiation, 4 patches, 800 elements, 1 month, ρ = 0.5

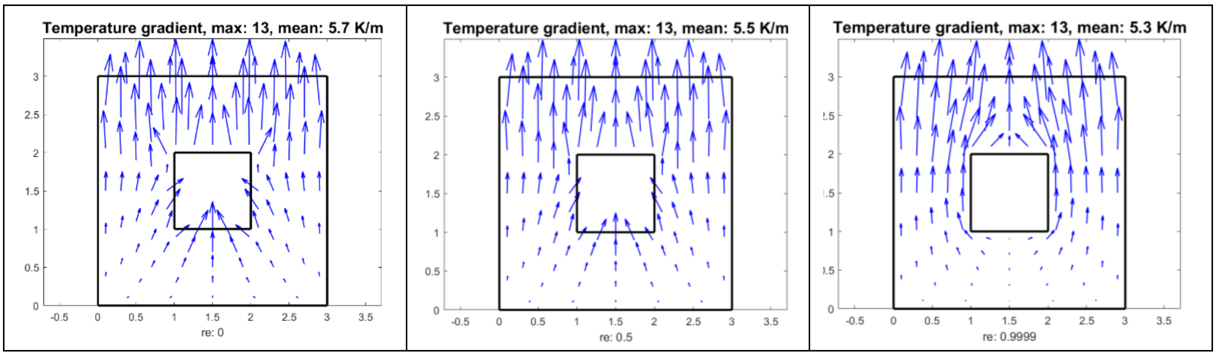


Figure 59: Temperatures gradients, black, gray body & mirror, 30 days

The test of *Figure 60* is the same, but with more patches. The elements are rectangular. The goal is to show the heat fluxes on a regular mesh like in *Figure 62*.

```
Fiammetta
Standard cavity,  Gi: 42, Di : 0
L   4, Meth., Ne, ca: 5  2  1
L   5, Co nnr nvn   : 0  0  0
L   6, rc, ra, cs   : 1  1  7
L   7, CAD interf.  : 24
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  15, np, nvertices: 8  16
L  16, num. nod side: 4
L  17, num elem side: 5
L  28, Domain area  : 8 m2
L  27, Num. elements: 200
L  33, Num. of DOF  : 240
L  68, Domain perim.: 16 m
L  70, St. Boltzmann: 5.6e-08 W/(m2K4)
LD  2, num. nod side: 4
LD  3, Dirich. label: 0
LD  4, N. virt. nod.: 0
LD  7, Numb. fix nod: 0
N  03, param Ne     : 2
N  25, Loaded  area : 0.3 m2
N  62, N. load. nod.: 18
L  90, Rad. vertices: 7   6  10  11
L  96, Anis. index  : 0
sc 04, Initial temp.: 280 K
L 132, Rad. edg. len: 1  1  1  1 m
L 133, Radiativ area: 0.4 m2
L 190, Det. R. mat M: 0.73587
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.: 2    MJ/K
sc 31, area*th*ro*Cp: 2    MJ/K
sc 36, Imposed Heat : 50   Wm-2
sc 59, size(K) nfi  : 240  240    0
sc 96, iteration    : 180 360 540 720
sc 97, Line vect. gt: 280.4 1 297.84 K
sc119, DTm Tm - Tini: 6.27 K
sc133, Min obs. temp: 280  K
sc134, Max obs. temp: 301  K
sc135, Final temper.: 280 286  298 K
sc124, Capacity* DTm: 12.5   MJ
sc131, Injected heat: 13.5   MJ
sc132, Expl. heat in: 12.4   MJ
L 2dm, He fl. K*tca : -0.256 W
L 306, Max. T(lcont): 287    K
L 308, Date, CPU, 29-Aug-2023, 5.7957 s
g  18, Max temp grad: 13, mean: 5.4 K/m
hf 24, Max heat flow: 13, mean: 5.4 W/m2
```

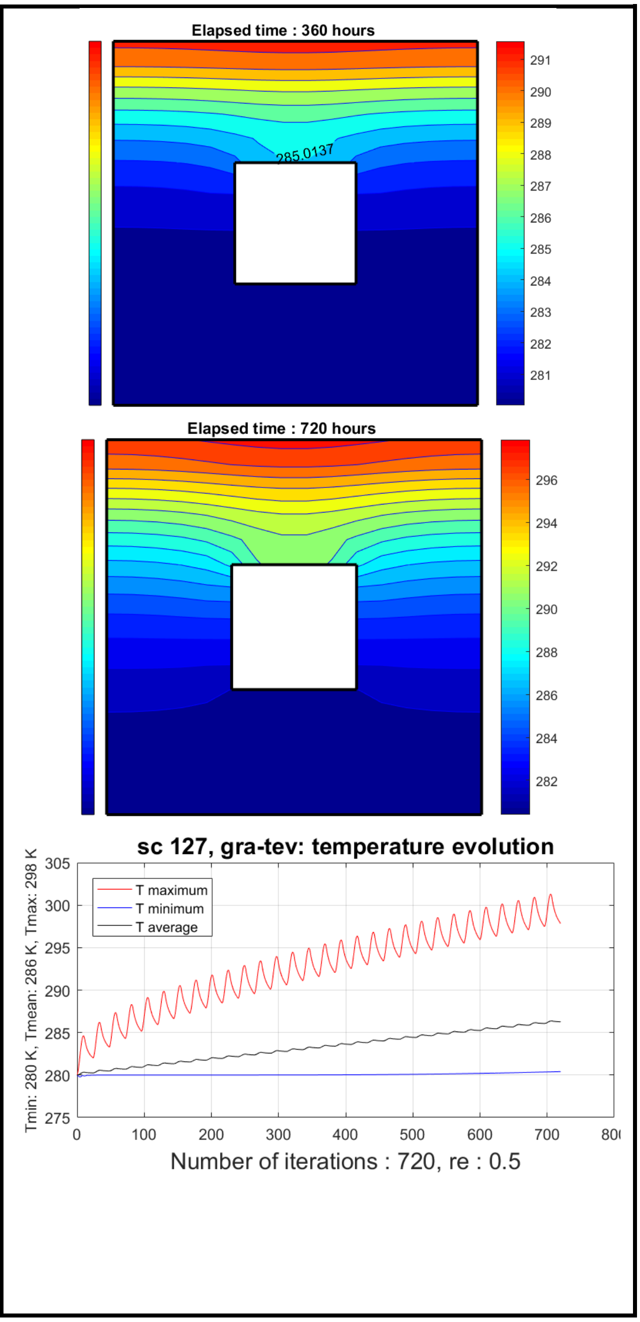


Figure 60: Isotherms, radiation, 8 patches, 200 elements, 1 month, ρ = 0.5

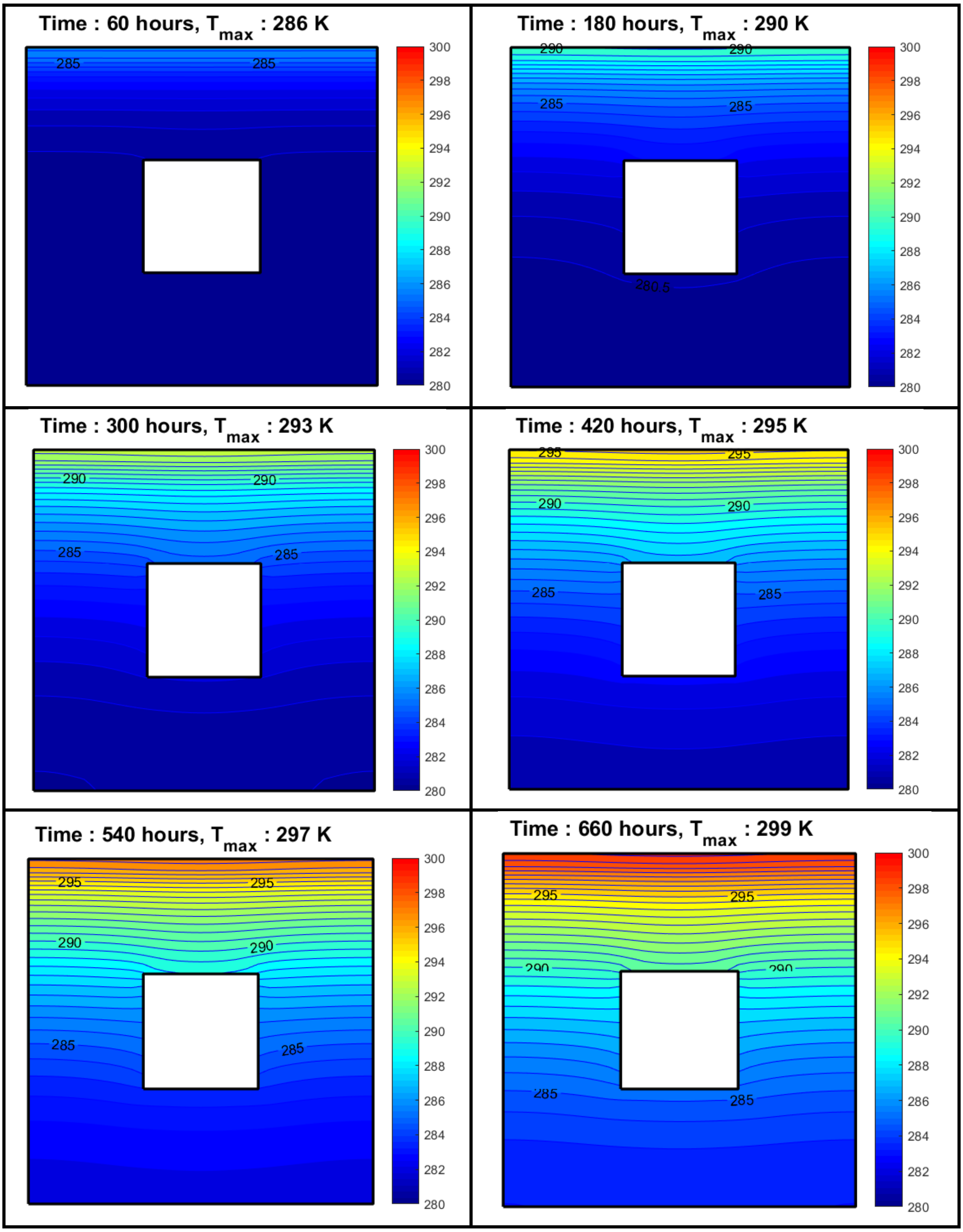


Figure 61: Isotherms, gray body, evolution after 60, 120, …, 660 hours, ρ = 0.5

d) Radiosity method, comparison of gray cavities

In *Figure 62*, 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 will help the user to better understand these physical phenomena in their four dimensions (space and time). In *Figure 63* the temperature distribution is shown along the cavity border. The profiles are rather different.

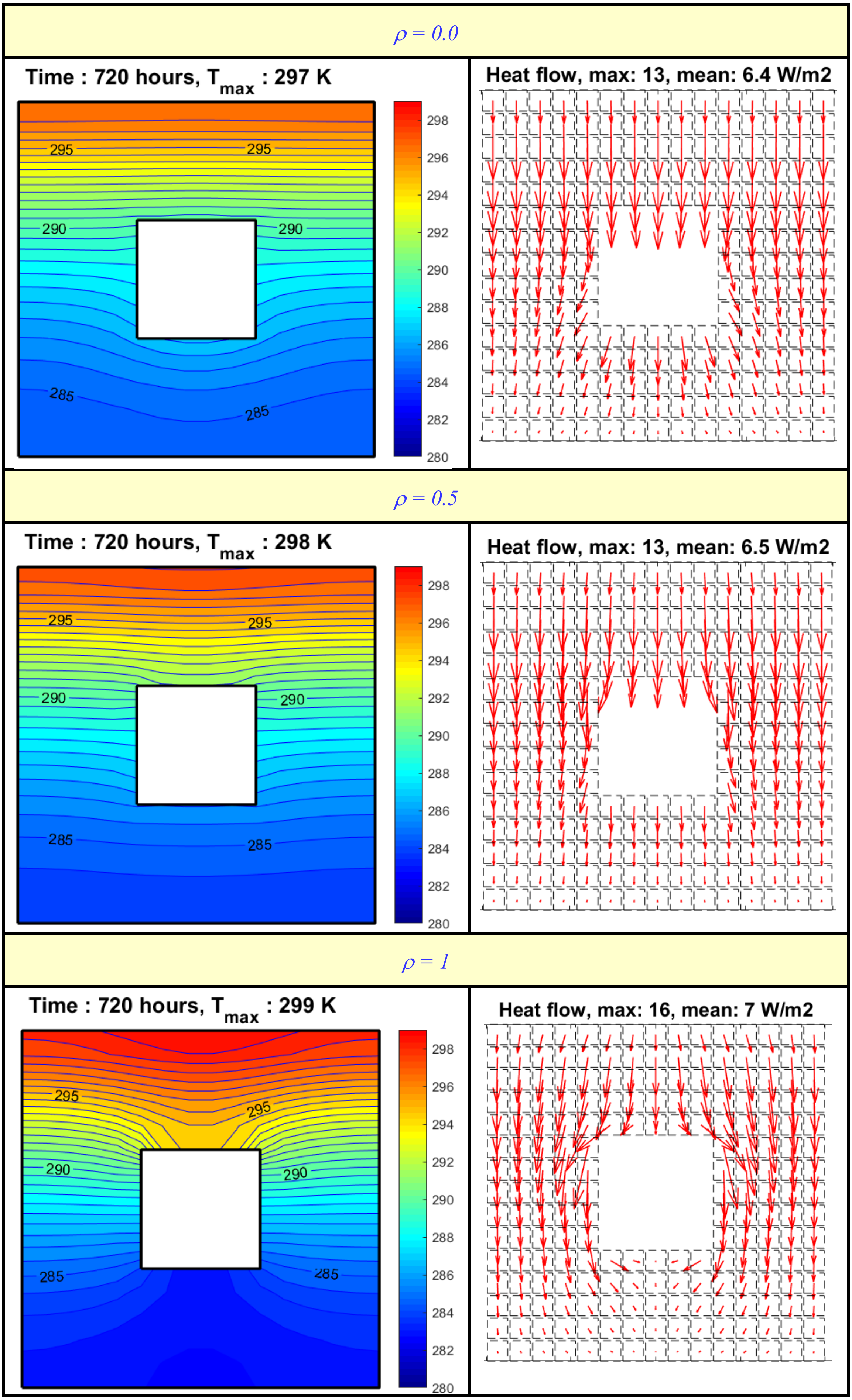


Figure 62: Isotherms and heat flows, black, gray body & mirror, 30 days

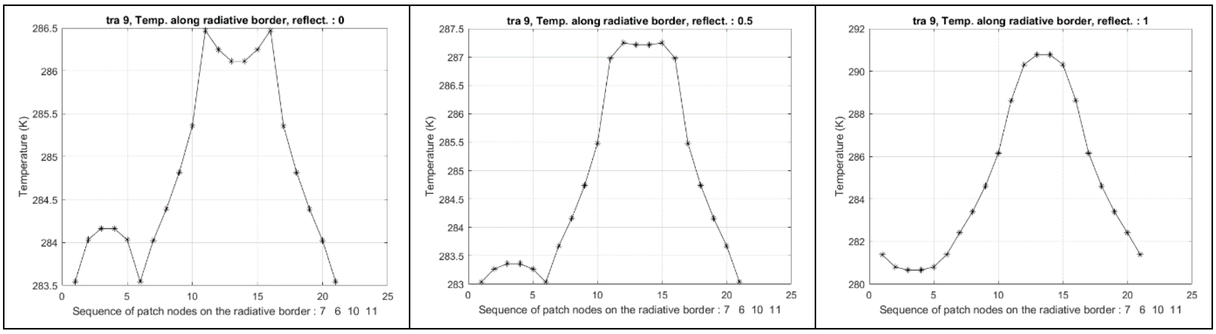


Figure 63: Temperature on the cavity wall, black, gray body & mirror, 30 days

6.3 Rectangular cavity

```
Fiammetta
Rectang. cavity,  Gi: 13, Di : 8
L   4, Meth., Ne, ca: 3  0  1
L   5, Co nnr nvn   : 7  0  2
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  12, Convection h : 9 W/(m2K)
L  14, np, nvertices: 4  8
L  16, num. nod side: 7
L  17, num elem side: 8
L  28, Domain area  : 14 m2
L  27, Num. elements: 256
L  33, Num. of DOF  : 290
L  63, Domain perim.: 28 m
LD 61, N. fix. nodes: 11
LD 62, Fix. DOF, lfi: 1 9 ... 2  289  290
LD 63, imposed temp.: 270 ... 270 300 300 K
N  03, param Ne     : 0
c. 03, dK = no + nvn: 290
c. 04, N.virt c.nod.: 2
c. 05, Variable Co  : 7
c. 40, Convect. sid.: 2  3  4  1
c 125, Nu. conv. el.: 16
L  94, Anis. index  : 0
st 08, Ini. tmi, tma: 270  300 K
st 09, Numb. fixat. : 11
st 27, N. iter. nit : 720
st 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))  : 3.5 MJ/K
st 48, area*th*ro*Cp: 3.5 MJ/K
st 52, Imposed Heat : 0 W/m-2
st 57, size(K) nfi  : 290  290   11
st 82, iteration    : 180
st 82, iteration    : 360
st 82, iteration    : 540
st 82, iteration    : 720
st104, ddt Tm - Tin : 21.2   K
st105, ddt*sumsum(C): 74.2   MJ
st106, Min obs. temp: 269    K
st107, Max obs. temp: 299.4 K
L 272, Date, CPU, 23-Jul-2023, 3.799 s
ch 03, coef. red. dt: 25      W/m2
ch 19, temp. grad.  : -15.9697 K
ch 20, Mean conv. fl: 0  0 0   W/m2
hf 24, Max heat flow: 80, mean: 15 W/m2
```

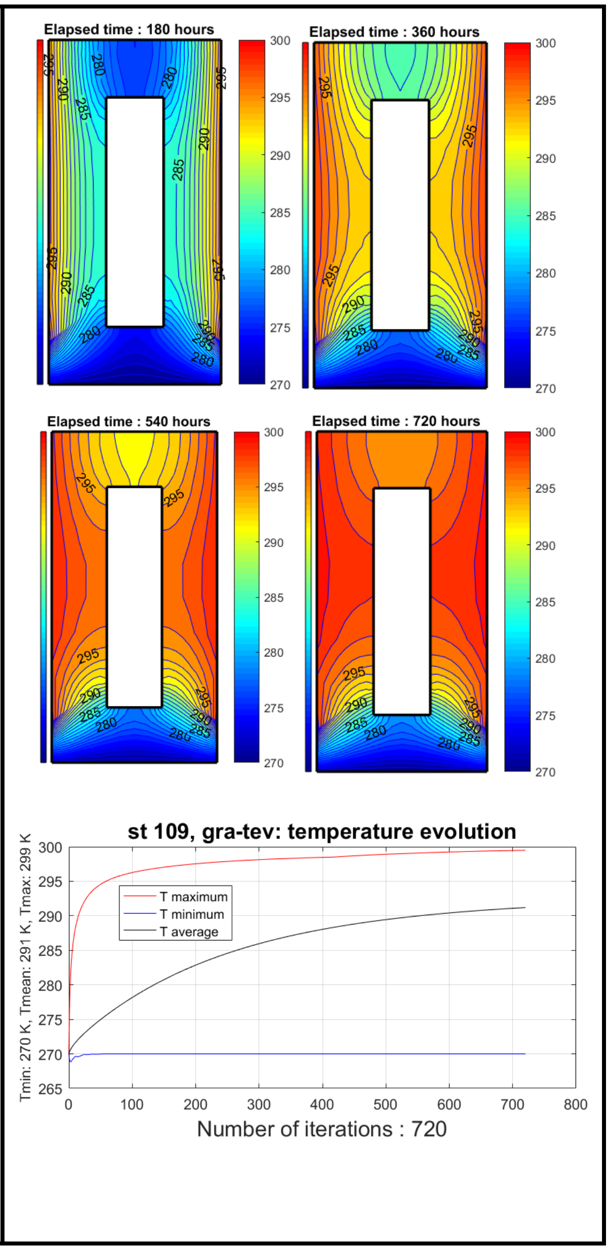


Figure 64: Adiabatic rectangular cavity, method 3, 2 virt. conv. nodes

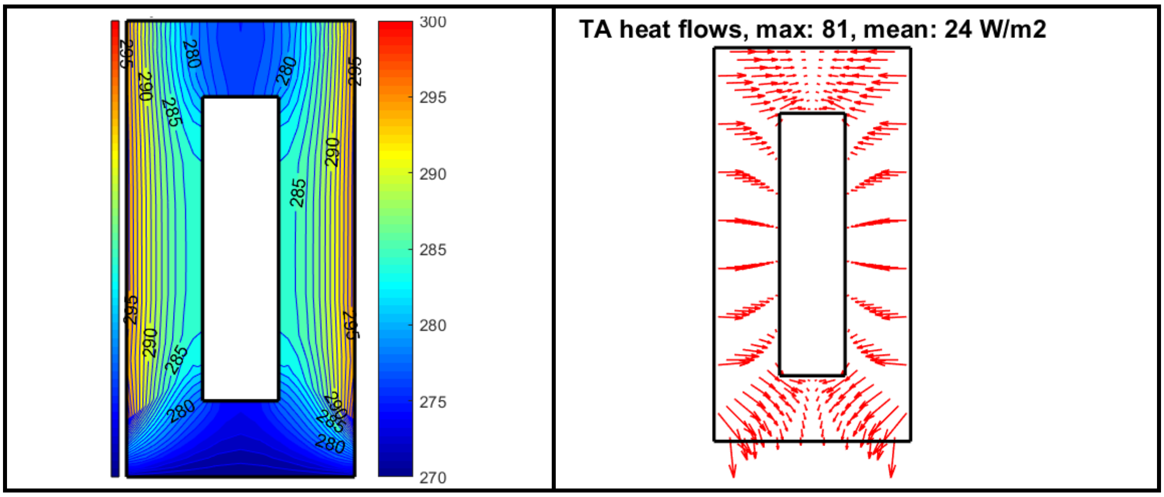
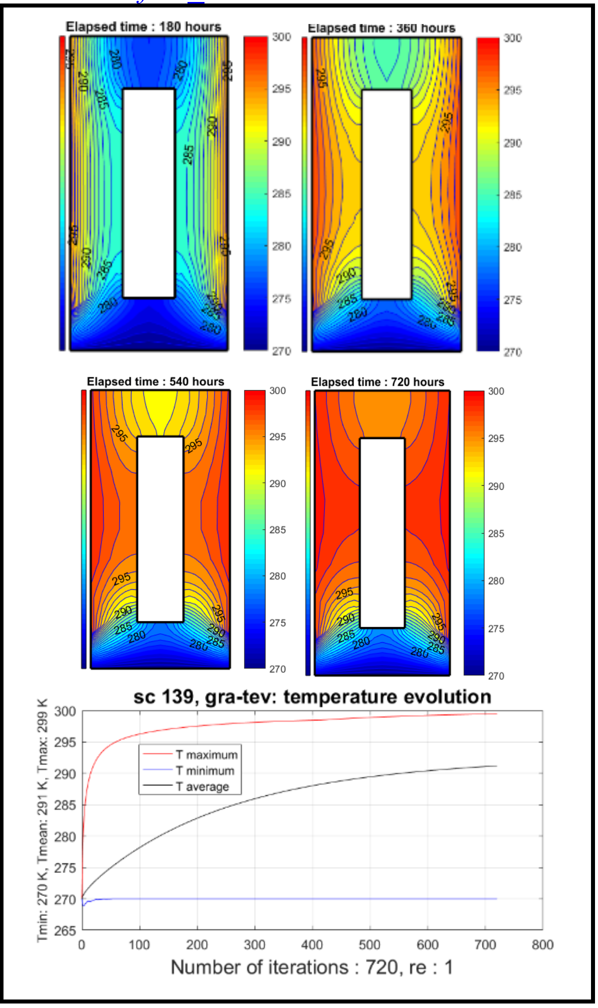


Figure 65: Adiabatic rectangular cavity, isotherms & heat flows after 180 hours

We observe (*Figure 65*) 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 64*, 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 66* are the same as those of *Figure 64*. Note the settings of *lines* 94 & *101* in *fem_smc.m*

```
Fiammetta
Rectang. cavity,  Gi: 12, Di : 8
L   4, Meth., Ne, ca: 5  0  1
L   5, Co nnr nvn    : 7  0  2
L   6, rc, ra, cs    : 1  0  1
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  11, Refl. coeff.  : 1
L  13, Convection h  : 9 W/(m2K)
L  15, np, nvertices: 4  8
L  17, num. nod side: 7
L  29, Domain area   : 14 m2
L  33, Num. elements: 256
L  35, Num. of nodes: 288
L  36, Num. of DOF   : 290
L  63, Domain perim.: 28 m
L  66, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD 63, N. fix. nodes: 11
LD 64, Fix. DOF, lfi: 1 9 ... 15 2 289 290
LD 65, Imposed temp.: 270 ... 270 300 300 K
N  03, param Ne      : 0
c. 03, dK = no + nvn: 290
c. 04, N.virt c.nod.: 2
c. 05, Variable Co   : 7
c. 52, Convect. sid.: 2  3  4  1
c 125, Nu. conv. el.: 16
L  89, numb ra.nodes: 32
L  90, Rad. vertices: 6  5  8  7
L  94, Anis. index   : 0
L 125, Sky temper.   : 280 K
L 130, Rad. edg. len: 1  4  1  4 m
L 131, Radiativ area: 1 m2
L 137, D. rad. mat M: 0
sc 04, Initial temp.: 270 K
sc 06, Numb. fixat. : 11
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.: 3.5  MJ/K
sc 31, area*th*ro*Cp: 3.5  MJ/K
sc 36, Imposed Heat : 0    Wm-2
sc 59, size(K) nfi  : 290  290   11
sc 95, iteration    : 180
sc 96, Line vect. gt: 270    1  300 K
sc 95, iteration    : 360 ... 540 ... 720
sc119, DTm Tm - Tini: 21.2 K
sc120, Min obs. temp: 269  K
sc121, Max obs. temp: 299  K
sc122, Final temper.: 270 291 299 K
sc124, Capacity* DTm: 74.2  MJ
sc135, Ejected heat : 81.5  MJ
L 2dm, He fl. K*tca : -0.25 W
L 303, Max. T(lcont): 298   K
L 305, Date, CPU, 23-Jul-2023, 3.9709 s
g  18, Max temp grad: 40, mean: 7.5 K/m
ch 03, coef. red. dt: 25        W/m2
ch 19, temp. grad.  : -15.5955 K
ch 20, Mean conv. fl: 1.0788e-05 0 0 W/m2
hf 25, Max heat flow: 80, mean: 15   W/m2
```

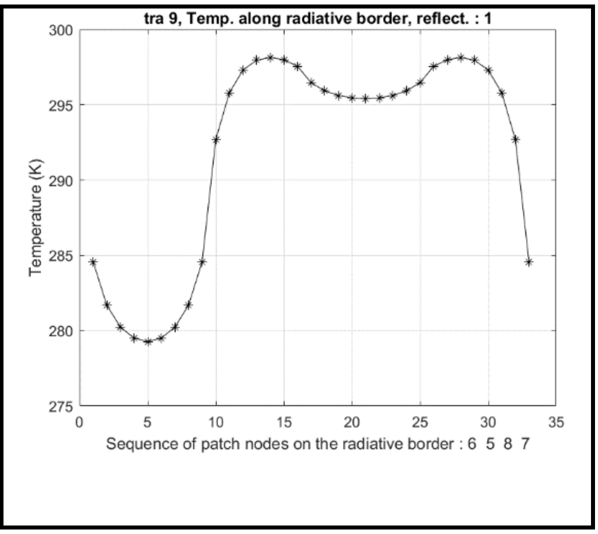


Figure 66: Cavity with adiabatic boundary $\varepsilon = 0$, $\rho = 1$

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

6.4 Street section

6.4.1 Periodic 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*).

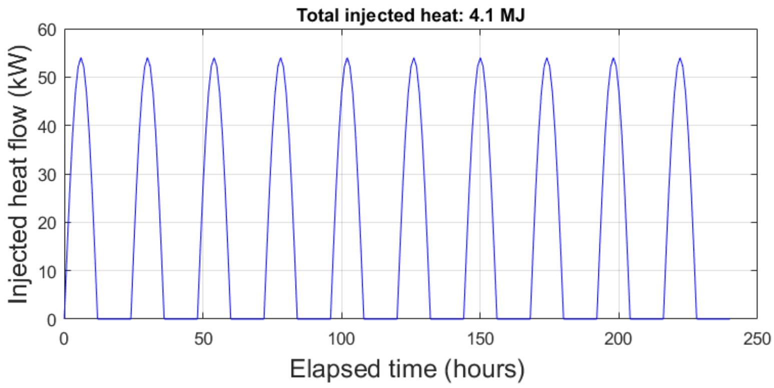


Figure 67: Thermal loading over a period of 10 days (240 hours)

The drawing of *Figure 67* 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 130* of *fem_smc.m*. The time function *f(t)* used to weight the heat flow input is given by:

$$f(t) = \begin{cases} \sin\left(\frac{2t}{p}\pi\right) & if \quad 0 \le t \le \frac{p}{2} \\ 0 & if \quad \frac{p}{2} \le t \le p \end{cases} \tag{132}$$

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:

$$\frac{2}{p}\int_0^{p/2} \sin\left(\frac{2t}{p}\pi\right)\, dx = -\frac{2}{p}\left[\cos\left(\frac{2t}{p}\pi\right)\frac{p}{2\pi}\right]_0^{p/2} = \frac{-1}{2\pi}(-1\ -1) = \frac{2}{\pi} \simeq 0.6366 \tag{133}$$

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 $i_h = 50\ Wm^{-2}$ and if the time is expressed in seconds, the heat flow is:

$$\bar{q}_n = i_h\, f(t)\ Wm^{-2} = i_h\ \sin\frac{t\ \pi}{12*3600}\ Wm^{-2} = i_h\ \sin\frac{t\ \pi}{12*3600}\ Wm^{-2} \qquad (134)$$

The area of the boundary where heat is injected is:

$$s_b = e \text{ x } L = 0.1*3 = 0.3\ m^2 \qquad (135)$$

In this expression, e is the thickness of the solid, L the length of the border and i_h is the imposed heat flow density. The imposed heat flow (W) is:

$$\bar{q}_n b_s = i_h\ e\,L\,f(t)\ W = \ i_h\ e\,L\,f(t)\ W = i_h\ e\,L \sin\frac{t\pi}{12*3600}\ W = i_h\ e\ L\ \sin\frac{t\pi}{43200}\ W \qquad (136)$$

In one day, the injected heat is equal to h_d, now expressed in joules:

$$h_d = \int_0^{43200} \bar{q}_n b_s\ dt\ J = i_h\ e\ L \int_0^{43200} \sin\frac{t\pi}{43200} dt\ J = i_h\ e\ L\left[-\frac{43200}{\pi}\cos\frac{t\pi}{43200}\right]_0^{43200} J$$
$$h_d = \frac{i_h\ e\ L\ 12*3600\ *\ 2}{\pi}\ J = 0.2475\ i_h\ MJ \qquad (137)$$

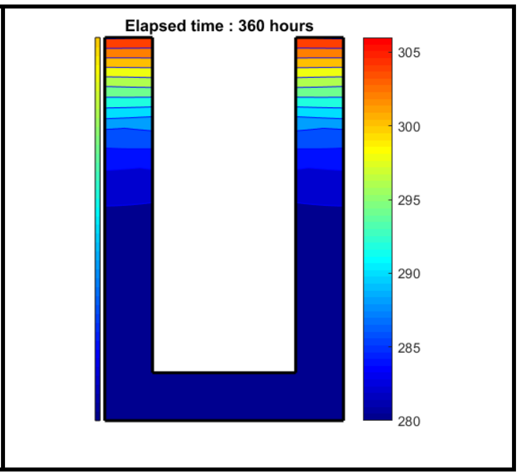
With a heat flow density $i_h = 50\ Wm^{-2}$ and a loaded area $eL = 0.3\ m^2$, the total incoming heat flow after 30 days is: $h_d * 30 = 12.3759\ MJ$.
According to the input data of *Figure 60*, the instructions: “tt = 720*3600, ih = 50, bos = .3, hdm = tt*ih/pi*bos*1e-6; the instruction: “disp (['Lt142, Ex.heat input: ', num2str(hdm,3), ' MJ'])” gives the result: “Lt142,   Heat input: 12.4 MJ”

A periodic heat input is introduced, the imposed heat is increasing during 6 hours, decreasing to 0 during 6 hours and not effective during 12 hours (*Figure 67*). It is similar to the situation on Equator at equinoctial time.

6.4.2 Adiabatic or pure reflective walls, ρ = 1

In an adiabatic rectangular street section, with a finite element model of 225 *DOF*, the temperature is always greater than the initial one (> 280 *K*). At the end of the integration process of 720 hours, *Tmin* = 280 *K*, *Tmax* = 318 *K* and *Tmean* = 288 *K*. This example is shown in *Figure 69*, etc. The method number 3 (Matlab© 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 42*).

```
Fiammetta
Street section,   Gi: 15, Di : 0
L    4, Meth., Ne, ca: 3   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   13, np, nvertices: 3   8
L   15, num. nod side: 7
L   17, num elem side: 8
L   27, Domain area   : 19 m2
L   27, Num. elements: 192
L   32, Num. of DOF   : 225
L   63, Domain perim.: 40  m
LD   7, Numb. fix nod: 0
N   03, param Ne      : 111
N   51, Total load    : 1  W
```



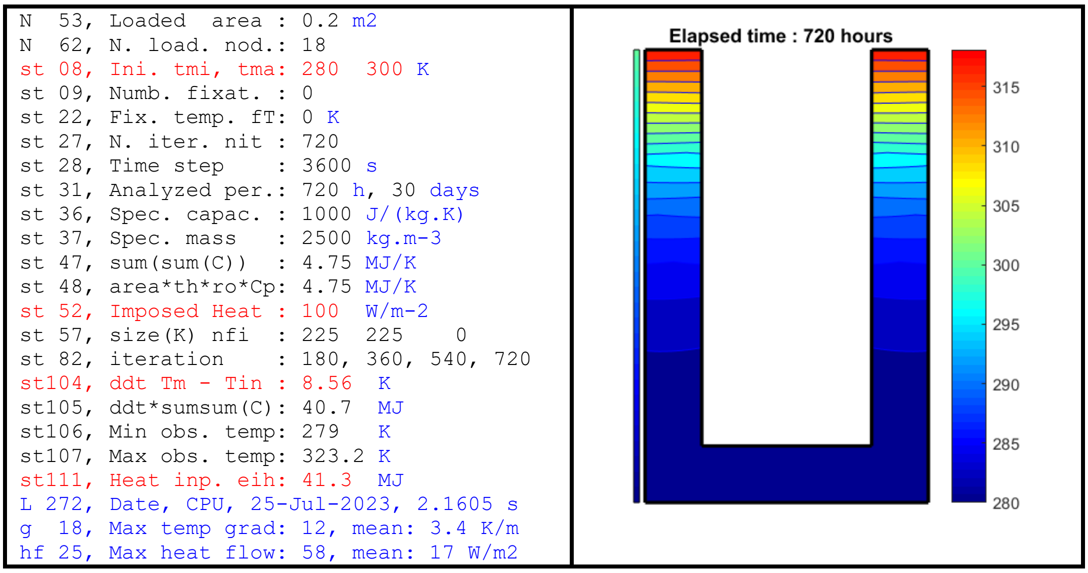


Figure 68: Adiabatic Street Section

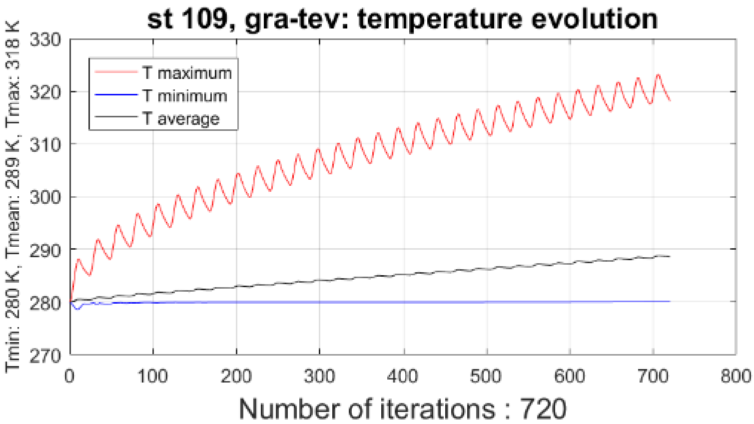


Figure 69: Street section – adiabatic walls - 225 DOF

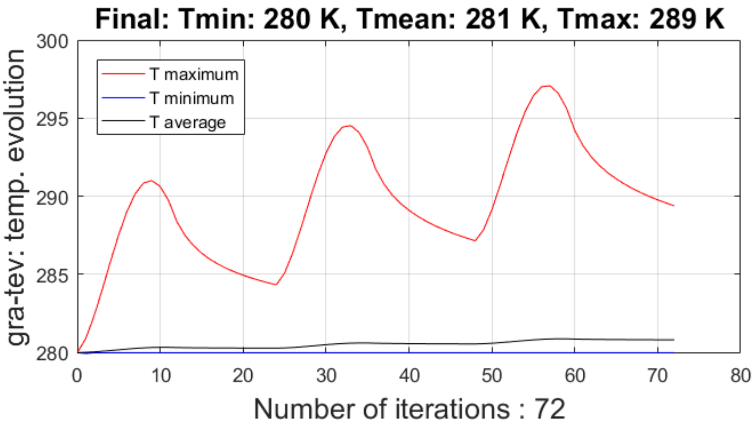


Figure 70: Temperature evolution, adiabatic walls, 3 days, 32 elements per side

The color scales of both drawings of *Figure 68* are the same when the instructions *78 - 80* of the Matlab© 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*).

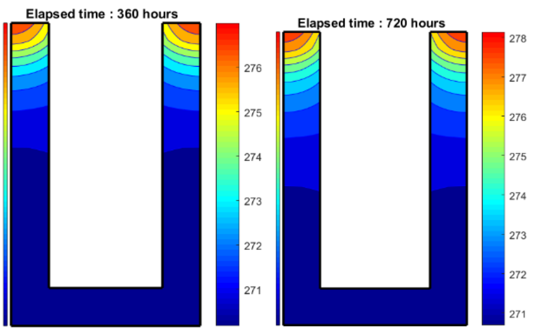
A perfectly reflective wall is identical to an adiabatic one (*Figure 71*).The latter needs less data and is easier to run because it does not use radiative exchanges parameters or methods. The method used in *Fiammetta* for this problem is the number 5 (*line 266-298*, *Table 28*), entitled: "Solution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

Adiabatic wall	Perfectly reflective wall ($\rho = 1$)
Fiammetta	**Fiammetta**
Street section, Gi: 35, Di : 0	Street section, Gi: 35, Di : 0
L 4, Meth., Ne, ca: 5 111 0	L 4, Method, Ne : 5 111
L 5, Co nnr nvn : 0 0 0	L 5, Co nnr nvn : 0 0 0
L 6, rc, ra, cs : 0 0 0	L 6, rc, ra, cs : 1 0 2
L 7, CAD interf. : 10	L 7, CAD interf. : 10
L 8, Thickness : 0.1 m	L 8, Thickness : 0.1 m
L 9, Conduction k : 5 W/(mK)	L 9, Conduction k : 5 W/(mK)
L 10, DT isotherms : 2 K	L 10, DT isotherms : 2 K
L 11, Refl. coeff. : 0	L 11, Refl. coeff. : 1
L 14, np, nvertices: 3 8	L 14, np, nvertices: 3 8
L 16, num. nod side: 15	L 16, num. nod side: 15
L 17, num elem side: 16	L 17, num elem side: 16
L 28, Domain area : 19 m2	L 28, Domain area : 19 m2
L 27, Num. elements: 768	L 27, Num. elements: 768
L 33, Num. of DOF : 833	L 33, Num. of DOF : 833
L 63, Domain perim.: 40 m	L 63, Domain perim.: 40 m
L 70, St. Boltzmann: 5.6704e-08 W/(m2K4)	L 66, St. Boltzmann: 5.67e-08 W/(m2K4)
LD 4, N. virt. nod.: 0	LD 4, N. virt. nod.: 0
LD 7, Numb. fix nod: 0	LD 7, Numb. fix nod: 0
N 03, param Ne : 111	N 03, param Ne : 111
N 51, Total load : 1 W	N 51, Total load : 1 W
N 53, Loaded area : 0.2 m2	N 53, Loaded area : 0.2 m2
N 62, N. load. nod.: 34	N 62, N. load. nod.: 34
	L 89, N. r.nodes ca: 49
	L 91, Rad. vertices: 2 4 6 8
L 94, Anis. index : 0	L 94, Anis. index : 0
sc 04, Initial temp.: 270 K	sc 04, Initial temp.: 270 K
sc 06, Numb. fixat. : 0	sc 06, Numb. fixat. : 0
sc 09, Time step : 3600 sec	sc 09, Time step : 3600 sec
sc 12, Analyzed per.: 720 h, 30 days	sc 12, Analyzed per.: 720 h, 30 days
sc 18, Spec. capac. : 1000 J/(kg.K)	sc 18, Spec. capac. : 1000 J/(kg.K)
sc 19, Spec. mass : 2500 kg.m-3	sc 19, Spec. mass : 2500 kg.m-3
sc 30, Dom. capacit.: 4.75 MJ/K	sc 30, Dom. capacit.: 4.75 MJ/K
sc 31, area*th*ro*Cp: 4.75 MJ/K	sc 31, area*th*ro*Cp: 4.75 MJ/K
sc 36, Imposed Heat : 100 Wm-2	sc 36, Imposed Heat : 100 Wm-2
sc 59, size(K) nfi : 833 833 0	sc 59, size(K) nfi : 833 833 0
sc 95, iteration : 180	sc 95, iteration : 180
sc 95, iteration : 360	sc 95, iteration : 360
sc 95, iteration : 540	sc 95, iteration : 540
sc 95, iteration : 720	sc 95, iteration : 720
sc 97, Line vect. gt: 270.05 0.92 284.9 K	sc 97, Line vect. gt: 270.05 0.92 284.9 K
sc119, Tmean - Tini : 3.18 K	sc119, DTm Tm - Tini: 3.18 K
sc133, Min obs. temp: 270 K	sc120, Min obs. temp: 270 K
sc134, Max obs. temp: 288 K	sc121, Max obs. temp: 298 K
sc135, Final temper.: 270 273 285	sc122, Final temper.: 270 273 285 K
sc124, Capacity* DTm: 15.1 MJ	sc124, Capac*DT mean: 15.1 MJ
sc131, Injected heat: 16.4 MJ	sc131, Injected heat: 16.4 MJ
sc132, Expl. heat in: 16.5 MJ	sc132, Ex.heat input: 16.5 MJ
L 305, Date, CPU, 26-Jul-2023, 11.5129 s	L 305, Date, CPU, 26-Jul-2023, 11.5199 s
g 18, Max temp grad: 5.3, mean: 1.3 K/m	g 18, Max temp grad: 5.3, mean: 1.3 K/m

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

6.4.3 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.  : 270 K
L 130, Rad. edg. len: 7  3  7 m
L 133, Radiativ area: 1.7 m2
L 142, Det. R. mat M: 1
L 145, SVF sum(Fsky): 8.73
L 159, Sum sky loads: -90.33 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 29,      sum(Mss) : 3.644 W
sc 95, iteration    : 720
sc 96, Line vect. gt: 270.7 0.7 278.14  K
sc119, DTm Tm - Tini: 1.94   K
sc120, Min obs. temp: 270    K
sc121, Max obs. temp: 281    K
sc122, Final temper.: 271 272 278 K
sc124, Capacity* DTm: 9.22   MJ
sc131, Injected heat: 16.4   MJ
sc132, Expl. heat in: 16.5   MJ
L 2dm, He fl. K*tca : -3.77  W
L 303, Max. T(lcont): 276    K
L 305, Date, CPU, 26-Jul-2023, 11.5732 s
g  18, Max temp grad: 4, mean: 0.72  K/m
hf 25, Max heat flow: 20, mean: 3.6  W/m2
```

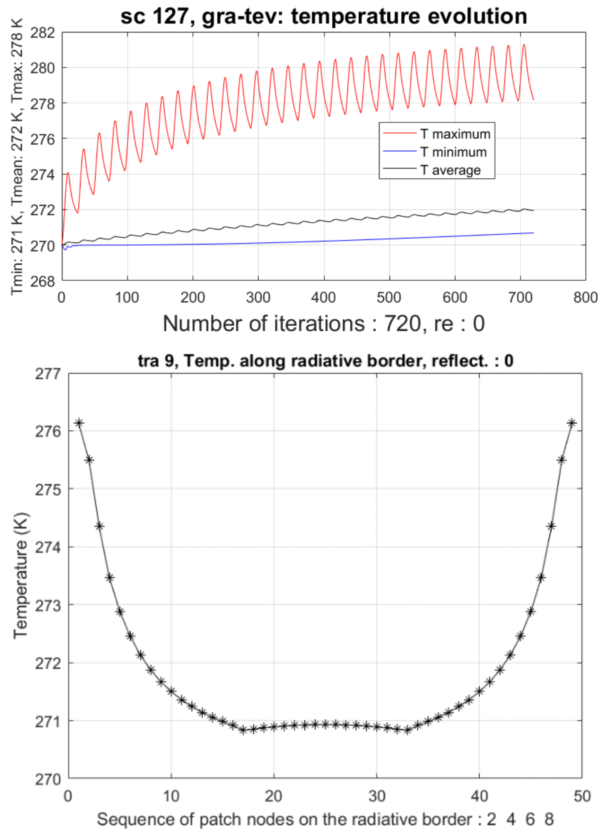
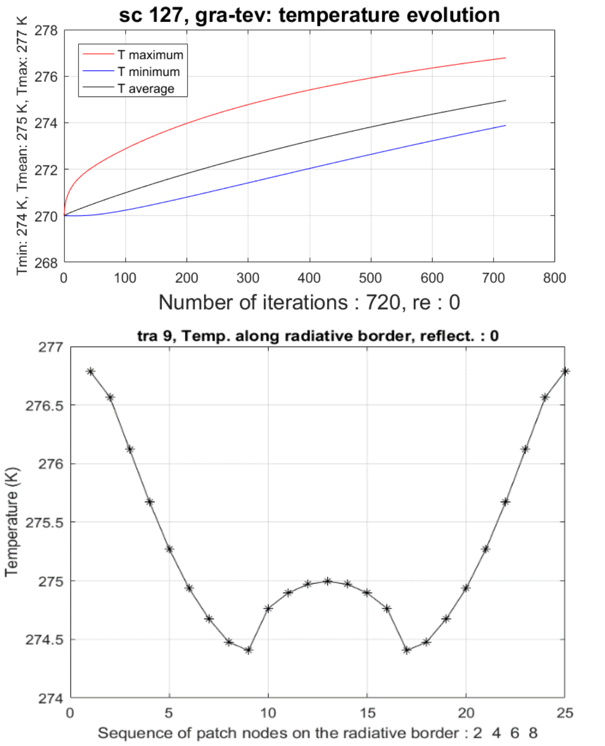


Figure 72: Street section with black body walls and injected heat of 100 Wm^{-2}, 16.4 MJ

If we impose again black body street walls, remove the imposed heat load and increase the sky temperature to 280 *K*, 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   7, CAD interf.  : 10
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 142, Det. R. mat M: 1
L 145, SVF sum(Fsky): 4.36
L 159, Sum sky loads: -1042 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, Dom. capacit.: 47.5 MJ/K
sc 31, area*th*ro*Cp: 47.5 MJ/K
sc 36, Imposed Heat : 0    Wm-2
sc 59, size(K) nfi  : 225  225    0
sc 95, iteration    : 180, 360, 540, 720
rs 29,      sum(Mss) : -73.81 W
sc 96, Line vect. gt: 273.9 0.3 276.8 K
sc119, DTm Tm - Tini: 4.96 K
sc120, Min obs. temp: 270  K
sc121, Max obs. temp: 277  K
sc122, Final temper.: 274 275 277 K
sc124, Capacity* DTm: 235  MJ
L 2dm, He fl. K*tca : 58.9 W
L 303, Max. T(lcont): 277  K
L 305, Date, CPU, 26-Jul-2023, 2.2544 s
g  18, Max temp grad: 1.2, mean: 0.52 K/m
hf 25, Max heat flow: 5.9, mean: 2.6 W/m2
```

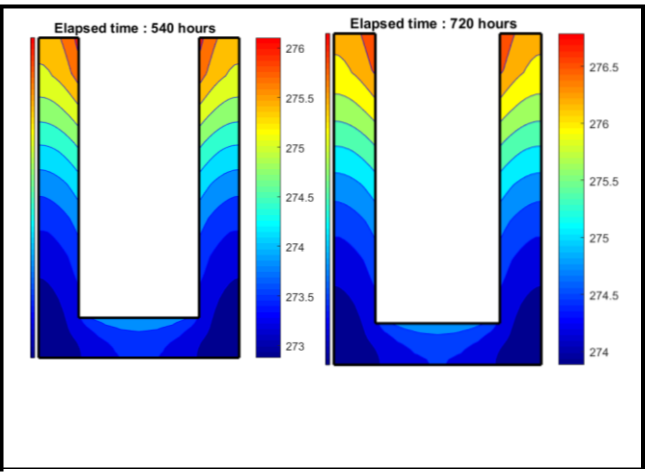


Figure 73: Street section with black body walls and without injected heat

From the pure geometric characteristics [F_{sky}], we deduce the impact of the sky radiation, which is proportional to its temperature (*L 145, SVF sum(Fsky): 4.36*).

6.4.4 Street walls emissivity equal to .5

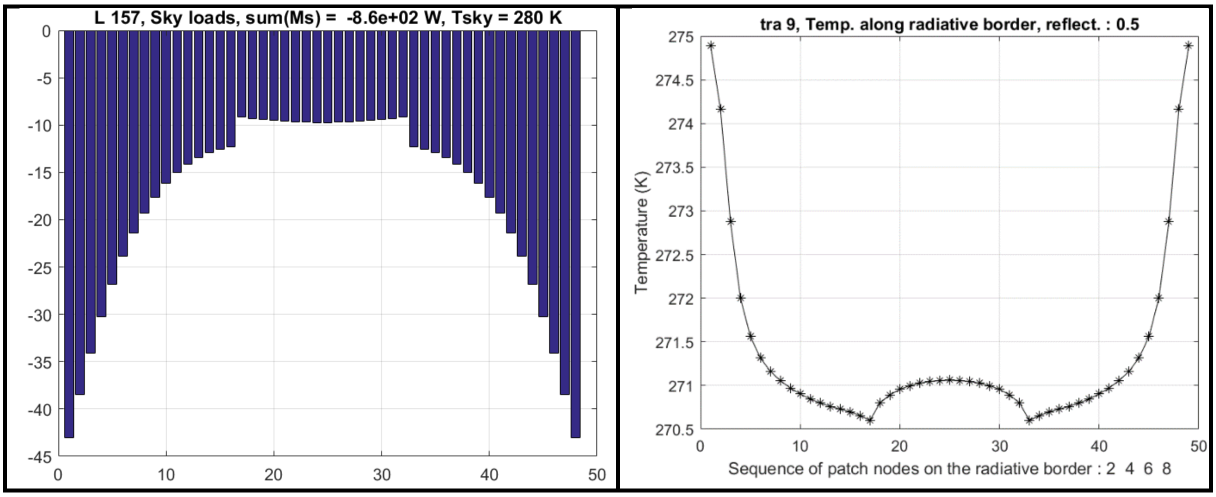


Figure 74: Street section, ρ = 0.5, sky loads and wall temperatures after 72 hours

```
Fiammetta
Street section,  Gi: 35, Di : 0
L   4, Method, Ne    : 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     : 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 171, Sum sky loads: -855 W
sc 04, Initial temp.: 270 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
sc 95, iteration    : 18 … 36… 54
rs 29,       sum(Mss) : -99.28 W
sc 95, iteration    : 54 …72
sc119, DTm Tm - Tini: 0.938 K
sc120, Min obs. temp: 270  K
sc121, Max obs. temp: 278  K
sc122, Final temper.: 270 271 275 K
sc124, Capacity* DTm: 44.6  MJ
sc131, Injected heat: 16.4 MJ
sc132, Expl. heat in: 16.5 MJ
L 2dm, He fl. K*tca : 93.9  W
L 303, Max. T(lcont): 275  K
L 305, Date, CPU, 26-Jul-2023, 2.7685 s
g  18, Max temp grad: 3.2, mean: 0.82 K/m
hf 25, Max heat flow: 16, mean: 4.1 W/m2
```

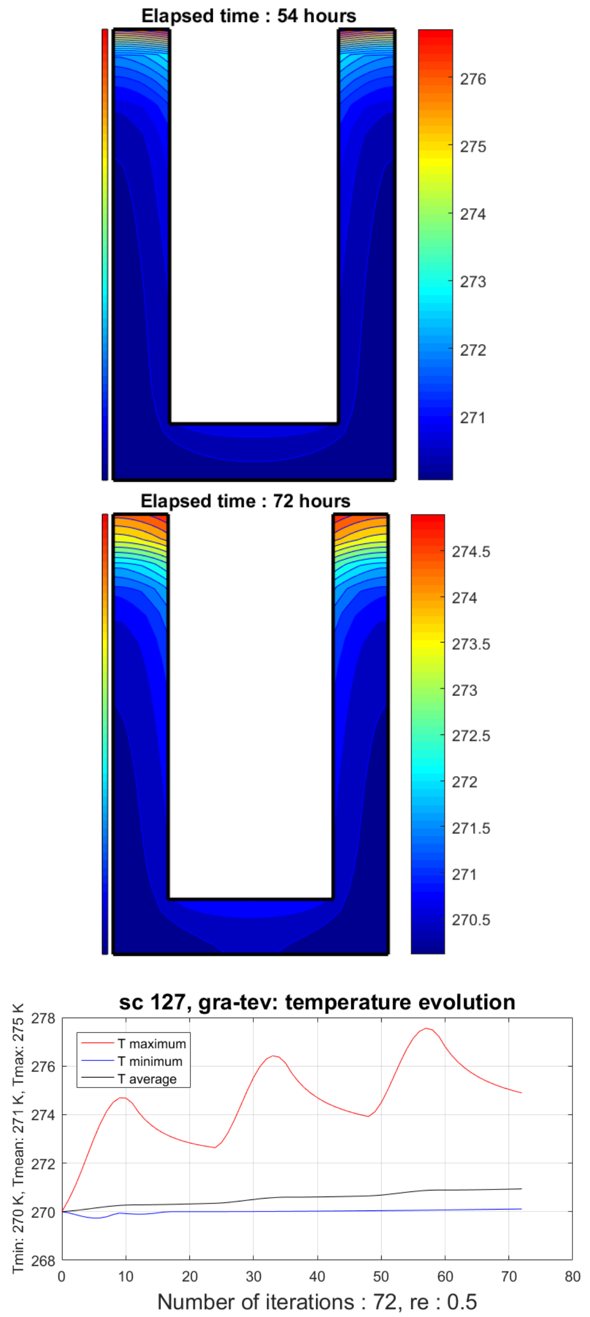


Figure 75: Street section, ρ = 0.5, injected heat: 16.4 MJ, sky temperature = 280 K

6.5 Thermal bridge

6.5.1 Stationary heat flow - 2 convective virtual nodes

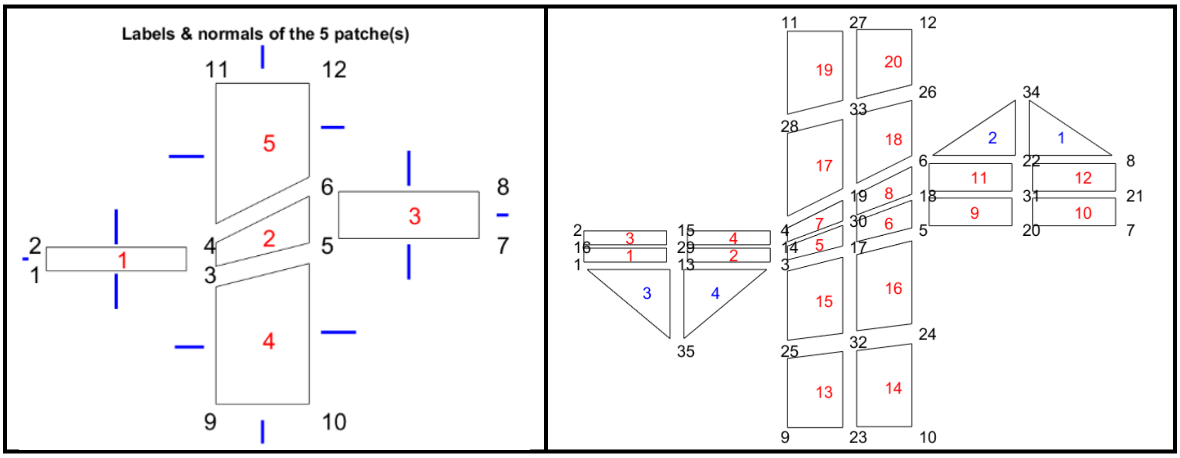


Figure 76: Thermal bridge, topological configuration

The next example relates to a domain involving convective boundaries on sides 1-3 and 6-8 of the main body (*Figure 76*). The convection conditions 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*).

```
Fiammetta
Thermal bridge, Gi: 18, Di : 2
L   4, Meth., Ne, ca: 1  0  0
L   5, Co nnr nvn   : 5  0  2
L   6, rc, ra, cs   : 0  0  0
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  13, Convection h : 20 W/(m2K)
L  15, np, nvertices: 5  12
L  17, num. nod side: 15
L  19, num elem side: 16
L  29, Domain area  : 66 m2
L  32, Num. elements: 1280
L  34, Num. of DOF  : 1379
L  63, Domain perim.: 56 m
LD 16, Fixed nodes  : 1378  1379
LD 17, Fix. temper. : 300  280 K
N  03, param Ne     : 0
c. 03, dK = no + nvn: 1379
c. 04, N.virt c.nod.: 2
c. 05, Variable Co  : 5
c. 40, Convect. sid.: 8  6  1  3
c 187, Nu. conv. el.: 32
L  96, Anis. index  : 0
L 216, Total dissip.: 69.5 WK
L 219, Dis. in solid: 56.5 WK
L 220, DT   in solid: 19.9 K
L 230, Fixed DOF    : 1378  1379
L 231, Imposed temp.: 300        280 K
L 250, React. flows : 6.95      -6.95 W
L 255, Date, CPU, 26-Jul-2023, 0.84384 s
g  18, Max temp grad: 8.3, mean: 1.1 K/m
ch 03, coef. red. dt: 25 W/m2
ch 19, temp. grad.  : 3.7306 K
ch 20, Mean conv. fl: 0 -0.57915 0 W/m2

L  17, num. nod side: 15
hf 25, Max heat flow: 83, mean: 11 W/m2
L  17, num. nod side: 4
hf 25, Max heat flow: 42, mean: 11 W/m2
```

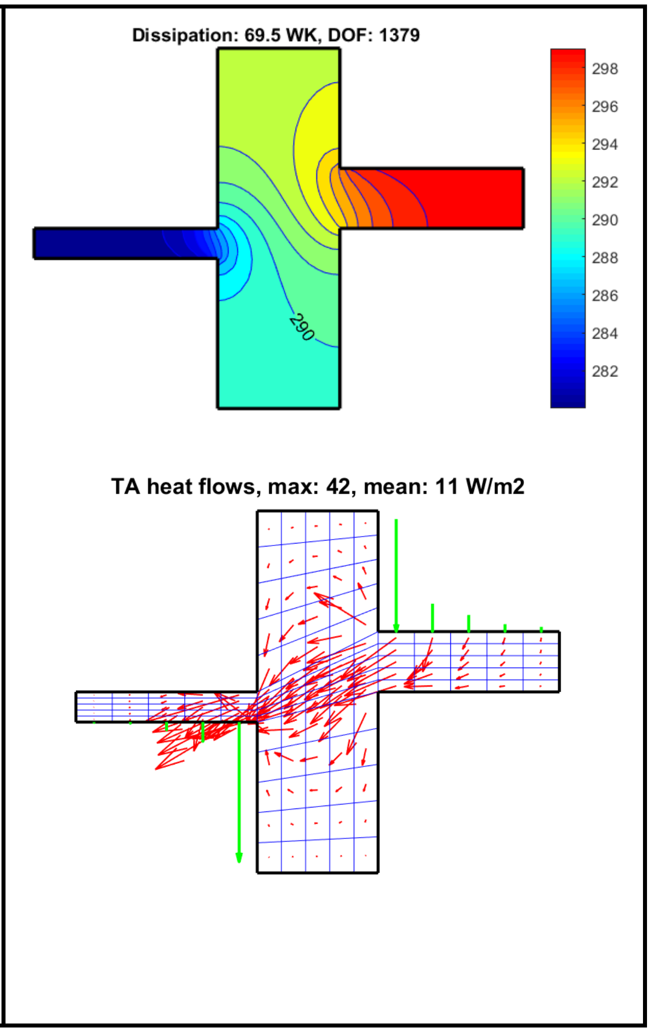


Figure 77: Thermal bridge with 2 convective virtual nodes, steady state

- 1 radiative virtual node & 1 convective virtual node

In the example of *Figure 78*, 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 78*) by the difference of temperature: 30 *K* (*sd 55, Figure 78*) is equal to the dissipation: 693.1 *WK* (*sd 39, Figure 78*). 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_
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 238, Date, CPU, 26-Jul-2023, 1.1 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
```

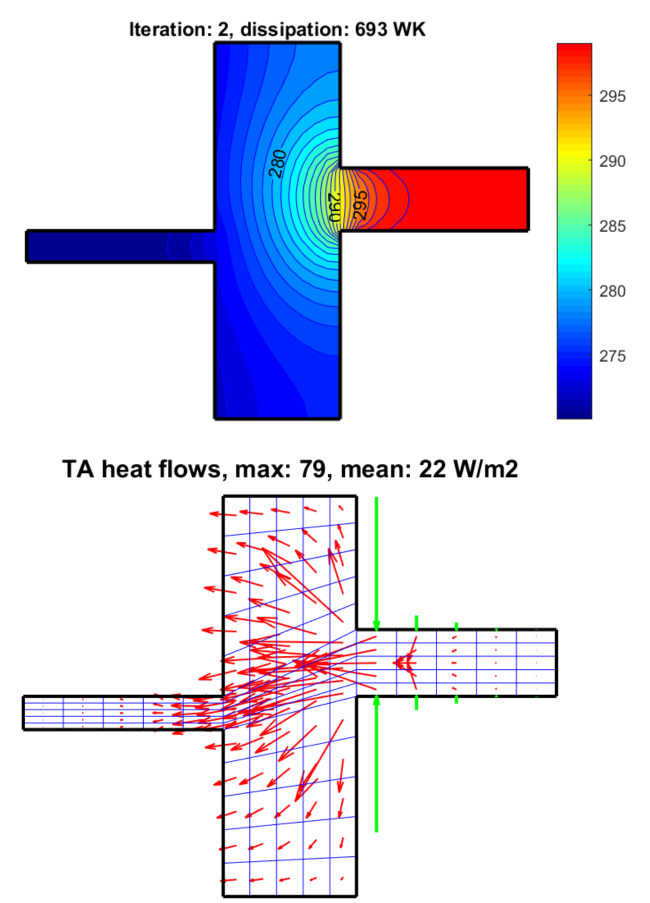


Figure 78: Thermal bridge, mixed b. c., 1 convective virtual node, 1 radiative v. node (ρ = .5)

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

```
Fiammetta
Thermal bridge, Gi: 20, Di : 7
L   4, Method, Ne    : 3  0  0
L   5, Co nnr nvn    : 5  0  2
L   6, rc, ra, cs    : 0  0  0
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  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 m2
L  29, Num. elements: 320
L  31, Num. of DOF  : 371
L  62, Domain perim.: 56 m
N  03, param Ne      : 0
c. 03, dK = no + nvn: 371
c. 04, N.virt c.nod.: 2
c. 05, Variable Co  : 5
c. 40, Convect. sid.: 8  6  1  3
c 125, Nu. conv. el.: 16
L  96, Anis. index  : 0
st 08, Ini. tmi, tma: 280  300 K
st 09, Numb. fixat. : 2
st 22, Fix. temp. fT: 300  280 K
st 23, ddl fix. lfi : 370  371
st 27, N. iter. nit : 720
st 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))  : 16.5 MJ/K
st 48, area*th*ro*Cp: 16.5 MJ/K
st 52, Imposed Heat : 100 W/m-2
st 57, size(K) nfi  : 371  371    2
st 82, iteration     : 180
st 82, iteration     : 360
st 82, iteration     : 540
st 82, iteration     : 720
st104, ddt Tm - Tin : 5.56 K
st105, ddt*sumsum(C): 91.7 MJ
st106, Min obs. temp: 280 K
st107, Max obs. temp: 299.7 K
st111, Heat inp.eih: 132 MJ
L 272, Date, CPU, 26-Jul-2023, 8.4566 s
g  18, Max temp grad: 7.8, mean: 1.2 K/m
hf 24, Max heat flow: 78, mean: 12 W/m2
```

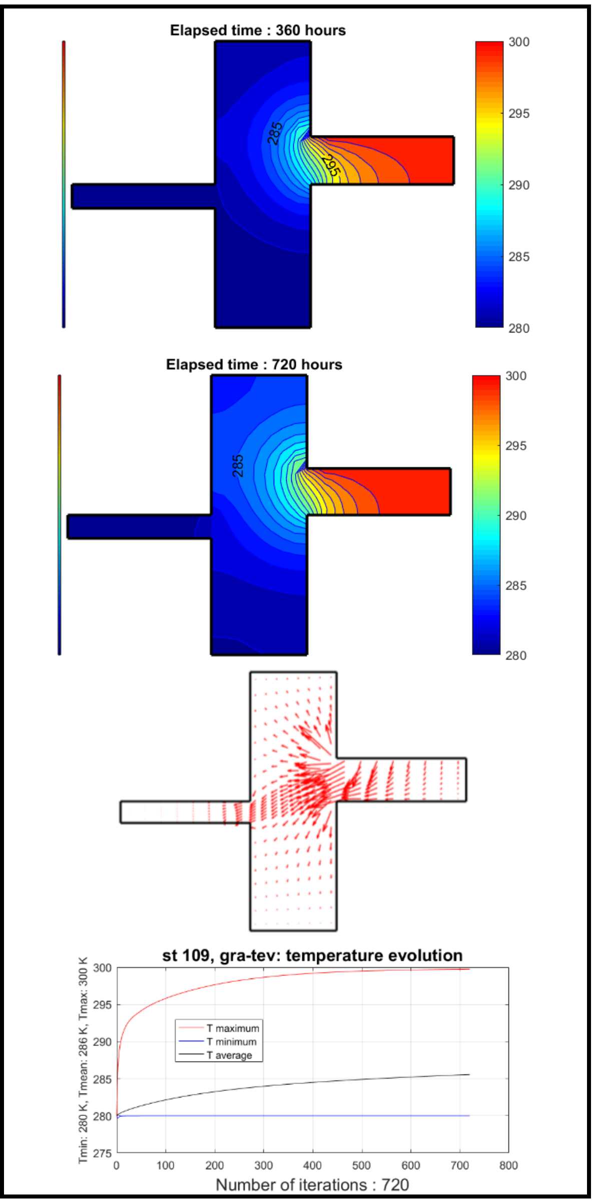
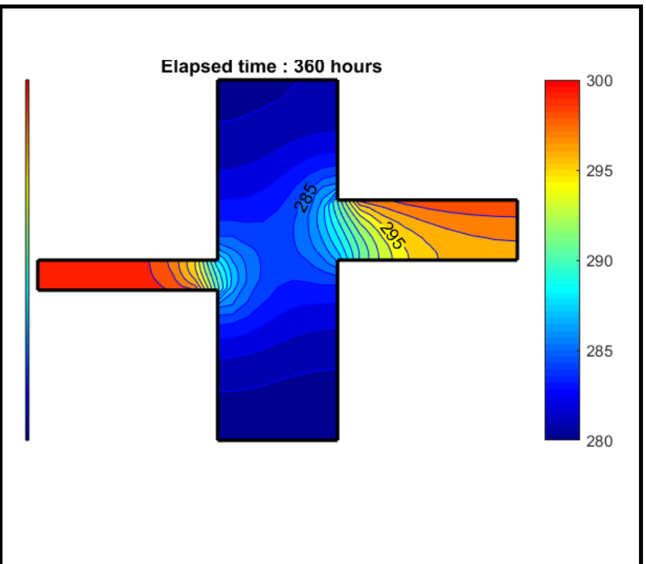


Figure 79: Thermal bridge, transient analysis, 2 convective virtual nodes

b. One convective and one radiative virtual node

```
Fiammetta
Thermal bridge, Gi: 36, Di : 16
L   4, Me, Di, Ne    : 4  0  0
L   5, Co nnr nvn    : 5  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 m2
L  29, Num. elements: 320
L  31, Num. of DOF  : 371
L  62, Domain perim.: 56 m
L  66, St. Boltzmann: 5.67e-08 W/(m2K4)
LD 76, Fixed nodes  : 370   371
LD 77, Fix. temper. : 280  300 K
N  03, param Ne      : 0
c. 03, dim. K Co cs : 371
```



```
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 285, Date, CPU, 26-Jul-2023, 3.6486 s
hf 24, Max heat flow: 77, mean: 16 W/m2
```

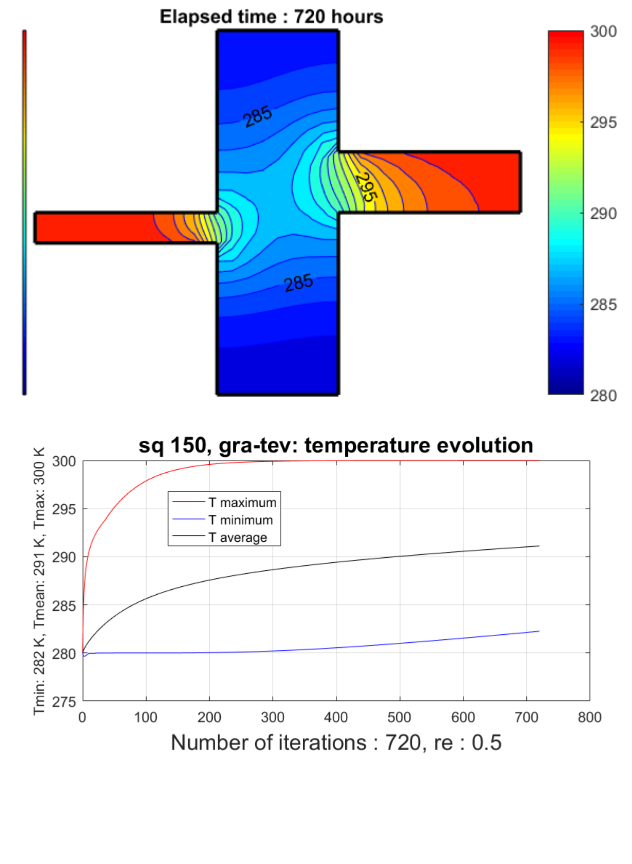


Figure 80: Thermal bridge, transient analysis, 1 convective & 1 radiative virt. node, $\rho = .5$

c. Radiative edge

```
Fiammetta
Thermal bridge, Gi: 21, Di : 2
L   4, Me, Di, Ne   : 4  0  0
L   5, Co nnr 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. : 1
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  30, Domain area  : 66 m2
L  33, Num. elements: 320
L  35, Num. of nodes: 369
L  36, 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
LD 17, Fix. temper. : 280  300 K
N  03, param Ne     : 0
c. 03, Numb. var. dK: 371
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  96, Anis. index  : 0
L 127, Sky temper.  : 280 K
L 131, Rad. edg. len: 6  6  1  6  5 m
L 132, Radiativ area: 2.4 m2



L 239, Num. fix. DOF: 370  371
L 240, T. fix. nod. : 280  300 K
sq 04, Initial temp.: 280 K
sq 05, Fixed DOF    : 370  371
sq 09, Time step    : 3600 s
```

```
Fiammetta
Thermal bridge, Gi: 22, Di : 12
L   4, Me, Di, Ne   : 4  0  0
L   5, Co nnr 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. : 1
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  30, Domain area  : 66 m2
L  33, Num. elements: 320
L  35, Num. of nodes: 369
L  36, Num. of DOF  : 370
L  59, Domain perim.: 56 m
L  66, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD110, Fixed nodes  : 370
LD111, Fix. temper. : 300 K
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  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
L 166, dim F = nv   : 42
L 175, determinant M: 0.6935
L 179, Sum 2d member: -0.0836 W
L 253, Num. fix. DOF: 370
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-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 : 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 283, Diss in solid: 111 WK
L 284, Total dissip.: 140 WK
L 285, Date, CPU, 26-Jul-2023, 3.606 s
hf 24, Max heat flow: 75, mean: 13 W/m2
```

```
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  42
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
sq143, 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 283, Diss in solid: 111 WK
L 284, Total dissip.: 140 WK
L 285, Date, CPU, 26-Jul-2023, 3.901 s
hf 24, Max heat flow: 75, mean: 13 W/m2
```

Figure 81: 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 ρ more or less in the range $0 \leq \rho \leq .9$.

```
Fiammetta
Thermal bridge, Gi: 21, Di : 2
L   4, Me, Di, Ne    : 4  0  0
L   5, Co nnr 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 m2
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
LD 17, Fix. temper.  : 280  300 K
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 m2



L 253, Num. fix. DOF: 370  371
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-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
```

```
Fiammetta
Thermal bridge, Gi: 22, Di : 12
L   4, Me, Di, Ne    : 4  0  0
L   5, Co nnr 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 m2
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
LD111, Fix. temper.  : 300 K
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 m2
L 166, dim F = nv    : 42
L 177, determinant M: 0.9161
L 181, Sum 2d member: -348 W
L 253, Num. fix. DOF: 370
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-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  42
sq 44, N. rad. edges: 5
sq 45, rad seg leng.: 6  6  1  6  5 m
```

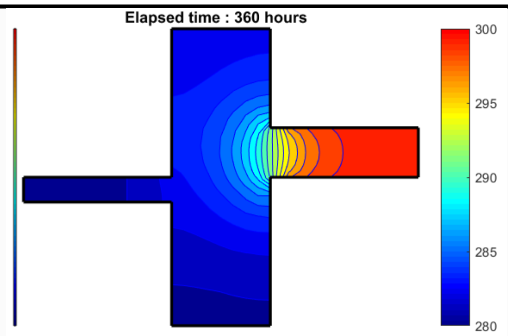
```
sq 46, N. rad. elem.: 40


sq122, Iteration    : 360 gt: 280 1 300 K
sq122, Iteration    : 720 gt: 280 1 300 K
sq143, 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 283, Diss in solid: 111 WK
L 284, Total dissip.: 140 WK
L 285, Date, CPU, 26-Jul-2023, 3.8247 s
hf 24, Max heat flow: 75, mean: 13 W/m2
```

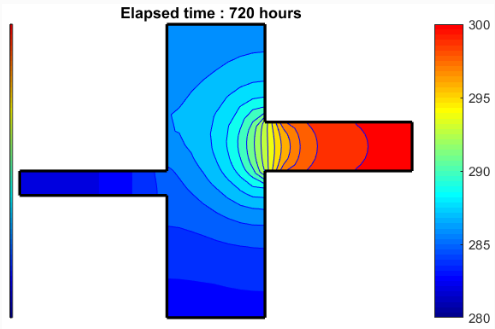
```
sq 46, N. rad. elem.: 40
sq 63, sum(gr)      : 0 W
sq 77, sum(Mn)      : 0 W
sq122, Iteration    : 360 gt: 280. 1 300 K
sq122, Iteration    : 720 gt: 280. 1 300 K
sq143, Tmean - Tini : 6.89 K
sq144, Stored heat  : 114 MJ
sq146, Min it+1 temp: 280.3 K
sq147, Max it+1 temp: 287 K
sq148, Max it+1 temp: 300 K
sq154, Ejected heat : 0.202 MJ
L 283, Diss in solid: 111 WK
L 284, Total dissip.: 140 WK
L 285, Date, CPU, 26-Jul-2023, 3.8439 s
hf 24, Max heat flow: 75, mean: 13 W/m2
```

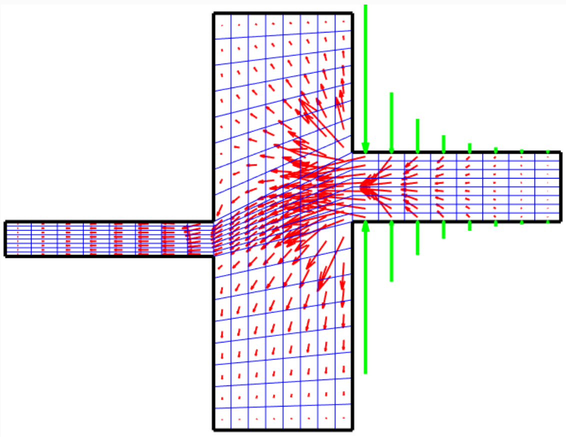
Figure 82: Thermal bridge comparison, radiative virtual node – radiative edge, $\rho = .5$

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.

```
Fiammetta
Thermal bridge, Gi: 22, Di : 12
L   4, Method, Ne    : 4  0  0
L   5, Co nnr 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  : 20  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: 7
L  19, num elem side: 8
L  30, Domain area   : 66 m2
L  33, Num. elements: 320
L  35, Num. of nodes: 369
L  36, Num. of DOF   : 370
L  66, Domain perim.: 56 m
L  66, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD110, Fixed nodes   : 370
LD111, Fix. temper.  : 300 K
N  03, param Ne      : 0
c. 03, dK = no + nvn: 370
c. 04, N.virt c.nod.: 1
c. 05, Variable Co   : 6
c. 54, Convect. sid.: 8  6  5  7
c 179; 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, Radiativ area: 2.4 m2
L 166, dim F = nv    : 42
L 177, determinant M: 0.9161
L 181, Sum 2d member: -348 W
L 253, Num. fix. DOF: 370
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-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  42
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: 280.25 1 300 K
sq122, Iteration     : 720 gt: 281.78 1 300 K
```







```
sq143, Tmean - Tini : 8.36  K
sq144, Stored heat  : 138   MJ
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 283, Diss in solid: 138   WK
L 284, Total dissip.: 181   WK
L 285, Date, CPU, 26-Jul-2023, 3.8827 s
hf 24, Max heat flow: 110, mean: 22 W/m2
```

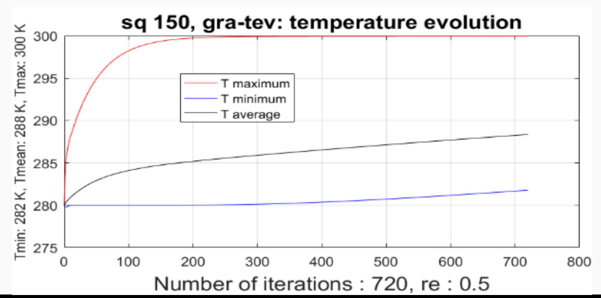


Figure 83: Thermal bridge, transient analysis, 1 convective node & 1 radiative side

7 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© procedure *Fiammetta.m* and, secondly, comment a scheme of the procedure, the meaning of each variable and a description of the output format.

Procedure *Fiammetta.m*

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

```
[xyt,lK,bor,pbo] = cad_mes(xyz_cao,car_cao,nni,nbo);     % Mesh generation
nel  = size(lK,1) ; disp(['L  33, Num. elements: ',num2str(nel)])
no   = size(xyt,1);dK = no + nvn + nnr;
       disp(['L  35, Num. of nodes: ',num2str(no)])
       disp(['L  36, Num. of DOF  : ',num2str(dK)])
for i    = 1:nel         % Enforce anticlockwize ordering of element nodes
    n    = cross([xyt(lK(i,3),:)-xyt(lK(i,1),:) 0],...
           [xyt(lK(i,4),:)-xyt(lK(i,1),:) 0]);
    if n(3) < 0;iq = lK(i,2);lK(i,2) = lK(i,4);lK(i,4) = iq;end
end
bov   = zeros(nbo,npv+2); bov(:,1:2)=bor(:,1:2); pe = 0.;% Outward normals
if deb==1;disp('L  43, Call function: ..... gra_mnl .....');end
gra_mnl(xyz_cao,car_cao,[0 0 0],15);axis equal;%Drawing normals: + line 56
axis off;title(['Labels & normals of the ',num2str(np),' patche(s)'])
for i = 1 : nbo % ........................ Loop on the nin CAD interfaces
    if bor(i,4)==0% interface i must be a single one to be on the boundary
        ne = cross([xyt(bor(i,2),:)-xyt(bor(i,1),:) 0],[0 0 1]);
        nn = ne / norm(ne);
        mi = [(xyt(bor(i,6),:) + xyt(bor(i,5),:))/2 0];         % Mid edge
        li = norm(xyt(bor(i,2),:) - xyt(bor(i,1),:));        % Edge length
        pe = pe + li;            % Update the perimeter of the CAD domain
        po = (mi + nn*li/5 );                   % Normal vector: mi - po
        plot([mi(1) po(1)],[mi(2) po(2)],'b','LineWidth',2);hold on
        for j      = 1:npv     % Loop on npv vertices not in patch bor(i,3)
            yes    = 0;
            for c = 1:4            % Check if vertex j is in patch bor(i,3)
                if j == car_cao(bor(i,3),c);yes = 1;end
            end
            if yes == 0  % Find vertices visible from interface bor(i,1:2)
                pn  = [xyz_cao(j,:) 0] - mi; vis = dot(po-mi,pn);
                if vis > 0; bov(i,j+2)=j; end
            end
        end
    end
end % .....End loop on the nbo CAD interfaces to draw the outwards normals
disp(['L  66, Domain perim.: ',num2str(pe),' m'])    % End outward normals
xyz  = zeros(dK,3); xyz(1:no,1:2) = xyt;     % Coordinates expressed in 3D
SB   =  5.6704e-8;                        % Stefan-Boltzmann constant Wm-2K-4
disp(['L  70, St. Boltzmann: ',num2str(SB),' W/(m2K4)'])
if deb    == 1;disp( 'L  71, Call function: ..... cad_Dir ....');end
[lfi, fT] =  cad_Dir(Di,no,nvn,car_cao,bor,pbo,nni);      % Dirichlet b.c.
if lfi(1) == 0;nf = 0;else;nf=size(lfi,2);end
if deb    == 1; disp( 'L  74, Call function: ..... cad_Neu .....');end
[gh,lg,bos]= cad_Neu(Ne,dK,car_cao,pbo,bor,th,xyz_cao);      % Neumann b.c.
if deb    == 1;disp('L  81, Call function: ..... gra_mnl .....');end
if Co > 0
    if deb == 1;disp('L  78  , Call function: ..... cad_con ....');end
    [lc,vc,he] = cad_con(car_cao,bor,pbo,h,dK,nci,deb,nvn,cs,Co,xyz);
    xyz(no+(1:nvn+nnr),1:2) =  vc(1:nvn+nnr,1:2);% Virtual nodes coord.
end % End option Co > 0. Some sides are linked to virtual convective nodes
if nel < 100;gra_mnl(xyz,lK,lc,10);axis equal,axis off;hold on;end
lcont = zeros(1,4*nci);lv=zeros(1,4);     % DOF and vertices of the cavity
if cs == 0                    % Compute the radiative nodes, vertices & number
    mcr = 0;                             % mcr is the number of radiative nodes
else                  % Analysis of the 3 situations: cavity, street, balcony
    if deb==1; disp( 'L  88, Call function: ..... cad_ban .....');end
    [lcont,lv] = cad_ban(car_cao,bor,pbo,nni,cs);mcr = max(size(lcont));
    disp(['L  89, Numb ra nodes: ',num2str(mcr)])
    if size(lv,2) > 1;disp(['L  90, Rad. vert. lv: ',num2str(lv)]);end
end
%  ...................... Assembling the nel element conductivity matrices
Kk  = zeros(dK,dK);            % Global conductivity matrix initialization
    disp(['L  94, Anis. index  : ',num2str(Ai)])
if Ai  == 0;co = ones(1,nel)*k;else;co = mat_cok(Ai,nci,fa,xyz,lK,deb);end
if deb == 1; disp('L  98, Call function: ..... fem_Kco .....');end
for n    = 1:nel        % Loop on the elements for computing global K matrix
    Kel = fem_Kco(xyz,lK(n,:))*co(n)*th;      % Element conductivity matrix
    for i       = 1:4;i1 = lK(n,i);
        for j  = 1:4;j1 = lK(n,j);Kk(i1,j1) = Kk(i1,j1) + Kel(i,j);end
    end
end                           % End assembling the nel conductivity matrices
K         = Kk;
if Co > 0     % Assembling the nco = size(lc,1) element convective matrices
    if deb    == 1;disp('L 107, Call function: ..... fem_Kcv .....');end
    for n      = 1:size(lc,1)                              % loop on elements
        Kec    = fem_Kcv(xyz,lc,he(n)*th);      % Element convective matrices
        for i = 1:3
            for j=1:3;K(lc(n,i),lc(n,j))=K(lc(n,i),lc(n,j))+Kec(i,j);end
        end
    end
end                             % End assembling the nco convection matrices
```

```
if Gi==41;Lel=ones(4,1);end;if Gi==42;Lel=ones(4,1);end;
if rc      == 0                     % Radiative exchanges are not present
    Lel(1,1) = 0; ns = 0;                          % Init. edge elements lenghts
else             % View factor and radiosity matrices for radiative transfers
    if cs == 2;lcc =  lv;          ns = 3;end   % lcc & ns: street section
    if cs == 3;lcc =  lv;          ns = 5;end   % lcc & ns: Thermal bridge
    if cs == 4;lcc =  lv;          ns = 2;end   % lcc & ns: rectangle
    if cs == 1;lcc = [lv' ;lv(1)];ns = 4;end   % lcc & ns: quadril. cavity
%   if cs == 5;lcc = [lv' ;lv(1)];ns = 4;end   % lcc & ns: quadril. cavity
    if cs == 6;lcc = [lv' ;lv(1)];ns = 4;end   % lcc & ns: quadril. cavity
    if cs == 7;lcc = [lv' ;lv(1)];ns = 4;end   % lcc & ns: quadril. cavity
    if cs == 9;lcc = [lv' ;lv(1)];ns = 3;end   % lcc & ns: C shape
    Lel   = zeros(ns,1); % Compute the lengths of ns patch radiative edges
    Tsky  = 280; disp(['L 127, Sky temper.  : ',num2str(Tsky),' K'])
    for i = 1:ns      % Computaion of the lengths of ths n radiative edges
        Lel(i,1)= sqrt((xyz_cao(lcc(i+1),1)-xyz_cao(lcc(i),1))^2 + ...
            (xyz_cao(lcc(i+1),2)-xyz_cao(lcc(i),2))^2);
    end;
    disp(['L 132, Rad. edg. len: ',num2str(Lel(1:ns,1)') ,' m'])
    disp(['L 133, Radiativ area: ',num2str(sum(Lel(:,1)*th)),' m2'])
    if cs == 1  % View factor & radiosity matrices of quadrilateral cavity
        if deb==1;disp('L 136, Call function: ..... geo_vfc .....');end
        Fs = geo_vfc(nci,Lel,ns);I=eye(nr);Ms = zeros(nr,1);
        M  = (I-re*Fs);disp(['L 137, Det. R. mat M: ',num2str(det(M))])
    end
    if cs == 2                 % Street section view factors lines 150 -- 172
        if deb ==1;disp('L 141, Call function: .... . geo_stf .....');end
        Fs = geo_stf(nci,Lel(2:3));nr = size(Fs,1);I = eye(nr);
        M  = (I-re*Fs);disp(['L 142, Det. R. mat M: ',num2str(det(M))])
        Fsky = 1-sum(Fs,2)'; % Fsky is the line vector of sky view factors
        if nni ==1;disp(['L 144, Sky view fact: ',num2str(Fsky,3)]);end
        disp(['L 145, SVF sum(Fsky): ',num2str(sum(Fsky),3)])
        if re == 1      % Sky & Ground contributions to radiative exchanges
            Ms = zeros(nr,1);
        else
            lon            = zeros(1,nci*3);
            for i          = 1:nci          % Element lengths on the 3 sides
                lon(1,i)        = Lel(1)/(nci); lon(1,i+nci) =Lel(2)/(nci);
                lon(1,i+2*nci) = Lel(3)/(nci);
            end
            Ms =(re*(I-Fs)*M^(-1)-I)*Fsky'.*lon'*th*SB*Tsky^4; % Fiam. 126
         if nr<10;disp(['L 155,     Sky loads : ',num2str(Ms',2),' W']);end
            figure;bar(Ms');grid on
            title( ['L 157, Sky loads, sum(Ms) =  ',...
                    num2str(sum(Ms),2),' W, Tsky = ',num2str(Tsky),' K']);
            disp(['L 159, Sum sky loads: ',num2str(sum(Ms),4),' W'])
        end
    end
    if cs == 3                 % Thermal bridge view factors lines 171 -- 190
        if deb==1;disp('L 163, Call function: ..... geo_baf .....');end
        nco = size(Lel,1);kn = 0;lon = zeros(1,nco*nci);%Ftot  = 0    ;
        for i = 1:nco;for j = 1:nci;kn=kn+1;lon(1,kn) = Lel(i)/nci;end;end
        if ra == 1;F = geo_baf(nci,xyz_cao);
            nv = size(F,2);disp(['L 166, dim F = nv    : ',num2str(nv)])
            Fs     = F(1:nv-2,1:nv-2);I = eye(size(Fs,1));M = (I-re*Fs);
            disp(['L 169, determinant M: ',num2str(det(M))])
            Fgr    = F(1:nv-2,nv-1); Fsky=F(1:nv-2,nv);
            kn=0;%lon = zeros(1,nne
            Ms      =(re*(I-Fs)*M^(-1)-I)*SB*th*(Fsky+Fgr).*Tsky^4.*lon';
            disp(['L 173, Sum 2d member: ',num2str(sum(Ms),3),' W'])
            if nni == 1
                disp(['L 175, Ms  Eq. 104  : ',num2str(Ms')])
                disp(['L 176, unicol Fgr   : ',num2str(Fgr' ,3)]);
                disp(['L 177, unicol Fsky  : ',num2str(Fsky',3)])
                disp(['L 178, sum(F,2)     : ',num2str(sum(F,2)',3)])
            end
        end
    end
    if cs == 6  % View factor & radiosity matrices of quadrilateral cavity
        if deb==1;disp('L 183, Call function: ..... geo_vfc .....');end
%         Fs = geo_vfc(nci,Lel,ns);I=eye(nr);Ms = zeros(nr,1);
          Fs = geo_vfc(nci,Lel,4 );I=eye(nr);Ms = zeros(nr,1);
%      Fi = geo_vfc(nci,Lel,4);I=eye(nr);Fs = (Fi+Fi')/2;Ms = zeros(nr,1);
        M  = (I-re*Fs);disp(['L 187, Det. R. mat M: ',num2str(det(M))])
    end
    if cs == 7  % View factor & radiosity matrices of quadrilateral cavity
        if deb==1;disp('L 190, Call function: ..... geo_vfr .....');end
        Fs = geo_vfr(nci,Lel   );I=eye(nr);Ms = zeros(nr,1);
        M  = (I-re*Fs);disp(['L 192, Det. R. mat M: ',num2str(det(M))])
    end
    if cs == 9                                                  % C shape
        disp(['L 195, Radiat. nodes: ',num2str(lcont')])
```

```
        if deb==1;disp('L 196, Call function: ..... geo_stf .....');end
        Fs = geo_stf(nci,Lel(2:3));nr = size(Fs,1);I = eye(nr);
        M  = (I-re*Fs);disp(['L 198, Det. Radios M: ',num2str(det(M))])
        Fsky = 1-sum(Fs,2)'; % Fsky is the line vector of sky view factors
        disp(['L 200, 1-sum(Fs,2)  : ',num2str(sum(Fsky),3)])
        if nni ==1;disp(['L 201, Ext view fact: ',num2str(Fsky,3)]);end
        if re == 1      % Sky & Ground contributions to radiative exchanges
            Ms = zeros(nr,1);
        else
            lon            = zeros(1,nci*3);
            for i          = 1:nci         % Element lengths on the 3 sides
                lon(1,i)        = Lel(1)/(nci); lon(1,i+nci) =Lel(2)/(nci);
                lon(1,i+2*nci) = Lel(3)/(nci);
            end
            Ms =(re*(I-Fs)*M^(-1)-I)*Fsky'.*lon'*th*SB*Tsky^4; % Fiam. 119
            figure;bar(Ms');grid on
            title( ['L 212, Sky loads, equation (119), sum(Ms) =  ',...
                    num2str(sum(Ms),2),' W, ',num2str(Tsky),' K']);
            disp(['L 214, Sum sky loads: ',num2str(sum(Ms),4),' W'])
        end
    end
end
% ======================================================================
if Me == 1 % ..... Solution of linear steady state heat transfer problems
    nf     = size(lfi,2);  N   = zeros(nf,dK);% L. con. fix.%gh=zeros(nf,1);
    for i = 1:nf;N(i,lfi(i)) = 1;gh(dK+i) = fT(i); end;
    A     = [K N';N zeros(nf,nf)];B = A\gh;tca = B(1:dK);
    figure;gt =[min(tca) pai max(tca)];
    if deb ==1;disp(['L 224, DT isoth gt  : ',num2str(gt),' K']);end
    nc3    = (nci)^2;  % nc3 = numb elem / patch, same for all the patches
    if deb ==1;disp( 'L 226, Call function: ..... gra_ipa .....');end
    for i = 1 : np                                  % Loop on CAD patches
        gra_ipa(nci,nci,lK((i-1)*nc3+1:nel,:),tca,xyz,gt);hold on
    end
    axis equal;colorbar;axis off
    title (['Dissipation: ',num2str(.5*tca'*K*tca,3),' WK, DOF: ',...
            num2str(dK),' '])
    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 of the isotherms
    disp(['L 239, Total dissip.: ',num2str(tca'*K*tca/2,3),' WK'])
    dissol = .5*tca(1:no)'*Kk(1:no,1:no)*tca(1:no);
    disp(['L 241, Dis. in solid: ',num2str(dissol,3),' WK'])
    disp(['L 242, DT   in solid: ',num2str(max(tca(1:no))-...
            min(tca(1:no)),3),' K'])
    if deb==1;disp(['L 219, tca mi.ma.av.: ',...
            num2str([min(tca) max(tca) mean(tca)],3),' K']);end
    reac = K*tca;
    if nf < 6
        disp(['L 248, Fixed DOF     : ',num2str(lfi)])
        disp(['L 249, Imposed temp.: ',num2str( tca(lfi)',3),' K'])
        disp(['L 250, React. flows : ',num2str(reac(lfi)',3),' W'])
    end
   disp(['L 252, Date, CPU, ',num2str(date),', ',num2str(toc(CPU)),' s',])
    if deb ==1;disp('L 219-254, End method 1, stationary linear');end
end % ==================================================================
if Me == 2  % Nonlinear stat. rad. heat transfer - conv. like virtual node
    if nnr > 0                % nnr is the number of radiative virtual nodes
        if deb ==1;disp('L 236, Call function: ..... fem_snl .....');end
     tca= fem_smd(K,gh,lfi,fT,xyz,lK,lcont,nnr,nvn,nci,np,SB*th*(1-re),...
          ca,nbo,bor);
     disp(['L 238, T. vir. nod. : ',num2str(tca(dK),3),' K'])
    end
 disp(['L 238, Date, CPU, ',num2str(date),', ',num2str(toc(CPU),2),' s',])
if deb ==1;disp('L 231-239, End non lin. sol. with virt. rad. node');end
end % ==================================================================
if Me == 3  % ........ Solution of transient linear heat transfer problems
    if deb==1;disp ('L 251, Call function: ..... fem_smt .....');end
    [tca]=fem_smt(np,xyz,lK,dK,no,nci,deb,cs,area,th,pai,fT,lfi,gh,...
        nbo,bor,K,fmd,Di);
   disp(['L 245, Date, CPU, ',num2str(date),', ',num2str(toc(CPU)),' s',])
    if deb ==1;disp('L 241-246, End linear transient');end
end % ==================================================================
if Me == 4  % Solution of non linear transient rad. heat transfer problems
    disp(['L 251, Num. fix. DOF: ',num2str(lfi)]);% Ms=0;
    disp(['L 252, T. fix. nod. : ',num2str(fT),' K'])
    if deb==1,disp('L 260, Call function: ..... fem_smq .....');end
%     Ftot = zeros(size(lcont,1)-1,size(lcont,1)+1);
    [tca] = fem_smq(np,xyz,lK,dK,nci,deb,rc,ra,cs,area,th,xyz_cao,...
```

```
278          gh,lfi,fT,Tsky,nbo,bor,K,Lel,re,SB*th*(1-re),lcont,Ms,pai,F,lon);
279          rea = Kk*tca;reac = K*tca;
280            disp(['L 258, Diss in solid: ',num2str(rea' *tca/2,3),' WK'])
281        disp(['L 259, Total dissip.: ',num2str(reac'*tca/2,3),' WK'])
282        disp(['L 260, Date, CPU, ',num2str(date),', ',num2str(toc(CPU)),' s',])
283    if deb ==1;disp('L 250-262, End non lin radiat. trans. heat transf.');end
284    end % =====================================================================
285    if Me == 5          % Cavity with view factor matrix and radiative exchanges
286        if rc > 0                           % Radiative heat exchange is present
287            if re ==1                        % re = 1 : adiabatic wall or mirror
288                Mpr = eye(nr,nr);          % nr is the number of radiative edges
289            else                                  % re = 0 : black body walls
290                Mpr =(eye(nr,nr)-Fs)*M^(-1);
291            end
292        else
293            Mpr = zeros(nbo*nci,nbo*nci); % if cs~=2;Ms = zeros(nbo*nci,1);end
294        end %     disp('L 278, Mpr');disp(M);disp(Mpr),disp(Ms)
295        if deb==1;disp('L 271, Call function: ..... fem_smc .....');end
296        [tca]  = fem_smc(np,xyz,lK,dK,nci,deb,rc,cs,area,gh,lg,...
297            fT,lfi,nbo,bor,K,mcr,Lel,SB,re,Ms,Mpr,lcont,pai,th,ca,bos);
298        if rc > 0  % Additional output when radiative heat exchange is present
299            if deb==1;disp('L 275, Call function: ..... gra_2dm .....');end
300            gra_2dm(K,tca,re,lcont)
301            if deb==1;disp('L 277, Call function: ..... gra_tra .....');end
302            gra_tra(tca,lcont,lv,re,ca)
303            disp(['L 303, Max. T(lcont): ',num2str(max(tca(lcont)),3),' K'])
304        end
305        disp(['L 305, Date, CPU, ',num2str(date),', ',num2str(toc(CPU)),' s',])
306        if deb ==1;disp('L 261-282, End cavity, view factors, rad exch.');end
307    end % =====================================================================
```

Table 28: Matlab© procedure Fiammetta.m

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

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

Line	Occ.	Name	Meaning of the variables used in *Fiammetta.m*
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 *geo_baf.m* (*L 174*),
1	10	lon	Initialization of the vector of elements lengths on patch sides for radiative exchanges
2	2	Gi	input data of *cad_gin.m* used for the identification of an application
2	20	xyz_cao	Coordinates of the patch vertices, output of *cad_gin.m*.
2	17	car_cao	Localization of patches
2	12	nbo	Number of CAD patches interfaces
2	7	Me	Method used for the solution *Me* = 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 *cad_con.m* (*L 80*)
2	8	nvn	Number of virtual convection nodes, used in *cad_Dir.m* (*L 71*), *cad_con.m* (*L 80*), *fem_smd.m* (*L 236*)
2	7	nnr	Number of virtual radiative nodes, used in *fem_smd.m* (*L 226*)
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 *fem_smc.m* (*L 277*), *fem_smq.m* (*L 257*),
3	4	ra	Flag indicating the use of conductive radiative elements, used in *fem_smq.m* (*L 257*),
3	19	cs	View factor matrix flag: 1 = cavity, 2 = street, 3 = thermal bridge, 5 = both vertical sides are convective, used in *cad_con.m* (*L 80*), *cad_ban.m* (*L 88*), *fem_smt.m* (*L 246*), *fem_smq.m* (*L 257*), *fem_smc.m* (*L 277*),
8	13	th	Thickness, *m*, used in *cad_Neu.m*, *fem_smd.m* (*L 236*), *fem_smt.m* (*L 246*), *fem_smq.m* (*L 257*), *fem_smc.m* (*L 277*).
9	3	k	Conductivity coefficient $Wm^{-1}K^{-1}$.
10	6	pai	Temperature interval in isotherms drawing *K*
11	18	re	Coefficient of reflexion (adimensional), used in *fem_smd.m* (*L 236*), *fem_smq.m* (*L 257*), *fem_smc.m* (*L 277*).
13	3	h	Convection coefficient $Wm^{-2}K^{-1}$, used in *cad_con.m* (*L 80*)
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 *lines 22 - 30*
32	13	xyt	Nodal coordinates, computed in *cad_mes.m*, the mesh generation function

32	20	lK	Localization matrix of conductive elements (dimension: *nel* x 4), computed in *cad_mes.m*
32	23	bor	computed in *cad_edg.m*
32	5	pbo	computed in *cad_mes.m*
33	7	nel	Number of conductive elements = size (*lK*, 1)
34	11	no	Number of nodes of the mesh = size (*xyt*, 1)
34	15	dK	Number of *DOF,* dK = no + nvn + nnr
51	4	pe	Perimeter of the conductive domain, *m*. Initialized at *L 41*, and printed at *L 66*.
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 10^{-8} $Wm^{-2}K^{-4}$ used in *fem_smd.m* (*L 236*), *fem_smt.m* (*L 246*) , *fem_smc.m* (*L 277*),
71	13	lfi	Uni-column matrix: list of fixed nodes (Dirichlet), computed in *cad_Dir.m* (*L 71*), used in *fem_smd.m* (*L 236*), *fem_smt.m* (*L 246*), *fem_smq.m* (*L 257*), *fem_smc.m* (*L 277*).
71	7	fT	Uni-column matrix of fixed nodal temperatures, computed in *cad_Dir.m* (*L 71*), used in *fem_smd.m* (*L 236*), *fem_smt.m* (*L 246*), *fem_smq.m* (*L 257*), *fem_smc.m* (*L 277*).
72	8	nf	Number of fixations = number of columns of lfi, it is also the dimension of *fT* & *lfi*, computed in *cad_Dir.m* (*L 71*). 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 *cad_Neu.m*
74	2	lg	Lengths of the patch sides, used in *fem_smc.m* (*L 277*).
74	2	bos	Areas of the patch sides, used in *fem_smc.m* (*L 277*).
80	2	he	Uni-line matrix of the element convective coefficients, computed in *cad_con.m* (*L 80*)
80	7	lc	Localizations of convective elements defined in *cad_con.m*
88	14	lcont	Uni-column matrix of *mrr* radiative cavity, street and balcony nodes, computed in *cad_ban.m (L 88*). It is initialized in *L 82*.
88	13	lv	Uni-column matrix of cavity, street, balcony vertices, computed in *cad_ban.m (L 88*). It is initialized in *L 82*.
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 *nel* products of thickness by conductivity. coeff. *k*. It is also used in the procedure *P_flgr.m* and the function *gra_ahf.m* to compute the heat flow vectors. It is computed in *mat_cok.m*
100	2	Kel	Element conductive matrix WK^{-1}, computed in *fem_Kco.m*
109	2	Kec	Element "conductive – convective" matrix WK^{-1} , computed in *fem_Kcv.m.*
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 *ns* radiative edges' lengths *m,* used in geo_vfc.m (*L 136*), *geo_stf.m* (*L 146*), *fem_smq.m* (*L 257*), *fem_smc.m* (*L 277*).
118	10	lcc	Sequence of radiative CAD vertices + first repeated if cavity
126	6	Tsky	Sky temperature, used in *fem_smq.m* (*L 257*),
135	16	Fs	View factor matrix for radiative element from mesh border, computed in geo_vfc.m (*L 136*), geo_vfr.m (*L 145*), *geo_stf.m* (*L 150*),
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 *dK DOF* or nodal temperatures
196	3	gt	Uni-line matrix for isolines definition in *gra_ipa.m*
223	4	reac	Second member, reac = K * tca
269	4	Mpr	*(I - Fs)* M^{-1} matrix - flow out of a segment (adim.)

Table 30: Matlab© 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 *Fiammetta.m* with the function *cad_gin.m*

```
function[xyz_cao,car_cao,nbo,Me,Di,Ne,Co,nvn,nnr,fmd,ca] = cad_gin(Gi)
fmd = 0;ca=0;

if Gi == 8
% ....... 8. Standard rect. 1 rectangle .................................
    ha = 2;xyz_cao = [0 0;1 0;1 ha;0 ha];
           car_cao = [1 2 3 4 ];nbo=4;
    Me = 1;Di =13;Ne = 0;Co = 1;nvn = 2;nnr = 0;
    disp([ 'Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 9
% ....... 9. Standard rect. 1 rectangle .................................
    ha = 2;xyz_cao = [0 0;1 0;1 ha;0 ha];
           car_cao = [1 2 3 4 ];nbo=4;
    Me = 1;Di = 3;Ne = 0;Co = 0;nvn = 0;nnr = 0;
    disp([ 'Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi ==25
% ....... 8. Standard rect. 1 rectangle .................................
    ha = 2;xyz_cao = [0 0;1 0;1 ha;0 ha];
           car_cao = [1 2 3 4 ];nbo=4;
    Me = 1;Di =13;Ne = 0;Co = 1;nvn = 2;nnr = 0;
    disp([ 'Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi ==28
% ....... 8. Standard rect. 1 rectangle .................................
    ha = 2;xyz_cao = [0 0;1 0;1 ha;0 ha];
           car_cao = [1 2 3 4 ];nbo=4;
```

```
    Me = 1;Di =13;Ne = 0;Co = 1;nvn = 2;nnr = 0;
    disp([ 'Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 30
% ....... 9. Standard rect. 1 rectangle .................................
    ha = 2;xyz_cao = [0 0;1 0;1 ha;0 ha];
           car_cao = [1 2 3 4 ];nbo=4;fmd=1;
    Me = 3;Di = 3;Ne = 0;Co = 0;nvn = 0;nnr = 0;
    disp([ 'Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 7
% ....... 7. Standard rect. 2 squares ...................................
    xyz_cao  = [0 0;1 0;0 1;1 1;0 2;1 2];
    car_cao  = [1 2 4 3;3 4 6 5];nbo =7;
    Me = 3;Di = 7;Ne = 0;Co = 3;nvn = 2;nnr = 0;
disp([ 'Rectan. 2 squares Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 10
% ....... 7. Standard rect. 2 squares ...................................
    xyz_cao  = [0 0;1 0;0 1;1 1;0 2;1 2];
    car_cao  = [1 2 4 3;3 4 6 5];nbo =7;
    Me = 4;Di = 7;Ne = 0;Co = 4;nvn = 1;nnr = 1;
    disp(['g  72, 1 r., 2s. Gi : ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 24
% ....... 7. Standard rect. 2 squares ...................................
    xyz_cao  = [0 0;1 0;0 1;1 1;0 2;1 2];
    car_cao  = [1 2 4 3;3 4 6 5];nbo =7;
    Me = 4;Di =16;Ne = 0;Co = 4;nvn = 1;nnr = 1;
   disp([ 'Rectan. 2 squares Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 26
% ....... 7. Standard rect. 2 squares ...................................
    xyz_cao  = [0 0;1 0;0 1;1 1;0 2;1 2];
    car_cao  = [1 2 4 3;3 4 6 5];nbo =7;
    Me = 1;Di =10;Ne = 0;Co = 7;nvn = 3;nnr = 0;
disp([ 'Rectan. 2 squares Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 27
% ....... 7. Standard rect. 2 squares ...................................
    xyz_cao  = [0 0;1 0;0 1;1 1;0 2;1 2];
    car_cao  = [1 2 4 3;3 4 6 5];nbo =7;
    Me = 1;Di =19;Ne = 0;Co = 7;nvn = 3;nnr = 0;
disp([ 'Rectan. 2 squares Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 29
% ....... 7. Standard rect. 2 squares ...................................
    xyz_cao  = [0 0;1 0;0 1;1 1;0 2;1 2];
    car_cao  = [1 2 4 3;3 4 6 5];nbo =7;
    Me = 1;Di = 9;Ne =11;Co = 0;nvn = 0;nnr = 0;
    disp([ '1 rect., 2 squar. Gi : ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 31
% ....... 7. Standard rect. 2 squares ...................................
    xyz_cao  = [0 0;1 0;0 1;1 1;0 2;1 2];
    car_cao  = [1 2 4 3;3 4 6 5];nbo =7;
    Me = 3;Di =15;Ne = 0;Co = 7;nvn = 4;nnr = 0;
    disp([ 'Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 32
% ....... 7. Standard rect. 2 squares ...................................
    xyz_cao  = [0 0;1 0;0 1;1 1;0 2;1 2];
    car_cao  = [1 2 4 3;3 4 6 5];nbo =7;
    Me = 3;Di =14;Ne = 0;Co = 9;nvn = 4;nnr = 0;
    disp([ 'Standard 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 33
% ....... 7. Standard rect. 2 squares ...................................
    xyz_cao  = [0 0;1 0;0 1;1 1;0 2;1 2];
    car_cao  = [1 2 4 3;3 4 6 5];nbo =7;
    Me = 3;Di =17;Ne = 0;Co = 7;nvn = 3;nnr = 0;
    disp([ 'Rectan. 2 squares Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 23
% ...... 23. Vertical rectangle ........................................
    ha = 1;xyz_cao = [0 0;1 0;1 ha;0 ha];
           car_cao = [1 2 3 4 ];nbo=4;
    Me = 3;Di = 0;Ne = 0;Co = 0;nvn = 0;nnr = 0;fmd=1;
    disp([ 'Vertical. 1 rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 17
% ...... 17. Horizontal rectangle ........................................
```

```
    ha = 1;xyz_cao = [0 0;2 0;2 ha;0 ha];
           car_cao = [1 2 3 4 ];nbo=4;
    Me = 3;Di = 0;Ne = 0;Co = 0;nvn = 0;nnr = 0;fmd=1;
    disp([ '1 horizon. rect., Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 34
%.... .... 9. Standard trapezoidal .......................................
    ha = 1;xyz_cao = [0 0;1 0;.75 0.8;.25 ha];
           car_cao = [1 2 3 4 ];nbo = 4;
    Me = 1;Di = 7;Ne = 0;Co = 1;nvn = 2;nnr = 0;
    disp([ 'Trapezoidal shape Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 6
%.... .... 9. Standard trapezoidal .......................................
    ha = 1;xyz_cao = [0 0;1 0;.75 ha;.25 ha];
           car_cao = [1 2 3 4 ];nbo = 4;
    Me = 1;Di = 7;Ne = 0;Co = 1;nvn = 2;nnr = 0;
disp([ 'Trapezoidal shape Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 31
% ....... C shape six patches .........................................
    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
    car_cao  = [1 2 4 3;3 4 6 5;10 3 5 8;9 10 8 7;11 12 10 9]; nbo = 16;
    Me = 1;Di = 21;Ne = 0;Co = 0;nvn = 0;nnr = 0;
    disp(['C shape 3 patches Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 3
% ....... C shape three patches .........................................
    xyz_cao  = [2 2;2 3;1 2;0 3;0 0;1 1;3 0;3 1];% CAD
    car_cao  = [1 2 4 3;3 4 5 6;7 8 6 5]; nbo = 10;
    Me = 1;Di = 6;Ne = 0;Co = 0;nvn = 0;nnr = 0;
    disp(['C shape 3 patches Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 5
% ....... 5. C shape diag top left .....................................
    xyz_cao  = [3 2;3 3;0 3;1 2;0 1;1 1;2 1;0 0;1 0;2 0];
    car_cao  = [1 2 3 4;4 3 5 6;6 5 8 9; 6 9 10 7];nbo=13;
    Me = 1;Di = 5;Ne = 0;Co = 0;nvn = 0;nnr = 0;
    disp(['C shape 4 patches Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 1
% ....... 1. Standard cavity .............................................
    xyz_cao  = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
    car_cao  = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
    Me = 3;Di = 1;Ne = 0;Co = 0;nvn = 0;nnr = 0;ca = 1;
    disp([ 'Standard cavity,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 2
% ....... 1. Standard cavity .............................................
    xyz_cao  = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
    car_cao  = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
    Me = 3;Di = 1;Ne = 0;Co = 2;nvn = 1;nnr = 0;ca = 1;
    disp(['Standard cavity,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 4
% ....... 1. Standard cavity .............................................
    xyz_cao  = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
    car_cao  = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
    Me = 1;Di = 4;Ne = 0;Co = 0;nvn = 0;nnr = 0;ca = 1;
    disp([ 'Standard cavity,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 11
% ....... 1. Standard cavity .............................................
    xyz_cao   = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
    car_cao   = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
    Me = 3;Di = 1;Ne = 0;Co = 0;nvn = 0;nnr = 0;ca = 1;
    disp([ 'Standard cavity,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 39
% ....... 1. Standard cavity .............................................
    xyz_cao   = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
    car_cao   = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
    Me = 2;Di = 4;Ne = 0;Co = 0;nvn = 0;nnr = 1;ca = 1;
    disp([ 'Standard cavity,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 40
% ....... 1. Standard cavity .............................................
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];
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;...
    10 11 15 14;11 12 16 15];nbo=24;
    Me = 2;Di =20;Ne = 0;Co = 0;nvn = 0;nnr = 1;ca = 1;
    disp([ 'Standard cavity,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
```

```
end
if Gi == 41
% ....... 1. Standard cavity .............................................
    xyz_cao   = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
    car_cao   = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
    Me = 5;Di = 0;Ne = 1;Co = 0;nvn = 0;nnr = 0;ca = 1;
    disp([ 'Standard cavity,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 42
% ....... 1. Standard cavity .............................................
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];
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;...
    10 11 15 14;11 12 16 15];nbo=24;
    Me = 5;Di = 0;Ne = 2;Co = 0;nvn = 0;nnr = 0;ca = 1;
    disp([ 'Standard cavity,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 12
% ....... 12. Rectangular cavity ..radiative exchanges .................
    xyz_cao   = [0 0;3 0;3 6;0 6;1 1;2 1;2 5;1 5];
    car_cao   = [1 2 6 5;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
    Me = 5;Di = 8;Ne = 0;Co = 7;nvn = 2;nnr = 0;ca = 1;
    disp([ 'Rectang. cavity,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
    % cs : 1 = cavity: 2 = str.
end
if Gi == 13
% ....... 13. Rectangular cavity ..transient linear ......................
    xyz_cao   = [0 0;3 0;3 6;0 6;1 1;2 1;2 5;1 5];
    car_cao   = [1 2 6 5;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
    Me = 3;Di = 8;Ne = 0;Co = 7;nvn = 2;nnr = 0;ca = 1;
    disp([ 'Rectang. cavity,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 14
% ....... 14. Street section       ......................................
    xyz_cao  = [0 8;1 8;0 0;1 1;5 0;4 1;5 8;4 8];nbo = 10;
    car_cao  = [2 1 3 4;4 3 5 6; 6 5 7 8];
    Me = 5;Di = 0;Ne =  0;Co = 0;nvn = 0;nnr = 0;
disp(['Street section,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 15
% ....... 15. Street section       ......................................
    xyz_cao  = [0 8;1 8;0 0;1 1;5 0;4 1;5 8;4 8];nbo = 10;
    car_cao  = [2 1 3 4;4 3 5 6; 6 5 7 8];
    Me = 3;Di = 0;Ne = 111;Co = 0;nvn = 0;nnr = 0;
disp(['Street section,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 35
% ....... 14. Street section       ......................................
    xyz_cao  = [0 8;1 8;0 0;1 1;5 0;4 1;5 8;4 8];nbo = 10;
    car_cao  = [2 1 3 4;4 3 5 6; 6 5 7 8];
    Me = 5;Di = 0;Ne =111;Co = 0;nvn = 0;nnr = 0;
    disp(['Street section,  Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 16
% ....... 16. Thermal bridge ........................................
    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];
    car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
    Me = 2;Di = 2;Ne = 0;Co = 5;nvn = 1;nnr = 1;
    disp([ 'Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 18
% ....... 18. Thermal bridge ........................................
    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];
    car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
    Me = 1;Di = 2;Ne = 0;Co = 5;nvn = 2;nnr = 0;
    disp([ 'Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 19
% ....... 19. Thermal bridge ........................................
    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];
    car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
    Me = 1;Di =11;Ne = 0;Co = 5;nvn = 2;nnr = 0;
    disp([ 'Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 20
% ....... 19. Thermal bridge ........................................
    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];
    car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
    Me = 3;Di =7;Ne = 0;Co = 5;nvn = 2;nnr = 0;
    disp([ 'Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 21
% ....... 16. Thermal bridge ........................................
```

```
    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];
    car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
    Me = 4;Di = 2;Ne = 0;Co = 6;nvn = 1;nnr = 1;
    disp([ 'Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 22
% ....... 16. Thermal bridge ........................................
    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];
    car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
    Me = 4;Di = 12;Ne = 0;Co = 6;nvn = 1;nnr = 0;
    disp([ 'Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 36
% ....... 16. Thermal bridge ........................................
    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];
    car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
    Me =4;Di = 16;Ne = 0;Co = 5;nvn = 1;nnr = 1;
    disp([ 'Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 37
% ....... 16. Thermal bridge ........................................
    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];
    car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
    Me = 4;Di =16;Ne = 0;Co = 5;nvn = 1;nnr = 1;
    disp([ 'Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 38
% ....... 16. Thermal bridge ........................................
    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];
    car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
    Me =2;Di = 22;Ne = 0;Co = 8;nvn = 1;nnr = 1;
    disp([ 'Thermal bridge, Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 43
% ....... C shape six patches .........................................
    xyz_cao  = [0 0;0 1;0 2;0 3;1 0;1 1;1 2;1 3;2 0;2 1;2 2;2 3;3 2;3 3];
    car_cao  = [1 5 6 2;2 6 7 3;3 7 8 4;5 9 10 6;7 11 12 8 ;11 13 14 12];
    nbo = 19;Me = 1;Di = 23;Ne = 0;Co = 0;nvn = 0;nnr = 0;
    disp(['C shape 6 patches Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 44
% ....... C shape three patches ........................................
    xyz_cao  = [3 2;3 3;1 2;0 3;0 0;1 1;2 0;2 1];% CAD
    car_cao  = [1 2 4 3;3 4 5 6;7 8 6 5]; nbo = 10;
    Me = 1;Di = 6;Ne = 0;Co = 0;nvn = 0;nnr = 0;
    disp(['C shape 3 patches Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
if Gi == 45
% ....... C shape three patches ........................................
    xyz_cao  = [3 2;3 3;1 2;0 3;0 0;1 1;2 0;2 1];% CAD
    car_cao  = [1 2 4 3;3 4 5 6;7 8 6 5]; nbo = 10;
    Me = 1;Di = 6;Ne = 0;Co = 0;nvn = 0;nnr = 0;
    disp(['C shape 3 patches Gi: ',num2str(Gi),', Di : ',num2str(Di)])
end
end
```

Table 31: Matlab© 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 54*.

Dirichlet boundary conditions inserted with the function *cad_Dir.m*

```
function [lfi,fT] = cad_Dir(Di,no,nvn,car_cao,bor,pbo,nni)
% disp(['LD  2, num. nod side: ',num2str(nni)])
% General data .........................................................
% nnc  = 5 ;                % Number of fixed nodes on the horizontal sides
% disp(['L  25, N.fix h.-side: ',num2str(nnc)])
if Di == 0
    lfi(1)=0;fT(1)=0;nf=0;disp(['LD  7, Numb. fix nod: ',num2str(nf)])
end
if Di == 1                     % lfi = list of DOF on the top of the cavity
    lfi = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
    nf  = size(lfi,2);fT = ones(1,nf)*300;
    disp(['LD 12, Fixed nodes  : ',num2str(lfi)])
```

```
end
if Di == 2
    fT =[300 280];lfi=[no+1 no+2];% nf=2;  % Fixation of two virtual nodes
    disp(['LD 16, Fixed nodes  : ',num2str(lfi)])
    disp(['LD 17, Fix. temper. : ',num2str(fT),' K']);
end
if Di == 22
    fT =[270 300];lfi=[no+1 no+2];% nf=2;  % Fixation of two virtual nodes
    disp(['LD 21, Fixed nodes  : ',num2str(lfi)])
    disp(['LD 22, Fix. temper. : ',num2str(fT),' K']);
end
if Di == 3
    nnc =  5;        % nnc = number of fixed nodes on the horizontal sides
    if nni < nnc;nnc = nni;end
    a   = [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
    b   = [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
    nx  = min(nnc,nni+3);lfi=[b(nni+3-nx:nni+2) a(nni+3-nx:nni+2)];
    nf  = size(lfi,2);fT = [ones(1,nf/2)*270 ones(1,nf/2)*320];
    disp(['LD 23, N. fix. nodes: ',num2str(nf)])
end
if Di == 4
    lfi = [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)...
           car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
    nf = size(lfi,2);fT = [ones(1,nf/2)*270 ones(1,nf/2)*300];
    disp(['LD 36, N. fix. nodes: ',num2str(nf)])
    if nf < 20;disp(['LD 37, Fix. DOF, lfi: ',num2str(lfi)]);end
end
if Di == 20
    lf  = [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)...
           car_cao(2,1) bor(pbo(2,1),5):bor(pbo(2,1),6) car_cao(2,2)...
           car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)...
           car_cao(6,3) bor(pbo(6,3),5):bor(pbo(6,3),6) car_cao(6,4)...
           car_cao(7,3) bor(pbo(7,3),5):bor(pbo(7,3),6) car_cao(7,4)...
           car_cao(8,3) bor(pbo(8,3),5):bor(pbo(8,3),6) car_cao(8,4)];
     nt = size(lf,2);lfi = zeros(1,nt);k = 3;lfi(1:3)=lf(1:3);
     for i  = 4 : nt;n  = 0 ;
         for j = 1 : i-1;if lf(j) == lf(i);n = 1;end;end
         if n > 0;lf(i)=0;else;k=k+1;lfi(k)=lf(i);end
     end;nf = k;fT = [ones(1,nf/2)*270 ones(1,nf/2)*300];
    disp(['LD 52, N. fix. nodes: ',num2str(nf)])
    if nf < 20;disp(['LD 53, Fix. DOF, lfi: ',num2str(lfi(1:nf))]);end
end
if Di == 5
    lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)...
         car_cao(4,3) bor(pbo(4,3),5):bor(pbo(4,3),6) car_cao(4,4)];
    nf = size(lfi,2);
    if nf > 2 ;fT = [ones(1,nf/2)*300 ones(1,nf/2)*310 ];end
%     if nf > 2 ;fT = [ones(1,nf/2)*300 ones(1,nf/2)*270 ];end
end
if Di == 6
    lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)...
         car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)];
%     lfi=[car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)...
%          car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)];
    nf = size(lfi,2);
    if nf > 2 ;fT =[ones(1,nf/2)*300 ones(1,nf/2)*310];end
    if nf < 15;disp(['LD 69, Fixed nodes  : ',num2str(lfi)]);end
%     if nf > 2 ;fT = [ones(1,nf/2)*270 ones(1,nf/2)*300 ];end
%     if nf > 2 ;fT = [ones(1,nf/2)*300 ones(1,nf/2)*270 ];end% corps noir
end
if Di == 21
    lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)...
         car_cao(5,1) bor(pbo(5,1),5):bor(pbo(5,1),6) car_cao(5,2)];
    nf = size(lfi,2);
    if nf > 2 ;fT = [ones(1,nf/2)*270 ones(1,nf/2)*300 ];end% corps noir
end
if Di == 7
    fT =[300 280];lfi=[no+1 no+2];% nf=2;       % Fix. of both virt. nodes
end
if Di == 8                            % Fixation of low horizontal cavity side
    lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
    nf = size(lfi,2);fT = ones(1,nf)*270;  % Dirichlet boundary conditions
    lfi(1,nf+1:nf+2)=[no+1 no+2];fT(1,nf+1:nf+2)=[300 300];nf=nf+2;
    disp(['LD 61, N. fix. nodes: ',num2str(nf)])
    disp(['LD 62, Fix. DOF, lfi: ',num2str(lfi)])
    disp(['LD 63, Imposed temp.: ',num2str(fT),' K'])
end
if Di ==18                            % Fixation of low horizontal cavity side
    lfi=[car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
    nf = size(lfi,2);fT = ones(1,nf)*270;  % Dirichlet boundary conditions
    lfi(1,nf+1:nf+2)=[no+1 no+2];fT(1,nf+1:nf+2)=[300 300];nf=nf+2;
    disp(['LD 67, N. fix. nodes: ',num2str(nf)])
```

```
    disp(['LD 68, Imposed temp.: ',num2str(fT),' K'])
    disp(['LD 69, Fix. DOF, lfi: ',num2str(lfi)])
    disp(['LD 70, Av. imp. temp: ',num2str(mean(fT)),' K'])
end
if Di == 16
    fT =[280 300];lfi=[no+1 no+2];% nf=2;          % Fix. of 2 virt. nodes
    disp(['LD 76, Fixed nodes  : ',num2str(lfi)])
    disp(['LD 77, Fix. temper. : ',num2str(fT),' K']);
end
if Di == 9
    lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
    fT = ones(size(lfi,2),1)*273;
    if(size(lfi,2)) < 10;disp(['LD 62, Fixed nodes  : ',num2str(lfi)]);end
    if(size(lfi,2)) < 10;disp(['LD 63, Fixed temp.  : ',num2str(fT')]);end
    disp(['LD 99, Numb. of fix.: ',num2str(size(lfi,2))])
end
if Di == 10
    fT =[270 300 280 ];lfi=[no+1 no+2 no+3];% nf=3;% Fix. of 3 virt. nodes
    disp(['LD 68, Fixed nodes  : ',num2str(lfi)])
    disp(['LD 69, Fix. temper. : ',num2str(fT),' K']);
end
if Di == 17           % Dirichlet= fixation of 3 virt. nodes and the base
    lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2) ...
        no+1 no+2 no+3];
    nf = size(lfi,2);fT = [ones(1,nf-3)*280 300 290 300] ;
    if nf < 10
        disp(['LD 76, Fixed nodes  : ',num2str(lfi)])
        disp(['LD 77, Fix. temper. : ',num2str(fT),' K']);
    else
        disp(['LD 79, Fix. conv nod: ',num2str(lfi(nf-2:nf))])
        disp(['LD 80, Fix. conv nod: ',num2str(fT (nf-2:nf))])
    end
end
if Di == 11
    fT =[300 280];lfi=[no+1 no+2];% nf=2;  % Fixation of two virtual nodes
    disp(['LD 73, Fixed nodes  : ',num2str(lfi)])
    disp(['LD 74, Fix. temper. : ',num2str(fT),' K']);
end
if Di == 12
    fT =300;lfi=no+1 ;% nf=1;              % Fixation of one virtual nodes
    disp(['LD110, Fixed nodes  : ',num2str(lfi)])
    disp(['LD111, Fix. temper. : ',num2str(fT),' K']);
end
if Di == 13
    fT =[300 270];lfi=[no+1 no+2];% nf=2;  % Fixation of two virtual nodes
    disp(['LD 83, Fixed nodes  : ',num2str(lfi)])
    disp(['LD 84, Fix. temper. : ',num2str(fT),' K']);
end
if Di == 14
    fT=[300 300 300 300];lfi=[no+1 no+2 no+3 no+4];%nf=4;%Fix.4 virt. nod.
    disp(['LD 88, Fix. nod. lfi: ',num2str(lfi)])
    disp(['LD 89, Fix. temp. fT: ',num2str(fT),' K']);
end
if Di == 15
    fT=[280 280 280 280];lfi=[no+1 no+2 no+3 no+4];%nf=4;%Fix.4 virt. nod.
    disp(['LD 93, Fix. nod. lfi: ',num2str(lfi)])
    disp(['LD 94, Fix. temp. fT: ',num2str(fT),' K']);
end
if Di == 19                    % lfi = list of DOF on the top of the cavity
    lfi = [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)...
        no+1 no+2 no+3];
    nf  = size(lfi,2);fT = [ones(1,nf-3)*270 280 300 290];
end
% if nfi < 7;disp(['L  73, fix. top side: ',num2str(lfi)]);end
% bc = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
%       [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
%       [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
%       [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)]];
% disp(['L  78, n.fix. cavity: ',num2str(size(bc))])
% fT      = zeros(1,no); lfi = zeros(1,no);
if Di == 23
    lfi=[car_cao(6,2) bor(pbo(6,2),5):bor(pbo(6,2),6) car_cao(6,3)...
         car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];
    nf = size(lfi,2);
    if nf > 2 ;fT = [ones(1,nf/2)*300 ones(1,nf/2)*270 ];end% corps noir
end
if Di > 23
if nvn == 0                   % lfi = uni-line matrix, fT = uni-line matrix
    lfi = [[car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)]';
           [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)]']';
    nf = size(lfi,2)/2;fT=[ones(nf,1)*300 ;ones(nf,1)*270];
%     mfi = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
```

```
end
if nvn == 1
%     lfi = [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)]';
%     nfi=size(lfi,2);fT=ones(1,nfi)*280;
% % DOF of the 1. Standard cavity
% bc = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
%        [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
%        [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
%        [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)]];
end
% if nvn == 2               % Dirichlet boundary conditions for convection
%     fT =[270 300];lfi=[no+1 no+2];         % Fixation of 2 virtual nodes
% end
if nvn == 3
    fT=[270 285 300];lfi=[no+1 no+2 no+3];   % Fixation of 3 virtual nodes
end
nfi = size(lfi,2);disp(['LD186, N. fix. nodes: ',num2str(nfi)])
if nfi < 15;      disp(['LD187, Fixed nodes  : ',num2str(lfi)]);end
% if nfi > 0
%     if nfi < 15
%         disp(['LD109, Numb. fix. N.: ',num2str(nf)])
%         disp(['LD110, Fixed nodes  : ',num2str(lfi)])
%         disp(['LD111, Fix. temper. : ',num2str(fT),' K']);
%     end
% end
end
end
```

Table 32: Matlab© 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 *cad_Neu.m*

```
function [gh,lg,bos] = cad_Neu(Ne,dK,car_cao,pbo,bor,th,xyz_cao)
%........................................ Neumann boundary conditions
disp(['N  03, param Ne     : ',num2str(Ne)]); bos=0;lg=zeros(1,1);
if Ne == 0
    gh  = zeros(dK,1); bos=0;  % 2d member initialization always necessary
end
if Ne == 1
    gh = zeros(dK,1);                       % second member initialization
    lg = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
    nh = size(lg,2);np=nh-1;g(1)=1/(2*np);g(nh)=g(1);g(2:nh-1)=1/np;
    gh(lg(1:nh)) = g(1:nh)/sum(g);
    bos = (xyz_cao(2,1)-xyz_cao(1,1))*th;
    disp(['N  13, Loaded  area : ',num2str(bos),' m2'])
end
if Ne == 2
    gh = zeros(dK,1);                       % second member initialization
    lg =[ [car_cao(8,3) bor(pbo(8,3),5):bor(pbo(8,3),6) car_cao(8,4)]...
          [car_cao(7,3) bor(pbo(7,3),5):bor(pbo(7,3),6) car_cao(7,4)]...
          [car_cao(6,3) bor(pbo(6,3),5):bor(pbo(6,3),6) car_cao(6,4)]];
    ni=size(lg,2)/3;lg(ni)=0;lg(2*ni)=0;
    k=0;for i = 1:size(lg,2);if lg(i)> 0;k=k+1;lg(k)=lg(i);end;end
    nh = size(lg,2)-2;np=nh-1;g(1)=1/(2*np);g(nh)=g(1);g(2:nh-1)=1/np;
    gh(lg(1:nh)) = g(1:nh)/sum(g);
    bos = (xyz_cao(4,1)-xyz_cao(1,1))*th;
    disp(['N  25, Loaded  area : ',num2str(bos),' m2'])
end
if Ne == 11                 % Used with Gi = 29 & Di= 9, Gi = 14 & Di= 0
    gh  = zeros(dK,1);                      % second member initialization
    lg0=([[car_cao(2, 4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)]...
    [car_cao(1, 4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]]);
    nh  = size(lg0,2)/2;                 % Loaded nodes lg for CAD model 11
    lg  = [lg0(1:nh) lg0(nh+2:2*nh)];nh=2*nh-1;
    for i           = 1:nh;   gh(lg(1,i),1) = 2; end % Weights right side
    gh(lg(1,1),1)     = 1   ; gh(lg(1,nh),1)    = 1;
    gh = gh*25/sum(gh);                                % Total load = 25 W
    disp(['N  36, N. load. nod.: ',num2str(size(lg,2))])
    disp(['N  37, Total load   : ',num2str(sum(gh)),' W'])
end
if Ne == 111          % Used with Gi=15 Di= 0 or Gi=29 Di= 9 or Gi=35 Di= 0
    gh  = zeros(dK,1);                      % second member initialization
    lg  = [car_cao(3, 3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)...
          car_cao(1, 1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
    nh  = size(lg,2)/2;                  % Loaded nodes lg for CAD model 11
    for i           = 1:nh;   gh(lg(1,i),1) = 2; end % Weights right side
    gh(lg(1,1),1)     = 1   ; gh(lg(1,nh),1)    = 1;
```

```
    gh(lg(1:nh))      = gh(lg(1:nh))/sum(gh(lg(1:nh)));
    for i             = nh+1:2*nh;gh(lg(1,i),1) = 2; end% Weights left side
    gh(lg(1,nh+1),1)  = 1    ; gh(lg(1,2*nh),1) = 1;
    gh(lg(nh+1:2*nh))= gh(lg(nh+1:2*nh))/sum(gh(lg(nh+1:2*nh)));nh = 2*nh;
    gh=gh/2;
    disp(['N  51, Total load   : ',num2str(sum(gh)),' W'])
    bos = (xyz_cao(2,1)-xyz_cao(1,1))*th*2;
    disp(['N  53, Loaded  area : ',num2str(bos),' m2'])
end
if Ne == 0
    gh  = zeros(dK,1);%bos=0;  % 2d member initialization always necessary
else
    if nh < 10
        disp(['N  59, Loaded nodes : ',num2str(lg(1:nh))])
        disp(['N  60, Loads weights: ',num2str(gh(lg(1:nh))')])
    else
        disp(['N  62, N. load. nod.: ',num2str(size(lg,2))])
    end
end
end
```

Table 33: Matlab© 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© function *cad_con.m* – convection boundary conditions

```
function [lc,vc,hv]=cad_con(car_cao,bor,pbo,h,dK,nes,deb,nvn,cs,Co,xyz)
npa = size(car_cao,1);% npa = number patches ; nes number of elem per side
disp(['c. 03, dK = no + nvn: ',num2str(dK)])          % nes = n. elem/side
disp(['c. 04, N.virt c.nod.: ',num2str(nvn)]);
disp(['c. 05, Variable Co  : ',num2str(Co)]);
if Co == 1                     % Convective elements on 2 sides of patch 1
    bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
          [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
    cps = [car_cao(1,2) car_cao(1,3) car_cao(1,4) car_cao(1,1)];
    vc(1,1)= max (xyz(:,1))*2 ; vc(2,1)=-max (xyz(:,1));
    vc(1,2)= max (xyz(:,2))/2 ; vc(2,2)= vc(1,2);
    disp(['c. 12, Convect. sid.: ',num2str(cps)]);
end
if Co == 2  % Convective elements on the four internal sides of the cavity
    bt = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
          [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
          [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
          [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)]];
    cps = [car_cao(1,1) car_cao(1,2) car_cao(2,4) car_cao(2,1)...
           car_cao(3,4) car_cao(3,1) car_cao(4,2) car_cao(4,3)];
    vc(1,1)= mean (xyz(1:dK-nvn,1)) ; vc(1,2)= mean (xyz(1:dK-nvn,2)) ;
    disp(['c. 22, Convect. sid.: ',num2str(cps)]);
end
if Co == 3
    bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
          [car_cao(2,2) bor(pbo(2,2),5):bor(pbo(2,2),6) car_cao(2,3)];
          [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
          [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
    cps = [car_cao(1,2) car_cao(1,3) car_cao(2,2) car_cao(2,3)...
           car_cao(2,4) car_cao(2,1) car_cao(1,4) car_cao(1,1)];
    vc(1,1)= max (xyz(:,1))*2 ; vc(2,1)=-max (xyz(:,1));
    vc(1,2)= max (xyz(:,2))/2 ; vc(2,2)= vc(1,2);
    disp(['c. 33, Convect. sid.: ',num2str(cps)]);
end
if Co == 4
    bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
          [car_cao(2,2) bor(pbo(2,2),5):bor(pbo(2,2),6) car_cao(2,3)];];
    cps = [car_cao(1,2) car_cao(1,3) car_cao(2,2) car_cao(2,3)];
    vc(1,1)= max (xyz(:,1))*2 ;vc(1,2)= max (xyz(:,2))/2;
    vc(2,1)= vc(1,1)          ;vc(2,2)= vc(1,2);
    disp(['c. 41, Convect. sid.: ',num2str(cps)]);
end
if Co == 5             % Two external vertical sides of a four patches cavity
    bt = [[car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)];
          [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)]];
    cps = [car_cao(3,3) car_cao(3,4) car_cao(1,1) car_cao(1,2)];
    vc(1,1)= (xyz(6,1)+xyz(8,1))/2; vc(1,2)=(xyz(6,2)+xyz(12,2))/2;
```

```
    vc(2,1)= (xyz(6,1)+xyz(8,1))/2; vc(2,2)=(xyz(6,2)+xyz(12,2))/2;
    disp(['c. 48, Convect. sid.: ',num2str(cps)]);
end
if Co == 6      % Two horizontal sides on both sides of the thermal bridge
    bt = [[car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)];
          [car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)]];
    cps = [car_cao(3,3) car_cao(3,4) car_cao(3,1) car_cao(3,2)];
    vc(1,1)= (xyz(6,1)+xyz(8,1))/2; vc(1,2)=(xyz(6,2)+xyz(12,2))/2;
    vc(2,1)= vc(1,1);vc(2,2)=vc(1,2);
%     vc(2,1)= (xyz(1,1)+xyz(3,1))/2; vc(2,2)=(xyz(3,2)+xyz(9,2))/2;
    disp(['c. 51, Convect. sid.: ',num2str(cps)]);
end
if Co == 7          % Two external vertical sides of a four patches cavity
    bt = [[car_cao(2,2) bor(pbo(2,2),5):bor(pbo(2,2),6) car_cao(2,3)];
          [car_cao(4,4) bor(pbo(4,4),5):bor(pbo(4,4),6) car_cao(4,1)]];
    cps = [car_cao(2,2) car_cao(2,3) car_cao(4,4) car_cao(4,1)];
    vc(1,1)= max (xyz(:,1))*2 ; vc(2,1)=-max (xyz(:,1));
    vc(1,2)= max (xyz(:,2))/2 ; vc(2,2)= vc(1,2);
    disp(['c. 65, Convect. sid.: ',num2str(cps)]);
end
if Co == 8          % Two external vertical sides of a four patches cavity
    bt = [[car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)];
          [car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)]];
    cps = [car_cao(3,3) car_cao(3,4) car_cao(3,1) car_cao(3,2)];
    vc(1,1)= (xyz(6,1)+xyz(8,1))/2; vc(1,2)=(xyz(6,2)+xyz(12,2))/2;
    vc(2,1)= (xyz(5,1)+xyz(7,1))/2; vc(2,2)=(xyz(5,2)+xyz(10,2))/2;
    disp(['c. 73, Convect. sid.: ',num2str(cps)]);
end
if Co > 8
    if npa == 1                     % The domain contains a single patch
        if nvn == 2                                 % 2 convective sides
        bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
             [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
        cps = [car_cao(1,2) car_cao(1,3) car_cao(1,4) car_cao(1,1)];
        disp(['c. 81, Convect. sid.: ',num2str(cps)]);
        end
        if nvn == 3                                 % 3 convective sides
        bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
              [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)]];
        cps = [car_cao(1,2) car_cao(1,3) car_cao(1,3) car_cao(1,4)...
               car_cao(1,4) car_cao(1,1)];
        disp(['c. 89, Convect. sid.: ',num2str(cps)]);
        end
    end
    if npa == 2                       % The domain contains two patches
         if nvn == 3                                % 3 convective sides
       bt =[[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
            [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(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
            [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)];];
        cps = [car_cao(1,2) car_cao(1,3) car_cao(2,2) car_cao(2,3)...
               car_cao(2,3) car_cao(2,4) car_cao(2,4) car_cao(2,1)...
               car_cao(1,4) car_cao(1,1)];
        disp(['c.102, Convect. sid.: ',num2str(cps)]);
         end
         if nvn == 4        % 4 convective sides on the rect. with 2 patches
       bt = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
             [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(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
             [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)]];
         end
         cps = [car_cao(1,2) car_cao(1,3) car_cao(2,2) car_cao(2,3)...
                car_cao(2,3) car_cao(2,4) car_cao(2,4) car_cao(2,1)...
                car_cao(1,4) car_cao(1,1) car_cao(1,1) car_cao(1,2)];
        disp(['c.115, Convect. sid.: ',num2str(cps)]);
        vc(1,1)= max(xyz(:,1))*1.5;vc(1,2)=max (xyz(:,2))/2 ;
        vc(2,1)= max (xyz(:,1))/2 ;vc(2,2)=max (xyz(:,2))+max(xyz(:,1))/2;
        vc(3,1)= -max (xyz(:,1))/2;vc(3,2)= max (xyz(:,2))/2;
        vc(4,1)= max (xyz(:,1))/2 ;vc(4,2)= -max(xyz(:,1))/2;
    end
    if cs~=5
        disp(['c.122, cad_con cs   : ',num2str(cs)]);
    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*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 a*h];
                disp(['c.178, Conv. coeff. : ',num2str(hw),'  W/(m2K)'])
            end
        end
    end
    end
end
nec = (nes)*size(bt,1);                % There are nec convective elements
hv  = ones(1,nec)*h;
disp(['c.188, Nu. conv. el.: ',num2str(nec)])
if nec < 10;disp(['c.189, 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
        k1 = [1       npa*nes+1  ];
        k2 = [npa*nes (2*npa)*nes];
    end
    if nvn == 3                                       % 3 convective sides
        k1 = [1       2*nco/5+1 3*nco/5+1];
        k2 = [2*nco/5 3*nco/5   nco      ];
    end
```

```
    if nvn == 4                                          % 4 convective sides
        k1 = [1       2*nco/6+1 3*nco/6+1 5*nco/6+1];
        k2 = [2*nco/6 3*nco/6   5*nco/6   nco      ];
    end
    for i          = 1:nvn  % if nvn > 1, generation of the conv V. nodes
        for j      = k1(i):k2(i)
            lc(j,3) = dK + i - nvn;
        end
    end
end
if deb==1
    disp(['c.222, Co. virt. nod: ',num2str(dK-nvn+1:dK)])
end
end
```

Table 34: Matlab© function cad_con.m - localization of convection elements

Matlab© function *cad_mes.m*

```
function [xyc,lK,bor,pbo] = cad_mes(xyc,lca,ni,nbo)
nec         = size(lca,1);                          % Number of CAD patches
ndo         = size(xyc,1);nd = ndo;                   % Number of CAD nodes
[bor, pbs] = cad_edg(lca,nbo); % Compute the nbo patch sides in bor matrix
% =======================================================================
if ni                 == 1    % ni = 1 : one node generated on the interfaces
    lai                = zeros(1,nbo);                 % List of border edges
    for i              = 1: nbo
        lai(i)         = nd + i;
        bor(i,5)       = lai(i);
        bor(i,6)       = bor(i,5);
        xyc(nd+i,:)    = (xyc(bor(i,1),:)+xyc(bor(i,2),:))/2;
    end
else                 % More than 1 node have to be generated on the interfaces
    k                  = nd;
    for i              = 1 : nbo           % nbo is the number of interfaces
        bor(i,5)       = nd + (i-1)*ni+1;
        bor(i,6)       = nd + i*ni;
        for j          = 1 : ni
            A          = xyc(bor(i,1),:);B = xyc(bor(i,2),:);
            k          = k + 1;
            xyc(k,:) = A + (B-A)*j/(ni+1);   % coord. of the inerface nodes
        end
    end
end
pbo = sign(pbs(:,:)).*pbs;
nna                 = ndo + nbo*ni;   % Numb. of nodes after interfaces gen.
lK                  = zeros(nec*(ni+1)^2,4);nu = 0;
for n               = 1:nec % Loop on the CAD patches ......................
    to              = zeros(ni+2,ni+2);     % Gen. to = list of nodes matrix
    to(1,1)         = lca(n,1);            % Start with the 4 patch vertices
    to(1,ni+2)      = lca(n,2); to(ni+2,ni+2) = lca(n,3)
    to(ni+2,1)      = lca(n,4);                         % End patch vertices
    if bor(pbo(n,1),3) == n                     % Line 1 of patch matrix to
        to(1   ,2:ni+1) = bor(pbo(n,1),5)    :bor(pbo(n,1),6);
    else
        to(1   ,2:ni+1) = bor(pbo(n,1),6):-1:bor(pbo(n,1),5);
    end
    if bor(pbo(n,3),3) == n                  % Line ni+2 of patch matrix to
        to(ni+2,2:ni+1) = bor(pbo(n,3),6):-1:bor(pbo(n,3),5);
    else
        to(ni+2,2:ni+1) = bor(pbo(n,3),5)    :bor(pbo(n,3),6);
    end
    if bor(pbo(n,4),3) == n            % Side 4 = first column of matrix to
        to(2:ni+1,1)    = bor(pbo(n,4),6):-1:bor(pbo(n,4),5);
    else
        to(2:ni+1,1)    = bor(pbo(n,4),5)    :bor(pbo(n,4),6);
    end
    if bor(pbo(n,2),3) == n             % Side 2 = column ni+2 of matrix to
        to(2:ni+1,ni+2) = bor(pbo(n,2),5)   :bor(pbo(n,2),6);
    else
        to(2:ni+1,ni+2) = bor(pbo(n,2),6):-1:bor(pbo(n,2),5);
    end
    % Generation of internal nodes if ni > 0 ............................
    x1 = xyc(to(1,1),:)    ;x2 = xyc(to(1,ni+2),:);          % Patch vertices
    x3 = xyc(to(ni+2,1),:);x4 = xyc(to(ni+2,ni+2),:);
    for k                  =  1 : ni     %  ni x ni new nodes inside the patch
        for j              = 1: ni
            s              = k/(ni+1);t=j/(ni+1);% disp([s t])
            nna            = nna+1;
            to(j+1,k+1)    = nna;                         % Interior of the patch
```

```
                xyc(nna,:)  = x1*(1-s)*(1-t)+x2*s*(1-t)+x3*(1-s)*t+x4*s*t;
            end
        end
        % End of generation of the internal nodes ...........................
        for i = 1:ni+1% Mesh generation: (ni+1)(ni+1) elements per patch .....
            for j        = 1:ni+1
                nu       = nu+1;
                lK(nu,:) = [to(i,j) to(i+1,j) to(i+1,j+1) to(i,j+1)];
            end
        end % End of mesh generation .........................................
    end % End of loop on the CAD patches.....................................
    end
```

Table 35: Matlab© function cad_mes.m - construction of the CAD mesh topology

Matlab© function *cad_edg.m*

```
function [bor,pbo] = cad_edg(lca,nbo)
nec = size(lca,1);                                  % nec = number of elements
bor = zeros(nbo,8);boe=zeros(nec,8);pbo = zeros(nec,4);% bor = patch sides
for ie = 1:nec                                    % Loop ie on the nec patches
    boe(1,1) = lca (ie,1);boe(1,2) = lca (ie,2);boe(1,3) = ie;
    boe(2,1) = lca (ie,2);boe(2,2) = lca (ie,3);boe(2,3) = ie;
    boe(3,1) = lca (ie,3);boe(3,2) = lca (ie,4);boe(3,3) = ie;
    boe(4,1) = lca (ie,4);boe(4,2) = lca (ie,1);boe(4,3) = ie;
    if ie == 1
        bor(1:4,:) = boe(1:4,:);
        pbo(1,1:4) = 1:4;
        nbo        = 4;
    else
        for i      = 1 : 4                             % Loop on the new sides
            flag   = 0;
            for kl = 1:nbo                    % Loop on the yet detected sides
                if flag == 0
                    if boe(i,2)        == bor(kl,1)
                        if boe(i,1)    == bor(kl,2)
                            bor(kl,4)  = ie;          % Side detected before
                            pbo(ie,i)  = -kl;   % 2d occurence of a bor line
                            flag       = 1;
                        end
                    end
                end
            end
            if flag         == 0
                nbo         = nbo +1;
                bor(nbo,:)  = boe(i,:);
                pbo(ie,i)   = nbo;
            end;
        end
    end
end
end
```

Table 36: Matlab© function cad_edg.m – definition of the CAD mesh interfaces

Matlab© function *cad_ban.m* – Radiative nodes of cavity, street or balcony

```
function[lic,lv] = cad_ban(car_cao,bor,pbo,nni,cs)                 % 20210929
% For the top of the cavity, see Di = 1 in cad_Dir.m
if cs ==1                                                   % DOF on four sides
   lcont = [[car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
            [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];
            [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
            [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)]];
lv =[lcont(1,1) lcont(1,nni+2) lcont(2,nni+2) lcont(3,nni+2)];
lic=[lcont(1,1:nni+2) lcont(2,2:nni+2) lcont(3,2:nni+2) lcont(4,2:nni+1)];
end
if cs ==6                  % DOF on the four sides of the quadrilateral cavity
    lcont = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
             [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];
             [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
             [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)]];
lv =[lcont(1,1) lcont(1,nni+2) lcont(2,nni+2) lcont(3,nni+2)];
lic=[lcont(1,1:nni+2) lcont(2,2:nni+2) lcont(3,2:nni+2) lcont(4,2:nni+1)];
end
if cs ==7                 % DOF on the four sides of the square cavity Gi = 40
    lcont = [[car_cao(2,3) bor(pbo(2,3),5):bor(pbo(2,3),6) car_cao(2,4)];
             [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];
             [car_cao(7,1) bor(pbo(7,1),5):bor(pbo(7,1),6) car_cao(7,2)];
             [car_cao(5,4) bor(pbo(5,4),5):bor(pbo(5,4),6) car_cao(5,1)]];
```

```
lv =[lcont(1,1) lcont(1,nni+2) lcont(2,nni+2) lcont(3,nni+2)];
lic=[lcont(1,1:nni+2) lcont(2,2:nni+2) lcont(3,2:nni+2) lcont(4,2:nni+1)];
end
if cs ==2 % ....... 11. Street section - 3 patches  .....................
    c1   = [car_cao(1,1) bor(pbo(1,4),6):-1:bor(pbo(1,4),5) car_cao(1,4)];
    c2   = [car_cao(2,1) bor(pbo(2,4),6):-1:bor(pbo(2,4),5) car_cao(2,4)];
    c3   = [car_cao(3,1) bor(pbo(3,4),6):-1:bor(pbo(3,4),5) car_cao(3,4)];
    lic  = [c1 c2(2:size(c2,2)-1) c3]';    % List of street boundary nodes
    lv    = [lic(1) lic(nni+2) lic(2*nni+3) lic(3*nni+4)];% Vert.
end
if cs ==3 % List of balcony boundary nodes...............................
    bt    = [[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)]];
    lic = [bt(1,1:nni+2) bt(2,2:nni+2) bt(3,2:nni+2) bt(4,2:nni+2)...
             bt(5,2:nni+2)];    % DOF concerned by radiative heat transfer
    lv    = [lic(1) lic(nni+2) lic(2*nni+3) lic(3*nni+4)...
         lic(4*nni+5) lic(5*nni+6)]';                        % Vertices
end
if cs == 4 % List of rectangle right side nodes.........................
    bt   = [[car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
            [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
    lic   = [bt(1,1:nni+2) bt(2,1:nni+2)]';
    lv    = [lic(1) lic(nni+2) lic(2*nni+4)];
end
if cs ==5% Quadrilateral:!! previous version replaced by a new 20211021 !!
    bt    = [[car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
             [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
    lic = [bt(1,1:nni+2) bt(2,1:nni+2)]';
    lv = [lic(1) lic(nni+2) lic(nni+3) lic(2*nni+4)];
end
if cs ==9% C shape
    bt    = [[car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
             [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
             [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)]];
    lic = [bt(1,1:nni+2) bt(2,1:nni+2) bt(3,1:nni+2)]';
    lv = [lic(1) lic(nni+2) lic(2*nni+4) lic(size(lic,1)) ];
end
end
```

Table 37: Matlab© 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© function *fem_smd.m* - solution of the nonlinear radiative system

```
unction [tca] = fem_smd(Kk,gh,lfi,fT,xyz,lK,lcont,nnr,nnv,nci,np,SBte...
    ,ca,nbo,bor)
nf    = size(fT,2);% disp(['Ls  2, list f. nodes: ',num2str(lfi(1:nf))])
                   % disp(['Ls  3, Fix. temper. : ',num2str(fT),' K'])
disp(['sd  5, N. fixed nod.: ',num2str(nf)])
if ca == 1
    ner = max(size(lcont));disp(['sd  7, N. rad elem. : ',num2str(ner)])
    lcont(ner+1)=lcont(1);
else
    ner = max(size(lcont))-1;disp(['sd 10, N. rad elem. : ',num2str(ner)])
end
if ner < 11;disp(['sd 12, Radiat. nodes: ',num2str(lcont)]);end
dK    = size(Kk,1);                   % Kk is the pure conductivity matrix
no    = dK - nnr - nnv;        % Number of unknowns of the system to solved
% lic      = zeros(nnr,round(sqrt(no)));
% for ii=1:nnr               % Loop on the sides involving radiation conditions
%     j        = ii+2;if j==5;j=1;end
%     lic(ii,:) = [car_cao(1,ii+1) bor(pbo(1,ii+1),5):bor(pbo(1,ii+1),6)...
%         car_cao(1,j)];      % List of the DOF of the irradiated patch side
% end
lc        = zeros(ner,3);    % Localization matrix of the radiative elements
K         = Kk;                                  % Pure conductivity matrix
tcant     = [ones(1,no)*fT(1) fT];               % Initial temperature field
for iter = 1 : 2 ;disp(['sd 24, Iteration N. : ',num2str(iter),' / 2'])
    for n = 1:nnr                 % Generation of cond. rad. element matrices
        for i        = 1:ner
            lc(i,1) = lcont(1,i); lc(i,2) = lcont(1,i+1); lc(i,3) = no+n;
            Ker     =fem_Kcr(xyz,lc(i,:),SBte,tcant);% Elem cond rad matr.
            for ii  = 1:3
```

```
                for jj = 1:3
                    K(lc(i,ii),lc(i,jj))=K(lc(i,ii),lc(i,jj))+Ker(ii,jj);
                end
            end % End assembling the conductive radiative element matrices
        end
    end
    N = zeros(nf,no + nnr);              % Linear constraint for fixations
    for i = 1:nf  ; N(i,lfi(i))=1 ; gh(dK+i)=fT(i); end;
    A     = [K  N';N zeros(nf,nf)]; B = A\gh      ; tca = B(1:dK);
    disp(['sd 39, Dissipation  : ',num2str(-B'*gh,4),' WK'])
    disp(['sd 40, React. flows : ',num2str(B(dK+1:dK+nf)',5),' W'])
    gt    = [fT(1),1,fT(nf)];
    nc2   = nci*nci;figure;
    for i  = 1 : np           % ............. Loop on the np CAD patches
        gra_ipa(nci,nci,lK((i-1)*nc2+1:i*nc2,:),tca(1:no),xyz,gt);
        colorbar;hold on
    end
    title(['Iteration: ',num2str(iter),', dissipation: ',...
        num2str(tca'*K*tca,3),' WK']);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
    disp(['sd 55, max - min tca: ',num2str(max(tca)-min(tca),3),' K'])
    tcant = [tca(1:no,1); fT'];  % Store temp. of solid for next iteration
end                                                      % End of iterations
end
```

Table 38: Matlab© 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 *fem_smt.m* - solution of linear transient equations

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

```
    end
end                                   % End of capacity matrices assembling
cap  = area*th*Cp*ro*1e-6;         % Domain capcity computed from mat. data
disp(['st 47, sum(sum(C))  : ',num2str(sum(sum(C))*1e-6), ' MJ/K'])
disp(['st 48, area*th*ro*Cp: ',num2str(cap), ' MJ/K']);
tsmax = ones(1,nit+1)*tmi;tsmin=tmi;tmoy=tmi;tcav = tmi;
gou    = zeros(1,nit+1);
ih     =100;% 4.04008;%                        % Imposed heat load in Wm-2
disp(['st 52, Imposed Heat : ',num2str(ih), ' W/m-2'])
bos      = th*(max(xyz(:,1))-min(xyz(:,1)));% Cross section area upper edge
g        = zeros(dK,1);
K        = Kk; ip=0; % Conduct. matrix related to solid and convective part
for it   = 1:nit % ............................ Loop on the time ierations
    if it==1;disp(['st 57, size(K) nfi  : ',num2str([size(K) nfi] )]);end
    ip          = ip+1;
    if cs < 3;g = gh*f(it)*ih*bos; end     % Imposed generalized heat flows
    if nfi      == 0               % The problem does not include fixations
        tca     = (C + dti*K)\(C*tcan + dti*g); % Tutorial, pp 41, Equ. 67
    else
        if it   == 1
            if deb    ==1
                disp ('st 68, Call function: ..... fem_tra .....')
            end;
        end
        tca     = fem_tra(K,C,dti,g,lfi,tcan);
%         if it==1;disp(['t. 69, diag K       : ',num2str(diag(K)' )]);end
%         if it==1;disp(['t. 70, diag C       : ',num2str(diag(C)' )]);end
%         if it==1;disp(['t. 71, tca...       : ',num2str(tca' )]);end
        go      = K * tca(1:size(K,1));      % Second member of the system
        gou(it) = sum(go)*dti;                  % Reactions at iteration it
    end
    tcav (it+1) = tca (dK,1);
    tsmax(it+1) = max (tca(1:dK-nfi));
    tsmin(it+1) = min (tca(1:dK-nfi));
    tmoy (it+1) = mean(tca(1:dK-nfi));
    tcan        = tca;
    if ip == dtd       % In this iteration, isotherms drawing is generated
        ip = 0; gt=[min(tca) pai max(tca)];
        disp(['st 82, iteration    : ',num2str(it)])
        if deb==1;disp ('st 86, Call function: ..... gra_ipa .....');end;
        figure
        ori = min(xyz(:,1));
        a   = ori-.2;b=ori-.1;ha = max(xyz(:,2)); %.... drawing a left bar
        fill([a b b a],[0 0 ha ha],[tmi tmi tma tma]);hold on;
        for i   = 1 : np % .................... Loop on the np CAD patches
            gra_ipa(nci,nci,lK((i-1)*nci^2+1:i*nci^2,:),tca',xyz,gt);
            colorbar;axis off;hold on
        end
        xlabel(['st 77 : ',num2str([max(tca) min(tca)])]);hold on
        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 the border of the domain
        title(['Elapsed time : ',num2str(it*dti/3600),' hours']);hold on
    end
end % ........................................... End of time iterations
ddt = tmoy(nit+1)-min(tmi,tma);ah=ddt*cap;% Stor.heat: ah=DT*area*th*ro*Cp
      disp(['st104, ddt Tm - Tin : ',num2str(ddt,3),' K'])
      disp(['st105, ddt*sumsum(C): ',num2str(ah ,3),' MJ'])
      disp(['st106, Min obs. temp: ',num2str(ceil(min(tsmin)),4),' K'])
      disp(['st107, Max obs. temp: ',num2str(max(tsmax),4),' K'])
      if deb==1;disp( 't.100, Call function: ..... gra_tev .....');end
      gra_tev(nit,tt,0,tsmax,tsmin,tmoy,3); % tcav(1) = -1;% Temper. evol.
eih    = tt*ih/pi*bos*1e-6;                 % Equation eih  of tutorial
% disp(['t.103, S&G Heat in   : ',num2str(ga*nit,3),' MJ'])
if ih ~= 0;disp(['st108, Heat inp.eih : ',num2str(eih,3),' MJ']);end
if nfi > 0        % Results shown & displayed only in presence of fixations
    if sum(gou) > 1.e-6
        disp(['st115, Tot. reaction: ',num2str(sum(gou)*1.e-6,3),' MJ'])
        figure('Position',[100 100 700 300]);ylabel('(MJ)','fontsize',15);
        hold on;plot(gou(1:it)/1.e6);grid on;hold on
        title(['Reaction on fixed DOF, sum(gou): ',...
            num2str(sum(gou)/1.e6,3),' MJ'],'fontsize',15);hold on
    end
end
end
```

Table 39: Matlab© function fem_smt.m – solution of linear transient equations

Function *fem_smq.m* – solution of nonlinear transient equations

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

```matlab
for it  = 1:nit % ............................ Loop on the time ierations
    ip  = ip+1;
    if ra == 0
        if rc    == 1     % Add radiative matrix Kr to conductive matrix K
            for ir = 1:mcr
                Kr = fem_Kcr(xyz,lc(ir,:),SBt,tcan);
                for i=1:3
                    for j = 1:3
                        K(lc(ir,i),lc(ir,j))=K(lc(ir,i),lc(ir,j))+Kr(i,j);
                    end
                end
            end
        end
    end
    if ra == 1          % Computation of K due to inter element view factors
        [K,Qs,Qg] = fem_Kra(Kk,lcont,re,Ftot,SBt,tcan,Tsky,lon);
        g(lcont)   = Qs' + Qg'+Mn;
    end     % Injected heat = weights * periodic function f(t)* load * area
    if cs < 3; g = gh * f(it) * ih * bos; end              % Injected heat
    if nfi    == 0                        % The domain does not contain fixations
        tca   = (C + dti*K)\(C*tcan + dti*(g));      % Equation 68
    else
        gr     = zeros(ntca,1);    % Sequence t 114 - t 115 equiv to fem_tra
        for m = 1:size(lcont,1)
            gr(lcont(m,1)) = gr(lcont(m,1))+Mn(m,1);
        end
        K11    = K(1:ntca,1:ntca); K12= K(1:ntca,ntca+1:dK);
        CC     = C(1:ntca,1:ntca); A  = (CC + dti*K11);
        tcb    = A\(CC*tcan(1:ntca,1)-dti*(K12*fT'+ gr));
        tca    = [tcb ; fT' ];
%          go    = K * tca;                    % Second member of the system
%          gou(it) = sum(go(lfi))*dti;       % Outgoing heat at iteration it
    end
    tsmax(it+1) = max (tca(1:dK-nfi)); % min(300,max (tca(1:dK-nfi))); %
    tsmin(it+1) = min (tca(1:dK-nfi)); % max(270,min (tca(1:dK-nfi))); %
    tmoy (it+1) = mean(tca(1:dK-nfi)); %(tsmin(it+1)+tsmax(it+1))/2; %
    tcan        = tca;
    if ip == dtd        % In this iteration, isotherms drawing is generated
        tmin = min(tca);tmax = max(tca);
        gt    = [tmin pai tmax];ip = 0;
     disp(['sq122, Iteration    : ',num2str(it),' gt: ',num2str(gt),' K'])
        if deb==1;disp ('sq 129, Call function: ..... gra_ipa .....');end;
        figure
        ori = min(xyz(:,1))-.2;
        a   = ori-.2;b = ori-.1;ha=max(xyz(:,2)); %.... drawing a left bar
        fill([a b b a],[0 0 ha ha],[280 280 300 300]);hold on;
        for i   = 1 : np % .................... Loop on the np CAD patches
            gra_ipa(nci,nci,lK((i-1)*nci^2+1:i*nci^2,:),tca',xyz,gt);
            colorbar;hold on
        end
        xlabel(['sq139 : ',num2str([max(tca) min(tca)])]);hold on
        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 the border of the domain
        title(['Elapsed time : ',num2str(it*dti/3600),' hours']);hold on
    end
end % ........................................... End of time iterations
ddt = abs(tmoy(nit+1)-tmi);ah = ddt*cap;              % DT *th*area *Cp*ro
      disp(['sq143, Tmean - Tini : ',num2str(ddt,3),' K'])
      disp(['sq144, Stored heat  : ',num2str(ah ,3),' MJ'])
if ddt > .1
    disp(['sq146, Min it+1 temp: ',num2str(tsmin(it+1),4),' K'])
    disp(['sq147, Max it+1 temp: ',num2str(tmoy (it+1),3),' K'])
    disp(['sq148, Max it+1 temp: ',num2str(tsmax(it+1),3),' K'])
    if deb==1;disp( 'r.150, Call function: ..... gra_tev .....');end
    gra_tev(nit,tt,re,tsmax,tsmin,tmoy,4);          % Temperature evolution
end
if nfi > 0        % Results shown & displayed only in presence of fixations
    if sum(gou) > 0
        disp(['sq154, Ejected heat : ',num2str(sum(gou)*1.e-6,3),' MJ'])
    end
end
end
```

Table 40: Matlab© function fem_smq.m – solution of nonlinear transient equations

Function *fem_smc.m* - cavity, VF matrices, radiative transfers

```
function[tca]  = ...
    fem_smc(np,xyz,lK,dK,nci,deb,rc,cs,area,gh,lg,...
    fT,lfi,nbo,bor,Kk,mcr,Lel,SB,re,Ms,Mpr,lcont,pai,th,ca,bos)
tmi=300;nel=size(lK,1);disp(['sc 04, Initial temp.: ',num2str(tmi) ,' K'])
if lfi(1)== 0;nfi=0;else;nfi=size(lfi,2);end;
disp(['sc 06, Numb. fixat. : ',num2str(nfi)]);
% ng=size(lg,2);disp(['Lt 07, N. load. nod.: ',num2str(ng)]);
tcan  = ones(dK,1)*tmi;if nfi>0;tcan(lfi)=fT;end      % Initial conditions
nit   = 720;dth=1;dtd=nit/4;          % Numb. iter. & delta tau per it. (h)
dti   = dth*3600;disp(['sc 10, Time step     : ',num2str(dti),' sec'])
tt    = dti*nit;                               % Analyzed period in seconds
        disp(['sc 12, Analyzed per.: ',num2str(tt/3600) ,' h, ',...
              num2str(tt/3600/24) ,' days'])
nd    = tt/3600/24;f=zeros(1,tt/3600);                 % nd = number of days
for i = 1:nd             % f is the imposed periodic function, f(h = 1:12)
    f((i-1)*24+1:(i-1)*24+12) = sin((1:12)*pi/12);
end
Cp    = 1000 ;disp(['sc 18, Spec. capac. : ',num2str(Cp),' J/(kg.K)']);
ro    = 2500 ;disp(['sc 19, Spec. mass   : ',num2str(ro),' kg.m-3'])
C     = zeros(dK,dK);       % Initialization of the global capacity matrix
% C(ndK,ndK)        = 1;    % Capacity of the fluid virtual node Jkg-1K-1
if deb==1;disp('sc 22, Call function: ..... fem_Cae .....');end
for n      = 1:nel       % Global capacity mat. assemb., loop on nel elements
    Cae    = fem_Cae(xyz,lK(n,:))*th*Cp*ro;        % Element capacity matrix
    for i = 1:4 %      if n == 1;disp(Cae);end
        for j=1:4;C(lK(n,i),lK(n,j)) = C(lK(n,i),lK(n,j)) + Cae(i,j);end
    end
end                                  % End of capacity matrices assembling
cap        = area*th*Cp*ro*1e-6;      % Domain capacity comp. from mat. data
disp(['sc 30, Solid heat C.: ',num2str(sum(sum(C))*1e-6), ' MJ/K'])
disp(['sc 31, area*th*ro*Cp: ',num2str(cap), ' MJ/K']);%disp(C)
ze    = ones (1,nit+1)*tmi;tsmax=ze;tsmin=ze;tmoy=ze;% tcav=ze;
ga    = zeros(1,nit+1);gou = zeros(1,nit+1);
% tca  = zeros(ndK,1);% gtot = 0.;
ih    =   0;% 0;% 4.04008;%                        % Imposed heat load in Wm-2
       disp(['sc 36, Imposed Heat : ',num2str(ih), ' Wm-2'])
% gt       = [tmi pai tma];                % gt = [min(tcan) pai max(tcan)]
nc2   = nci*nci;                              % Number of elements in a CAD patch
% if cs< 2;bos=th*xext;end % Cross sect. area of upper edge, street canyon
%if cs==2;bos=(xyz_cao(2,1)+xyz_cao(7,1)-xyz_cao(1,1)-xyz_cao(8,1))*th;end
K        = Kk; ip=0; % Conduct. matrix related to solid and convective part
if ca > 0                              % Valid only for the rectangular cavity
    lon = [ones(nci,1)*Lel(1)/nci;ones(nci,1)*Lel(2)/nci;...
           ones(nci,1)*Lel(3)/nci;ones(nci,1)*Lel(4)/nci];
end
if rc*cs == 2                     % Radiation in the street section (3 sides)
    nne              = (mcr-1)/3;            % nne = numb segm per street side
    for i            = 1:nne  % Computing the element lengths on the 3 sides
        lon(i)       = Lel(1)/(nne); lon(mcr-i)   = Lel(3)/(nne);
        lon(i+nne) = Lel(2)/(nne);
    end
    if mcr < 10;disp(['sc 52, lon. rad. el.: ',num2str(lon ),' m']);end
end                     % It is important to observe that: sum(esm) = sum(sn)
if rc*cs == 3              % Radiation around the thermal bridge - 5 segments
    n     = 0;                                               % Segment lengths
    for k = 1:5;for i= 1:nci;n = n+1;lon(n) = Lel(k)/(nci); end;end;
end                     % It is important to observe that: sum(esm) = sum(sn)
% .............................................. Loop on the time ierations
Mn       = zeros(dK,1);% bos = max(xyz(:,1))-min(xyz(:,1));
          disp(['sc 59, size(K) nfi  : ',num2str([size(K) nfi] )])
if deb==1;disp( 'sc 65, Call function: ..... fem_rsm .....');end
% Loop on the nit iterations steps 62 -- 116 ============================
for it   = 1:nit                                % Loop on time iterations step
    if rc     == 1         % Add radiative matrix Kr to conductive matrix Kk
%         disp(['sc 64, Le. rad. sid.: ',num2str(lon)])
        [Kr,Mss] = fem_rsm(tcan,SB*th,re,Ms,Mpr,it,lcont,lon,ca,nit);
        for i= 1:mcr
            Mn(lcont(i)) = - Mss(i);
            for j = 1:mcr
                K(lcont(i),lcont(j)) = Kk(lcont(i),lcont(j)) + Kr(i,j);
            end
        end
    end    % Injected heat = weights * periodic function f(t)* load * area
    g            = gh * f(it) * ih * bos;                      % Injected heat
    if lg(1)>0;ga(it+1)=sum(g(lg))*dti;end          % Heat input at step it
    ip           = ip+1;
    if nfi       == 0                  % The problem does not include fixations
        tca      = (C + dti*K)\(C*tcan + dti*(g + Mn));        % Equation 69
    else                                      % Domain including some fixations
        tca      = fem_tra(K,C,dti,(g + Mn),lfi,tcan);
        if it    == 1
            if deb     ==1
```

```
                disp ('sc 82, Call function: ..... fem_tra .....')
            end
        end
        go       = K * tca;                     % Second member of the system
        gou(it)  = sum(go(lfi(1:nfi))*dti*1e-6);% Reactions at iteration it
    end                                          % tcav (it+1) = tca (ndK,1)
    tsmax(it+1) = max (tca(1:dK-nfi));
    tsmin(it+1) = min (tca(1:dK-nfi));
    tmoy (it+1) = mean(tca(1:dK-nfi));          %(tsmin(it+1)+tsmax(it+1))/2;
    tcan        = tca;
    if ip == dtd       % In this iteration, isotherms drawing is generated
        pai = min((max(tca)-min(tca))/10,pai);
        ip = 0;gt = [min(tca) pai max(tca)];%gt=[290 pai 300]; %
        disp(['sc 95, iteration    : ',num2str(it)])
        if deb==1;disp(['sc 96, uniline gt    : ',num2str(gt)]);end;
        if deb==1;disp(['sc 97, size(tca)     : ',num2str(size(tca))]);end;
        if deb==1;disp ('sc 98, Call function: ..... gra_ipa .....');end;
        figure
%         a=-.2;b=-.1;ha=max(xyz(:,2));ti=281;ta=297;%..drawing a left bar
        a=-.2;b=-.1;ha=max(xyz(:,2));ti=287;ta=308;%....drawing a left bar
        fill([a b b a],[0 0 ha ha],[ti ti ta ta]);hold on;
        for i   = 1 : np % .................... Loop on the np CAD patches
            gra_ipa(nci,nci,lK((i-1)*nc2+1:i*nc2,:),tca',xyz,gt);
            colorbar;hold on
        end
        xlabel(['sc107 : ',num2str([max(tca) min(tca)])]);hold on
        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 the border of the domain
        title(['Elapsed time : ',num2str(it*dti/3600),' hours']);hold on
    end
end % ............................................. End of time iterations
ddt = tmoy(nit+1)-tmi;ah = ddt*cap;      % Stored heat: ah=DT*area*th*ro*Cp
    disp(['sc119, DTm Tm - Tini: ',num2str(ddt,3),' K'])
    disp(['sc120, Min obs. temp: ',num2str(min(tsmin),3),' K'])
    disp(['sc121, Max obs. temp: ',num2str(max(tsmax),3),' K'])
    disp(['sc122, Final temper.: ',num2str([tsmin(nit+1) tmoy(nit+1)...
        tsmax(nit+1)],3)])
    disp(['sc124, Capacity* DTm: ',num2str(ah ,3),' MJ'])
if deb==1;disp( 'Lc125, Call function: ..... gra_tev .....');end
    gra_tev(nit,tt,re,tsmax,tsmin,tmoy,5)        % Drawing temp. evolution
if deb == 1;disp( 'sc127, Call function: ..... gra_hie .....');end
if ih > 0
    gra_hie(nit,tt,ga)
    hdm = tt*ih/pi*bos*1e-6; % Explicit exact injected heat Equ. (80) pp 49
    disp(['sc131, Injected heat: ',num2str(sum(ga)/1.e06,3),' MJ'])
    disp(['sc132, Ex.heat input: ',num2str(hdm,3),' MJ'])
end             % Drawing heat input evolution
if nfi > 0        % Results shown & displayed only in presence of fixations
    disp(['sc135, Ejected heat : ',num2str(sum(gou),3),' MJ'])
    figure('Position',[100 100 700 300]);
    plot(gou(1:it));grid on;hold on
    title(['sc138 - Ejected heat: ',num2str(sum(gou),4),' MJ'],...
        'fontsize',15);hold on
    ylabel('Ejected heat (J)','fontsize',15);hold on
end
end
```

Table 41: Matlab© function fem_smc.m – Cavity, VF matrices, radiative transfers

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

15	28	4	tsmin	Uni-line matrix, lowest temp. in solid at each time step *K*
16	28	4	tsmax	Uni-line matrix, highest temp. in solid at each time step *K*
17	29	4	ga	Uni-line matrix, heat input at each step sum(g)*dti, in *J*
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© variables used in the function fem_smc.m

Matlab© function *fem_Cae.m* – element capacity matrix

```
function [C] = fem_Cae(xyz,lo)          % Capacity matrix of a quadrilateral
Q  = [xyz(lo(1),1:3); xyz(lo(2),1:3); xyz(lo(3),1:3); xyz(lo(4),1:3)];
s  = [.5-sqrt(3)/6 .5+sqrt(3)/6 .5+sqrt(3)/6 .5-sqrt(3)/6];  % 4 Gauss pts
t  = [.5-sqrt(3)/6 .5-sqrt(3)/6 .5+sqrt(3)/6 .5+sqrt(3)/6];  % 4 Gauss pts
C  = zeros(4,4);% area = 0.;
for i=1:4                                       % Loop on the 4 Gauss points
    f     = [(1-s(i))*(1-t(i)) s(i)*(1-t(i)) s(i)*t(i) (1-s(i))*t(i)];
    fs    = [-(1-t(i)) (1-t(i)) t(i) -t(i)    ];              % Derivative s
    ft    = [-(1-s(i)) -s(i)     s(i) (1-s(i))];              % Derivative t
    ds    = fs * Q;
    dt    = ft * Q;
    C     = C + f’*f* sqrt(dot(cross(ds,dt),cross(ds,dt)))/4;
end
end
```

Table 43: Matlab© function fem_Cae.m – capacity matrix of a quadrilateral

Matlab© function *fem_Kco.m* – conductivity matrix of isoparamatric element

```
function [K] = fem_Kco(xyz,lo)  % Conductivity matrix K, 3D surf. 20211001
Q  = [xyz(lo(1),: ); xyz(lo(2),: ); xyz(lo(3),: ); xyz(lo(4),: )];
s  = [.5-sqrt(3)/6 .5+sqrt(3)/6 .5+sqrt(3)/6 .5-sqrt(3)/6];% s 4 Gauss pts
t  = [.5-sqrt(3)/6 .5-sqrt(3)/6 .5+sqrt(3)/6 .5+sqrt(3)/6];% t 4 Gauss pts
K  = zeros(4,4); % area = 0.;
for i=1:4                                       % Loop on the 4 Gauss points
    fs    = [-(1-t(i)) (1-t(i)) t(i) -t(i)    ];              % Derivative s
    ft    = [-(1-s(i)) -s(i)     s(i) (1-s(i))];              % Derivative t
    gra   = [fs;ft];           % Gradient of the scalar bilinear function
    ds    = fs * Q;                          % Differencial in the s direction
    dt    = ft * Q;                          % Differencial in the t direction
%   area = area + sqrt(dot(cross(ds,dt),cross(ds,dt)))/4;
    J     = [fs*Q(:,1) fs*Q(:,2);ft*Q(:,1) ft*Q(:,2)];     % Jacobian matrix
    K=K+((J^(-1)*gra)’*J^(-1)*gra)*sqrt(dot(cross(ds,dt),cross(ds,dt)))/4;
end
%   disp([‘Patch area : ’,num2str(area)])
end       % Multiplied by k and the thickness, the K matrix is adimensional
```

Table 44: Matlab© function fem_Kco.m – element conduction matrix

Matlab© function *fem_Kcv.m* – *conduction-convection* matrix

```
function [K] = fem_Kcv(xyz,lc,hh)     % Convection matrix on a line segment
Q      = [xyz(lc(1),1:3); xyz(lc(2),1:3)];
L      = norm(Q(2,: )-Q(1,: ));
```

```
K      = [2 1 -3;1 2 -3;-3 -3 6]*hh*L/6;
end
```

Table 45: Matlab© 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 σ ($Wm^{-2}K^{-4}$). 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 ($\varepsilon = 1 - \rho$). 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© function *fem_smd.m* is needed to solve the system (*Table 38*).

Matlab© function *fem_Kcr.m* – radiative boundary element matrix

```
function [K] = fem_Kcr(xyz,lc,SBt,tca)%K is integrated on the bound. Segm.
Q      = [xyz(lc(1),1:3); xyz(lc(2),1:3)];  % Vector of element extremities
L      = norm(Q(2,: )-Q(1,: );                          % Length of the element
T1     = (tca(lc(1))+tca(lc(2)))/2;
tv     = tca(lc(3));
co     = (T1^2+tv^2)*(T1+tv)*L*SBt;
K      = [2 1 -3;1 2 -3;-3 -3 6]*co/6;
end
```

Table 46: Matlab© function fem_Kcr.m – 3 x 3 element radiation matrix

Matlab© function *fem_Kra.m* – inter elements view factors

```
function[K,Qs,Qg] = fem_Kra(Kk,lcont,re,Ftot,SBt,tcan,Tsky,lon) % 20211118
K      = Kk;
nd     = size(lcont,1)-1;                       % nd = number radiatve nodes
M      = eye(nd)-re*Ftot(1:nd,1:nd);                      % Radiosity matrix
% S      = eye(nd)*SBt*(1-re).*lon(1:nd);
S      = eye(nd)*SBt.*lon(1:nd);
tb     = zeros(nd,1);
for I = 1:nd; tb(i) = (tcan(lcont(i))+tcan(lcont(i+1)))/2; end
for I = 1:nd; S(I,i)= S(I,i)*tb(i)^3                    ; end
KI     = (eye(nd)-Ftot(1:nd,1:nd))*M^(-1)*S;
N      = zeros(nd,nd+1);for I = 1:nd;N(I,i)=.5;N(I,i+1)=.5; end% edg to nod
KJ     = N'*KI*N;
for I      = 1:nd
    for j = 1:nd;K(lcont(i),lcont(j))=K(lcont(i),lcont(j))+KJ(I,j);end
end
Qs     = (Ftot(1:nd,1:nd)*M^(-1)*Ftot(1:nd,nd+1)*SBt*Tsky^4)'*N;
Qg     = (Ftot(1:nd,1:nd)*M^(-1)*Ftot(1:nd,nd+2)*SBt*Tsky^4)'*N;
end
```

Table 47: Matlab© function fem_Kra.m – Inter elements view factors

Matlab© function *fem_rsm.m* second member in a radiative section

```
function[Mss] = fem_rsm(tca,SBt,re,Ms,Mpr,it,lcont,lon,ca,nit,deb)
mcr            = max(size(lcont));              % Number of radiative nodes
cat            = tca(lcont);              % Nodal temperatures of radiative nodes
SBte           = SBt*(1-re);if re > .999;SBte = 0;end;% Kr = zeros(mcr,mcr);
if SBte == 0                                                % re = 1, mirror
    Mss = zeros(mcr,1);
else                          % Emissivity is positive on the concerned boundary
    if ca   == 0                              % if ca == 0: Street or open zone
        si = mcr-1;Lo = zeros(si,mcr);     % si = number of radiative sides
        for i = 1:si;for j = i:i;Lo(i,j)=.5;Lo(i,j+1)=.5;end;end;
        cam(1:si,1)   = (cat(1:si,1)+cat(2:mcr,1))/2;      % Mid-segm. temp.
        Mss = Lo'*(Ms + (eye(si).*lon)'*SBte*Mpr*eye(si)*cam.^4);
    else         % if ca greater than 0, cavity or closed zone is concerned
        Lc = zeros(mcr,mcr);Lc(mcr,mcr)   = .5;Lc(mcr,1) = .5;
        for i = 1:mcr-1; Lc(i,i)=.5; Lc(i,i+1)=.5;end ;
        cam    = (Lc*cat);
        Mss    =  (eye(mcr).*lon)'*SBte*Lc'*Mpr*eye(mcr)*cam.^4;
    end
end
if deb==2;disp([it nit]);end
```

```
end
```

Table 48: Matlab© 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© function *fem_tra.m* – linear transient problem

```
function [tca]  = fem_tra(K,C,dti,g,lfi,tcan)
dK              = size(K,1);                    % Size of matrices K and C
nfi             = size(lfi,2);                      % Number of fixed DOF
N               = zeros(nfi,dK);for i=1:nfi;N(i,lfi(i))=1;end
A               = [C+dti*K N';N zeros(nfi,nfi)];
G               = [C*tcan+g ; tcan(lfi)]; B = A\G; tca = B(1:dK);
end
```

Table 49: Matlab© 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© 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 to 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© function *gra_ist.m* - isotherm drawing in a *nx* x *ny* mesh

```
function [] = gra_ist (nx,ny,z,xyz,gt)
no = (nx+1)*(ny+1);ii = 0;                          % Number of points of the grid
xx = zeros(ny+1,nx+1);yy = xx;tn = ones(ny+1,nx+1)*z(1); % Initializations
for i           = 1:ny
    for j       = 1:nx+1; ii = ii+1; tn(i,j) = z(ii); end;
end
tn(ny+1,:)      = z(ii+1:no); jj = 0;
for i           = 1:ny+1
    for j = 1:nx+1;jj = jj+1;xx(i,j) = xyz(jj,1);yy(i,j) = xyz(jj,2);end
end
colormap(gra_cob);                                          % Color map definition
[CS,H]     = contourf(xx,yy,tn,(gt(1):gt(2):gt(3)),'b');hold on;axis equal
             clabel(CS,H,[ 280 285 290 295 300 305 310 315 320]);
plot ([0 nx nx 0 0],[0 0 ny ny 0],'k','LineWidth',2);hold on;axis equal
end
```

Table 51: Matlab© 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© function *gra_mel.m* - drawing a shrunk mesh

```
function [] = gra_mel(xyz,lK,dn,sh,fs)                 % Drawing the shrunk mesh
% sh is the shrinking coefficient 0 < sh <= 1
nel     = size(lK,1);nn=size(lK,2);X = zeros(nn+1,1);Y = zeros(nn+1,1);
for j   = 1:nel                  % Nodes are numbered left - right, top - bottom
    ce = zeros(2,1);
    for i      = 1:nn                  % Loop on the nn vertices of the elements
        ce(1)  = ce(1)+xyz(lK(j,i),1)/nn;
        ce(2)  = ce(2)+xyz(lK(j,i),2)/nn;
        X(i)   = xyz(lK(j,i),1);
        Y(i)   = xyz(lK(j,i),2);
    end
    X(nn+1)  = X(1);Y(nn+1)=Y(1);  % First node repeated at the end of list
    if dn == 1
        plot((1-sh)*ce(1)+sh*X,(1-sh)*ce(2)+sh*Y,'k');hold on
        text(ce(1),ce(2),num2str(j),'Color','r','Fontsize',fs);hold on
    end
    if dn == 2
        plot((1-sh)*ce(1)+sh*X,(1-sh)*ce(2)+sh*Y,'k');hold on
        text(ce(1),ce(2),num2str(j),'Color','b','Fontsize',fs);hold on
    end
    if dn == 3;plot((1-sh)*ce(1)+sh*X,(1-sh)*ce(2)+sh*Y,'-b');hold on;end
end
end
```

Table 52: Matlab© function gra_mel.m - drawing a shrunk mesh

Matlab© function *gra_mnl.m* - displaying nodes and element labels

```
function [] = gra_mnl(xyz,lK,lc,fs)        % Display node and elements labels
figure;gra_mel(xyz,lK,1,1,fs)  % Draw conductive elements and nodes labels
if lc(1,3) > 0; gra_mel(xyz,lc,2,.8); axis equal; axis off;end
for i=1:size(xyz,1)
    text(xyz(i,1),xyz(i,2),num2str(i),'Fontsize',fs);hold on
end
end
```

Table 53: Matlab© function gra_mnl.m - displaying node & element labels

Matlab© function *gra_tra.m* – Temperature along radiative border

```
function [] = gra_tra(tca,lcont,re,ca)
figure;                    % Drawing the temperature along the radiative border
```

```
if ca > 0
    plot([tca(lcont); tca(lcont(1))],'k*-')
else
    plot(tca(lcont),'k*-');
end
if re > .999;re = 1;end
title(['tra 9, Temp. along radiative border, reflect.: ',num2str(re)]);
xlabel('From bottom right to left, to top left & top right')
ylabel('Temperature (K)');grid on;
end
```

Table 54: Matlab© function gra_tra.m - Temperature along radiative border

Matlab© function *gra_2dm.m* – Second member along radiative border

```
function[]=gra_2dm(K,tca,re,lcont)
gsm = K *tca;                         % second members on the cavity border
figure ( 'Position', [100 100 800 300]);bar(gsm(lcont)');grid on
xlabel (['Reflectivity:  ', num2str(re,2)])
title  (['L 2dm, Final radiative loads Kk*tca, min max sum:  ',...
 num2str([min(gsm(lcont)) max(gsm(lcont)) sum(gsm(lcont))],3),...
 ' W']);hold on
disp(['L 2dm, He fl. K*tca : ',num2str(sum(gsm(lcont)),3),' W'])
end
```

Table 55: Matlab© function gra_2dm – Second member along radiative border

Matlab© function *gra_atg.m* – temperature gradients

```
function [] = gra_atg (xyz,lK,tca)  % Element temperatures gradient arrows
nel     = size(lK,1);% nel = 1;%
u       = zeros(nel,1);v=zeros(nel,1);xx=zeros(nel,1);yy=zeros(nel,1);
area    = zeros(nel,1);
for ii = 1:nel                                  % Loop on the nel elements
    Q  = [xyz(lK(ii,1),1:2); xyz(lK(ii,2),1:2); xyz(lK(ii,3),1:2); ...
        xyz(lK(ii,4),1:2)];
    X  = Q(:,1);Y = Q(:,2);xx(ii) = sum(Q(:,1))/4;yy(ii) = sum(Q(:,2))/4;
    area(ii) = (X(2)-X(1)+X(3)-X(4))/2*(Y(3)-Y(1));
    te        = [tca(lK(ii,1)) tca(lK(ii,2)) tca(lK(ii,3)) tca(lK(ii,4))];
    J         = [[-1 1 1 -1]*X  [-1 1 1 -1]*Y;...% Jacob. matrix barycenter
                [-1 -1 1 1]*X  [-1 -1 1 1]*Y]/2;
    gr        = [-1 1 1 -1;-1 -1 1 1]*te'/2;  % Parametric grad. barycenter
    g         = J^(-1)*gr;
    u(ii)     = g(1);
    v(ii)     = g(2);
end
scale = 2;quiver(xx,yy,u,v,scale,'b','LineWidth',1);hold on;
% ad      = sum(area);                          % Maximum & mean heat flow
gm     = [max(sqrt(u.*u+v.*v)) mean(sqrt(u.*u+v.*v))];% grad: max & average
% fm      = mean(sqrt(u.*u+v.*v));                                  % W/m2
title(['Temperature gradient, max: ',num2str(gm(1),2),', mean: ',...
    num2str(gm(2),2),' K/m'],'fontsize',15);axis off;hold on
disp(['g  18, Max temp grad: ', num2str(gm(1),2),', mean: ',...
    num2str(gm(2),2),' K/m'])
end
```

Table 56: Matlab© function gra_atg.m - visualization of temperature gradient arrows

Matlab© function *gra_ahf.m* - visualization of heat flow arrows in a mesh

```
function [fm] = gra_ahf (xyz,lK,tca,co)        % Element heat flows arrows
nel     = size(lK,1);
u       = zeros(nel,1);v=zeros(nel,1);xx=zeros(nel,1);yy=zeros(nel,1);
area    = zeros(nel,1);
for ii = 1:nel                                  % Loop on the nel elements
    Q  = [xyz(lK(ii,1),1:2);xyz(lK(ii,2),1:2);xyz(lK(ii,3),1:2);...
          xyz(lK(ii,4),1:2)];
    X  = Q(:,1);Y = Q(:,2);xx(ii) = sum(Q(:,1))/4;yy(ii) = sum(Q(:,2))/4;
    area(ii) = (X(2)-X(1)+X(3)-X(4))/2*(Y(3)-Y(1));
    te        = [tca(lK(ii,1)) tca(lK(ii,2)) tca(lK(ii,3)) tca(lK(ii,4))];
    J         = 1/2*[[-1 1 1 -1]*X    [-1 1 1 -1]*Y;...
                    [-1 -1 1 1 ]*X    [-1 -1 1 1]*Y];
    gr        = [-1 1 1 -1; -1 -1 1 1]*te'/2; % Parametric grad. barycenter
    g         = -co(ii)*J^(-1)*gr;
    u(ii)     = g(1);
    v(ii)     = g(2);
end
scale  = 2;quiver(xx,yy,u,v,scale,'r','LineWidth',1);hold on;
ad     = sum(area);                             % Maximum & mean heat flow
gm     = [max(sqrt(u.*u+v.*v)) sqrt(u.*u+v.*v)'*area/ad];  % disp(gm(2)^2)
fm     = mean(sqrt(u.*u+v.*v));                                     % W/m2
title(['TA heat flows, max: ',num2str(gm(1),2),', mean: ',...
```

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

Table 57: Matlab© function gra_ahf.m - visualization of heat flow arrows

Matlab© function *gra_chf.m* - visualization temperature arrows for convection

```
function [] = gra_chf(xyz,lc,tca,h,fm)% Temperatures arrows for convection
nel    = size(lc,1); qn = zeros(nel,3);
disp(['ch 03, coef. red. dt: ',num2str(fm,2),' W/m2'])
for ii = 1:nel                       % Loop on the nel convective elements
    tgt = [xyz(lc(ii,2),1)-xyz(lc(ii,1),1);...
           xyz(lc(ii,2),2)-xyz(lc(ii,1),2); 0];    % Vector tangent to edge
    xm  = xyz(lc(ii,1),1)+(xyz(lc(ii,2),1)-xyz(lc(ii,1),1))/2;
    ym  = xyz(lc(ii,1),2)+(xyz(lc(ii,2),2)-xyz(lc(ii,1),2))/2;
    dt  = ((tca(lc(ii,1))+tca(lc(ii,2)))/2-tca(lc(ii,3)));    % Temp. diff.
    nor = cross(tgt,[0 0 1])/norm(tgt)*h*norm(dt)/fm;
    qn(ii,:) = dt/norm(tgt)*cross(tgt,[0 0 1]);        % h*dt x unit normal
    if dt > 0
        quiver(xm(1),ym(1),nor(1),nor(2),'r','LineWidth',1);hold on
    else
        quiver(xm(1)+nor(1),ym(1)+nor(2),-nor(1),-nor(2),'r',...
            'LineWidth',1);hold on
    end
end
title('Convective heat flows')
disp(['ch 19, temp. grad.  : ',num2str(dt),' K'])
disp(['ch 20, Mean conv. fl: ',num2str(mean(qn,1)),' W/m2'])
% if size(lc,1) < 10;disp(qn);end
end
```

Table 58: Matlab© function gra_chf.m - visualization of heat flow arrows

Matlab© function *gra_cob.m*

```
function [bar] = gra_cob
bar=[ 0      0          0.5625
0         0          0.6250
0         0          0.6875
0         0          0.7500
0         0          0.8125
0         0          0.8750
0         0          0.9375
0         0          1.0000
0         0.0625     1.0000
0         0.1250     1.0000
0         0.1875     1.0000
0         0.2500     1.0000
0         0.3125     1.0000
0         0.3750     1.0000
0         0.4375     1.0000
0         0.5000     1.0000
0         0.5625     1.0000
0         0.6250     1.0000
0         0.6875     1.0000
0         0.7500     1.0000
0         0.8125     1.0000
0         0.8750     1.0000
0         0.9375     1.0000
0         1.0000     1.0000
0.0625    1.0000     0.9375
0.1250    1.0000     0.8750
0.1875    1.0000     0.8125
0.2500    1.0000     0.7500
0.3125    1.0000     0.6875
0.3750    1.0000     0.6250
0.4375    1.0000     0.5625
0.5000    1.0000     0.5000
0.5625    1.0000     0.4375
0.6250    1.0000     0.3750
0.6875    1.0000     0.3125
0.7500    1.0000     0.2500
0.8125    1.0000     0.1875
0.8750    1.0000     0.1250
0.9375    1.0000     0.0625
1.0000    1.0000          0
1.0000    0.9375          0
1.0000    0.8750          0
1.0000    0.8125          0
```

```
1.0000    0.7500         0
1.0000    0.6875         0
1.0000    0.6250         0
1.0000    0.5625         0
1.0000    0.5000         0
1.0000    0.4375         0
1.0000    0.3750         0
1.0000    0.3125         0
1.0000    0.2500         0
1.0000    0.1875         0
1.0000    0.1250         0
1.0000    0.0625         0
1.0000         0         0];
end
```

Table 59: Matlab© function gra_cob.m - color bar

Matlab© function *gra_ipa.m* - drawing isotherm lines in a Coons patch

```
function [] = gra_ipa (nx,ny,el,z,xyz,gt)      % Isotherms lines in a patch
xx  = zeros(ny+1,nx+1);yy = xx;mp = xx ;tn  = ones(ny+1,nx+1)*z(1);ii = 0;
for j       = 1:ny
    for i   = 1:nx
        ii = ii+1;
        mp(j  , i)   = el(ii,1); mp(j  ,i+1)  = el(ii,2);
        mp(j+1, i+1) = el(ii,3); mp(j+1, i)   = el(ii,4);
    end
end
for j             = 1 : nx+1
    for i         = 1 : ny+1
        xx(i,j)   = xyz(mp(i,j),1);yy(i,j)   = xyz(mp(i,j),2);
        tn(i,j)   = z(mp(i,j));
    end;
end
colormap(gra_cob);                                           % Color map definition
[CS,H]      = contourf(xx,yy,tn,(gt(1):gt(2):gt(3)),'b');hold on;axis equal
             clabel(CS,H,[ 280 285 290 295 300 305 310 315 320]);
end
```

Table 60: Matlab© function gra_ipa.m - drawing isotherms in a Coons patch

Matlab© function *gra_lin.m* - visualization of the levels of a function

```
function[]      = gra_lin(xyz,nel,lK,tca)% Visualization element by element
colormap(gra_cob)
for i       = 1:nel
    fill(xyz(lK(i,:),1)',xyz(lK(i,:),2)',tca(lK(i,:)));hold on;
end;
% fill([-1 -.9 -.9 -1],[0 0 1 1],[300 300 320 320]);hold on;
colorbar;axis equal;axis off
end
```

Table 61: Matlab© function gra_lin.m - visualization of the levels of a scalar function

Matlab© function *gra_hie.m* – visualization of a periodic heat load

```
function [] = gra_hie(ni,dt,ga)                        % Evolution of incoming heat
tem             = (0:ni)*dt/3600/ni;             % Time steps for the time graphics
figure('Position',[100 100 700 300]);
plot   (tem,ga*1.e-3','b');hold on;grid on
xlabel('Elapsed time (hours)  ','fontsize',15)
ylabel('Injected heat flow (kW)','fontsize',15)
title (['gra-hie: Total injected heat: ',num2str(sum(ga)*1.e-6,3),...
    ' MJ'],'fontsize',15)
end
```

Table 62: Matlab© function gra_hie.m – visualization of a periodic scalar field

Function Matlab© *gra_tev.m* – temperature evolution in transient applications

```
function [] = gra_tev(ni,dt,re,tsmax,tsmin,tmoy,met)      % Temp. evolution
tem            = (0:ni)*dt/3600/ni;                 % Time steps for the graphics
st             = size(tsmin,2);
figure('Position',[100 100 600 300])
plot   (tem,tsmax','r');hold on;         % Plotting the 3 evolutive functions
plot   (tem,tsmin','b');hold on;
plot   (tem,tmoy   ,'k');hold on;
ree=re;if re>.999;ree=1;end
if met == 3
    xlabel(['Number of iterations : ',num2str(ni),],'fontsize',15);grid on
```

```
        title  ( 'st 109, gra-tev: temperature evolution','fontsize',15)
elseif met == 4
    xlabel (['Number of iterations : ',num2str(ni),', re : ',...
            num2str(ree)],'fontsize',15);grid on
        title  ( 'sq 150, gra-tev: temperature evolution','fontsize',15)
elseif met == 5
    xlabel (['Number of iterations : ',num2str(ni),', re : ',...
            num2str(ree)],'fontsize',15);grid on
        title  ( 'sc 139, gra-tev: temperature evolution','fontsize',15)
end
legend ('T maximum ','T minimum ','T average','Location','northwest')
ylabel (['Tmin: ',num2str(tsmin(st)*10/10,3),' K, Tmean: ',...
    num2str(tmoy(st)*10/10,3),' K, Tmax: ',num2str(tsmax(st)*10/10,3),...
    ' K'],'fontsize',10);hold on;grid on
end
```

Table 63: Matlab© 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© procedure *P_flgr.m* - drawing heat flows & temperature gradients

```
figure               % Drawing the element heat flows in the meshed domain
[fm]=gra_ahf(xyz,lK,tca,co);gra_mel(xyz,lK,3,1,10);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;axis off;hold on
end                          % End drawing element heat flows and domain border
if lc(1,3)> 0
    gra_chf(xyz,lc,tca,h,fm);     % fm is the averzge of element heat flows
end
figure          % Drawing the temperature gradients in the meshed domain
gra_atg(xyz,lK,tca);gra_mel(xyz,lK,2,.9,10);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
% disp(['re..................: ',num2str(re)]);
```

Table 64: Matlab© 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© function *geo_baf.m* – view factor matrix of a wall with balcony

```
function[F] = geo_baf(nci,xyz_cao) % Balcony view factor matrix % 20210929
n3          = nci*5+2;% Size of view factors matrix including sky and ground
F           = zeros(n3,n3); f=zeros(n3,1);
xc          = zeros(n3,1);
yc          = xc;
Le          = yc;                                   %  Edges definition
xn          = zeros(n3,1);yn = xn;                 % vertices definition
t           = 0:1/nci:1;b = zeros(1,n3-2);
for i               = 1:nci
    xn(i,1)         = xyz_cao(11,1)*(1-t(i))+xyz_cao(4,1)*t(i);
    xn(i+nci,1)     = xyz_cao(4,1)*(1-t(i))+xyz_cao(2,1)*t(i);
    xn(i+2*nci,1)   = xyz_cao(2,1)*(1-t(i))+xyz_cao(1,1)*t(i);
    xn(i+3*nci,1)   = xyz_cao(1,1)*(1-t(i))+xyz_cao(3,1)*t(i);
    xn(i+4*nci,1)   = xyz_cao(3,1)*(1-t(i))+xyz_cao(9,1)*t(i);
end
    xn(n3-2,1)      = xyz_cao(9,1);
    xn(n3-1  ,1)    = xyz_cao(9,1);
for i               = 1:nci
    yn(i,1)         = xyz_cao(11,2)*(1-t(i))+xyz_cao(4,2)*t(i);
    yn(i+nci,1)     = xyz_cao(4,2)  *(1-t(i))+xyz_cao(2,2)*t(i);
    yn(i+2*nci,1)   = xyz_cao(2,2)  *(1-t(i))+xyz_cao(1,2)*t(i);
```

```
    yn(i+3*nci,1) = xyz_cao(1,2) *(1-t(i))+xyz_cao(3,2)*t(i);
    yn(i+4*nci,1) = xyz_cao(3,2) *(1-t(i))+xyz_cao(9,2)*t(i);
end
    yn(n3-2,1)    = xyz_cao(3,2) *(1-t(i))+xyz_cao(9,2)*t(i);
    yn(n3-1,1)    = xyz_cao(9,2);
for i      = 1: n3-2;Le(i) = sqrt((xn(i+1)-xn(i))^2+(yn(i+1)-yn(i))^2);end
for i      = 1: n3-2;xc(i) = (xn(i)+xn(i+1))/2;yc(i)=(yn(i)+yn(i+1))/2;end
tsb        = [0 -1 0 1 0;-1 0 -1 0 -1];
k          = 0;for i=1:5;for j=1:nci;k=k+1; b(k)=i;end;end
ts(1,:)    = tsb(1,b);ts(2,:) = tsb(2,b);
F(1:3*nci ,n3)=.5;F(2*nci+1:5*nci,n3-1)=.5;
for np          = 1  : nci % Form fact. matrix for nci (upper L) vert. elem.
    for i       = nci+1  : 2*nci                    % Loop on the nci points
        r0      = sqrt((xn(i)   -xc(np))^2+(yn(i)   -yc(np))^2);
        r1      = sqrt((xn(i+1)-xc(np))^2+(yn(i+1)-yc(np))^2);
        f(i)    = -(((xn(i)-xc(np))/r0 -(xn(i+1)-xc(np))/r1 ) *ts(1,np)-...
                     ((yn(i)-yc(np))/r0 -(yn(i+1)-yc(np))/r1 ) *ts(2,np))/2;
    end
    f(n3-1)     = -(((xn(i)-xc(np))/r1-1) *ts(1,np)-...
                      ((yn(i)-yc(np))/r1  ) *ts(2,np))/2;
    f(n3)       = 0.5;
    F(np,:)     = f';
end
f               = zeros(n3,1);
for np          = nci+1:2*nci    % Form fact. for nci (upper L) horiz. elem.
    for i       =  1   :   nci                     % Loop on the nci points
        r0      = sqrt((xn(i)   -xc(np))^2+(yn(i)   -yc(np))^2);
        r1      = sqrt((xn(i+1)-xc(np))^2+(yn(i+1)-yc(np))^2);
        f(i)    =  (((xn(i)-xc(np))/r0 -(xn(i+1)-xc(np))/r1 )*ts(1,np)-...
                     ((yn(i)-yc(np))/r0 -(yn(i+1)-yc(np))/r1 ) *ts(2,np))/2;
    end
    f(n3-1)     = 0.;
    f(n3)       = 1-sum(f(1:n3-2));
    F(np,:)     = f';
end
f               = zeros(n3,1);
for np          = 3*nci+1:4*nci  % Form fact. for nci (lower L) horiz. elem.
    for i       = 4*nci+1  :5*nci                   % Loop on the nci points
        r0      = sqrt((xn(i)   -xc(np))^2+(yn(i)   -yc(np))^2);
        r1      = sqrt((xn(i+1)-xc(np))^2+(yn(i+1)-yc(np))^2);
        f(i)    = (((xn(i)-xc(np))/r0 -(xn(i+1)-xc(np))/r1 ) *ts(1,np)-...
                     ((yn(i)-yc(np))/r0 -(yn(i+1)-yc(np))/r1 ) *ts(2,np))/2;
    end
    f(n3-1)     = 1-sum(f(1:n3-2)); f(n3)    = 0.;
    F(np,:)     = f';
end
f               = zeros(n3,1);
for np          = 4*nci+1:5*nci   % Form fact. for nci (lower L) vert. elem.
    for i       = 3*nci+1:4*nci                    % Loop on the nci points
        r0      = sqrt((xn(i)   -xc(np))^2+(yn(i)   -yc(np))^2);
        r1      = sqrt((xn(i+1)-xc(np))^2+(yn(i+1)-yc(np))^2);
        f(i)    = -(((xn(i)-xc(np))/r0 -(xn(i+1)-xc(np))/r1 )*ts(1,np)-...
                     ((yn(i)-yc(np))/r0 -(yn(i+1)-yc(np))/r1 ) *ts(2,np))/2;
    end
    f(n3-1)     = .5 ;f(n3) = 1-sum(f(1:n3-1));  F(np,:) = f';
end
end
```

Table 65: Matlab© function geo_baf.m – view factor matrix – wall with balcony

Matlab© function *geo_stf.m* – view factor matrix of a street section

```
function[F] = geo_stf(n,Lel)
xs          = Lel(1); ys = Lel(2);          % Lel = vector of the side lengths
no          = n-1;% n is the number of el. on a side, no the number of nodes
n3          = n*3;% *                % View factors matrix: 3 sides and the sky
F           = zeros(n3,n3); r0=zeros(1,n3); r1=zeros(1,n3); f = zeros(1,n3);
xc          = zeros(1,n3)   ;yc    = xc; Le = yc;             %  Edges definition
xn          = zeros(1,n3+1);yn     = xn;                   % vertices definition
xn(1:no)    = 0;   k = 0; for i = n+2:2*n+1; k=k+1; xn(i)=xs/(n)*k;end;
xn(2*n+2:3*n+1) = xs;% disp(['xn : ',num2str(xn)])
for i=1:n;yn(i )= ys-(i-1)*ys/(n);end;yn(n+2:2*n )=0;k=-1;
for i=2*n+1:3*n+1;k=k+1;yn(i)=k*ys/(n);end;
for i      = 1: n3;Le(i) = sqrt((xn(i+1)-xn(i))^2+(yn(i+1)-yn(i))^2);end
for i      = 1: n3;xc(i) = (xn(i)+xn(i+1))/2;yc(i)=(yn(i)+yn(i+1))/2;end
ts         = [0 -1 -1;1 0 1;0 1 -1;];       % Tangents of edges & their sign
for np          = 1 : n3            % Form factor matrix for the n3 elements
    i0          = 0; i1 = 1;
    for i       = 1 : n3                               % Loop on the n4 points
        i0      = i0+1;
        i1      = i1+1;
        nuc     = ceil(np/n);
```

```
            r0(i)  = sqrt((xn(i0)-xc(np))^2+(yn(i0)-yc(np))^2);
            r1(i)  = sqrt((xn(i1)-xc(np))^2+(yn(i1)-yc(np))^2);
            f(i)   = (((xn(i0)-xc(np))/r0(i)-(xn(i1)-xc(np))/r1(i))*...
                ts(nuc,1)-((yn(i0)-yc(np))/r0(i)-(yn(i1)-yc(np))/r1(i))*...
                ts(nuc,2))*ts(nuc,3)/2;
        end
        for i      = 1:n;f((nuc-1)*n+i) = 0.;end     % View factor of anal. face
        F (np,:)   = f;
    end
end
```

Table 66: Matlab© function geo_stf.m – view factor matrix of a street section

Matlab© function *geo_vfc.m*

```
function[F] = geo_vfs(n,Lel,ns)% n is the numb. of elements on the 4 sides
xs          = Lel(1); ys = Lel(2);          % Lel = vector of the side lengths
no          = n+1;                  % no is the number of nodes on each side
if ns       == 4;n4=n*ns;else;n4=n*ns+2;end % VF mat.: ns sides or ns + sky
F           = zeros(n4,n4); r0=zeros(1,n4); r1=zeros(1,n4); f = zeros(1,n4);
xc          = zeros(1,n4)  ;yc   = zeros(1,n4);Le = yc;  %  Edges definition
xn          = zeros(1,n4+1);yn   = zeros(1,n4+1);      % vertices definition
for i = 1:n      % Ordered nodes in the street: from bottom-left, area left
    xn(i + 1)   = xs/n*i; xn(no + i)    = xs; xn(no + n+i)  = xs-xs/n*i;
    yn(i + no)  = ys/n*i; yn(no +n+i)   = ys; yn(3*n+1+i)   = ys-ys/n*i;
end
for i       = 1: n4;Le(i)=sqrt((xn(i+1)-xn(i))^2+(yn(i+1)-yn(i))^2);end
for i       = 1: n4;xc(i)=(xn(i)+xn(i+1))/2; yc(i)=(yn(i)+yn(i+1))/2;end
ts          = [1 0 1;0 1 -1;-1 0 1;0 -1 -1];% Tangents of edges & their sign
for np          = 1 : n4% Compute the form factor matrix for the np elements
    i0          = 0; i1 = 1;
    for i       = 1 : n4                            % Loop on the n4 points
        i0      = i0+1;
        i1      = i1+1;
        nuc     = ceil(np/n);
        r0(i)   = sqrt((xn(i0)-xc(np))^2+(yn(i0)-yc(np))^2);
        r1(i)   = sqrt((xn(i1)-xc(np))^2+(yn(i1)-yc(np))^2);
        f(i)    = (((xn(i0)-xc(np))/r0(i)-(xn(i1)-xc(np))/r1(i))*...
            ts(nuc,1)-((yn(i0)-yc(np))/r0(i)-(yn(i1)-yc(np))/r1(i))*...
            ts(nuc,2))*ts(nuc,3)/2;
    end
    for i           = 1:n;f((nuc-1)*n+i) = 0.;end  % View factor of anal. face
    F(:,np)         = f;
end
end
```

Table 67: Matlab© function geo_vfc.m – view factor matrix – ns sides domain

Matlab© function *geo_vfr.m*

```
function[F] = geo_vfr(n,Lel)         % Rectangular cavity view factor matrix
xs          = Lel(1); ys = Lel(2);          % Lel = vector of the side lengths
no          = n+1;% n: number of elem. & no : number of nodes per patch side
nf          = n*4;                           % Size of the view factors matrix
F           = zeros(nf,nf);r0 = zeros(1,nf);r1 = r0;f = r0;xc = r0;yc = r0;
xn          = zeros(1,nf+1);yn = xn;                    % vertices definition
for I = 1:n                       % Ordered nodes from bottom-left, area left
    xn(I + 1)   = xs/n*I; xn(no + i)    = xs; xn(no + n+i)  = xs-xs/n*I;
    yn(I + no)  = ys/n*I; yn(no +n+i)  = ys; yn(3*n+1+i)   = ys-ys/n*I;
end
for I       = 1: nf;xc(i)=(xn(i)+xn(i+1))/2; yc(i)=(yn(i)+yn(i+1))/2;end
ts          = [1 0 1;0 1 -1;-1 0 1;0 -1 -1];% Tangents of edges & their sign
for np          = 1 : nf% Compute the form factor matrix for the nf elements
    i0          = 0;
    xc1=(.5-sqrt(3)/6)*(xn(np+1)-xn(np))+xn(np);
    xc2=(.5+sqrt(3)/6)*(xn(np+1)-xn(np))+xn(np);
    yc1=(.5-sqrt(3)/6)*(yn(np+1)-yn(np))+yn(np);
    yc2=(.5+sqrt(3)/6)*(yn(np+1)-yn(np))+yn(np);
    for I       = 1 : nf                  % Loop on the nf visible segments
        i0      = i0+1;i1 = i0+1;
        nuc     = ceil(np/n);                     % nuc is the patch side number
        r0(i)   = sqrt((xn(i0)-xc1)^2+(yn(i0)-yc1)^2);
        r1(i)   = sqrt((xn(i1)-xc1)^2+(yn(i1)-yc1)^2);
        f1      = (((xn(i0)-xc1)/r0(i)-(xn(i1)-xc1)/r1(i))*...
            ts(nuc,1)-((yn(i0)-yc1)/r0(i)-(yn(i1)-yc1)/r1(i))*...
            ts(nuc,2))*ts(nuc,3)/2;
        r0(i)   = sqrt((xn(i0)-xc2)^2+(yn(i0)-yc2)^2);
        r1(i)   = sqrt((xn(i1)-xc2)^2+(yn(i1)-yc2)^2);
        f2      = (((xn(i0)-xc2)/r0(i)-(xn(i1)-xc2)/r1(i))*...
            ts(nuc,1)-((yn(i0)-yc2)/r0(i)-(yn(i1)-yc2)/r1(i))*...
            ts(nuc,2))*ts(nuc,3)/2;
```

```
    end
    for I       = 1:n;f((nuc-1)*n+i) = 0.;end  % View factor of anal. face
    F(:,np)     = f;
end
end
```

Table 68: Matlab© 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© function *mat_cok.m* – non homogeneous conductivity

```
function [co] = mat_cok (Ai,nci,fa,xy,lK,deb)   % Non uniform conductivity
k   = 1 ;                                                        % W/(m K)
nel = nci * nci;                              % Number of elements per patch
co  = ones(nel,1)*k;     % By default, the system is isotropic, k constant
if Ai == 1
    if floor(nci/2) < ceil(nci/2)                               % nci is odd
        tr=1/nci;
        for i =floor(nci/2)*nci+1:floor(nci/2)*nci+nci;co(i)=k*fa;end% (n)
    else
        tr=2/nci;                                                % nci is even
        for i =(nci/2-1)*nci+1:(nci/2*nci)+nci;co(i)=k*fa;end         % (W)
    end
end
if Ai == 3
    if floor(nci/2) < ceil(nci/2)                               % nci is odd
        m = -nci; tr=1/nci;
        for j = 1:nci        % Definition of 1 element wide vertical strip
            m = m+nci;for i =((nci+1)/2):((nci+1)/2);co(m+i)=k*fa;end% (W)
        end
    else
        m = -nci; tr=2/nci;                                     % nci is even
        for j = 1:nci       % Definition of 2 elements wide vertical strip
            m = m+nci;for i =(nci/2):(nci/2+1);co(m+i)=k*fa;end% (W)
        end
    end
end
if Ai == 4
    if floor(nci/2) < ceil(nci/2)                               % nci is odd
        m = -nci; tr=3/nci;
        for j = 1:nci       % Definition of 3 elements wide vertical strip
            m=m+nci;for i=((nci+1)/2-1):((nci+1)/2+1);co(m+i)=k*fa;end%(W)
        end
    else
        m = -nci; tr=4/nci;                                     % nci is even
        for j = 1:nci       % Definition of 4 elements wide vertical strip
            m = m+nci;for i =(nci/2-1):(nci/2+2);co(m+i)=k*fa;end% (W)
        end
    end
end
if deb == 1
    figure;nu=0;% colormap(gra_cob)                               % Isotherms
    for i=1:nel
        nu=nu+1;fill(xy(lK(i,:),1)',xy(lK(i,:),2)',co(nu));hold on
    end
    colorbar;axis equal;axis off
end
disp(['Lm 47, Anis. index  : ',num2str(Ai)])
disp(['Lm 48, k & coef*k   : ',num2str([k k*fa]),' W/(m K)'])
disp(['Lm 49, Thickn. ratio: ',num2str(tr)])
if nel < 50;disp(['Lm 26, Anis. vector : ',num2str(co')]);end
end
```

Table 69: Matlab© function mat_cok.m - non homogeneous conductivity

8.6 List of Matlab© procedures and functions

Fia_20221023.m	*Table 1*
Fiam_20221023.m	*Table 2*
P_flgr.m	*Table 64*
Fiammetta.m	*Table 28*
cad_ban.m	*Table 37*

cad_Dir.m *cad_Neu.m* *cad_con.m* *cad_edg.m* *cad_gin.m* *cad_mes.m*	*Table 32* *Table 33* *Table 34* *Table 36* *Table 31* *Table 35*
fem_Cae.m *fem_Kco.m* *fem_Kcr.m* *fem_Kcv.m* *fem_Kra.m* *fem_rsm.m* *fem_smd.m* *fem_smt.m* *fem_smq.m* *fem_smc.m* *fem_tra.m*	*Table 43* *Table 44* *Table 46* *Table 45* *Table 47* *Table 48* *Table 38* *Table 39* *Table 40* *Table 41* *Table 49*
geo_baf.m *geo_stf.m* *geo_vfc.m* *geo_vfr.m*	*Table 65* *Table 66* *Table 67* *Table 68*
gra_atg.m *gra_ahf.m* *gra_chf.m* *gra_cob.m* *gra_hie.m* *gra_ipa.m* *gra_ist.m* *gra_lin.m* *gra_mel.m* *gra_mnl.m* *gra_tev.m* *gra_tra.m* *gra_2dm.m*	*Table 56* *Table 57* *Table 58* *Table 59* *Table 62* *Table 60* *Table 51* *Table 61* *Table 52* *Table 53* *Table 63* *Table 54* *Table 55*
mat_cok.m	*Table 69*

Table 70: Matlab© 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© 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 immerged in three fluids with low temperature on the top and bottom parts. The middle part is immerged 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

[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

[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

11. Content

12. End