Fiammetta, Finite Element Method Applied to Heat Transfer

– Tutorials –

Benoit Beckers
September 2021
Urban Physics Joint Laboratory
Université de Pau et des Pays de l'Adour (France)

The six lectures on Finite Elements applied to heat transfer, 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 rectangle meshed by squares. The last lecture is devoted to the radiative exchanges.

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 express the surface temperature field, such that it could be captured, in the real world, by a thermal camera placed in front of these buildings.

The tutorial presents short programs, written in Matlab©, which are progressively developed in conduction, convection, radiation and transient regime. Each step is completed by an exercise that will allow the reader becoming familiar with the main features presented in the lectures.

1. Tutorial I: Basic problem of thermal conduction

1.1 Finite element model

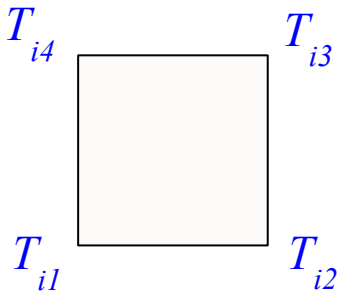
In each sub-domain or finite element numbered i, a Rayleigh-Ritz is applied. The temperature τ_i of element i is discretized by bilinear polynomial functions associated to four nodal temperatures T_{ij} of the vertices (*Figure 1*).

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

Explicitly, for a square of dimension a, we have:

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

This definition allows computing the conduction matrix of the square (*Figure 1*). This matrix does not depend on its size, but only on the conductivity coefficient k and the thickness e. The figure shows the square element and its degrees of freedom (*DOF*). 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 1: A square element and its conductivity matrix

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

$$< I(T) = \sum_{i=1}^{nel}\left(\int_{\Omega j}\frac{1}{2}k_i\ \vec{\nabla}\sum_{j=1}^{4}T_{ij}f_{ij}.\ \vec{\nabla}\sum_{j=1}^{4}T_{ij}f_{ij}\ d\Omega_i + \int_{S_{2i}}\bar{q}_n\ \sum_{j=1}^{4}T_{ij}f_{ij}\ dS_i\right) > \tag{3}$$

After introducing the polynomial trial functions given in (2), we can write (3) in matrix form:

$$< I(T) = \sum_{i=1}^{nel}\left[T_i\right]^T\left[K_i\right]\left[T_i\right]+\left[T_i\right]^T\left[F_i\right] > \tag{4}$$

The last term $[F_i]$ of (4) is the vector of generalized heat loads.

In the next step, we have to express the continuity of the temperature field across the domain. At each interface between two elements, it must be stated that the nodal temperature field is identical, which means that the nodal temperatures of the elements sharing a same node are the same. In the mesh of *Figure 2*, 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.

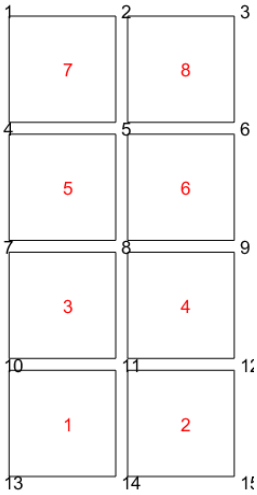


Figure 2: Node and element numbering

The temperature gradients are obtained by derivation of the element temperature field *Figure 1*, but now, we extend the shape to a rectangle of dimensions a x b:

$$\tau = T_1\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right)+T_2\frac{x}{a}\left(1-\frac{y}{b}\right)+T_3\frac{x}{a}\frac{y}{b}+T_4\left(1-\frac{x}{a}\right)\frac{y}{b} \tag{5}$$

It is easy to derive this polynomial expression:

$$\begin{bmatrix}\frac{\partial\tau}{\partial x}\\ \frac{\partial\tau}{\partial y}\end{bmatrix}=\begin{bmatrix}\frac{1}{a}\left(\frac{y}{b}-1\right) & \frac{1}{a}\left(1-\frac{y}{b}\right) & \frac{y}{ab} & -\frac{y}{ab}\\ \frac{1}{b}\left(\frac{x}{a}-1\right] & -\frac{x}{ab} & \frac{x}{ab} & \frac{1}{b}\left(1-\frac{x}{a}\right)\end{bmatrix}\begin{bmatrix}T_1\\T_2\\T_3\\T_4\end{bmatrix} \tag{6}$$

$$\begin{bmatrix}\frac{\partial\tau}{\partial x}\\ \frac{\partial\tau}{\partial y}\end{bmatrix}=\frac{1}{ab}\begin{bmatrix}y-b & b-y & y & -y\\ x-a & -x & x & a-x\end{bmatrix}\begin{bmatrix}T_1\\T_2\\T_3\\T_4\end{bmatrix}$$

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

$$\begin{aligned}\frac{\partial\tau}{\partial x}&=\frac{1}{2a}\left(\left(T_2+T_3\right)-\left(T_1+T_4\right)\right)\\ \frac{\partial\tau}{\partial y}&=\frac{1}{2b}\left(\left(T_3+T_4\right)-\left(T_1+T_2\right)\right)\end{aligned} \tag{7}$$

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

$$\begin{bmatrix}q_x\\q_y\end{bmatrix}=-k\begin{bmatrix}\frac{\partial\tau}{\partial x}\\ \frac{\partial\tau}{\partial y}\end{bmatrix} \tag{8}$$

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

1.2 Innovative handling of the Dirichlet boundary conditions

The Matlab© procedure of *Table 1* exhibits a basic 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:

$$[N]\,[T] = [\bar{T}] \tag{9}$$

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 $[\Lambda]$:

$$< I(T) = [T]^T [K][T] + [\Lambda]^T \left([N][T] - [\bar{T}]\right) \tag{10}$$

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 $[T]$:

$$[K][T] + [N]^T [\Lambda] = [0] \tag{11}$$

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:

$$\begin{bmatrix} K & N^T \\ N & 0 \end{bmatrix} \begin{bmatrix} T \\ \Lambda \end{bmatrix} = \begin{bmatrix} 0 \\ \bar{T} \end{bmatrix} \quad \rightarrow \quad [A][B] = [G] \tag{12}$$

The uni-column matrix $[G]$ contains a sequence with zero values of the same size as $[T]$ 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 and the Lagrange multipliers which represent the generalized heat flows through the nodes. Unlike the matrix $[K]$, the matrix $[A]$ is nonsingular if the restraints are independent.

Matlab© procedure *Fiammetta.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
end % ;if nn<20;disp([(1:nn)' xy]);end          % Display nodal coordinates
disp(['Number of nodes     : ',num2str(nn)])     % End geometry definition
ne   = nx*ny; lK = zeros(ne,4); m = 0;          % Topology: mesh definition
for j       = 1:ny
    for i   = 1:nx
       m    = m+1;lK(m,1) = nn-nx-1+i-(j-1)*(nx+1); lK(m,2) = lK(m,1) + 1;
                  lK(m,3) = lK(m,2)-nx-1          ; lK(m,4) = lK(m,3)-1;
    end
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;for i=1:ne;fill(xy(lK(i,:),1)',xy(lK(i,:),2)',T(lK(i,:)));
28     hold on;axis equal;end;colorbar;axis off
29 disp(['Dissipation ....... : ',num2str(T'*K*T/2,3),' WK'])
30 disp(['CPU ............... : ',num2str(toc(CPU),3),' sec. '])
31 % figure;colormap(gra_cob);gra_ist(nx,ny,T,xy,[min(T) 2 max(T)])%Isotherms
32 % colorbar;axis equal;axis off
33 % if ne < 100;gra_mnl(xy,lK,[0 0 0]);axis equal;axis off;end
```

Table 1: Matlab© procedure Fiammetta.m - Conduction problems

With the parameter r = 1, (*line 1* of *Table 1*), the result (*Figure 3*) is the exact solution, independent of the mesh.

```
Fiammetta
Number of nodes      : 6
Number of elements   : 2
Numb. of fix. temp.  : 4
Dissipation ....... : 625 WK
CPU ............... : 0.153 sec.
```

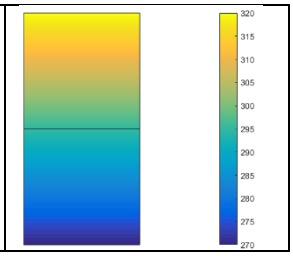


Figure 3: Temperature levels obtained in a program using exclusively Matlab© functions

Disabling *lines 27 & 28* and enabling the use of funtion *gra_cob.m* (*Table 2*) and *gra_ist.m* (*Table 3*) in *line 31 & 32*, we get better colors (*Figure 4*, *left*). The use of the function *gra_ist.m* instead of the Matlab© function *fill*, gives the graphics with isotherm lines (*Figure 4*, *right*).

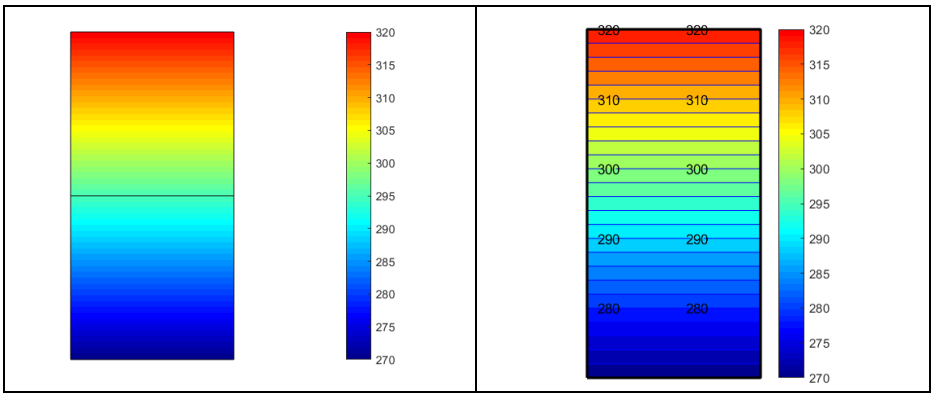


Figure 4: Example with imposed temperatures producing a vertical gradient

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:

$$le\, k \frac{\Delta T}{\Delta y} = k \frac{50}{2} = 25\ W \qquad (13)$$

Added at the end of the procedure, the following line provides the input and output nodal generalized heat flows 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*.

```
34  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 *nx* +1 first terms of [*Λ*], *nx* is the number of elements in the *x* direction.
In *Figure 4*, 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, in Matlab© notation, the reactive generalized heat flows of the top horizontal side given by: `disp(['hf = ',num2str(B(nn+1:nn+nf/2)')])`
are: `hf = 2.5 5 5 5 5 2.5 W`
With Dirichlet boundary conditions (*lines 13 - 15*) we obtain the solutions shown in *Figure 5*.

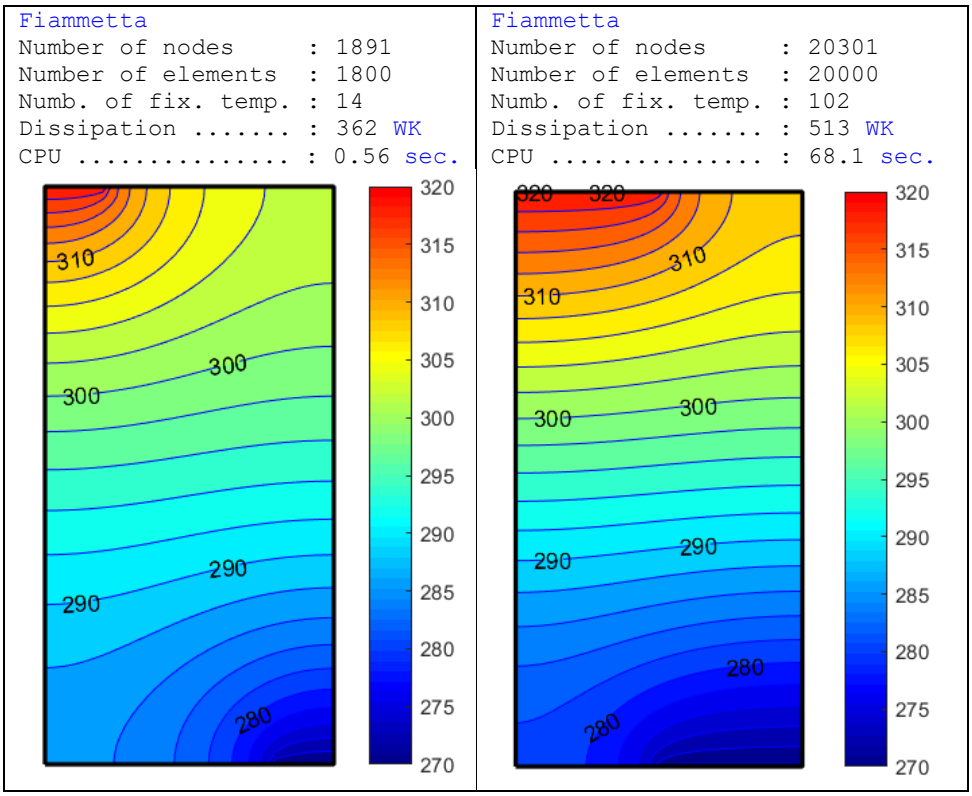


Figure 5: Imposed temperatures on parts of the horizontal faces (nx = 30 & 50)

1.3 Matlab procedure for the basic conduction problem

To perform tests on conduction, we use the procedure of *Table 1*, first in a 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 the beginning of this document, the finite element model is limited to a rectangle composed of squares (*Figure 2*). The temperature is always expressed in Kelvin (*K*).
The Matlab© procedure of *Table 1* is providing one single output: the visualization of the domain with colored temperature levels (*Figure 6*). This graphics is generated in the *lines 27* –

29 of *Fiammetta_33_20210830.m*, using the standard Matlab© function *fill*. In *line 27*, to obtain a high-quality color bar, we are also using the function *gra_cob.m* (*Table 2*), which is providing a very effective color bar at the right of *Figure 6*. 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).

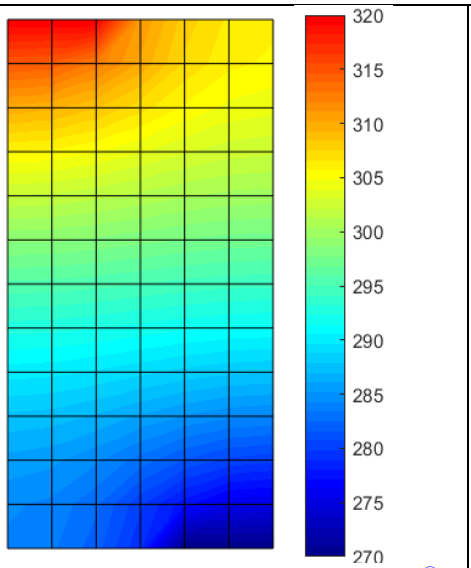


Figure 6: Standard output of Matlab© procedure Fiammetta.m

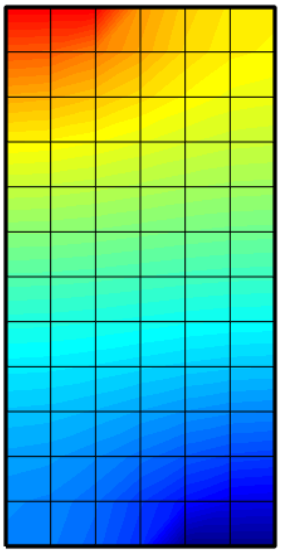


Figure 7: Fiammetta.m output after running the instruction below.

The extra command described below can be inserted after running *Fiammetta.m* for drawing the border of the domain with thick black lines (*Figure 7*, *right*).

```
34  plot ([0 nx nx 0 0],[0 0 ny ny 0],'k','LineWidth',2);hold on;axis equal
```

Matlab© function *gra_cob.m*

```
 1  function [bar] = gra_cob
 2  bar=[ 0   0         0.5625
 3  0         0         0.6250
 4  0         0         0.6875
 5  0         0         0.7500
 6  0         0         0.8125
 7  0         0         0.8750
 8  0         0         0.9375
 9  0         0         1.0000
10  0         0.0625    1.0000
11  0         0.1250    1.0000
12  0         0.1875    1.0000
13  0         0.2500    1.0000
14  0         0.3125    1.0000
15  0         0.3750    1.0000
16  0         0.4375    1.0000
17  0         0.5000    1.0000
18  0         0.5625    1.0000
19  0         0.6250    1.0000
20  0         0.6875    1.0000
21  0         0.7500    1.0000
22  0         0.8125    1.0000
23  0         0.8750    1.0000
24  0         0.9375    1.0000
25  0         1.0000    1.0000
26  0.0625    1.0000    0.9375
27  0.1250    1.0000    0.8750
28  0.1875    1.0000    0.8125
29  0.2500    1.0000    0.7500
30  0.3125    1.0000    0.6875
```

```
 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 2: Matlab© function gra_cob.m - color bar

Matlab© function *gra_ist.m* - isotherm drawing

```
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 3: Matlab© function gra_ist.m - isotherm drawing

It is useful to remember some specifications of the variables: *B, G, T* are uni-column matrices, while, *lfi* and *fT* are uni-line matrices.
The drawings of *Figure 8* are obtained with a command calling *gra_ist.m* (*Table 3*), which is more efficient to represent isotherms than the Matlab command *fill*.

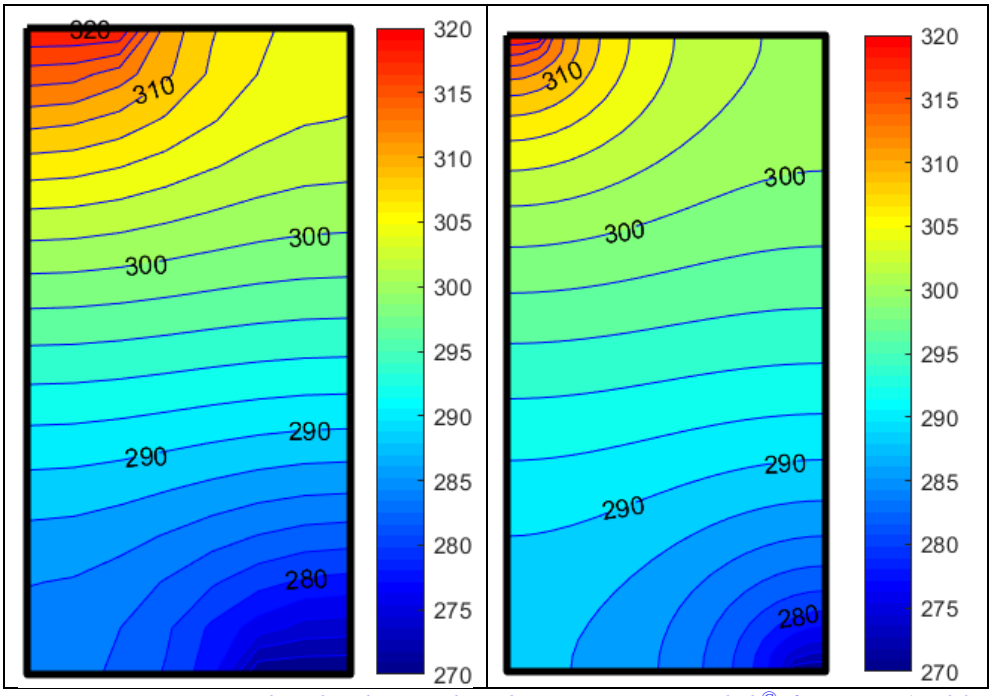


Figure 8: Temperature levels obtained with gra_ipa.m Matlab© function (Table 69)

On the left of *Figure 8*, we see the isothermal curves for a 7 x 14 mesh. On the right, with the same Dirichlet boundary conditions, we have the result for a 20 x 40 mesh. As explained in *Table 72*, the number of fixed nodes is the same for both tests, which means that the temperature peaks are sharper in the finer mesh. Both drawings are obtained with the function *gra_ist.m* (*Table 3*).

1.4 Neumann boundary conditions

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

$$< I(\tau) = \int_{\Omega} \frac{1}{2} k \ \vec{\nabla}\tau . \vec{\nabla}\tau \ d\Omega + \int_{S_2} \bar{q}_n \ \tau dS \ > \quad minimum \tag{14}$$

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

$$\int_{S_{2edge}} \bar{q}_n \tau \ d \ S_{el} = g_1 T_1 + g_2 T_2 \tag{15}$$

We express the edge temperature in term of edge weight functions

$$\tau_{edge} = T_1 \left(1 - \frac{x}{l} \right) + T_2 \frac{x}{l} \tag{16}$$

We can write the discretized functional:

$$\int_{S_{2edge}} \bar{q}_n \tau dS_{el} = \int_{S_{2edge}} \bar{q}_n \left(T_1 \left(1 - \frac{x}{l} \right) + T_2 \frac{x}{l} \right) dS_{el} \tag{17}$$

In matrix form, we have:

$$\text{With : } [T]=\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \text{ we have : } \int_{S_{2edge}} \bar{q}_n \tau dS_{el} = \int_{S_{2edge}} \bar{q}_n \begin{bmatrix} \left(1-\frac{x}{l}\right) & \frac{x}{l} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} dS_{el} \tag{18}$$

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

$$\int_{S_{2edge}} \bar{q}_n \tau dS_{el} = \int_{S_{2edge}} \bar{q}_n \begin{bmatrix} \left(1-\frac{x}{l}\right) & \frac{x}{l} \end{bmatrix} dS_{el} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \tag{19}$$

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

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix}^T = \int_{S_{2edge}} \bar{q}_n \begin{bmatrix} \left(1-\frac{x}{l}\right) & \frac{x}{l} \end{bmatrix} dS_{el} \tag{20}$$

To run the example of *Figure 9*, we replace the *lines 13* to *15* in *Fiammetta.m* by the instructions of *Table 4*.

```
13  n1 = nx+1;lfi = nn:-1:nn-n1+1;fix=ones(1,n1)*270;nf = n1;      % Imposed T
14  gh = zeros(nn,1);gh(1) = 20/ny;gh(nx+2:(nx+1):(nx+1)*(ny+1)-nx)= 40/ny;...
15      gh((nx+1)*(ny+1)-nx)=20/ny;                    % von Neumann conditions
16  disp(['Heat loading        : ',num2str(sum(gh),3),' W'])
```

Table 4: Instructions substituted to lines 16, 17 & 19 in Fiammetta_33_20210830.m

The high quality isotherms of *Figure 9* are obtained by running the instructions *34* to *36* of the procedure *Fiammetta.m*.
We have obtained the general method to build the second members of the heat transfer equations corresponding to the 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:

$$F_1 = \bar{q}\frac{el}{2} \quad ; \quad F_2 = \bar{q}\frac{el}{2} \tag{21}$$

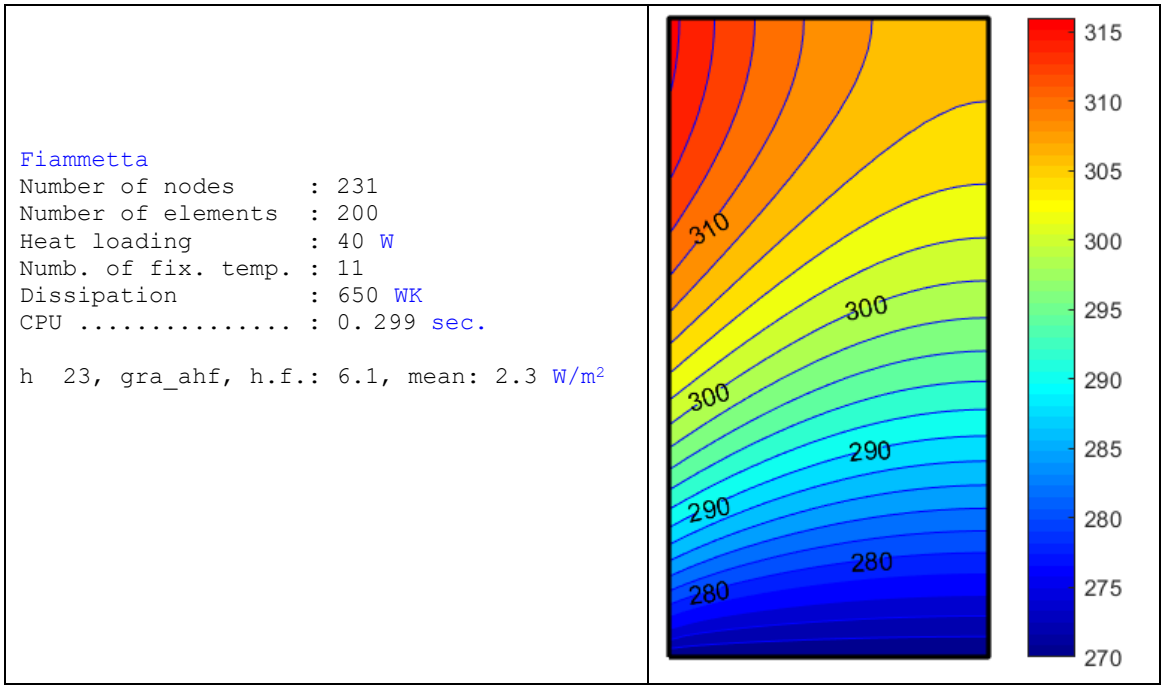


Figure 9: Isocurves for the problem with prescribed heat flows

1.5 Matlab© procedure: *Fiammetta.m* (*Table 1*)

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 boundary conditions, *nn* terms of the second member may be imposed.
Line 17 : Material properties: thickness in *m* and conductivity in $WK^{-1}m^{-1}$.
Lines 18 - 23 : Creation of the element conductivity of a square and assembly of the global conductivity matrix.

```
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
```

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 1*)

```
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 2*).
Line 28 : Drawing the temperature levels with the Matlab function *fill*.
Line 29 - 30 : End of the drawing sequence. Display of global results and processing time.
Lines 31 - 32 : Alternative visualization of the isotherms
Line 33 : Optional display of the mesh with node and element numbers (*Table 5: Localization matrix for the 2 x 4 mesh of Figure 2Table 5*).

```
disp([(1:8)' lK]):
     1     13     14     11     10
     2     14     15     12     11
     3     10     11      8      7
     4     11     12      9      8
     5      7      8      5      4
     6      8      9      6      5
     7      4      5      2      1
     8      5      6      3      2
```

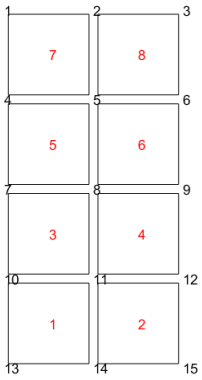


Table 5: Localization matrix for the 2 x 4 mesh of Figure 2

2. Tutorial II: Convection

In this chapter, we consider that the conductive model is 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) 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:

$$< I(\tau) = \int_{\Omega} \frac{1}{2} k \, \vec{\nabla}\tau \, . \, \vec{\nabla}\tau \, d\Omega + \frac{1}{2} \int_{S_3} h\left(\tau - \tau_f\right)^2 dS + \int_{S_2} \bar{q}_n \, \tau dS \; > \quad minimum \tag{22}$$

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.

$$\text{Functional at element level: } I_{el} = \frac{1}{2} \int_{S_3} h\left(\tau - \tau_f\right)^2 dS \tag{23}$$

In (22) and (23), h represents the convection coefficient. The fluid temperature τ_f is assumed uniform. We study an element edge that is a line segment of length L. We discretize the edge temperature as follows:

$$\tau = T_0\left(1 - \frac{x}{L}\right) + T_1 \frac{x}{L} \tag{24}$$

In this expression x varies between 0 and L. Replacing in the functional (23), we obtain:

$$\begin{aligned} I_{el} &= \frac{1}{2} \int_0^L h \left(\begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} - t_f \right)^2 dx \\ &= \frac{1}{2} h \int_0^L \left\{ [T]^T \begin{bmatrix} 1 - \frac{x}{L} \\ \frac{x}{L} \end{bmatrix} \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} [T] - 2 \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} [T] t_f + t_f^2 \right\} dx \end{aligned} \tag{25}$$

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]:

$$[T_{el}] = \begin{bmatrix} T_0 & T_1 & T_f \end{bmatrix}^T \tag{26}$$

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

$$I_{el} = \frac{1}{2} h T_{el}^T \left(\int_0^L \begin{bmatrix} 1-\frac{x}{L} \\ \frac{x}{L} \\ -1 \end{bmatrix} \begin{bmatrix} 1-\frac{x}{L} & \frac{x}{L} & -1 \end{bmatrix} dx \right) T_{el} \tag{27}$$

Developing the expression:

$$I_{el} = \frac{1}{2} h T_{el}^T \int_0^L \begin{bmatrix} \left(1-\frac{x}{L}\right)^2 & \left(1-\frac{x}{L}\right)\frac{x}{L} & -\left(1-\frac{x}{L}\right) \\ \left(1-\frac{x}{L}\right)\frac{x}{L} & \left(\frac{x}{L}\right)^2 & -\frac{x}{L} \\ -\left(1-\frac{x}{L}\right) & -\frac{x}{L} & 1 \end{bmatrix} dx \; T_{el} \tag{28}$$

After integrating and including the thickness e to ensure the coherence of units, we transform the functional (28) into an algebraic function of the nodal temperatures:

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

The integrals present in the discretized functional are easily computed:

$$\int_0^L \left(1-\frac{x}{L}\right)^2 dx = \int_0^L \left(1-2\frac{x}{L}+\frac{x^2}{L^2}\right) dx = L - L + \frac{L}{3} = \frac{L}{3} \tag{30}$$

$$\int_0^L \left(1-\frac{x}{L}\right)\frac{x}{L} dx = \int_0^L \left(\frac{x}{L}-\frac{x^2}{L^2}\right) dx = \frac{L}{2} - \frac{L}{3} = \frac{L}{6} \tag{31}$$

[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.

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

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

$$I_{el} = \frac{1}{2} h\ eL\ T_{el}^T \begin{bmatrix} \frac{1}{3} & \frac{1}{6} & -\frac{1}{2} \\ \frac{1}{6} & \frac{1}{3} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix} T_{el} = \frac{1}{2} T_{el}^T\ K_h\ T_{el} \tag{33}$$

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

$$K_h = \frac{ehL}{6} \begin{bmatrix} 2 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & 6 \end{bmatrix} \tag{34}$$

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.

$$K = K_k + K_h \tag{35}$$

The convection virtual nodes may be free or fixed. The convection matrix defined in (34) is computed in the function *fem_Kcv.m* (*Table 6*). 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.

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

```
1  function [K] = fem_Kcv(xyz,lc,hh) % K integration on a line segment
2  Q      = [xyz(lc(1),1:3); xyz(lc(2),1:3)];
3  L      = norm(Q(2,:)-Q(1,:));
4  K      = [2 1 -3;1 2 -3;-3 -3 6]*hh*L/6;
5  end
```

Table 6: Matlab© function fem_Kcv.m - convection matrix

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

```
2   % ....... 8. Standard rect. 1 rectangle ..............................
3   ha = 2;xyz_cao  = [0 0;1 0;1 ha;0 ha];car_cao =[1 2 3 4 ];nbo=4;
4   disp('L   4, Standard rect. 1 rectangle: ')
6   cs=5;nnv=2;
26  nni  = 1;if nni<1;nni=1;end     % Number nodes inserted on patches borders
33  pai  = .5;disp(['L  69, DT isotherms : ',num2str(pai), ' K'])
75  nfi=2;fT =[300 270];lfi=[no+1 no+2];          % Fix. of both virt. nodes
```

The above table shows the lines of Fiammetta enabled to run the test (*Figure 10*) on a rectangle (*Table 71*). It corresponds to a constant conductivity coefficient (*mat_cok.m*, *Table 8*).

```
Fiammetta_20211111
Standard 1 rect., Gi: 8
L   4, Me, Di, Ne    : 1  13   0
L   5, Co nnr nvn    : 1  0  2
L   6, rc, ra, cs    : 0  0  5
L   7, CAD interf.   : 4
L   9, N. pa., vert.: 1  4
L  11, num. nod side: 1
L  13, num elem side: 2
L  23, Domain area   : 2 m2
L  26, Num. elements: 4
L  28, Num. of DOF   : 11
L  59, Domain perim.: 6 m
L  61, Thickness     : 0.1 m
L  62, Conduction k  : 1 W/(mK)
L  63, DT isotherms  : 0.5 K
LD 73, Fixed nodes   : 10  11
LD 74, Fix. temper.  : 300  270 K
LD106, N. fix. nodes: 2
LD109, Fixed nodes   : 10  11
LD110, Fix. temper.  : 300  270 K
L  69, Fixed DOF     : 10  11
L  76, Convection h  : 1 W/(m2K)
c. 03, Numb. var. dK: 11
c. 04, N.virt c.nod.: 2
L  89, Numb. r.nodes: 6
L  90, R. nod. lcont: 2  6  3  4  8  1
L  91, Rad. vertices: 2  3  4  1
L  96, Anis. index   : 0
L 188, Total dissip.: 30 WK
L 191, Dis. in solid: 10 WK
L 192, DT   in solid: 10 K
L 201, Fixed DOF     : 10  11
L 202, Imposed temp.: 300       270 K
L 203, React. flows : 2        -2 W
L 205, End Me=1, CPU: 0.499 sec.
```

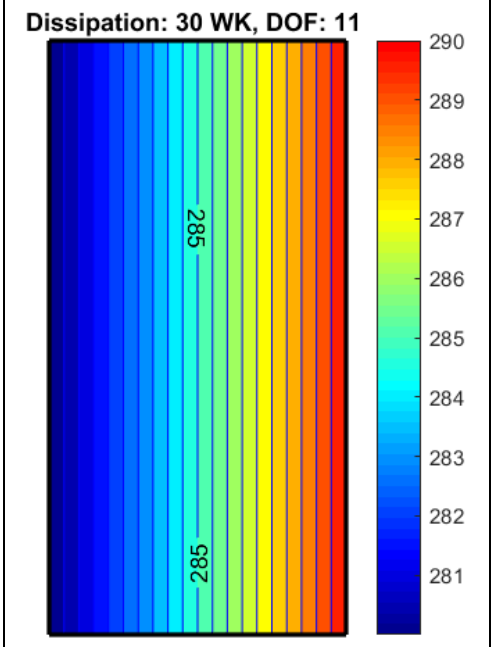


Figure 10: Isotherms orthogonal to both adiabatic boundaries. Biot number = 1

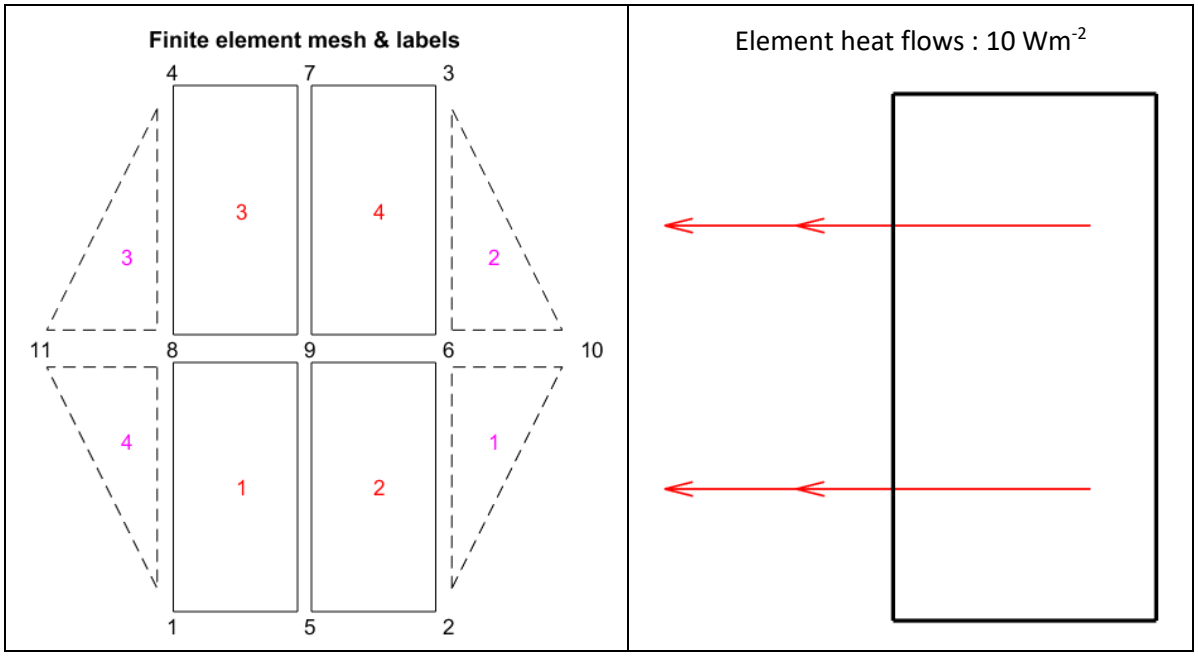


Figure 11: Nodes and elements numbering: heat flows with convective boundary conditions

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. 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*	$t_2 < t_3$

$$q_x = h(t_0 - t_1) = k\frac{(t_1 - t_2)}{w} = h(t_2 - t_3) \tag{36}$$

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

$$\beta = \frac{h\,w}{k}, \qquad \beta\,(t_0 - t_1) = t_1 - t_2 = \beta\,(t_2 - t_3) \tag{37}$$

From the first relation, we deduce:

$$t_1 = \frac{\beta\,t_0 + t_2}{1+\beta} \tag{38}$$

Now, we develop the second one:

$$\frac{\beta\,t_0 + t_2}{1+\beta} - t_2 = \beta\,t_2 - \beta\,t_3 \tag{39}$$

We obtain:

$$t_2 = \frac{t_0 + (1+\beta)t_3}{2+\beta} \tag{40}$$

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

$$t_1 = \frac{\beta\,t_0}{1+\beta} + \frac{t_0 + (1+\beta)\,t_3}{(2+\beta)\,(1+\beta)} = \frac{t_0}{1+\beta}\left(\beta + \frac{1}{(2+\beta)}\right) + \frac{t_3}{(2+\beta)} = \frac{(1+\beta)\,t_0 + t_3}{(2+\beta)} \tag{41}$$

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

$$(t_2 - t_1) = (t_3 - t_0)\frac{\beta}{2+\beta} \tag{42}$$

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 7*.

Biot numb. 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)	Dissip. Sol.
.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 7: 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_20211111
Standard 1 rect., Gi: 8
L   4, Me, Di, Ne   : 1  2  0
L   5, Co nnr nvn   : 1  0  2
L   6, rc, ra, cs   : 0  0  5
L   7, CAD interf.  : 4
L   9, N. pa., vert.: 1  4
L  11, num. nod side: 1
L  13, num elem side: 2
L  23, Domain area  : 2 m2
L  29, Num. elements: 4
L  31, Num. of DOF  : 11
L  62, Domain perim.: 6 m
L  64, Thickness    : 0.1 m
L  65, Conduction k : 1 W/(mK)
L  66, DT isotherms : 2 K
L  72, Fixed DOF    : 10  11
L  79, Convection h : 18 W/(m2K)
c. 03, dim. K Co cs : 11   1   5
c. 04, N.virt c.nod.: 2
L  92, Nu rad. nodes: 6
L  93, Rad. vertices: 2  3  4  1
L  94, R. nod. lcont: 2  6  3  4  8  1
L 101, Anis. index  : 0
L 188, Total dissip.: 81 WK
L 191, Dis. in solid: 72.9 WK
L 192, DT   in solid: 27 K
L 201, Fixed DOF    : 10  11
L 202, Imposed temp.: 300       270 K
L 203, React. flows : 5.4      -5.4 W
L 205, End Me=1, CPU: 0.434 sec.
```

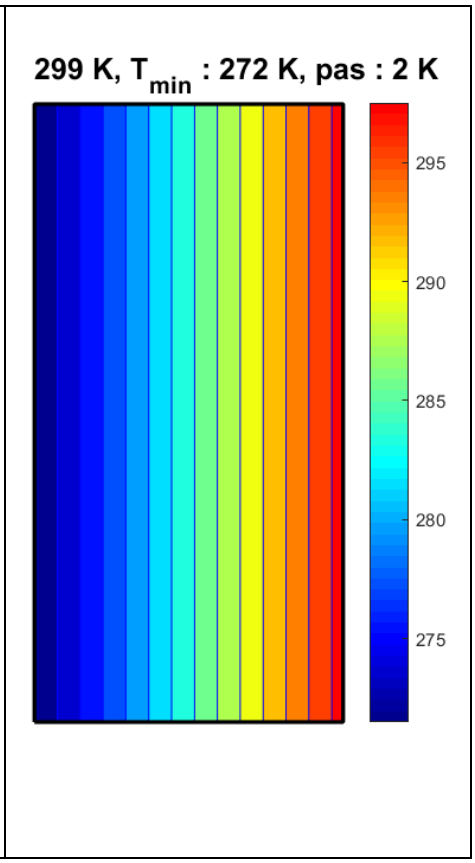


Figure 12: Example of heat transfer through a wall with a high Biot number $\beta = 18$

2.3 Material anisotropy

The *mat_cok.m* function is subdivided into two sequences. In *Table 8*, 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*.

Matlab© function *mat_cok.m* – non homogeneous conductivity

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

Table 8: Matlab© function mat_cok.m - non homogeneous conductivity

We now analyze a rectangular domain with part of the horizontal edges fixed at values of 320 *K* and 270 *K*. On these edges, *nnc* nodes are fixed. To identify the *DOF* of a patch edge, we use an instruction giving the number 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_20211111
Standard 1 rect., Gi: 9
L   4, Me, Di, Ne    : 1   3   0
L   5, Co nnr nvn    : 0   0   0
L   6, rc, ra, cs    : 0   0   5
L   7, CAD interf.   : 4
L   9, N. pa., vert.: 1   4
L  11, num. nod side: 16
L  13, num elem side: 17
L  23, Domain area   : 2 m2
L  26, Num. elements: 289
L  28, Num. of DOF   : 324
L  59, Domain perim.: 6 m
L  61, Thickness     : 1 m
L  62, Conduction k : 1 W/(mK)
L  63, DT isotherms : 2 K
LD 23, N. fix. nodes: 10
L  89, Numb. r.nodes: 36
L  91, Rad. vertices: 2   3   4   1
L  96, Anis. index   : 0
L 188, Total dissip.: 390 WK
L 191, Dis. in solid: 390 WK
L 192, DT    in solid: 50 K
L 205, End Me=1, CPU: 0.354 sec.
```

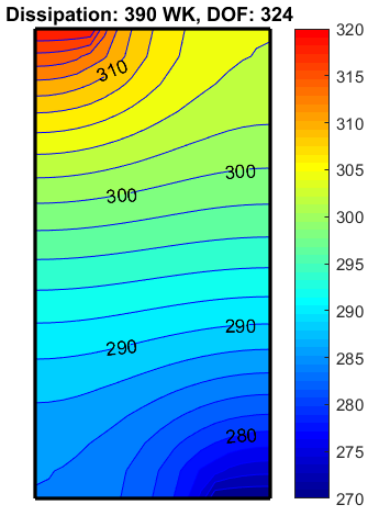


Figure 13: Isocurves in a rectangle with fixed DOF on horizontal edges, uniform k

It is now 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 an horizontal strip. This modification has to be performed by modifying the function *mat_cok.m*. The elements are numbered from left to right and from bottom to top (*Table 8*).
When we are using the function used to detect quantities along the boundary of a rectangular meshed domain is *cao_bou.m* (*Table 9*). To achieve this operation, we use the input matrices of the procedure *cad_bou.m*: *car_cao* is the matrix of the input patches, *bor* and *pbo* are matrices created in *cad_mes.m* to store the characteristics of the edges.

Matlab© function *cad_bou.m* – selection of nodal quantity *gh* along the border

```
1  function [gperi,li] = cad_bou(car_cao,bor,pbo,gh)% Loads on patch boundary
2  a     = [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
3  b     = [car_cao(1,2) bor(pbo(1,2),5):bor(pbo(1,2),6) car_cao(1,3)];
4  c     = [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
5  d     = [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)];
6  li    = [a(1:size(a,2)-1)  b(1:size(b,2)-1) c(1:size(c,2)-1) ...
7           d(1:size(d,2)-1)];     % List of the nodes from the 4 patch sides
8  gperi = gh(li);
9  end
```

Table 9: Matlab© function cad_bou.m - nodal quantity along patch 1 boundary

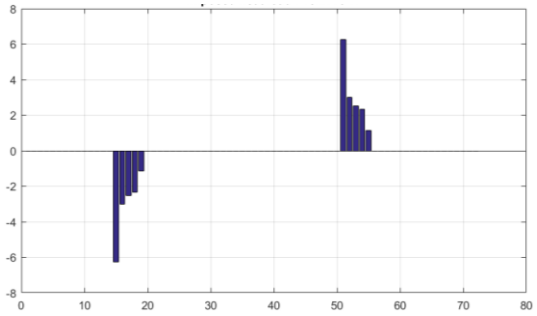


Figure 14: Heat input and output for example of Figure 13 - boundary load: 15.3 W

```
Fiammetta_20211111
Standard 1 rect., Gi: 9
L    4, Me, Di, Ne    : 1   3   0
L    5, Co nnr nvn    : 0   0   0
L    6, rc, ra, cs    : 0   0   5
L    7, CAD interf.   : 4
L    9, N. pa., vert.: 1   4
L   11, num. nod side: 49
L   13, num elem side: 50
L   23, Domain area   : 2 m2
L   26, Num. elements: 2500
L   28, Num. of DOF   : 2601
L   59, Domain perim.: 6 m
L   61, Thickness     : 1 m
L   62, Conduction k : 1 W/(mK)
L   63, DT isotherms : 2 K
LD 23, N. fix. nodes: 10
L   89, Numb. r.nodes: 102
L   91, Rad. vertices: 2   3   4   1
L   96, Anis. index   : 0
L 188, Total dissip.: 275 WK
L 191, Dis. in solid: 275 WK
L 192, DT    in solid: 50 K
L 205, End Me=1, CPU: 1.13 sec.
```

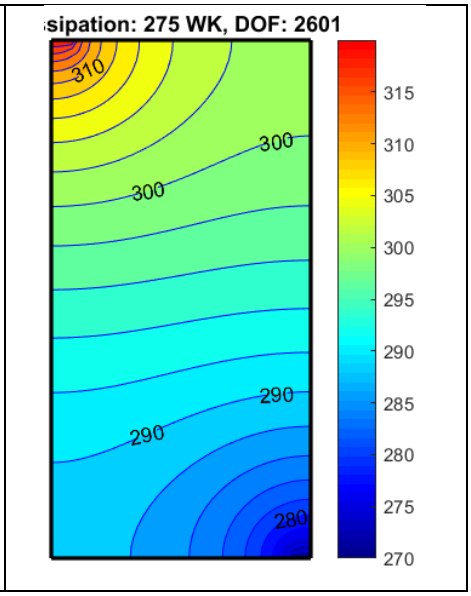


Figure 15: Isocurves for isotropic material

```
Fiammetta_20211111
Standard 1 rect., Gi: 9
L    4, Me, Di, Ne    : 1   3   0
L    5, Co nnr nvn    : 0   0   0
L    6, rc, ra, cs    : 0   0   5
L    7, CAD interf.   : 4
L    9, N. pa., vert.: 1   4
L   11, num. nod side: 49
L   13, num elem side: 50
L   23, Domain area   : 2 m2
L   26, Num. elements: 2500
L   28, Num. of DOF   : 2601
L   59, Domain perim.: 6 m
L   61, Thickness     : 1 m
L   62, Conduction k : 1 W/(mK)
L   63, DT isotherms : 2 K
LD 23, N. fix. nodes: 10
L   89, Numb. r.nodes: 102
L   91, Rad. vertices: 2   3   4   1
L   96, Anis. index   : 3
Lm 47, Anis. index   : 3
Lm 48, k & coef*k    : 1 0.1 W/(m K)
Lm 49, Thickn. ratio: 0.04
L 188, Total dissip.: 259 WK
L 191, Dis. in solid: 259 WK
L 192, DT    in solid: 50 K
L 205, End Me=1, CPU: 1.15 sec.
```

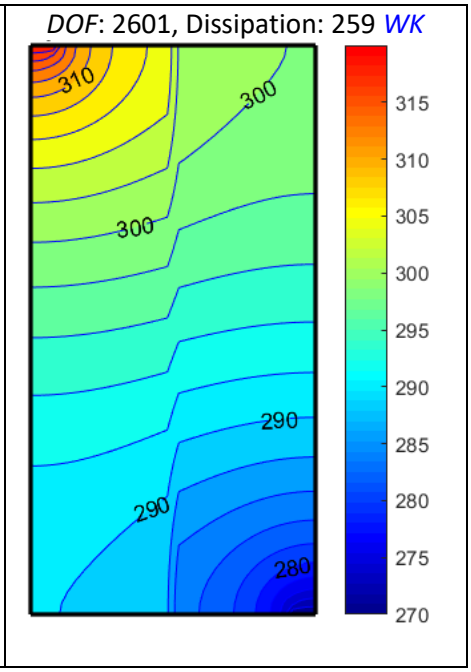


Figure 16: Isocurves for material with vertical strip, 0.1 k (variable fa = .1)

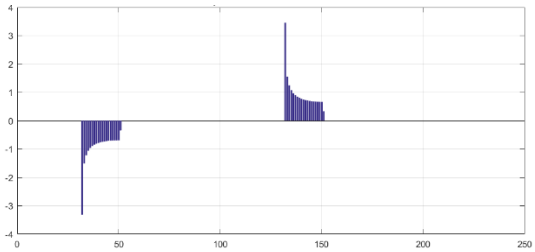


Figure 17: Heat input and output for horizontal strip – heat load: 18.9 W

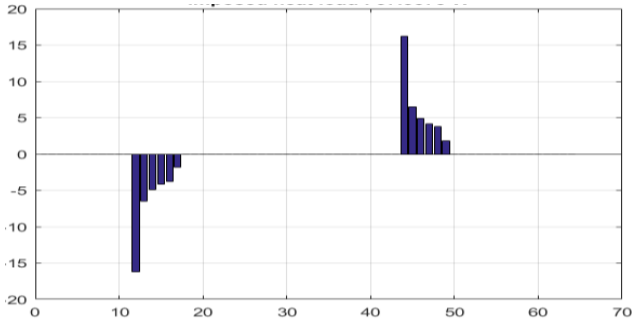


Figure 18: Boundary heat loads: 37.4 W - vertical strip of high conductivity

```
Fiammetta_20211111
Standard 1 rect., Gi: 9
L    4, Me, Di, Ne    : 1   3   0
L    5, Co nnr nvn    : 0   0   0
L    6, rc, ra, cs    : 0   0   5
L    7, CAD interf.   : 4
L    9, N. pa., vert.: 1   4
L   11, num. nod side: 15
L   13, num elem side: 16
L   23, Domain area   : 2 m2
L   29, Num. elements: 256
L   31, Num. of DOF   : 289
L   62, Domain perim.: 6 m
L   64, Thickness     : 1 m
L   65, Conduction k : 1 W/(mK)
L   66, DT isotherms : 2.5 K
LD 22, N. fix. nodes: 10
L   92, Number. nodes: 34
L   93, Num. vertices: 2   3   4   1
Lm 47, Anis. index   : 3
Lm 48, k & coef*k    : 1   10000 W/(m K)
Lm 49, Thickn. ratio: 0.125
L 188, Total dissip.: 741 WK
L 191, Dis. in solid: 741 WK
L 192, DT    in solid: 50 K
L 205, End Me=1, CPU: 0.343 sec.
```

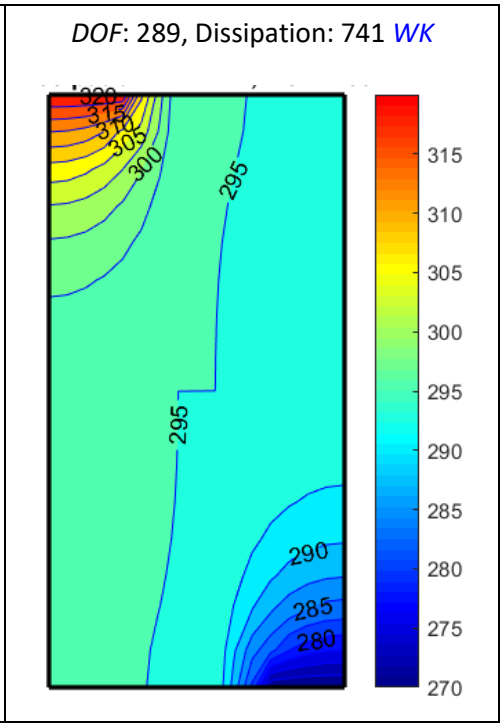


Figure 19: Isocurves for the vertical strip 2 elements wide, 16 x 16 mesh

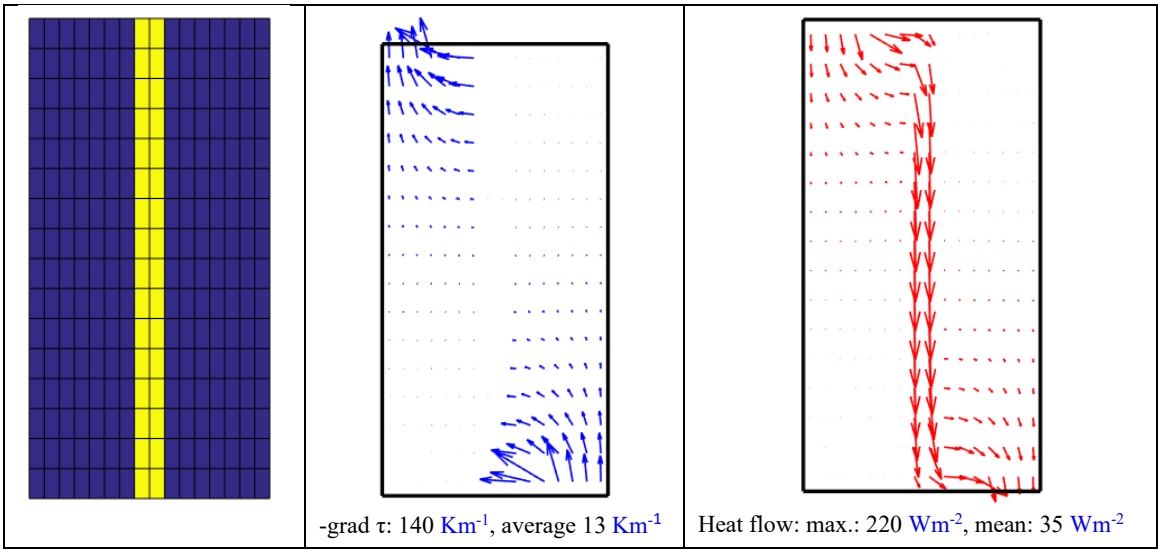


Figure 20: Heat flows and temperature gradients for a vertical thermal bridge

The heat flows on top and bottom horizontal sides are progressing from 15.3 *W* (*Figure 14*) in the homogeneous case to 18.9 *W* in the case of the horizontal strip (*Figure 17*) and finally to 37.4 *W* in the case of the vertical one (*Figure 18*). In the example shown in *Figure 19*, we have tested the function on a domain involving a vertical strip in which the ratio of conductivities is equal to 10000. In *Figure 21*, we test the same mesh with a vertical strip of insulating material for which the conductivity is ten times smaller than the general one.

```
Fiammetta_20211111
Standard 1 rect., Gi: 9
L    4, Me, Di, Ne    : 1   3   0
L    5, Co nnr nvn    : 0   0   0
L    6, rc, ra, cs    : 0   0   5
L    7, CAD interf.   : 4
L    9, N. pa., vert.: 1   4
L   11, num. nod side: 31
L   13, num elem side: 32
L   23, Domain area   : 2 m2
L   29, Num. elements: 1024
L   31, Num. of DOF   : 1089
L   62, Domain perim.: 6 m
L   64, Thickness     : 1 m
L   65, Conduction k  : 1 W/(mK)
L   66, DT isotherms : 2.5 K
LD 22, N. fix. nodes: 10
L   92, Number. nodes: 66
L   93, Num. vertices: 2   3   4   1
Lm 47, Anis. index   : 3
Lm 48, k & coef*k    : 1 0.1 W/(m K)
Lm 49, Thickn. ratio: 0.0625
L 188, Total dissip.: 283 WK
L 191, Dis. in solid: 283 WK
L 192, DT    in solid: 50 K
L 205, End Me=1, CPU: 0.482 sec.
```

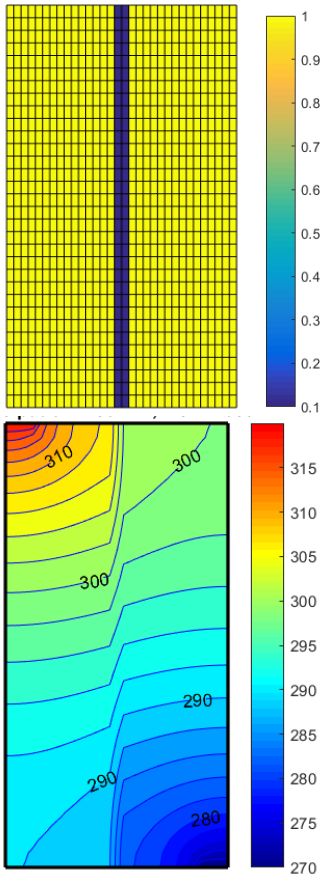


Figure 21: Isocurves in presence of a vertical small conductivity strip (0.1 k)

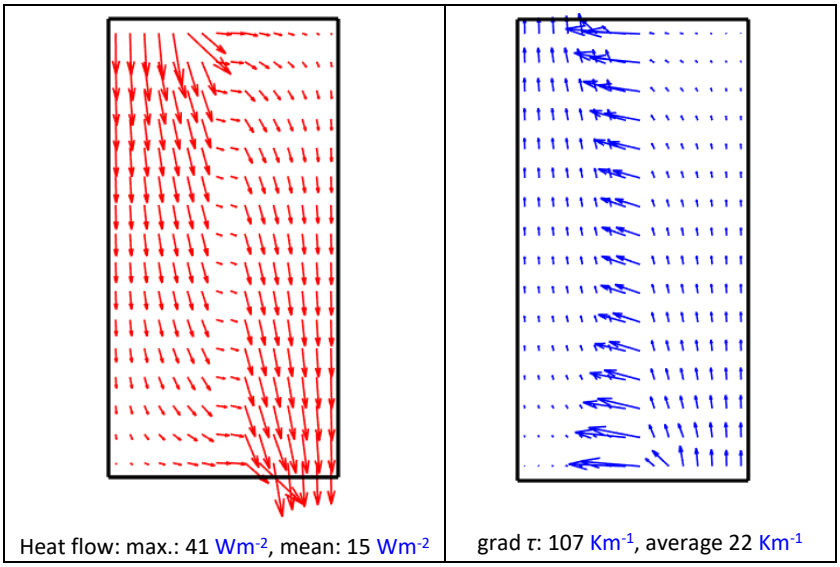


Figure 22: Heat flows and temperature gradients (vertical strip with small conductivity)

It is possible to use two additional visualizations (*Figure 23*), the first concerns the anisotropy of material characteristics (conductivity coefficient and thickness): This illustration is created in *mat_coK.m* (*Table 8*). The second represents scalars defined element by element, like the module of the heat flow.

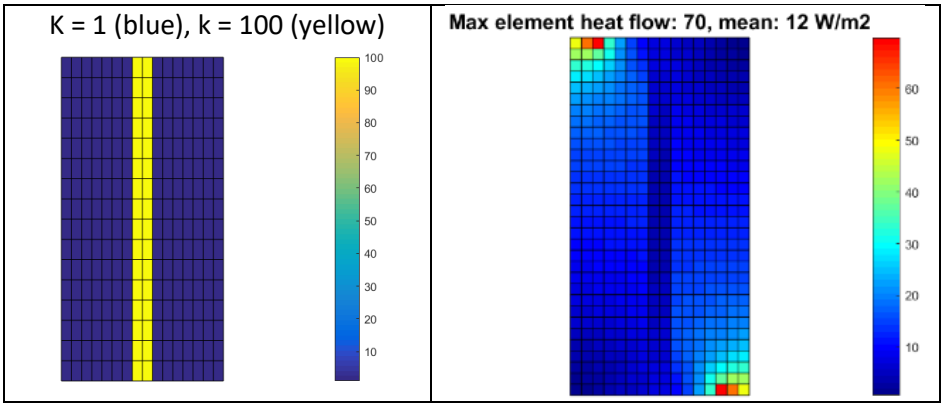


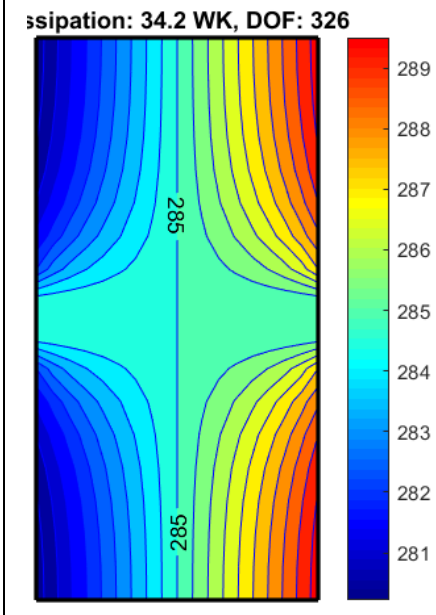
Figure 23: Visualizations of scalars defined element by element

2.4 Thermal bridge

With respect to the example of *Figure 10*, we simply modify the function *mat_cok.m* (*Table 8*). The effect of the thermal bridge is important, but here, the conductivity ratio is 1000. If this ratio falls to 10, the effect is still visible. This test shows the importance of thermal bridges in building design (*Figure 24* & *Figure 25*).

We observe that the heat rate crossing the domain is equal to 3.6 *W* when the conductivity is uniform. It reaches the value of 9.4 *W* if the conductivity in the thermal bridge is 1000 times the conductivity in the other part of the domain. Specific data for this test are present in the next figures. Two tests are realized: with conductivity of 1000 $Wm^{-1}K^{-1}$ and 10 $Wm^{-1}K^{-1}$ in the horizontal thermal bridge.

```
Fiammetta_20211111
Standard 1 rect., Gi: 8
L    4, Me, Di, Ne    : 1   13    0
L    5, Co nnr nvn    : 1   0   2
L    6, rc, ra, cs    : 0   0   5
L    7, CAD interf.   : 4
L    9, N. pa., vert.: 1   4
L   11, num. nod side: 16
L   13, num elem side: 17
L   23, Domain area   : 2 m2
L   26, Num. elements: 289
L   28, Num. of DOF   : 326
L   59, Domain perim.: 6 m
L   61, Thickness     : 0.1 m
L   62, Conduction k  : 1 W/(mK)
L   63, DT isotherms  : 0.5 K
LD 73, Fixed nodes   : 325   326
LD 74, Fix. temper.  : 300   270 K
LD106, N. fix. nodes: 2
LD109, Fixed nodes   : 325   326
LD110, Fix. temper.  : 300   270 K
L   69, Fixed DOF     : 325   326
L   76, Convection h  : 1 W/(m2K)
c. 03, Numb. var. dK: 326
c. 04, N.virt c.nod.: 2
L   89, Numb. r.nodes: 36
```



```
L  91, Rad. vertices: 2  3  4  1
L  96, Anis. index  : 1
Lm 48, k & coef*k   : 1 1000 W/(m K)
Lm 49, Thickn. ratio: 0.058824
L 188, Total dissip.: 34.2 WK
L 191, Dis. in solid: 7.81 WK
L 192, DT   in solid: 9.56 K
L 201, Fixed DOF    : 325  326
L 202, Imposed temp.: 300  270 K
L 203, React. flows : 2.28  -2.28 WL
205, End Me=1, CPU: 0.349 sec.
```

Figure 24: Isocurves - horizontal thermal bridge with high conductivity (1000 k)

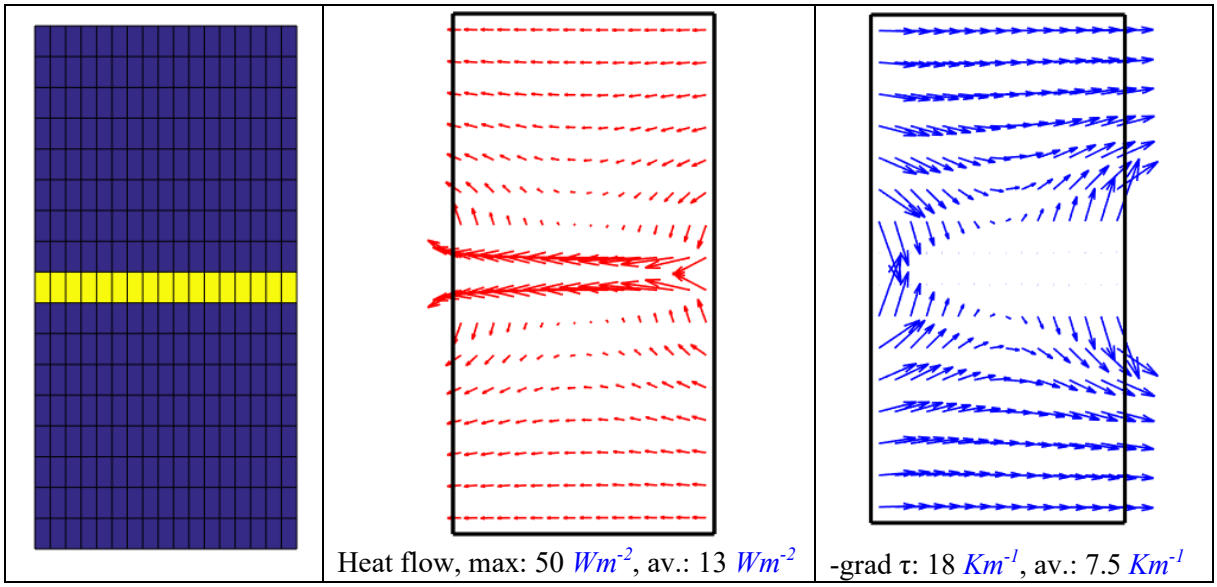
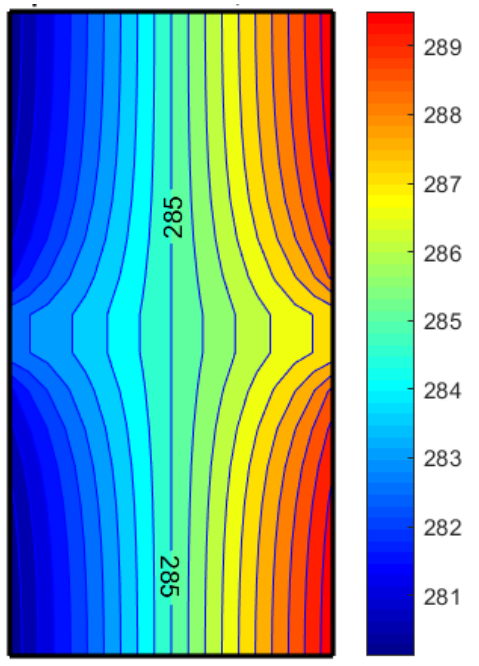


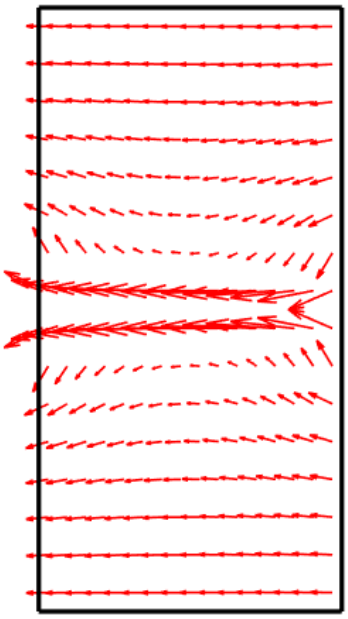
Figure 25: Heat flows and temperature gradients (horizontal strip with high conductivity)

```
Fiammetta_20211111
Standard 1 rect., Gi: 8
L   4, Me, Di, Ne   : 1  13   0
L   5, Co nnr nvn   : 1  0  2
L   6, rc, ra, cs   : 0  0  5
L   7, CAD interf.  : 4
L   9, N. pa., vert.: 1  4
L  11, num. nod side: 16
L  13, num elem side: 17
L  23, Domain area  : 2 m2
L  26, Num. elements: 289
L  28, Num. of DOF  : 326
L  59, Domain perim.: 6 m
L  61, Thickness    : 0.1 m
L  62, Conduction k : 1 W/(mK)
L  63, DT isotherms : 0.5 K
LD 73, Fixed nodes  : 325  326
LD 74, Fix. temper. : 300  270 K
LD106, N. fix. nodes: 2
LD109, Fixed nodes  : 325  326
LD110, Fix. temper. : 300  270 K
L  69, Fixed DOF    : 325  326
L  76, Convection h : 1 W/(m2K)
c. 03, Numb. var. dK: 326
c. 04, N.virt c.nod.: 2
L  89, Numb. r.nodes: 36
L  91, Rad. vertices: 2  3  4  1
```

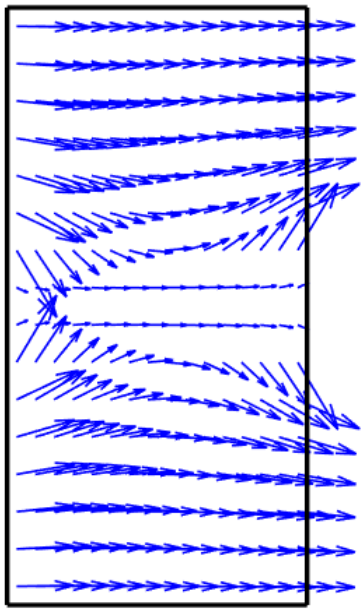


```
L  96, Anis. index  : 1
Lm 48, k & coef*k   : 1  10 W/(m K)
Lm 49, Thickn. ratio: 0.058824
L 188, Total dissip.: 32.3 WK
L 191, Dis. in solid: 8.99 WK
L 192, DT   in solid: 9.77 K
L 201, Fixed DOF    : 325  326
L 202, Imposed temp.: 300     270 K
L 203, React. flows : 2.15  -2.15 W
L 205, End Me=1, CPU: 0.352 sec.
```

Figure 26: Isocurves with the presence of horizontal smooth thermal bridge (10 k)



Heat flow, max: 36 Wm^{-2}, mean: 12 Wm^{-2}



-grad τ: 14 Km^{-1}, mean: 8.1 Km^{-1}

Figure 27: Heat flows and temperature gradients (horizontal smooth thermal bridge)

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*].

$$[Q]=\begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{bmatrix}=\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}=[X \quad Y] \tag{43}$$

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*]:

$$\begin{aligned} x(s,t)&=[F][X]=\begin{bmatrix}(1-s)(1-t) & s(1-t) & st & (1-s)t\end{bmatrix}[X] \\ y(s,t)&=[F][Y]=\begin{bmatrix}(1-s)(1-t) & s(1-t) & st & (1-s)t\end{bmatrix}[Y] \end{aligned} \tag{44}$$

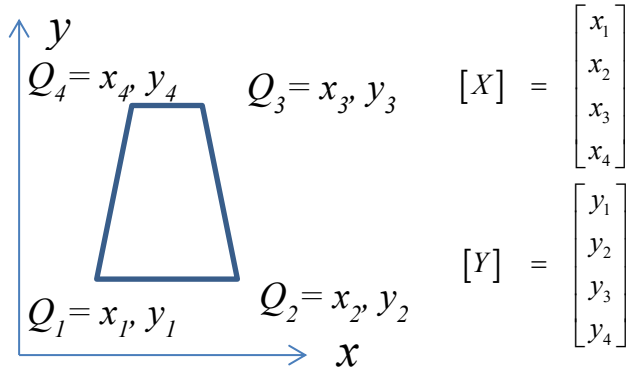
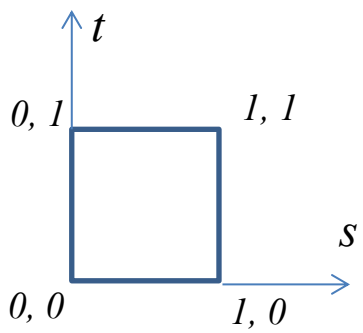
Cartesian space *x, y* *dS = dx dy*	Parametric space *s, t* *dS = J(s, t) ds dt*
	

Table 10: Coons patch definition in Cartesian and parametric spaces

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

$$\begin{aligned} x(s=1/2,t=1/2)&=\begin{bmatrix}1/4 & 1/4 & 1/4 & 1/4\end{bmatrix}[X]=(x_1+x_2+x_3+x_4)/4 \\ y(s=1/2,t=1/2)&=\begin{bmatrix}1/4 & 1/4 & 1/4 & 1/4\end{bmatrix}[Y]=(y_1+y_2+y_3+y_4)/4 \end{aligned} \tag{45}$$

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

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

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

$$[J] = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} -(1-t) & (1-t) & t & -t \end{bmatrix}[X] & \begin{bmatrix} -(1-t) & (1-t) & t & -t \end{bmatrix}[Y] \\ \begin{bmatrix} -(1-s) & -s & s & (1-s) \end{bmatrix}[X] & \begin{bmatrix} -(1-s) & -s & s & (1-s) \end{bmatrix}[Y] \end{bmatrix} \tag{47}$$

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

$$[J]_{s=t=\frac{1}{2}} = \frac{1}{2}\begin{bmatrix} [-1 & 1 & 1 & -1][X] & [-1 & 1 & 1 & -1][Y] \\ [-1 & -1 & 1 & 1][X] & [-1 & -1 & 1 & 1][Y] \end{bmatrix} \tag{48}$$

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

$$[J]_{s=t=\frac{1}{2}} = \frac{1}{2}\begin{bmatrix} x_2 + x_3 - x_1 - x_4 & y_2 + y_3 - y_1 - y_4 \\ x_4 + x_3 - x_1 - x_2 & y_4 + y_3 - y_1 - y_2 \end{bmatrix} \tag{49}$$

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:

$$J_{s=t=\frac{1}{2}} = \frac{1}{2}\left((x_2 + x_3 - x_1 - x_4)(y_4 + y_3 - y_1 - y_2) - (x_4 + x_3 - x_1 - x_2)(y_2 + y_3 - y_1 - y_4)\right) \tag{50}$$

Finally:

$$J_{s=t=\frac{1}{2}} = \frac{1}{2}\left((x_2 - x_4)(y_3 - y_1) + (x_3 - x_1)(y_4 - y_2)\right) \tag{51}$$

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).

$$\begin{bmatrix} \frac{\partial \tau}{\partial s} \\ \frac{\partial \tau}{\partial t} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \end{bmatrix} = [J] \begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \end{bmatrix} = [J]\vec{\nabla}\tau \tag{52}$$

After inverting (52), we obtain:

$$\vec{\nabla}\tau = \begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \end{bmatrix} = [J]^{-1} \begin{bmatrix} \frac{\partial \tau}{\partial s} \\ \frac{\partial \tau}{\partial t} \end{bmatrix} \tag{53}$$

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:

$$\tau = \begin{bmatrix} (1-s)(1-t) & s(1-t) & st & (1-s)t \end{bmatrix} [T] \tag{54}$$

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

$$\begin{bmatrix} \frac{\partial \tau}{\partial s} \\ \frac{\partial \tau}{\partial t} \end{bmatrix} = \begin{bmatrix} -(1-t) & (1-t) & t & -t \\ -(1-s) & -s & s & (1-s) \end{bmatrix} [T] \tag{55}$$

In the barycenter:

$$\begin{bmatrix} \frac{\partial \tau}{\partial s} \\ \frac{\partial \tau}{\partial t} \end{bmatrix}_{s=t=\frac{1}{2}} = \frac{1}{2} \begin{bmatrix} -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix} [T] \tag{56}$$

The two components of the following equation correspond to the *lines 11 & 12* of the function *gra_atg.m* (*Table 67*). They represent the temperature gradient

$$\vec{\nabla}\tau = \begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \end{bmatrix} = [J]^{-1} \frac{1}{2} \begin{bmatrix} -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix} [T] \tag{57}$$

Being able to calculate the temperature gradients, it is now possible to compute the conductivity matrices. The Matlab© function *fem_Kco.m* (*Table 11*) 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*.

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

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

Table 11: Matlab© function fem_Kco.m - element conductivity matrix

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 12*, 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 12: Numerically integrated conductivity matrix

To obtain the effective conductivity matrix, the result displayed in *Table 12* is multiplied by *k th* / 6, where *k* is the conductivity coefficient and *th* the thickness.

3.2 *CAD* model of the domain to be analyzed

The inputs of a CAD model involve three kinds of data. The *xyz_cao* matrix contains the coordinates of the nodes, *car_cao* is giving the patches definition and *nbo* is the number of interfaces limiting the patches. The size of the matrix *xyz_cao* must be the maximum numbering of the nodes defined in the matrix *car_cao*. Because these numbers represent *DOF*, they must all be present in the matrix *car_cao*. For the example of *Figure 36*, we have, on the left of *Table 13*, 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

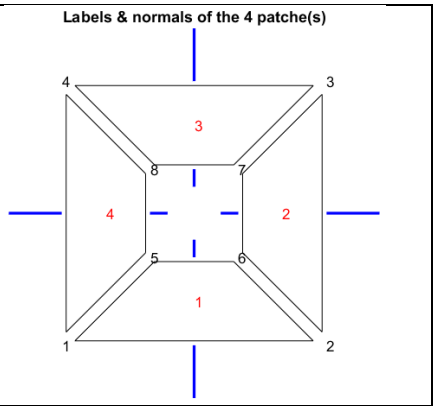
Table 13: Instructions used to display input data of Figure 36

Lines 16 - 19 of *Table 71* are generating the Coons patches displayed in *Table 13* and drawn in *Figure 36*.

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 14*, we see the four instructions used to determine the *DOF* of the cavity of *Figure **28***. The cavity is defined by *lines 1 - 4* of *Table 71*. The four *CAD* patches and their related data are shown in *Figure 28*. The matrix *bor* corresponds to a mesh of 100 elements counting four nodes on each patch side.

```
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 28: CAD data defining a domain surrounding a cavity

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 nodes 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 14*, we select the nodes of the second side of the third patch (*Figure 28*). 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 29*.

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 14: 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 14*):

```
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 14*):

```
lg =       3   37   38   39   40   4
```

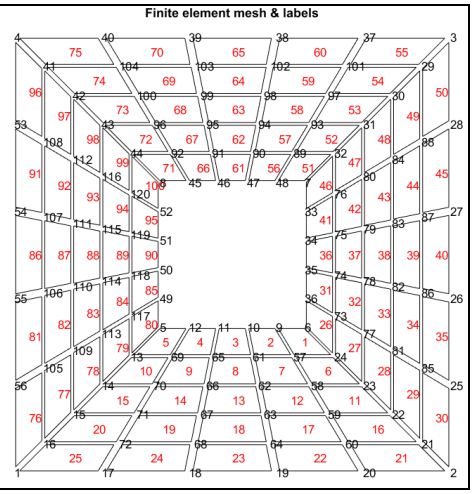
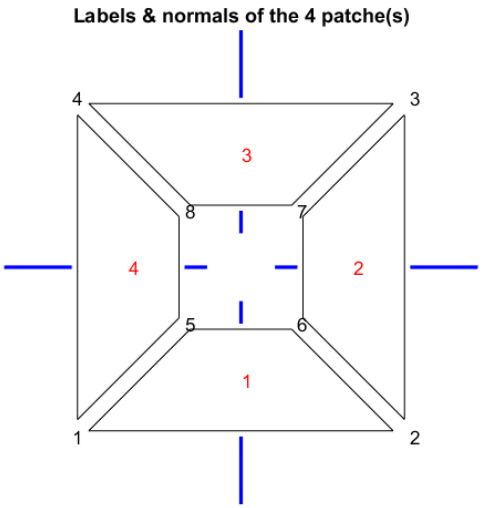


Figure 29: Finite element mesh corresponding to the CAD definition of Figure 28

3.4 Cavity with adiabatic hole

```
Fiammetta_20211111
Standard cavity,  Gi: 4
L    4, Me, Di, Ne    : 1   4   0
L    5, Co nnr nvn    : 0   0   0
L    6, rc, ra, cs    : 0   0   5
L    7, CAD interf.   : 12
L    9, N. pa., vert.: 4   8
L   11, num. nod side: 4
L   13, num elem side: 5
L   23, Domain area  : 8 m2
L   26, Num. elements: 100
L   28, Num. of DOF  : 120
L   59, Domain perim.: 16 m
L   61, Thickness    : 1 m
L   62, Conduction k : 1 W/(mK)
L   63, DT isotherms : 1 K
L   89, Numb. r.nodes: 12
L   91, Rad. vertices: 5   1   2   6
L   96, Anis. index  : 0
L 188, Total dissip.: 357 WK
L 191, Dis. in solid: 357 WK
L 192, DT   in solid: 30 K
L 205, End Me=1, CPU: 0.688 sec.

h   23, gra_ahf, h.f.: ma: 14, av: 9.7 W/m²
```



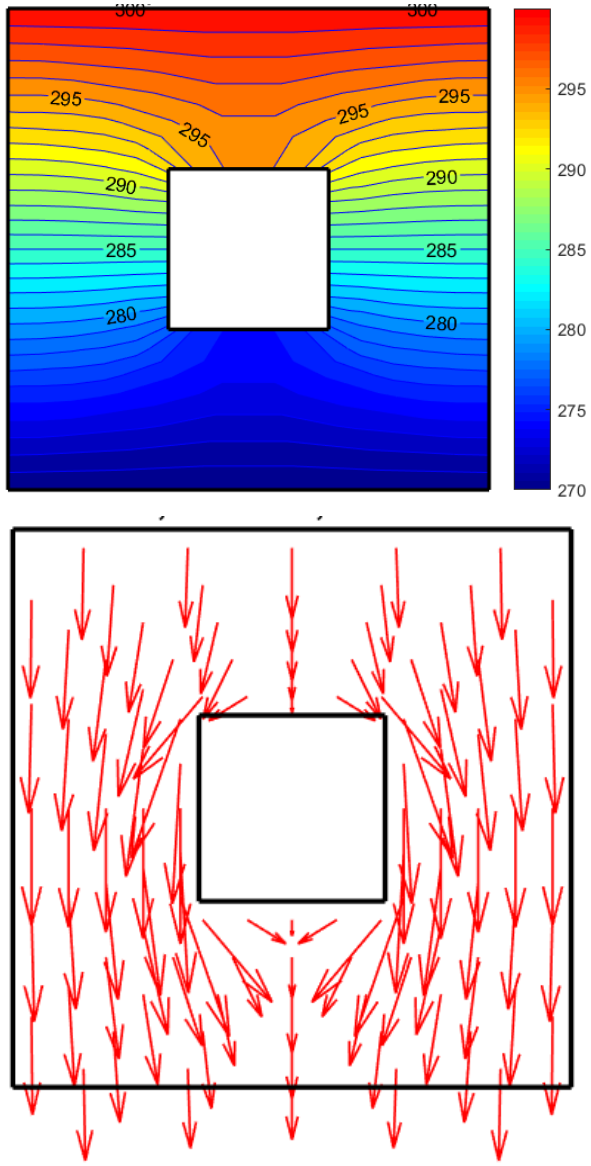


Figure 30: Heat flow around an adiabatic cavity

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 65* & *Figure 31*). Hereafter, the same domain will be defined with different *CAD* models: the long or the short horizontal parts of the model are trapezoidal in *Figure 34*, the domain is only composed of rectangular patches in *Figure 35* and, finally, it is composed of three trapezoidal patches. In these four situations, the dissipative function is respectively equal to 0.993 *WK*, 0.991 *WK*, 0.987 *WK* and, finally, 0.979 *WK*. Due to the convergence property of a pure conduction model with Dirichlet boundary conditions, the lowest value of the dissipative function is the best one [Debongnie, Zhong & Beckers 1995]. With the last model, it converges to 0.979 *WK* when we have 8241 *DOF*. So, the dissipation is decreasing when the mesh is finer [Debongnie & Beckers 2001]. The Matlab© functions

gra_mel.m (*Table 65*) and *gra_mnl.m* (*Table 66*) enable the visualization of the finite element mesh (*Figure 32*).

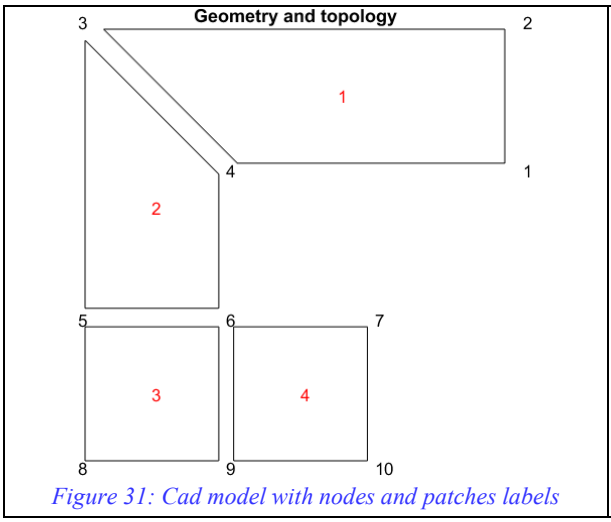


Figure 31: Cad model with nodes and patches labels

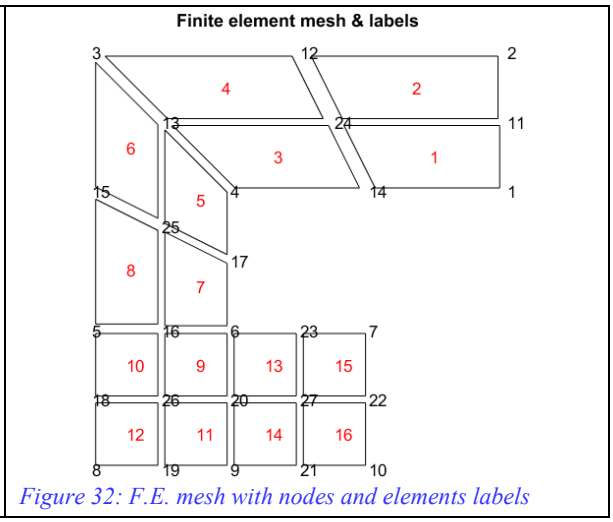


Figure 32: F.E. mesh with nodes and elements labels

Two functions are fundamental in *Fiammetta.m*: *cad_mes.m* (*Table 15*) and *cad_edg.m* (*Table 16*), which is called in *cad_mes.m*. They allow defining the topology of the *CAD* model through the construction of matrices *bor* and *pbo* that describe the patch interfaces.

Matlab© function *cad_mes.m*

```
1   function [xyc,lK,bor,pbo] = cad_mes(xyc,lca,ni,nbo)
2   nec         = size(lca,1);                          % Number of CAD patches
3   ndo         = size(xyc,1);nd = ndo;                   % Number of CAD nodes
4   [bor, pbs] = cad_edg(lca,nbo); % Compute the nbo patch sides in bor matrix
5   % =========================================================================
6   if ni                == 1   % ni = 1 : one node generated on the interfaces
7       lai               = zeros(1,nbo);                % List of border edges
8       for i             = 1: nbo
9           lai(i)        = nd + i;
10          bor(i,5)      = lai(i);
11          bor(i,6)      = bor(i,5);
12          xyc(nd+i,:)   = (xyc(bor(i,1),:)+xyc(bor(i,2),:))/2;
13      end
14  else              % More than 1 node have to be generated on the interfaces
15      k                 = nd;
16      for i             = 1 : nbo           % nbo is the number of interfaces
17          bor(i,5)      = nd + (i-1)*ni+1;
18          bor(i,6)      = nd + i*ni;
19          for j         = 1 : ni
20              A         = xyc(bor(i,1),:);B = xyc(bor(i,2),:);
21              k         = k + 1;
22              xyc(k,:) = A + (B-A)*j/(ni+1);  % coord. of the inerface nodes
23          end
24      end
25  end
26  pbo = sign(pbs(:,:)).*pbs;
27  nna               = ndo + nbo*ni;   % Numb. of nodes after interfaces gen.
28  lK                = zeros(nec*(ni+1)^2,4);nu = 0;
29  for n             = 1:nec % Loop on the CAD patches ......................
30      to            = zeros(ni+2,ni+2);     % Gen. to = list of nodes matrix
31      to(1,1)       = lca(n,1);            % Start with the 4 patch vertices
32      to(1,ni+2)    = lca(n,2); to(ni+2,ni+2) = lca(n,3)
33      to(ni+2,1)    = lca(n,4);                          % End patch vertices
34      if bor(pbo(n,1),3) == n                      % Line 1 of patch matrix to
35          to(1    ,2:ni+1) = bor(pbo(n,1),5)    :bor(pbo(n,1),6);
36      else
37          to(1    ,2:ni+1) = bor(pbo(n,1),6):-1:bor(pbo(n,1),5);
38      end
39      if bor(pbo(n,3),3) == n                 % Line ni+2 of patch matrix to
40          to(ni+2,2:ni+1) = bor(pbo(n,3),6):-1:bor(pbo(n,3),5);
41      else
42          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 15: 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 16: Matlab© function cad_edg.m - CAD mesh interfaces

```
Fiammetta_20211111
C shape, 4 patches, Gi: 5
L    4, Me, Di, Ne    : 1   5   0
L    5, Co nnr nvn    : 0   0   0
L    6, rc, ra, cs    : 0   0   5
L    7, CAD interf.   : 13
L    9, N. pa., vert.: 4   10
L   11, num. nod side: 4
L   13, num elem side: 5
L   23, Domain area   : 6 m2
L   29, Num. elements: 100
L   31, Num. of DOF   : 126
L   62, Domain perim.: 14 m
L   64, Thickness     : 0.1 m
L   65, Conduction k : 1 W/(mK)
L   66, DT isotherms : 0.5 K
L 188, Total dissip.: 0.993 WK
L 191, Dis. in solid: 0.993 WK
L 192, DT   in solid: 10 K
L 205, End Me=1, CPU: 1.47 sec.
```

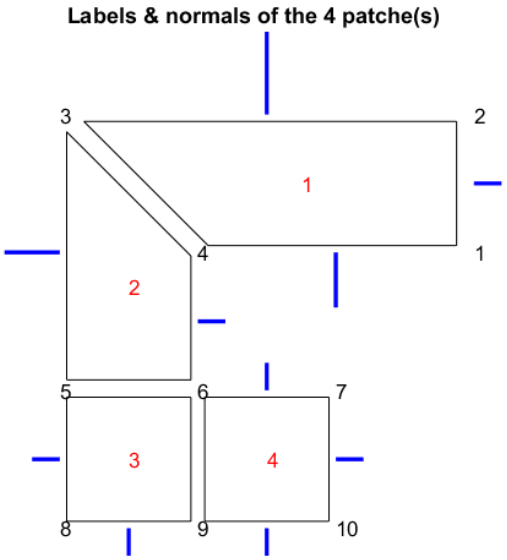


Figure 33: CAD models with diagonal on the long horizontal side

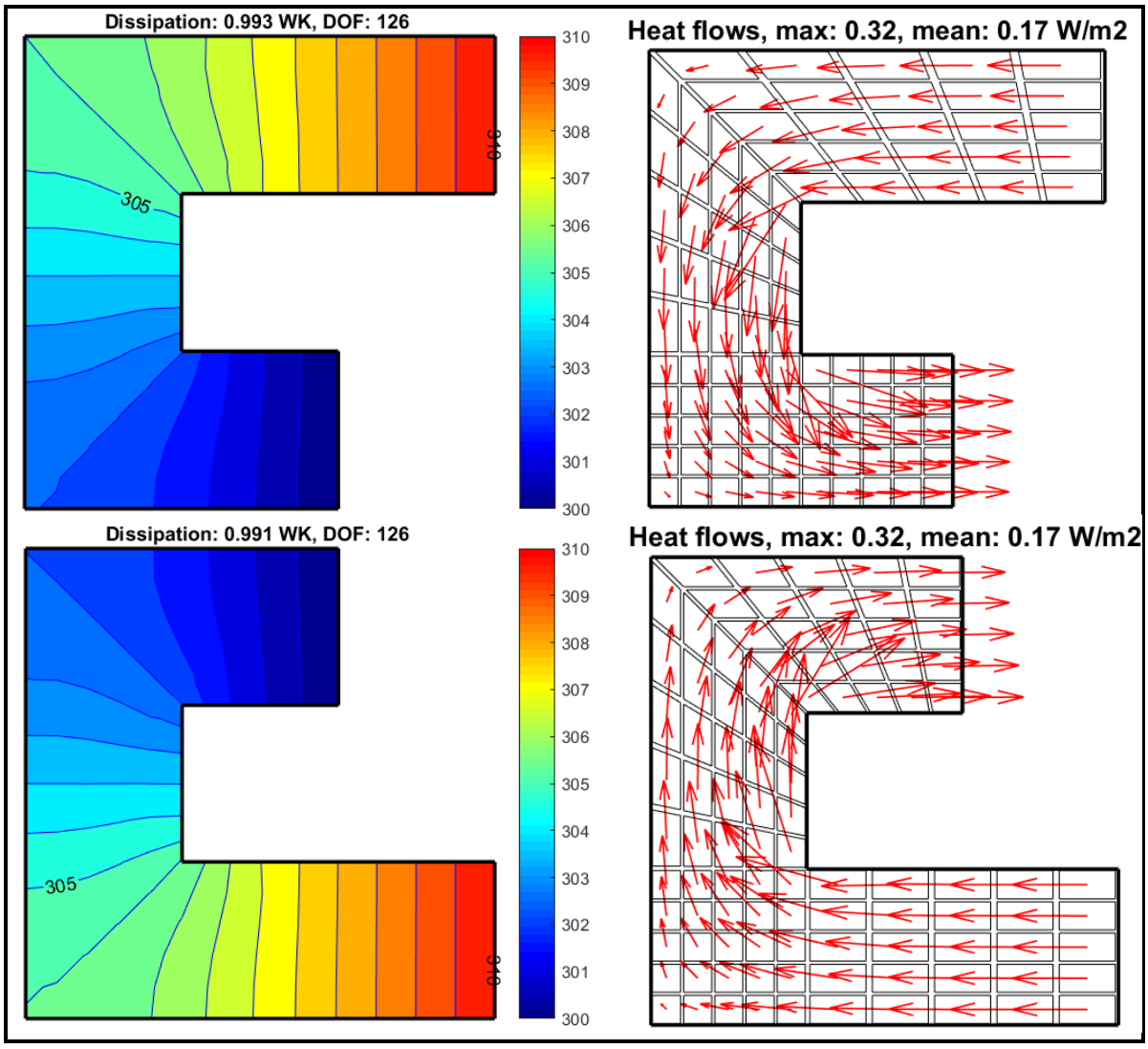


Figure 34: CAD models with diagonal on the short and the long horizontal side

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

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

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

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

69   gh   = zeros(ndK,1);      % Second member initialization always necessary

83       h = 18;hh = h*th;nec   = nni+1;

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

The result is given for two meshes:

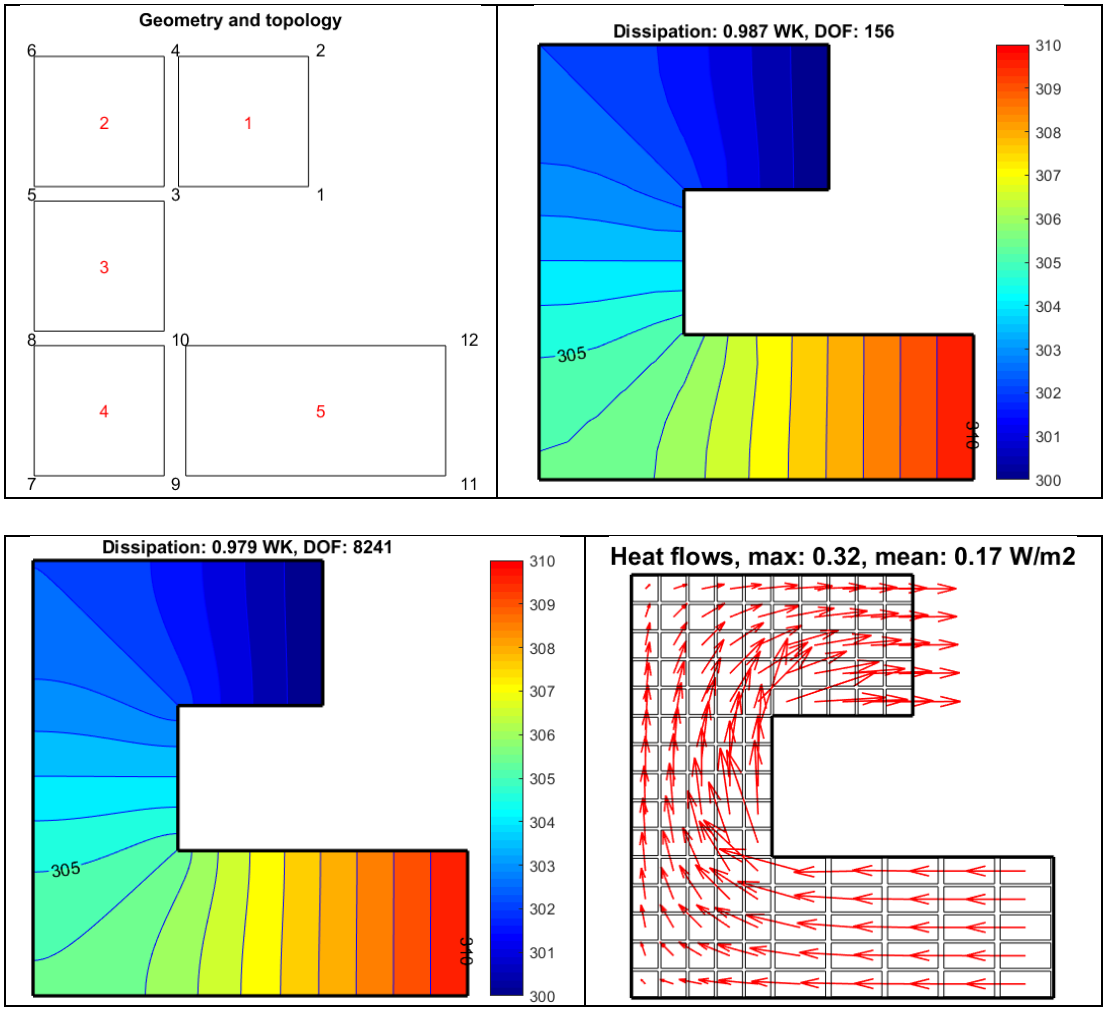


Figure 35: CAD model fully based on rectangular patches

```
Fiammetta_20211111
C shape, 3 patches, Gi: 3
L   4, Me, Di, Ne    : 1   6   0
L   5, Co nnr nvn    : 0   0   0
L   6, rc, ra, cs    : 0   0   5
L   7, CAD interf.   : 10
L   9, N. pa., vert.: 3   8
L  11, num. nod side: 49
L  13, num elem side: 50
L  23, Domain area   : 6 m2
```

```
L  29, Num. elements: 7500
L  31, Num. of DOF  : 7701
L  62, Domain perim.: 14 m
L  64, Thickness    : 0.1 m
L  65, Conduction k : 1 W/(mK)
L  66, DT isotherms : 0.5 K
L  92, Number. nodes: 102
L  93, Num. vertices: 2  4  3  1
L 188, Total dissip.: 0.979 WK
L 191, Dis. in solid: 0.979 WK
L 192, DT   in solid: 10 K
L 205, End Me=1, CPU: 6.78 sec.
```

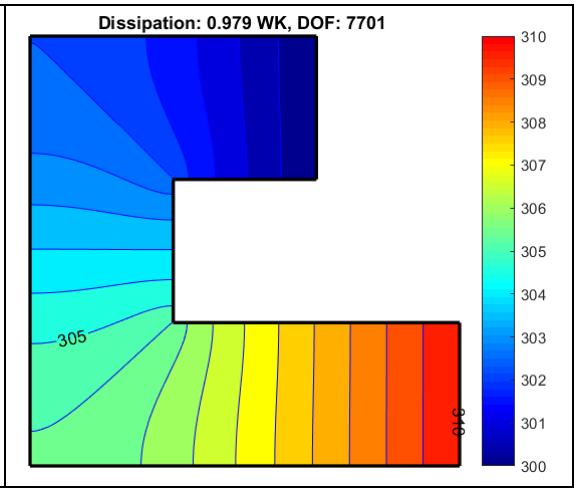


Figure 36: CAD model based on 3 patches, 7500 elements, CPU: 7 sec.

In the next test (*Figure 37*), with the linear constraints, we have 10353 unknowns.

```
Fiammetta_20211111
C shape, 4 patches, Gi: 5
L   4, Me, Di, Ne   : 1  5  0
L   5, Co nnr nvn   : 0  0  0
L   6, rc, ra, cs   : 0  0  5
L   7, CAD interf.  : 13
L   9, N. pa., vert.: 4  10
L  11, num. nod side: 49
L  13, num elem side: 50
L  23, Domain area  : 6 m2
L  29, Num. elements: 10000
L  31, Num. of DOF  : 10251
L  62, Domain perim.: 14 m
L  64, Thickness    : 0.1 m
L  65, Conduction k : 1 W/(mK)
L  66, DT isotherms : 0.5 K
L 188, Total dissip.: 0.979 WK
L 191, Dis. in solid: 0.979 WK
L 192, DT   in solid: 10 K
L 205, End Me=1, CPU: 15 sec.
```

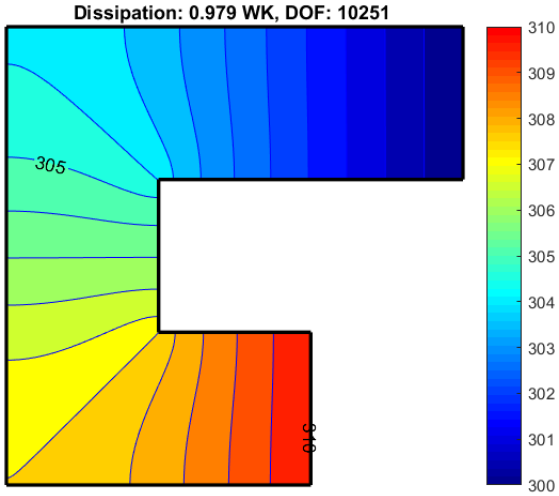


Figure 37: CAD model based on 4 patches, 10000 elements

3.6 Boundary conditions

We show here how we can introduce Dirichlet, von Neumann and convective boundary conditions. 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*).

Matlab© function *cad_Dir.m* – Dirichlet boundary conditions

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

```
13  if Di == 2
14      fT =[280 300];lfi=[no+1 no+2];nf=2;   % Fixation of two virtual nodes
15      disp(['LD 15, Fixed nodes  : ',num2str(lfi)])
16      disp(['LD 16, Fix. temper. : ',num2str(fT),' K']);
17  end
18  if Di == 3
19      nnc =  5;    % nnc = the number of fixed nodes on the horizontal sides
20      if nni < nnc;nnc = nni;end
21      a   = [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
22      b   = [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
23      nx  = min(nnc,nni+3);lfi=[b(nni+3-nx:nni+2) a(nni+3-nx:nni+2)];
24      nf  = size(lfi,2);fT = [ones(1,nf/2)*270 ones(1,nf/2)*320];
25      disp(['LD 23, N. fix. nodes: ',num2str(nf)])
26  end
27  if Di == 4
28      lfi = [[car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)]';
29             [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)]']';
30      nf = size(lfi,2);fT = [ones(1,nf/2)*270 ones(1,nf/2)*300];
31  end
32  if Di == 5
33      lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)...
34           car_cao(4,3) bor(pbo(4,3),5):bor(pbo(4,3),6) car_cao(4,4)];
35      nf = size(lfi,2);
36      if nf > 2 ;fT = [ones(1,nf/2)*300 ones(1,nf/2)*310 ];end
37  end
38  if Di == 6
39      lfi=[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)...
40           car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)];
41      nf = size(lfi,2);
42      if nf > 2 ;fT = [ones(1,nf/2)*300 ones(1,nf/2)*310 ];end
43  end
44  if Di == 7
45      fT =[300 280];lfi=[no+1 no+2];nf=2;        % Fix. of both virt. nodes
46  end
47  if Di == 8
48      lfi=[car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
49      nf = size(lfi,2);fT = ones(1,nf)*270;  % Dirichlet boundary conditions
50  end
51  if Di == 9
52      fT =[280 300];lfi=[no+1 no+2];nf=2;           % Fix. of 2 virt. nodes
53      disp(['LD 53, Fixed nodes  : ',num2str(lfi)])
54      disp(['LD 54, Fix. temper. : ',num2str(fT),' K']);
55  end
56  if Di == 10
57      fT =[270 270 300];lfi=[no+1 no+2 no+3];nf=3;   % Fix. of 3 virt. nodes
58      disp(['LD 58, Fixed nodes  : ',num2str(lfi)])
59      disp(['LD 59, Fix. temper. : ',num2str(fT),' K']);
60  end
61  if Di == 11
62      fT =[300 280];lfi=[no+1 no+2];nf=2;    % Fixation of two virtual nodes
63      disp(['LD 63, Fixed nodes  : ',num2str(lfi)])
64      disp(['LD 64, Fix. temper. : ',num2str(fT),' K']);
65  end
66  if Di == 12
67      fT =300;lfi=no+1 ;nf=1;                % Fixation of one virtual nodes
68      disp(['LD 68, Fixed nodes  : ',num2str(lfi)])
69      disp(['LD 69, Fix. temper. : ',num2str(fT),' K']);
70  end
71  if Di == 13
72      fT =[300 270];lfi=[no+1 no+2];nf=2;   % Fixation of two virtual nodes
73      disp(['LD 73, Fixed nodes  : ',num2str(lfi)])
74      disp(['LD 74, Fix. temper. : ',num2str(fT),' K']);
75  end
76  % if nfi < 7;disp(['L  73, fix. top side: ',num2str(lfi)]);end
77  % bc = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
78  %        [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
79  %        [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
80  %        [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)]];
81  % disp(['L  78, n.fix. cavity: ',num2str(size(bc))])
82  % fT      = zeros(1,no); lfi = zeros(1,no);
83  if Di > 12
84  if nvn == 0                  % lfi = uni-line matrix, fT = uni-line matrix
85      lfi = [[car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)]';
86             [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)]']';
87      nf = size(lfi,2)/2;fT=[ones(nf,1)*270 ;ones(nf,1)*300];
88  %     mfi = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
89  end
```

```
90  if nvn == 1
91  %     lfi = [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)]';
92  %     nfi=size(lfi,2);fT=ones(1,nfi)*280;
93  % % DOF of the 1. Standard cavity
94  % bc = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
95  %        [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
96  %        [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
97  %        [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)]];
98
99  end
100 % if nvn == 2                  % Dirichlet boundary conditions for convection
101 %     fT =[270 300];lfi=[no+1 no+2];           % Fixation of 2 virtual nodes
102 % end
103 if nvn == 3
104     fT=[270 285 300];lfi=[no+1 no+2 no+3];   % Fixation of 3 virtual nodes
105 end
106 nfi = size(lfi,2);disp(['LD106, N. fix. nodes: ',num2str(nfi)])
107 if nfi > 0
108     if nfi < 15
109         disp(['LD109, Numb. fix. N.: ',num2str(nf)])
110         disp(['LD110, Fixed nodes  : ',num2str(lfi)])
111         disp(['LD111, Fix. temper. : ',num2str(fT),' K']);
112     end
113 end
114 end
115 end
```

Table 17: Matlab© function cad_Dir.m - Dirichlet boundary conditions

Matlab© function *cad_Neu.m* – von Neumann boundary conditions

```
1  function [gh,bos] = cad_Neu(Ne,dK,car_cao,pbo,bor,th,xyz_cao)
2  %........................................... Neumann boundary conditions
3  if Ne == 1
4      gh = zeros(dK,1);                          % second member initialization
5      lg = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
6      nh = size(lg,2);np=nh-1;g(1)=1/(2*np);g(nh)=g(1);g(2:nh-1)=1/np;
7      gh(lg(1:nh)) = g(1:nh)/sum(g);
8      bos = (xyz_cao(2,1)-xyz_cao(1,1))*th;
9      disp(['N   9, Loaded  area : ',num2str(bos),' m2'])
10 end
11 if Ne == 11
12     gh = zeros(dK,1);
13     lg  = [car_cao(3, 3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)...
14           car_cao(1, 1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
15     nh  = size(lg,2)/2;                  % Loaded nodes lg for CAD model 11
16 %     lg = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
17 %     nh = size(lg,2);
18     for i            = 1:nh;   gh(lg(1,i),1) = 2; end % Weights right side
19     gh(lg(1,1),1)    = 1   ; gh(lg(1,nh),1)   = 1;
20     gh(lg(1:nh))     = gh(lg(1:nh))/sum(gh(lg(1:nh)));
21     for i            = nh+1:2*nh;gh(lg(1,i),1) = 2; end% Weights left side
22     gh(lg(1,nh+1),1) = 1   ; gh(lg(1,2*nh),1) = 1;
23     gh(lg(nh+1:2*nh))= gh(lg(nh+1:2*nh))/sum(gh(lg(nh+1:2*nh)));nh = 2*nh;
24     gh=gh/2;
25     bos = (xyz_cao(2,1)-xyz_cao(1,1)+xyz_cao(7,1)-xyz_cao(8,1))*th;
26     disp(['N  26, N. load. nod.: ',num2str(size(lg,2))])
27 end
28 if Ne == 0
29     gh  = zeros(dK,1);bos=0;   % 2d member initialization always necessary
30 else
31     if nh < 10
32         disp(['N  32, Loaded nodes : ',num2str(lg)])
33         disp(['N  33, Loads weights: ',num2str(gh(lg)')])
34     end
35 end
36 end
```

Table 18: Matlab© function cad_Neu.m - von Neumann boundary conditions

Matlab© function *cad_con.m* – convection boundary conditions

```
1  function [lc,hv]=cad_con(car_cao,bor,pbo,h,dK,nes,deb,nvn,cs,Co)% 20210929
2  npa = size(car_cao,1);% npa = number patches ; nes number of elem per side
3  disp(['c. 03, Numb. var. dK: ',num2str(dK)]) % nes = n. elem/side
4  disp(['c. 04, N.virt c.nod.: ',num2str(nvn)]);
```

```
if Co == 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)]];
    hv = ones(1,nvn*npa*nes)*h;
end
if Co == 2
    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)]];
    a  = ones(1,nes);
    hv = [a*h a*h a*h a*h];
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)]];
    hv = ones(1,nvn*npa*nes)*h;
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)];];
    hv = ones(1,nvn*npa*nes)*h;
end
if Co > 4
    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)]];
        hv = ones(1,nvn*npa*nes)*h;
        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)]];
        hv = ones(1,nvn*npa*nes)*h;
        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)];];
       hv = ones(1,nvn*npa+2*npa)*h;
         end
    end
    if cs~=5
    if npa > 2                 % The domain contains more than two patches
        if nvn == 1                                     % 1 convective side
     bt = [[car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];
           [car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)];
           [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
           [car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)];
           [car_cao(5,2) bor(pbo(5,2),5):bor(pbo(5,2),6) car_cao(5,3)]];
            a  = (1:nes);
            hv = [a*0 a*0 a*0 a*h a*0];
            if deb == 1
                a=1;hw = [a*0 a*0 a*0 a*h a*0];
                disp(['c. 64, 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)]];
            a  = (1:nes);
            hv = [a*0 a*0 a*h/2 a*h a*0 a*0 a*h a*h/2 a*0 a*0];
```

```
            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. 82, 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)];
            a  = (1:nes);
            hv = [a*h a*h a*h a*h a*h a*h a*h a*h a*h];
            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. 94, Conv. coeff. : ',num2str(hw),'  W/(m2K)'])
            end
        end
    end
    end
end
nec = (nes)*size(bt,1);                % There are nec convective elements
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 deb==1;disp(['c.112, N. conv. ele.: ',num2str(nco)]);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
        k1 = [1 npa*nes+1 ];
        k2 = [npa*nes (2*npa)*nes];
    end
    if nvn == 3                                         % 3 convective sides
        k1 = [1 nco/3+1 2*nco/3+1];
        k2 = [nco/3 2*nco/3 nco];
    end
    for i           = 1:nvn   %if nvn > 1, generation of the conv V. numb.
        for j       = k1(i):k2(i)
            lc(j,3) = dK+i-nvn;
        end
    end
end
if deb==1;disp(['c.132, conv v.N numb: ',num2str(dK-nvn+1:dK)]);end
end
```

Table 19: Matlab© function cad_con.m - localization of convection elements

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 69*). Another method for drawing the levels of a scalar function consists in working independently in each element with the function *gra_lin.m* (*Table 70*).
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 67*). 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 15*).

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.
The dissipation D in the solid is computed from the temperature gradient average, the conductivity coefficient and the volume V of the mesh:

$$D = \frac{1}{2} k \left(\nabla \tau\right)^2_{average} V \tag{58}$$

D = 1/2 * 1 $Wm^{-1}K^{-1}$ * $(23.3)^2$ K^2m^{-2} * V m^3
= 0.5 $Wm^{-1}K^{-1}$ * 576 K^2m^{-2} 0.075 m^3
= 20.3584 WK.

```
Fiammetta_20211111
Standard trapezoidal shape, Gi:  6
L   4, Me, Di, Ne    : 1   7   0
L   5, Co nnr nvn    : 1   0   2
L   6, rc, ra, cs    : 0   0   5
L   7, CAD interf.   : 4
L   9, N. pa., vert.: 1   4
L  11, num. nod side: 7
L  13, num elem side: 8
L  23, Domain area   : 0.75 m2
L  29, Num. elements: 64
L  31, Num. of DOF   : 83
L  62, Domain perim.: 3.5616 m
L  64, Thickness     : 0.1 m
L  65, Conduction k : 1 W/(mK)
L  66, DT isotherms : 1 K
L  72, Fixed DOF     : 82  83
L  79, Convection h : 18 W/(m2K)
c. 03, dim. K Co cs : 83   1   5
c. 04, N.virt c.nod.: 2
L  92, Number. nodes: 18
L  93, Num. vertices: 2  3  4  1
L 188, Total dissip.: 24.3 WK
L 191, Dis. in solid: 20.9 WK
L 192, DT    in solid: 18.5 K
L 201, Fixed DOF     : 82  83
L 202, Imposed temp.: 300        280 K
L 203, React. flows : 2.43      -2.43 W
L 205, End Me=1, CPU: 0.492 sec.
```

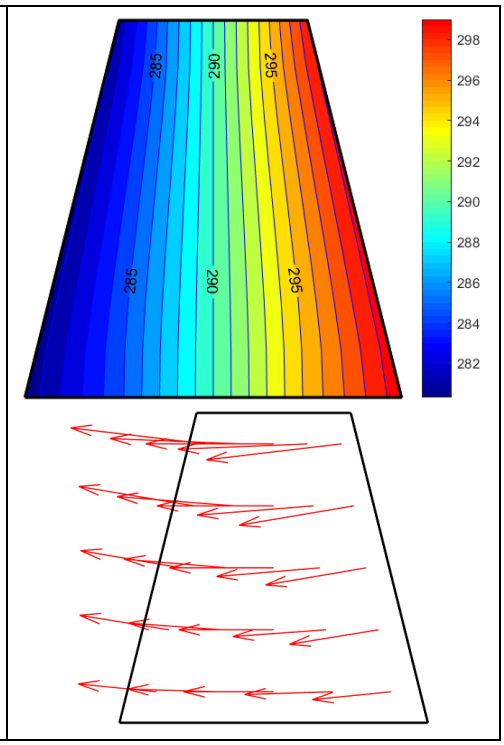
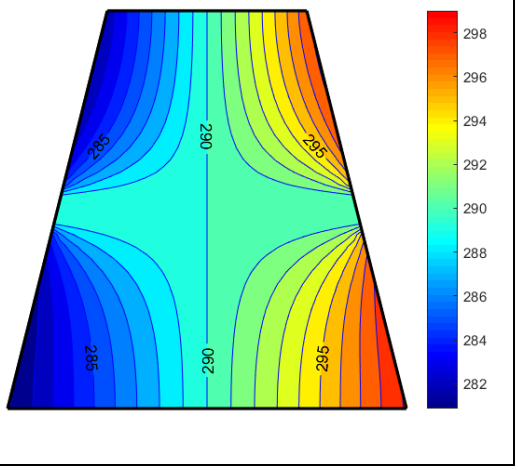


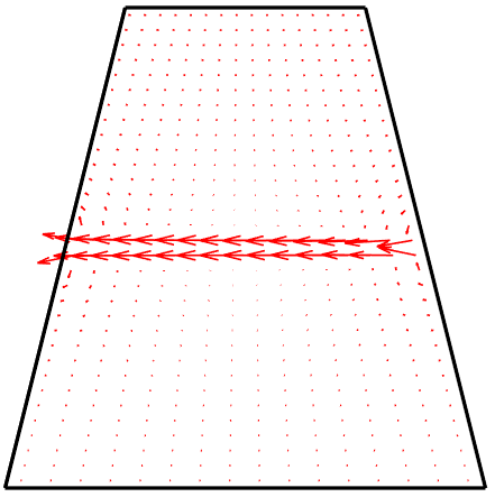
Figure 38: 2 imposed temperatures, 2 adiabatic faces in a trapezoidal domain

```
Fiammetta_20211111
Standard trapezoidal shape, Gi:  6
L   4, Me, Di, Ne    : 1   7   0
L   5, Co nnr nvn    : 1   0   2
L   6, rc, ra, cs    : 0   0   5
L   7, CAD interf.   : 4
L   9, N. pa., vert.: 1   4
L  11, num. nod side: 31
L  13, num elem side: 32
L  23, Domain area   : 0.75 m2
L  26, Num. elements: 1024
L  28, Num. of DOF   : 1091
L  59, Domain perim.: 3.5616 m
L  61, Thickness     : 0.1 m
L  65, Conduction k : 1 W/(mK)
L  66, DT isotherms : 1 K
L  72, Fixed DOF     : 1090  1091
L  79, Convection h : 18 W/(m2K)
c. 03, Numb. var. dK: 1091
c. 04, N.virt c.nod.: 2
L  96, Anis. index   : 1
```

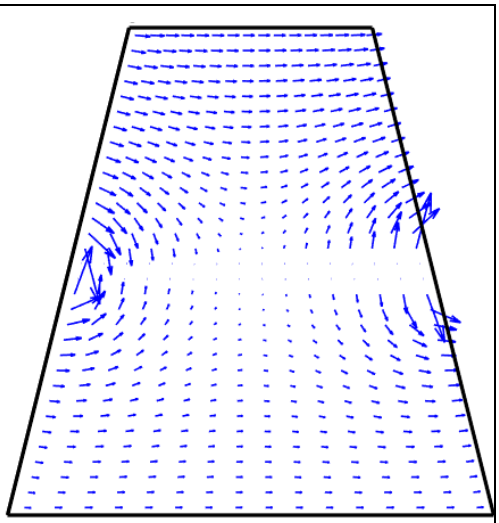


```
Lm 48, k & coef*k    : 1  1000 W/(m K)
Lm 49, Thickn. ratio: 0.0625
L 188, Total dissip.: 53.6 WK
L 191, Dis. in solid: 27.7 WK
L 192, DT   in solid: 18.3 K
L 201, Fixed DOF    : 1090  1091
L 202, Imposed temp.: 300       280 K
L 203, React. flows : 5.36     -5.36 W
L 205, End Me=1, CPU: 0.526 sec.
```

Figure 39: Non homogeneous trapezoidal domain



Heat flow, max: 476 Wm^{-2}, mean: 39 Wm^{-2}



-grad τ: 76 Km^{-1}, mean: 13 Km^{-1}

Figure 40: Horizontal strip with high conductivity 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 (*Table 8*). The heat flow picture is dominated by the arrows in the central zone, while in the gradient one the same zone of high flows is disappearing due to the small value of the gradients.

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.

$$\alpha = \frac{k}{\rho\, c_p} \tag{59}$$

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.

$$\left([C]+\theta\,\Delta t\ [K]\right)\left[T^{n+1}\right]=\left([C]-(1-\theta)\,\Delta t\ [K]\right)\left[T^{n}\right]+\Delta t\ \left(\theta\left[f^{n+1}\right]+(1-\theta)\left[f^{n}\right]\right) \tag{60}$$

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

$$\left([C]+\Delta t\ [K]\right)\left[T^{n+1}\right]=[C]\left[T^{n}\right]+\Delta t\ \left[f^{n+1}\right] \tag{61}$$

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

$$[T]=\begin{bmatrix} T_1 \\ T_f \end{bmatrix} \tag{62}$$

Equation (62) becomes:

$$\left(\begin{bmatrix} C_{11} & C_{1f} \\ C_{f1} & C_{ff} \end{bmatrix}+\Delta\, t \begin{bmatrix} K_{11} & K_{1f} \\ K_{f1} & K_{ff} \end{bmatrix}\right)\begin{bmatrix} T_1^{n+1} \\ T_f^{n+1} \end{bmatrix}=\begin{bmatrix} C_{11} & C_{1f} \\ C_{f1} & C_{ff} \end{bmatrix}\begin{bmatrix} T_1^{n} \\ T_f^{n} \end{bmatrix}+\Delta\, t \begin{bmatrix} f^{n+1} \\ r^{n+1} \end{bmatrix} \tag{63}$$

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 (63) is:

$$\left(\begin{bmatrix} C_{11} & C_{1f} \end{bmatrix}+\Delta\, t \begin{bmatrix} K_{11} & K_{1f} \end{bmatrix}\right)\begin{bmatrix} T_1^{n+1} \\ T_f^{n+1} \end{bmatrix}=\begin{bmatrix} C_{11} & C_{1f} \end{bmatrix}\begin{bmatrix} T_1^{n} \\ T_f^{n} \end{bmatrix}+\Delta t \begin{bmatrix} f^{n+1} \end{bmatrix} \tag{64}$$

Developing this relation gives:

$$\left([C_{11}]+\Delta t\ [K_{11}]\right)\left[T_1^{n+1}\right]=[C_{11}]\left[T_1^{n}\right]+\Delta t\left(\left[f^{n+1}\right]-\left[K_{1f}\right]\left[T_f\right]\right) \tag{65}$$

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

$$\left[T_1^{n+1}\right]=\left([C_{11}]+\Delta t\ [K_{11}]\right)^{-1}\ \left\{[C_{11}]\left[T_1^{n}\right]+\Delta t\left(\ \left[f^{n+1}\right]-\left[K_{1f}\right]\left[T_f\right]\ \right)\right\} \tag{66}$$

Without fixation:

$$\left[T^{n+1}\right]=\left(\left[C\right]+\Delta t\ \left[K\right]\right)^{-1}\left\{\left[C\right]\left[T^{n}\right]+\Delta t\ \left[f^{n+1}\right]\right\} \tag{67}$$

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

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

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

$$\begin{aligned}&\left(\left[C_{f1}\quad C_{ff}\right]+\Delta t\ \left[K_{f1}\quad K_{ff}\right]\right)\begin{bmatrix}T_1^{n+1}\\T_f\end{bmatrix}=\left[C_{f1}\quad C_{ff}\right]\begin{bmatrix}T_1^{n}\\T_f\end{bmatrix}+\Delta t\ \left[r^{n+1}\right]\\&\left[r^{n+1}\right]=\frac{1}{\Delta t}\left[C_{f1}\right]\left(\left[T_1^{n+1}\right]-\left[T_1^{n}\right]\right)+\left[K_{f1}\right]\left[T_1^{n+1}\right]+\left[K_{ff}\right]\left[T_f\right]\end{aligned} \tag{69}$$

4.2 Element capacity matrix

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

$$\tau=T_1\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right)+T_2\frac{x}{a}\left(1-\frac{y}{b}\right)+T_3\frac{x}{a}\frac{y}{b}+T_4\left(1-\frac{x}{a}\right)\frac{y}{b} \tag{70}$$

$$\begin{aligned}\tau &= [F][T]\\ [F] &= \left[\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right)\quad\frac{x}{a}\left(1-\frac{y}{b}\right)\quad\frac{x}{a}\frac{y}{b}\quad\left(1-\frac{x}{a}\right)\frac{y}{b}\right]\\ [T]^T &= [T_1\quad T_2\quad T_3\quad T_4]\end{aligned} \tag{71}$$

$$[C]=\int_V \rho c_p [F]^T [F]\,dV \tag{72}$$

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

$$\begin{aligned}C_{33}&=e\int_0^a\left(\int_0^b \rho c_p\ \frac{x^2y^2}{a^2b^2}\ dy\right)dx\\&=\rho e c_p\int_0^a\frac{b^3x^2}{3a^2b^2}\ dx=\rho e c_p\ \frac{b}{3}\int_0^a\frac{x^2}{a^2}dx=\rho e c_p\ \frac{ab}{9}=\frac{\rho V c_p}{9}\end{aligned} \tag{73}$$

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}.

$$[C] = \frac{\rho V c_p}{36} \begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix} \tag{74}$$

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 20*).

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

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

Table 20: Matlab© function fem_Cae.m – element capacity matrix

*Table **21*** shows Matlab© instructions allowing to compute element capacity matrices in various geometrical situations. To obtain the effective capacity matrix, the output of fem_Cae.m has to be multiplied by the thickness th (m), the capacity cp (Jkg-1K-1) and the specific mass ro (kgm-3). The total capacity for the cases presented in Table 21 is equal to th (0.1) x cp (1000) x ro (2500) x sum (sum (C)) / 36 = 250000 JK-1 for the first case and 106 JK-1 for the second one (*Table 21*).

Matlab 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` `sum(sum(C))`

Matlab Output	`C =  4  2  1  2  36.`
	`2  4  2  1`
	`1  2  4  2`
	`2  1  2  4`

Matlab 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` `sum(sum(C))`

Matlab Output	`C = 16  8  4  8  144`
	`8  16  8  4`
	`4  8  16  8`
	`8  4  8  16`

Matlab 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` `sum(sum(C))`

Matlab Output	`C =  4  2  1  2  36.`
	`2  4  2  1`
	`1  2  4  2`
	`2  1  2  4`

Matlab 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` `sum(sum(C))`

Matlab Output	`C =  4  2  1  2  36.`
	`2  4  2  1`
	`1  2  4  2`

	2 1 2 4
Matlab 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 sum(sum(C))
Matlab Output	C = 5 2.5 1 2 36. 2.5 5 2 1 1 2 3 1.5 2 1 1.5 3

Table 21: Numerically integrated capacity matrix

4.3 Temperature evolution

The procedure *gra_tev.m* (*Table 22*) 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 41*.

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

```
 1  function [] = gra_tev(ni,dt,re,tsmax,tsmin,tmoy)    % Temperature evolution
 2  tem          = (0:ni)*dt/3600/ni;                  % Time steps for the graphics
 3  st           = size(tsmin,2);%nj = ni+1;
 4  figure('Position',[100 100 600 300])
 5  plot  (tem,tsmax','r');hold on;       % Plotting the 3 evolutive functions
 6  plot  (tem,tsmin','b');hold on;
 7  plot  (tem,tmoy  ,'k');hold on;
 8  ylabel( 'gra-tev: temp. evolution','fontsize',15)
 9  xlabel(['Number of iterations : ',num2str(ni),', re : ',num2str(re)],...
10      'fontsize',15);grid on
11  legend('T maximum ','T minimum ','T average','Location','northwest')
12  title (['Final: Tmin: ',num2str(tsmin(st)*10/10,3),' K, Tmean: ',...
13      num2str(tmoy(st)*10/10,3),' K, Tmax: ',num2str(tsmax(st)*10/10,3),...
14      ' K'],'fontsize',15);hold on;grid on
15  end
```

Table 22: Matlab© function gra_tev.m – temperature evolution

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 58*, *Table 62*). 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 41*). 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*.

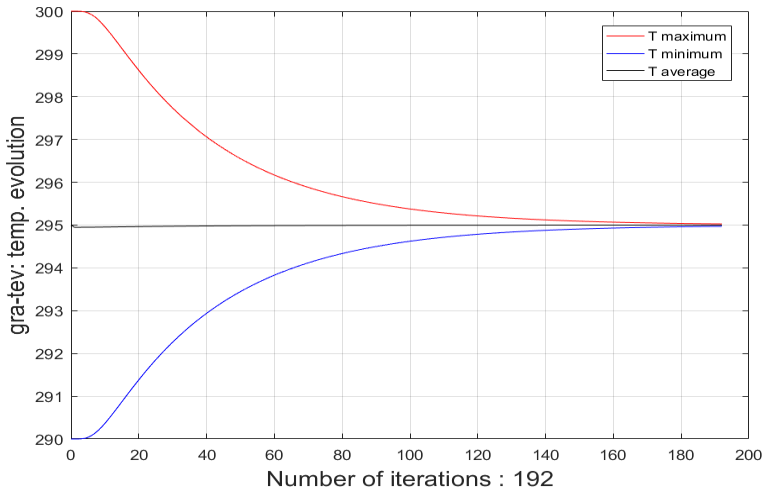


Figure 41: Smoothing process convergence in temperature homogenization

```
Fiammetta_20211111
Horizon. 1 rect., Gi: 17
L   4, Me, Di, Ne    : 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   9, N. pa., vert.: 1  4
L  11, num. nod side: 8
L  13, num elem side: 9
L  23, Domain area   : 2 m2
L  29, Num. elements: 81
L  31, Num. of DOF   : 100
L  62, Domain perim.: 6 m
L  64, Thickness     : 0.5 m
L  65, Conduction k : 2 W/(mK)
L  66, DT isotherms : 0.1 K
L  72, Fixed DOF     : 0
L  96, Anis. index   : 0
t. 04, Initial temp.: 290 K
t. 20, nit  N. iter.: 96
t. 17, Time step     : 3600 s
t. 15, Analyzed per.: 96 h, 4 days
t. 21, Spec. capac. : 1000 J/(kg.K)
t. 22, Spec. mass    : 2500 kg.m-3
t. 40, sum(sum(C))   : 2.5 MJ/K
t. 41, area*th*ro*Cp: 2.5 MJ/K
t. 45, Imposed Heat : 0 W/m-2
t  67, iteration     : 24
t  67, iteration     : 48
t  67, iteration     : 72
t  67, iteration     : 96
t. 80, Tmean - Tini : 5 K
t. 81, DT* sumsum(C): 12.5 MJ
t. 83, Min obs. temp: 290 K
t. 84, Max obs. temp: 300 K
L 231, End Me=3. CPU: 1.18 sec.
```

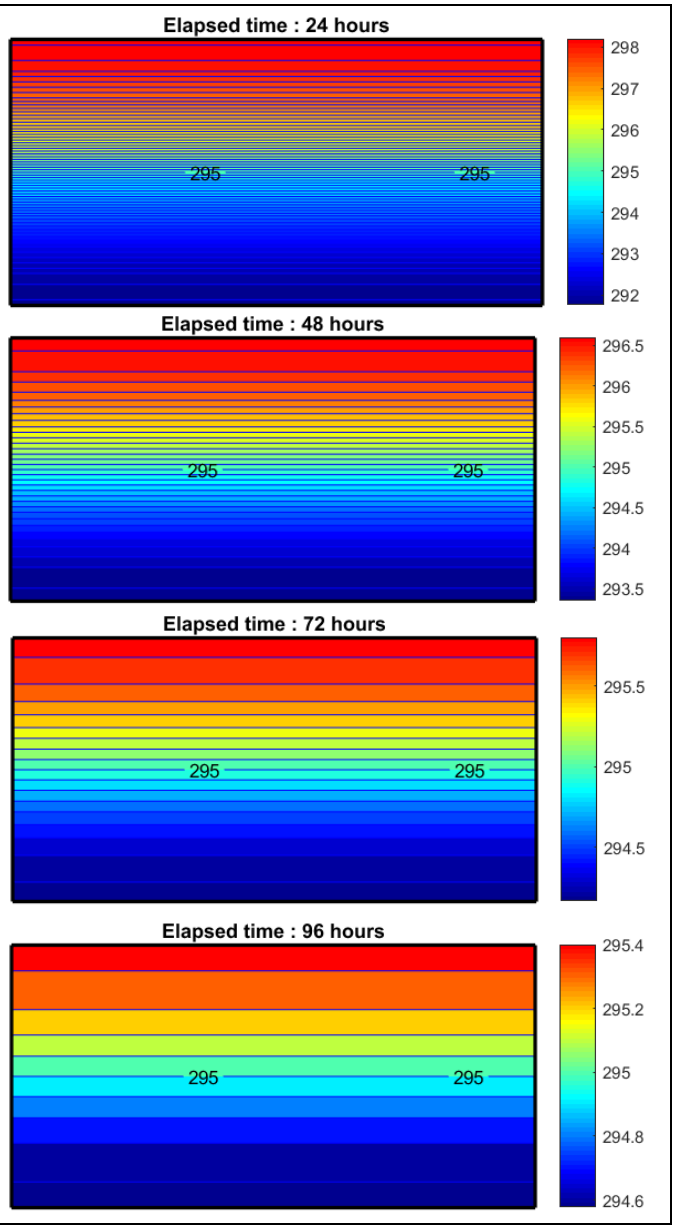


Figure 42: Evolution of the temperature field in a smoothing process

4.5 Heating of a solid at initial uniform temperature

The next example (*Figure 43*) 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, the temperature of the solid is 280 *K*. After 100 *h*, the mean temperature is 296 *K*.

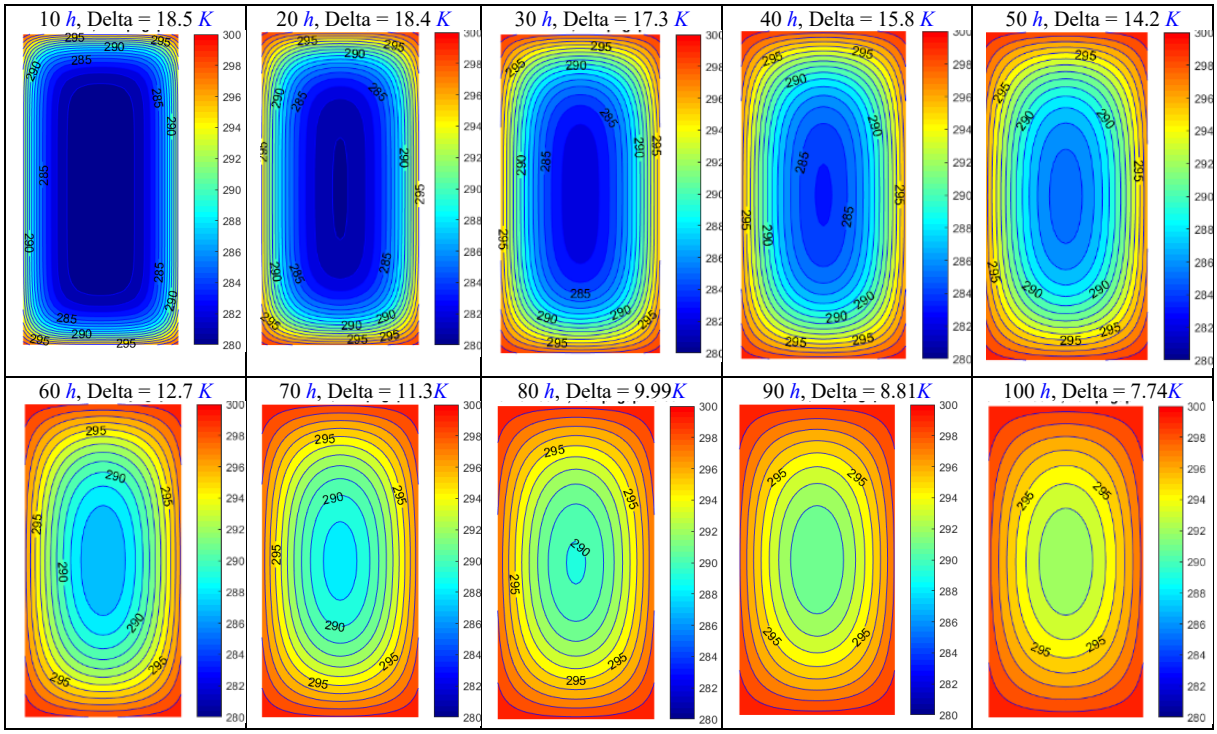


Figure 43: Evolution of isotherms in a heating operation: 100h

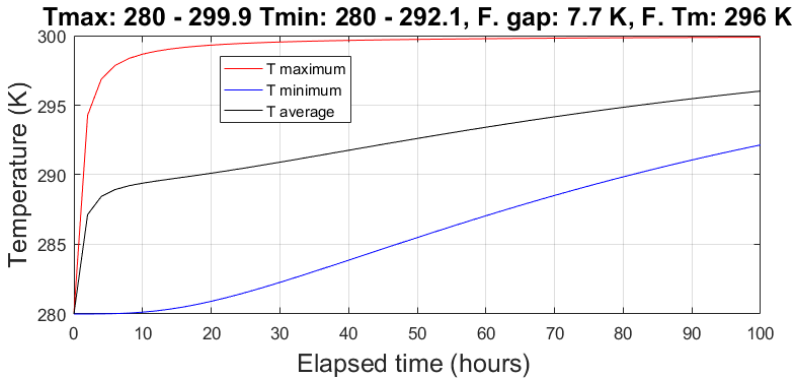


Figure 44: Evolution of specific temperatures in a heating operation

```
Fiammetta
Elem. per patch side: 40
Thickness           : 0.1 m
Conductivity coeff. : 1 W/(mK)
Stefan Boltzmann cst: 5.6704e-08 W/(m2K4)
N. elements         : 1600
N. DDL              : 1685
Convection coeff.   : 25 W/(m2K)
Specific capacity   : 1000 J/(kg.K)
Specific mass       : 2500 kg.m-3
Tot. cap.sum(sumI)  : 500000 J/K
Interv. Drawing ist : 100 h
Time step           : 2 h
Analyzed period     : 200 h
CPU ............... : 14.8 sec.
```

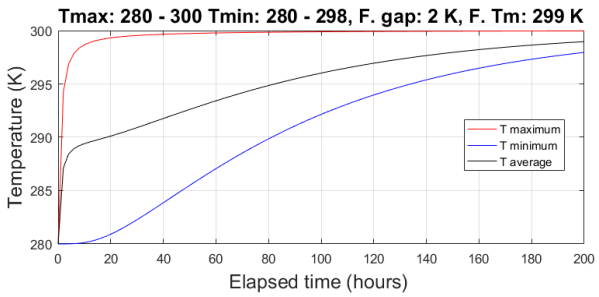


Figure 45: Evolution of max, min and mean temperatures in a heating operation: 200h

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.

4.6 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 14*).

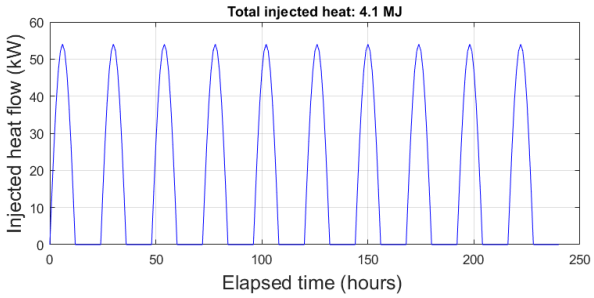


Figure 46 : Thermal loading over a period of 10 days (240 hours)

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

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

Table 23: Matlab© function gra_hie.m – visualization of a periodic scalar field

The drawing of *Figure 46* is created only if the time step (variable *dth*) is equal to 1 hour (*line 233* in *Fiammetta.m*, *Table 58*). The function *gra_hie.m* (*Table 23*) is called at *line 334* of *Fiammetta.m*. The time function *f(t)* used to weight the heat flow input is given by:

$$f(t) = \begin{cases} \sin\left(\frac{2t}{p}\pi\right) & \text{if } 0 \le t \le \frac{p}{2} \\ 0 & \text{if } \frac{p}{2} \le t \le p \end{cases} \quad (75)$$

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:

$$\frac{2}{p}\int_0^{p/2} \sin\left(\frac{2t}{p}\pi\right) dx = -\frac{2}{p}\left[\cos\left(\frac{2t}{p}\pi\right)\frac{p}{2\pi}\right]_0^{p/2} = \frac{-1}{2\pi}(-1-1) = \frac{2}{\pi} \simeq 0.6366 \quad (76)$$

Then, over the period *p*, the average of the time weighting function *f(t)* is 1/π 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:

$$\overline{q}_n = i_h f(t)\ Wm^{-2} = i_h \sin\frac{t\ \pi}{12*3600}\ Wm^{-2} = i_h \sin\frac{t\ \pi}{12*3600}\ Wm^{-2} \quad (77)$$

The area of the boundary where heat is injected is:

$$s_b = e \text{ x } L = 0.1*3 = 0.3\ m^2 \quad (78)$$

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

$$\bar{q}_n b_s = i_h\ e\ L\ f(t)\ W = \ i_h\ e\ L\ f(t)\ W = i_h\ e\ L \sin\frac{t\pi}{12*3600}\ W = i_h\ e\ L\ \sin\frac{t\pi}{43200}\ W \qquad (79)$$

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

$$h_d = \int_0^{43200} \bar{q}_n b_s\ dt\ J = i_h\ e\ L \int_0^{43200} \sin\frac{t\pi}{43200} dt\ J = i_h\ e\ L \left[-\frac{43200}{\pi}\cos\frac{t\pi}{43200}\right]_0^{43200}\ J$$
$$h_d = \frac{i_h\ e\ L\ 43200 * 2}{\pi}\ J = 0.2475\ i_h\ MJ \qquad (80)$$

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: $h_d * 30 = 12.4\ MJ$. The sequences of specific lines of the example shown in *Figure 53* are given in the following table.

```
1    % ....... 1. Standard cavity ..........................................
2    xyz_cao  = [0 0;3 0;3 3;0 3;1 1;2 1;2 2;1 2];
3    car_cao  = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12; nvn=1;vc=[1.5 1.5];
4    disp( 'L   5, 1. Standard cavity  : ')

49   % ........................................... Neumann boundary conditions
50   lg  = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
51   nh  = size(lg,2);
52   for I          = 1:nh;gh(lg(1,i),1) = 2;end% Weight functions on a CAD side
53   gh(lg(1,1),1) = 1;gh(lg(1,nh),1) = 1;gh = gh/(nni+1)/2;
54   if size(lg,2) < 12;disp(['List of loaded nodes: ',num2str(lg)]);end
55   if size(lg,2) < 12;disp(['Loads nodal weights : ',num2str(gh(lg(1:nh))'...
56           )]);end

117      tcan= ones(1,no)*280;tmi=280;  % Initial conditions transient analysis
119      nit =720;dth=1;dtd=180;  % Number iterations & delta tau per iteration

182      pai      = .5;            %pai      = (max(tcan)-min(tcan))/20;  %
184      gt       = [tmi pai 291]; %gt       = [min(tcan) pai max(tcan)]; %

232      tmi      = min(min(tcan),tmi);gt=[tmi  pai 291];
233      figure;ip=0;a=-.1;b=-.09;ha=max(xyz(:,2));
         fill([a b b a],[0 0 ha ha],[280 280 291 291]);hold on;
```

Table 24: Instructions for a test of time dependent heat loads

4.7 Interaction between spatial and temporal discretization

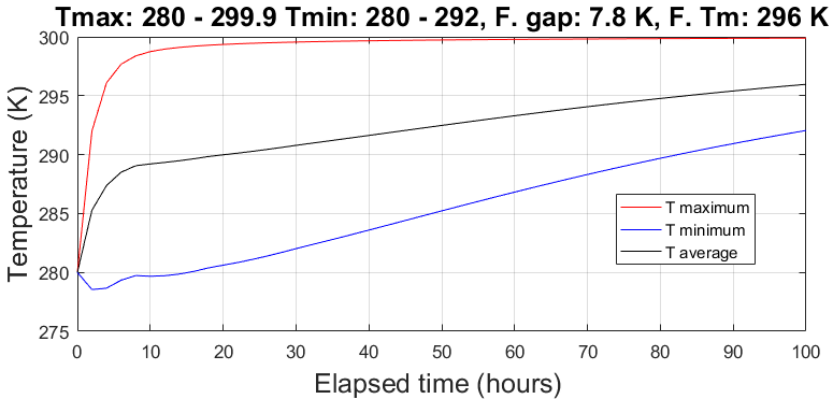


Figure 47: Temperature evolution in a coarse mesh

If we modify the mesh of the example shown in *Figure 43* to make it coarser (8 x 8 elements), we obtain some strange results. We observe in *Figure 47* that, at the beginning of the process, the minimum temperature inside the mesh is decreasing, and that 9 iterations are needed for the temperature to become again greater than the initial one. According to the second law of

thermodynamics, this behavior is not physically acceptable. However, the convergence is very similar to that of *Figure 44*; in both cases, the final temperature gap is equal to 7.8 *K*.

4.8 Adiabatic cavity & imposed temperatures on the top

```
Fiammetta_20211111
Standard cavity,  Gi: 1
L    4, Me, Di, Ne    : 3  1  0
L    5, Co nnr nvn    : 0  0  0
L    6, rc, ra, cs    : 0  0  5
L    7, CAD interf.   : 12
L    9, N. pa., vert.: 4  8
L   11, num. nod side: 7
L   13, num elem side: 8
L   23, Domain area  : 8 m2
L   29, Num. elements: 256
L   31, Num. of DOF  : 288
L   62, Domain perim.: 16 m
L   64, Thickness     : 0.1 m
L   65, Conduction k : 2 W/(mK)
L   66, DT isotherms : 1 K
L   96, Anis. index  : 0
t. 05, Initial temp.: 280 K
t. 20, nit  N. iter.: 720
t. 21, Time step     : 3600 s
t. 23, Analyzed per.: 720 h, 30 days
t. 21, Spec. capac. : 1000 J/(kg.K)
t. 22, Spec. mass    : 2500 kg.m-3
t. 40, sum(sum(C))   : 2 MJ/K
t. 41, area*th*ro*Cp: 2 MJ/K
t. 45, Imposed Heat : 0 W/m-2
t  67, iteration     : 180
t  67, iteration     : 360
t  67, iteration     : 540
t  67, iteration     : 720
t. 80, Tmean - Tini : 10.9 K
t. 81, DT* sumsum(C): 21.8 MJ
t. 83, Min obs. temp: 280 K
t. 84, Max obs. temp: 300 K
t. 93, Tot. reaction: 20.6 MJ
L 231, End Me=3. CPU: 4.44 sec.
```

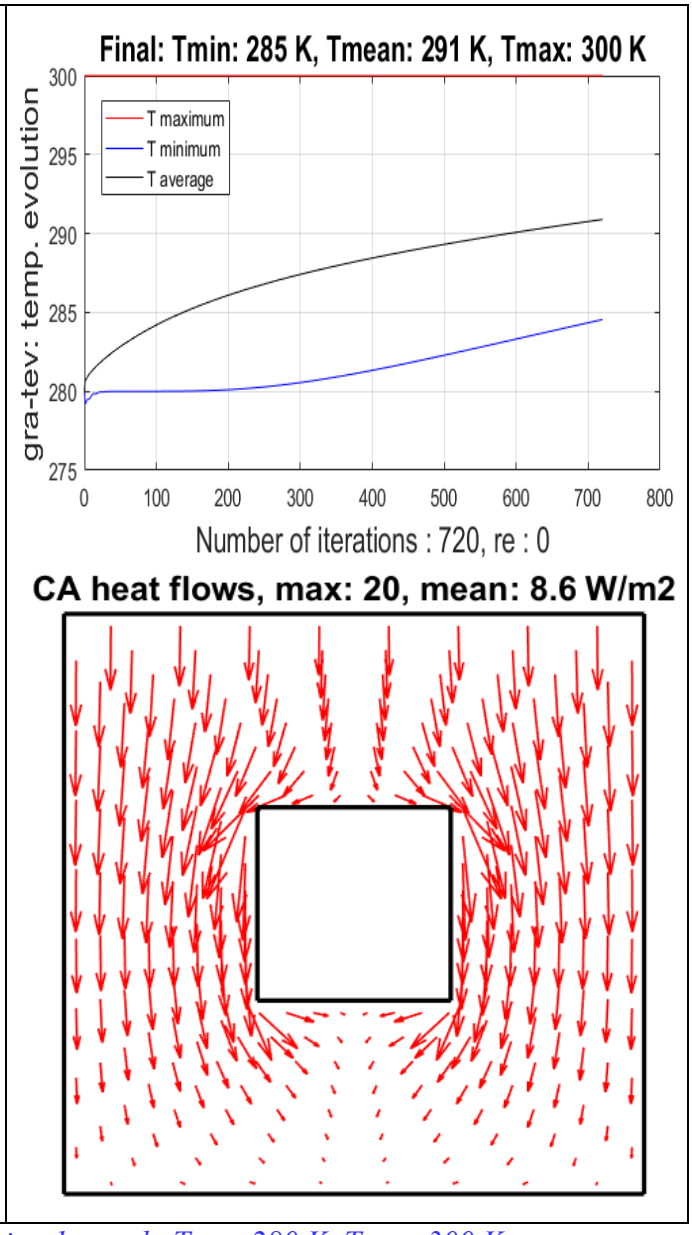


Figure 48: Adiabatic cavity, 1 month, T_{ini} = 280 K, T_{top} = 300 K

In *Figure 48*, we observe that the stored heat (`t.81, Stored heat: 21.8 MJ`), is, as expected, the product of (`t.40, sum(sum(C)):2 MJ/K`) and (`t.80, Tmean - Tini: 10.9 K`). 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 49* , 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*, we enable the Matlab© instructions *70* & *71* in *fem_til.m*.

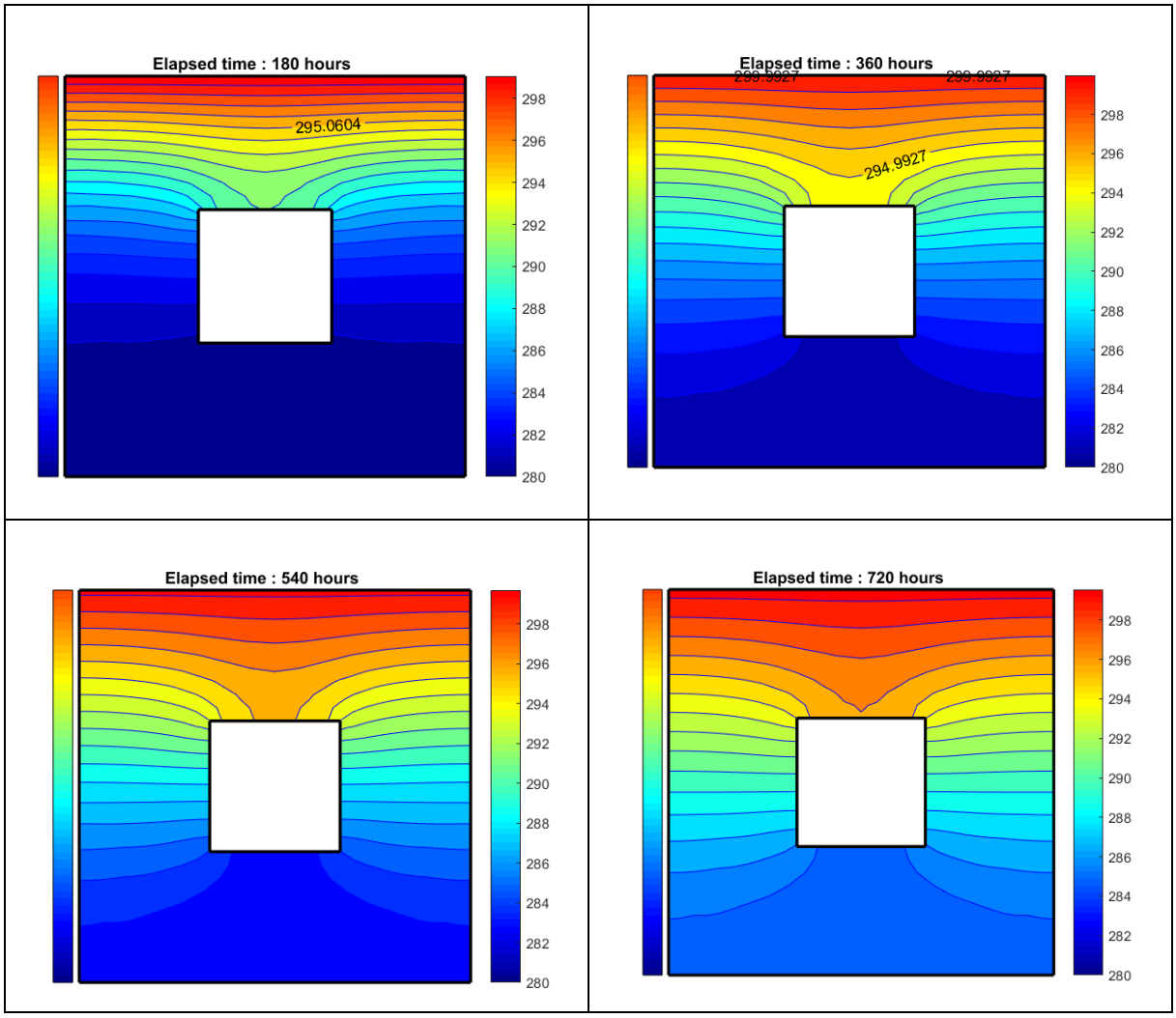
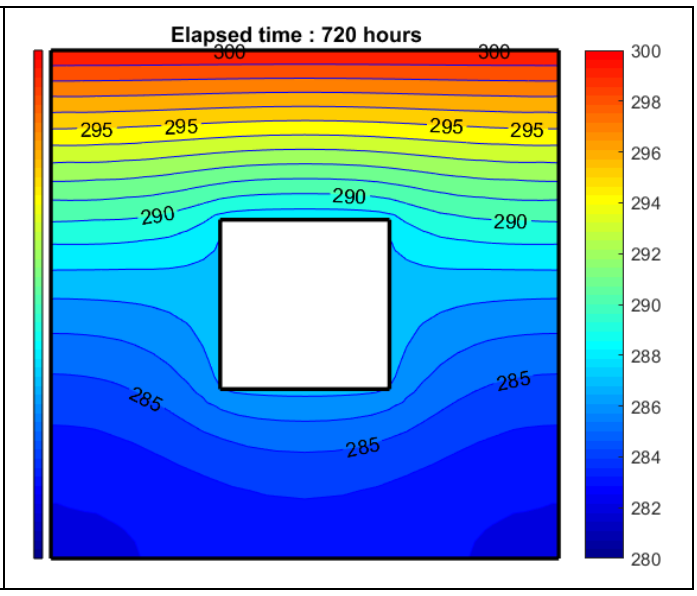


Figure 49: Adiabatic cavity, evolution of the temperature field

4.9 Convection in a square cavity

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

```
Fiammetta_20211111
Standard cavity,  Gi: 2
L    4, Me, Di, Ne    : 3   1   0
L    5, Co nnr nvn    : 2   0   1
L    6, rc, ra, cs    : 0   0   5
L    7, CAD interf.   : 12
L    9, N. pa., vert.: 4   8
L   11, num. nod side: 9
L   13, num elem side: 10
L   23, Domain area   : 8 m2
L   29, Num. elements: 400
L   31, Num. of DOF   : 441
L   62, Domain perim.: 16 m
L   64, Thickness     : 0.1 m
L   65, Conduction k : 1 W/(mK)
L   66, DT isotherms : 1 K
L   76, Convection h : 18 W/(m2K)
c. 03, dim. K Co cs : 441    2    5
c. 04, N.virt c.nod.: 1
t. 04, Initial temp.: 280 K
t. 20, nit  N. iter.: 720
t. 21, Time step     : 3600 s
t. 23, Analyzed per.: 720 h, 30 days
t. 21, Spec. capac. : 1000 J/(kg.K)
```



```
t. 22, Spec. mass    : 2500 kg.m-3
t. 40, sum(sum(C))   : 2 MJ/K
t. 41, area*th*ro*Cp: 2 MJ/K
t. 45, Imposed Heat : 0 W/m-2
t  67, iteration     : 180
t  67, iteration     : 360
t  67, iteration     : 540
t  67, iteration     : 720
t. 80, Tmean - Tini : 8.58 K
t. 81, DT* sumsum(C): 17.2 MJ
t. 83, Min obs. temp: 280 K
t. 84, Max obs. temp: 300 K
t. 93, Tot. reaction: 16.4 MJ
L 231, End Me=3. CPU: 9.49 sec.
```

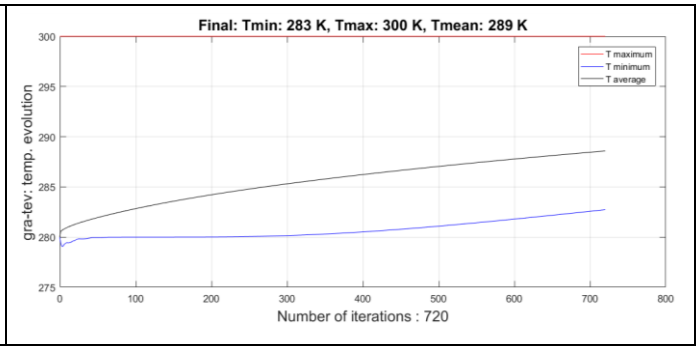


Figure 50: Isotherms, convection in the cavity, 1 month

The heat flow in the conductive medium appears more clearly in the element-by-element heat flows drawing. In *Figure 51*, the flow enters in the cavity essentially 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.

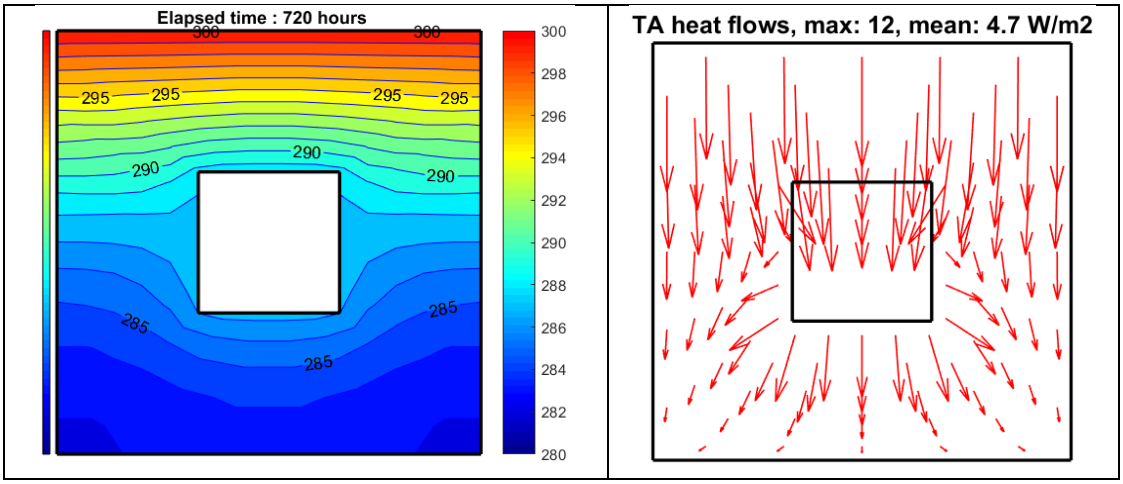


Figure 51: Two visualizations of the heat flow for a mesh of 100 elements

To compute the nodal contributions of the heat flow input equal to 50 x *f(t)* Wm^{-2}, we have to integrate the heat flow intensity over the side of the domain and to distribute it on the nodes. If we assume that the heat flow is constant on the side, and if the mesh is uniform (*n* elements on the side), the weight vector is: [.05 .1 … .1 .05]/*n*, it contains (*n*+1) terms whose sum is equal to 1. For a thickness *e* = 1 *m* and a length *L* =3 *m*, the weight vector has to be multiplied by *e L*. So, the nodal weights are equal to: [.05 .1 … .1 .05]/*n* * 3 m^2. Multiplying by the heat input: 50 Wm^{-2}, the sum of nodal input loads =150 *f(t)* *W*.

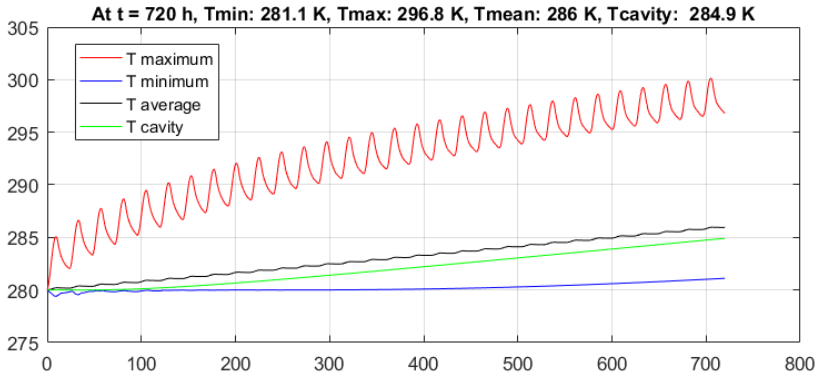


Figure 52: Temperature evolution, convection, 1 month

The loading is carried out by intoducing a sinusoidally varying heat flow over a half-period of twelve hours. During the following half-period the supplied heat is zero. Twenty-four hour cycles are built (*Figure 46*). The injected average power is therefore: 150 / π *W* = 28.648 *W*. the mean heat flow is equal to 28.648 / .8 = 35.81 Wm^{-2}. During each 24 *h* period, the domain is receiving 150 /(π * 3600 * 24) = 4125.3 *kJ*. In a month (720 *h* or 30 days), the input is 4125.3 * 30 = 123.759 *MJ*. If we multiply the difference between the final mean temperature (black curve 285.9 *K*) of the solid and the initial one (280 *K*) by the specific capacity (1000 $J\,kg^{-1}K^{-1}$), the specific mass (2500 $kg\,m^{-3}$) and the volume of the solid (.8 m^3), we obtain the heat quantity of 119 *MJ*.

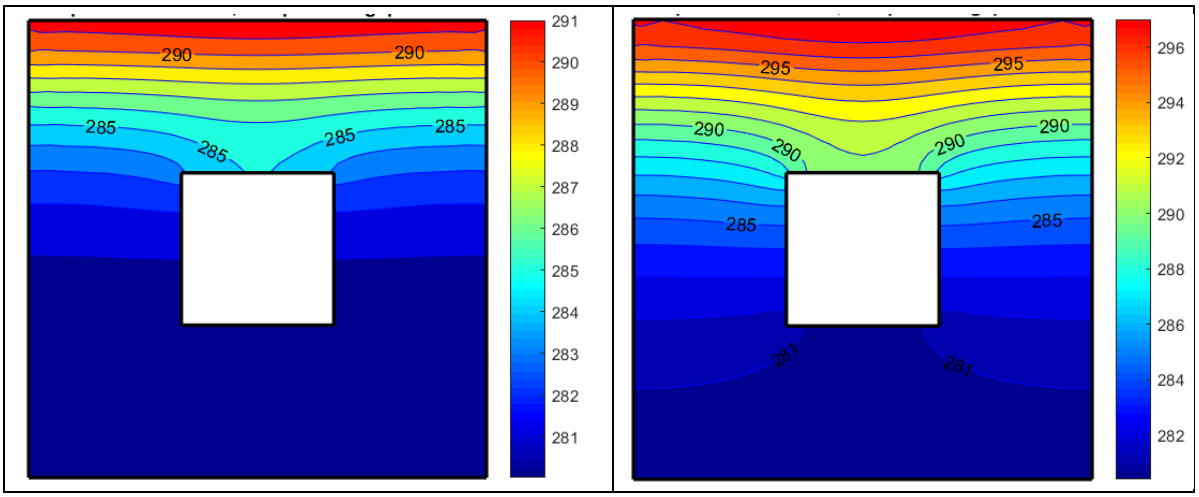


Figure 53: Isotherms after 15 days (T gap = 11.5 K) and 30 days (T gap = 17.8 K)

The Matlab© function *fem_tra.m* (*Table 25*) gives the solution of the iteration *it* of the transient problem when fixations are present.

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
ntca            = dK - nfi;                         % Number of unknowns
ldv             = 1:dK;       % ldv contains the sequence of node numbers
for k           = 1:nfi                        % Detection of the fixed DOF
    ldv(lfi(k)) = -ldv(lfi(k));% In ldv, signs of fixed nodes are negative
end
ldn             = zeros(1,dK);
ifi             = dK+1;
ili             = 0;
for k           = 1:dK
    if ldv(k)   < 0
        ifi     = ifi -1;
        ldn(ifi)= -ldv(k);
    else
        ili     = ili+1;
        ldn(ili)=  ldv(k);
    end
end         % Vector ldn allows moving the fixed DOF at the end of the list
tca             = zeros(dK,1);
Kn              = zeros(dK,dK); Cn = Kn;
tcb             = zeros(dK,1)  ; g2 = tcb;
for k           = 1:dK     % Dirichlet b.c. need shifts in: K, C, tcan & g
    for l       = 1:dK
        Kn(k,l) = K(ldn(k),ldn(l));      % Shifting lines and columns in K
        Cn(k,l) = C(ldn(k),ldn(l));      % Shifting lines and columns in C
    end
    tcb(k)      = tcan(ldn(k));                  % Shifting lines in tcan
    g2 (k)      = g   (ldn(k));                  % Shifting lines in g
end                                  % End shifting DOF in K, C, tcan, g
ta              = ntca+1; % ......................... Compute the solution
tcb (1:ntca)    = (Cn(1:ntca,1:ntca) + dti*Kn(1:ntca,1:ntca))\...
```

```
34      (Cn(1:ntca,1:ntca)*tcb(1:ntca,1)+ ...
35      dti*(g2(1:ntca,1)-Kn(1:ntca,ta:dK)*tcb(ta:dK,1)));% Tu. p. 40, equ. 66
36  for kk = 1:dK                % Restore right sequence of DOF in vector tca
37      tca(ldn(kk),1)=tcb(kk,1);  % tca is the single output of this function
38  end
39  end
```

Table 25: Matlab© function fem_tra.m – solution of the linear transient equations

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 view factor or form factor F_{ij} represents the fraction of radiant energy leaving surface *I* and impinging on surface *j* [Goral *et al* 1984].
In *2D* [Beckers 2011], the "*point – segment*" view factor relates a point to a segment (*Figure 54*) according to the formula:

$$F_{dL-j} = \int_{y \in L_j} \frac{\cos\theta_{dL} \cos\theta_j}{2r} \, dy \tag{81}$$

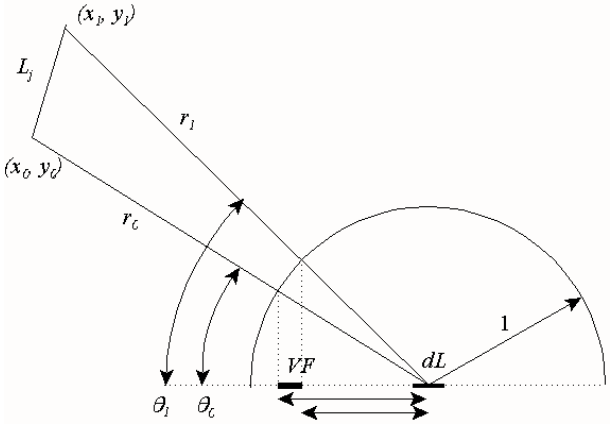


Figure 54: 2D "point – segment" view factor [Beckers 2011]

The explicit expression of the view factor is computed via the Nusselt analogy (*Figure 54*) [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:

$$F_{dL-j} = \frac{1}{2}\left(\frac{x_1}{r_1} - \frac{x_0}{r_0}\right) \tag{82}$$

To obtain the view factors of the elements of a surrounding square from a differential element *dL* (*Figure 54*) 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*]:

$$\sum_{j=1}^{N} F_{ij} = 1 \qquad closure \tag{83}$$

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

$$A_i F_{ij} = A_j F_{ji} \qquad reciprocity \tag{84}$$

In (85), 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 (82) to the side on which the element dL is located. Let be $\vec{r}_i / |r_i|$ the unit vector joining the studied segment to the extremities $I = 0$ and $I = 1$ of the target segments and $\vec{t}$ the tangent to the studied segments. The "*point – segment*" view factor is:

$$F_{dL-j} = \frac{1}{2}\left(\frac{\vec{r}_1}{|\vec{r}_1|} - \frac{\vec{r}_0}{|\vec{r}_0|} \right) \cdot \vec{t} \tag{85}$$

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

$$F_{ij} = \frac{1}{A_i} \int_i F_{dL-j} dA_j \tag{86}$$

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.

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

$$B = E + [R] J \tag{87}$$

In (87), the diagonal matrix $[R]$ contains the reflection coefficients. They satisfy the Kirchhoff' law which is relating, in each element i, the emissivity ε_i, the absorptivity α_i, and the reflectivity ρ_i:

$$R_{ii} = \rho_i \quad ; \quad \varepsilon_i = \alpha_i \quad ; \quad \rho_i = 1 - \varepsilon_i \tag{88}$$

As shown in (87), the radiosity component B_i is the radiant flux leaving the surface i of the solid domain, it is the sum of the emitted E_i and the reflected $\rho_i J_i$ fluxes. The incoming irradiance vector J is related to the other elements of the domain by the view factor matrix $[F]$ defined in (86). 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):

$$J = [F] B + F_{sky} e_{sky} + F_{gr} e_{gr} \tag{89}$$

Introducing the incoming irradiance (89) in (87), we have:

$$B = E + [R]\{\ [F]B\ +\ F_{sky}e_{sky} + F_{gr}e_{gr}\} \tag{90}$$

We now compute the exitances as functions of the radiosities

$$\begin{aligned} E &= B - [R][F]B - [R]F_{sky}e_{sky}\ -\ [R]F_{gr}e_{gr} \\ &= \{[I] - [R][F]\}B - [R]F_{sky}e_{sky} - [R]F_{gr}e_{gr} \\ &= [M]B - [R]F_{sky}e_{sky} - \ [R]F_{gr}e_{gr} \end{aligned} \tag{91}$$

In (91), we have introduced the *radiosity matrix*:

$$[M] = [I] - [R][F] \tag{92}$$

From (91), we deduce:

$$B = [M]^{-1}\{E + [R]F_{sky}e_{sky} + \ [R]F_{gr}e_{gr}\} \tag{93}$$

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

$$Q\ =\ B\ -\ J \tag{94}$$

Replacing in (94) J from (89), we obtain:

$$\begin{aligned} Q\ &=\ \ \ B\ \ \ \ -[F]B\ -F_{sky}\ e_{sky}\ \ -F_{gr}e_{gr} \\ &=\ \{[I]-[F]\}\ \ \ B\ \ \ -F_{sky}\ e_{sky}\ \ -F_{gr}e_{gr} \end{aligned} \tag{95}$$

Replacing B from (93), we finally have:

$$Q = \{[I]-[F]\}[M]^{-1}E + \{[I]-[F]\}[M]^{-1}[R]\{F_{sky}e_{sky} + F_{gr}e_{gr}\} - \{F_{sky}e_{sky} + F_{gr}e_{gr}\} \tag{96}$$

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

$$Q = \{[I]-[F]\}[M]^{-1}E + \left\{\{[I]-[F]\}[M]^{-1}[R] - \lfloor I \rfloor\right\}\{F_{sky}e_{sky} + F_{gr}e_{gr}\} \tag{97}$$

The exitances components, E_i (Wm^{-2}), of the vector E introduced in (87) may be calculated as function of the surface temperature field of the boundary elements i, the emissivity ε_i and the Stefan-Boltzmann constant σ ($Wm^{-2}K^{-4}$):

$$E\ \rightarrow\ E_i\ = \varepsilon_i\ \ \sigma\ \ T_i^4 \tag{98}$$

If present, the sky and ground exitances are obtained in the same way:

$$\begin{aligned} E_{sky}\ &\rightarrow\ E_{sky} = \varepsilon_{sky}\ \ \sigma\ T_{sky}^4 = \sigma\ T_{sky}^4 \text{ (if sky emissivity = 1)} \\ E_{gr}\ &\rightarrow\ E_{gr}\ \ = \ \varepsilon_{gr}\ \ \sigma\ T_{gr}^4\ \ = \sigma\ T_{gr}^4 \text{ (if sky emissivity = 1)} \end{aligned} \tag{99}$$

Throughout (93) to (96), Q is a function of the surface temperature of the radiating solid and can therefore be injected in the finite element model.

$$Q = \{[I]-[F]\}[M]^{-1}\ E(\tau) + \left\{\{[I]-[F]\}[M]^{-1}[R] - \lfloor I \rfloor\right\}\{F_{sky}e_{sky} + F_{gr}e_{gr}\} \tag{100}$$

In (100), the vector $E(\tau)$ is changing at each time iteration step. The last two terms of (99) correspond to the sky and ground radiations which have both imposed temperatures. These terms are classical second members of the system. In (100), 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 (100) 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). 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:

$$H = [W]\, Q \tag{101}$$

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:

$$E_i = \varepsilon_i\ \sigma\ \left(\tau_i^4\right)_{it-1} \ \rightarrow\ (E)_{it-1} \tag{102}$$

For the sky and the ground, the relations are:

$$e_{sky} = \sigma \tau_{sky}^4 \qquad \& \qquad e_{gr} = \sigma \tau_{gr}^4 \tag{103}$$

Equation (99) becomes:

$$Q = \{[I]-[F]\}[M]^{-1}(E)_{it-1} + \left\{\{[I]-[F]\}[M]^{-1}[R]-\lfloor I \rfloor\right\}\left\{F_{sky}\sigma\tau_{sky}^4 \ + \ F_{gr}\sigma\tau_{gr}^4\ \right\} \tag{104}$$

This relation exhibits three second member vectors that have to be introduced in the global finite element system after being expressed in nodal variable.

5.2 View factor matrix

Table 26 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 55*.

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

```
1  function[lcont,lv] = cad_ban(car_cao,bor,pbo,nni,cs)            % 20210929
2  if cs ==1                                                        % Cavity
3      c1    = [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
4      b1    = c1(size(c1,2):-1:1);
5      c2    = [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
6      b2    = c2(size(c1,2)-1:-1:1);
7      c3    = [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
```

```
    b3    = c3(size(c1,2)-1:-1:1);
    c4    = [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];
    b4    = c4(size(c1,2)-1:-1:2);
    lcont = [b1 b2 b3 b4]';                % List of cavity boundary nodes
    lv    = [lcont(1) lcont(nni+2) lcont(2*nni+3) lcont(3*nni+4)];% Vert.
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)];
    lcont = [c1 c2(2:size(c2,2)-1) c3]';   % List of street boundary nodes
    lv    = [lcont(1) lcont(nni+2) lcont(2*nni+3) lcont(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)]];
    lcont = [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    = [lcont(1) lcont(nni+2) lcont(2*nni+3) lcont(3*nni+4)...
         lcont(4*nni+5) lcont(5*nni+6)]';                    % Vertices
end
if cs == 4
    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)]];
    lcont = [bt(1,1:nni+2) bt(2,1:nni+2)]';
    lv   = [lcont(1) lcont(nni+2) lcont(2*nni+4)];
end
if cs ==5 % Quadrilateral:!! previous version replaced by a new 20211021!!
%     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)]];
%     lcont = [bt(1,1:nni+2) bt(2,1:nni+2)]';
%     lv   = [lcont(1) lcont(nni+2) lcont(2*nni+3)];
    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)]];
    lcont = [bt(1,1:nni+2) bt(2,1:nni+2)]';
    lv = [lcont(1) lcont(nni+2) lcont(nni+3) lcont(2*nni+4)];
end
end
```

Table 26: Matlab© function ***cad_ban.m*** *– radiative nodes of cavity, street or balcony*

→ a) Rectangular cavity

The function *geo_vfs.m* (*Table 27*) allows computing the view factor matrix for a rectangle whose length and height are stored in the 2 x 1 uni-column matrix *Lel*.

```
Fiammetta_20211111
Rectangular cavity,  Gi: 12
```

```
[(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
```

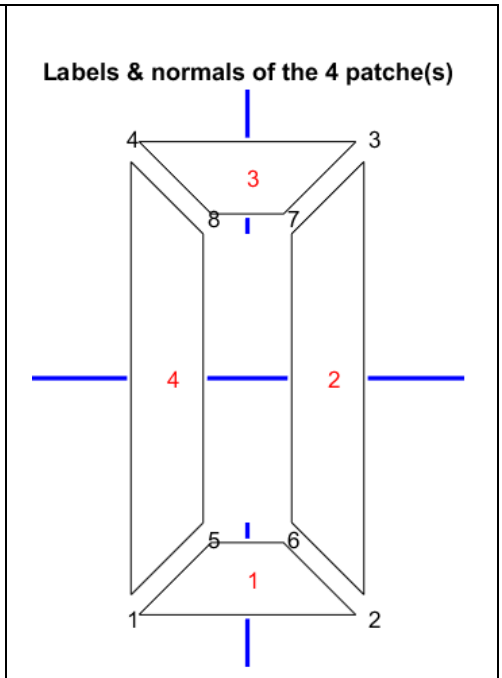


Figure 55: Input data – Gi = 12, Rectangular cavity

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;                % In geo_vfsold.m we have: F(np,:) = f;
end
end
```

Table 27: Matlab© function geo_vfc.m – view factor matrix – ns sides domain

The view factors are calculated using the "*point – segment*" method (86). The cavity is oriented with the sides parallel to the global axes. If all the segments have the same length, the view

factor matrix is symmetric according to the *reciprocity property* (85) and both the sums of columns and lines are equal to 1 (*closure property* (84)). *Table 28* shows the instructions used to visualize the view factor matrix and the *closure property* when there is only one element per square cavity side.

```
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

    1     1     1     1

    1
    1
    1
    1
```

```
Reciprocity property test

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 28: View factor matrix F in a square cavity – 4 segments

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 28*. In the square cavity, the results are the same, using either *geo_vfsold.m* or *geo_vfs.m* functions (*Table 27*).

To improve this evaluation, the best method is to perform a Gauss quadrature on the evaluated segment. The Matlab function *geo_vfr.m* is computing the view factor matrix with two Gauss points leading to the results of *Table 30*.

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);
```

```
29          f2     = (((xn(i0)-xc2)/r0(i)-(xn(i1)-xc2)/r1(i))*...
30              ts(nuc,1)-((yn(i0)-yc2)/r0(i)-(yn(i1)-yc2)/r1(i))*...
31              ts(nuc,2))*ts(nuc,3)/2;
32      end
33      for I       = 1:n;f((nuc-1)*n+i) = 0.;end  % View factor of anal. face
34      F(:,np)     = f; % In geo_vfsold.m we have: F(np,:) = f;
35  end
36  end
37
```

Table 29: Matlab© function geo_vfr.m – view factor matrix – 2 Gauss 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
    1.0000
```

Reciprocity property test (84)

```
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 30: Square cavity – 4 segments – **two** Gauss points per segment*

The difference between these functions is shown as a comment at the end of *line 35* of *Table 29*. Using numerical integration to compute the view factor of the opposite of a square side, we obtain the value of 0.4472 with 1 integration point (*Table 28*), and the value of 0.4130 with 2 Gauss points (*Table 30*). The exact value of 0.4142 is obtained with 3 integration points.

With two elements per cavity side (*Table 31*), the *reciprocity property* (85) 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.

```
F = geo_vfsold (2,[1 1]); disp(F); disp(sum(F)); disp(sum(F,2))
```

```
         0         0    0.0840    0.1160    0.1787    0.2425    0.1023    0.2764
         0         0    0.2764    0.1023    0.2425    0.1787    0.1160    0.0840
    0.1023    0.2764         0         0    0.0840    0.1160    0.1787    0.2425
    0.1160    0.0840         0         0    0.2764    0.1023    0.2425    0.1787
    0.1787    0.2425    0.1023    0.2764         0         0    0.0840    0.1160
    0.2425    0.1787    0.1160    0.0840         0         0    0.2764    0.1023
    0.0840    0.1160    0.1787    0.2425    0.1023    0.2764         0         0
    0.2764    0.1023    0.2425    0.1787    0.1160    0.0840         0         0

     1     1     1     1     1     1     1     1
     1
     1
     1
     1
     1
     1
     1
     1
```

*Table 31: View factor matrix F in a **square** cavity – 8 segments*

```
F = geo_vfsold(2,[1 1]); F-F'
```

```
         0         0   -0.0184         0         0         0    0.0184         0
         0         0         0    0.0184         0         0         0   -0.0184
    0.0184         0         0         0   -0.0184         0         0         0
         0   -0.0184         0         0         0    0.0184         0         0
         0         0    0.0184         0         0         0   -0.0184         0
         0         0         0   -0.0184         0         0         0    0.0184
   -0.0184         0         0         0    0.0184         0         0         0
         0    0.0184         0         0         0   -0.0184         0         0
```

```
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 32: Reciprocity check in a square cavity for result of Table 31

```
F  = geo_vfsold(1,[1 4]);disp(sum(F,1));disp(sum(F,2));
     lon = [1 4 1 4];disp([0 lon; (lon)' F]);
```

```
         0    1.0000    4.0000    1.0000    4.0000
    1.0000         0    0.4380    0.1240    0.4380
    4.0000    0.0528         0    0.0528    0.8944
    1.0000    0.1240    0.4380         0    0.4380
    4.0000    0.0528    0.8944    0.0528         0
```

Closure property:

```
disp(sum(F));columns  0.2296    1.7704    0.2296    1.7704
disp(sum(F,2)');lines   1   1   1   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(j,i); end;end;disp(Reci)
```

```
         0    0.2268         0    0.2268
   -0.2268         0   -0.2268         0
         0    0.2268         0    0.2268
   -0.2268         0   -0.2268         0
```

```
disp(round(Reci/0.2268))
```

```
     0     1     0     1
    -1     0    -1     0
     0     1     0     1
    -1     0    -1     0
```

```
F = geo_vfs(1,[1 4],4);disp(F);disp(sum(F,1));disp(sum(F,2))
```

```
         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

     1     1     1     1
    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(Reci)
```

```
         0   -1.6991         0   -1.6991
    1.6991         0    1.6991         0
         0   -1.6991         0   -1.6991
    1.6991         0    1.6991         0
```

```
disp(round(Reci/1.6991))
```

```
     0    -1     0    -1
     1     0     1     0
     0    -1     0    -1
     1     0     1     0
```

*Table 33: View factor matrix F in a **rectangular** cavity – 4 segments mesh*

```
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
```

```
    1     1     1     1
0.3237
1.6763
0.3237
1.6763
```

```
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(j,i); end;end;disp(Reci);disp(lon)
```

```
      0   -1.6535         0   -1.6535
 1.6535         0    1.6535         0
      0   -1.6535         0   -1.6535
 1.6535         0    1.6535         0
```

```
disp(round(Reci/1.6535))
     0    -1     0    -1
     1     0     1     0
     0    -1     0    -1
     1     0     1     0
```

*Table 34: **Rectangular** cavity – 4 segments mesh, 2 Gauss points*

```
F = geo_vfsold (2,[1 4]);
;lon=[.5 .5 2 2 .5 .5 2 2];disp([0 lon;(lon)' F]);
disp(sum(F)); disp(sum(F,2));disp(sum([sum(F,2)'])/8);
```

```
      0    0.5000    0.5000    2.0000    2.0000    0.5000    0.5000    2.0000    2.0000
 0.5000         0         0    0.3244    0.0834    0.0610    0.0624    0.0308    0.4380
 0.5000         0         0    0.4380    0.0308    0.0624    0.0610    0.0834    0.3244
 2.0000    0.0937    0.0528         0         0    0.0068    0.0189    0.1208    0.7071
 2.0000    0.0189    0.0068         0         0    0.0528    0.0937    0.7071    0.1208
 0.5000    0.0610    0.0624    0.0308    0.4380         0         0    0.3244    0.0834
 0.5000    0.0624    0.0610    0.0834    0.3244         0         0    0.4380    0.0308
 2.0000    0.0068    0.0189    0.1208    0.7071    0.0937    0.0528         0         0
 2.0000    0.0528    0.0937    0.7071    0.1208    0.0189    0.0068         0         0
```

Closure property:

```
disp(sum(F));columns 0.2954 0.2954 1.7046 1.7046    0.2954    0.2954    1.7046    1.7046
disp(sum(F,2)');lines
     1
     1
     1
     1
     1
     1
     1
     1
```

Table 35: View factor matrix F in a rectangular cavity – 8 segments mesh

```
F = geo_vfs(2,[1 4],4);lon=[.5 .5 2 2 .5 .5 2 2];
disp([0 lon;(lon)' F]) ; disp(sum(F)); disp(sum(F,2)');
                disp(sum([sum(F,2)'])/8);
```

```
      0    0.5000    0.5000    2.0000    2.0000    0.5000    0.5000    2.0000    2.0000
 0.5000         0         0    0.0937    0.0189    0.0610    0.0624    0.0068    0.0528
 0.5000         0         0    0.0528    0.0068    0.0624    0.0610    0.0189    0.0937
 2.0000    0.3244    0.4380         0         0    0.0308    0.0834    0.1208    0.7071
 2.0000    0.0834    0.0308         0         0    0.4380    0.3244    0.7071    0.1208
 0.5000    0.0610    0.0624    0.0068    0.0528         0         0    0.0937    0.0189
 0.5000    0.0624    0.0610    0.0189    0.0937         0         0    0.0528    0.0068
 2.0000    0.0308    0.0834    0.1208    0.7071    0.3244    0.4380         0         0
 2.0000    0.4380    0.3244    0.7071    0.1208    0.0834    0.0308         0         0
```

Closure property:

```
disp(sum(F) lines   1  1  1  1  1  1  1  1
disp(sum(F,2)) columns
    0.2954
    0.2954
    1.7046
    1.7046
    0.2954
    0.2954
    1.7046
    1.7046
```

```
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   -0.1405   -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));disp(sum(Reci,2)')
```

```
    1.3150    1.3150   -0.2582   -0.2582    1.3150    1.3150   -0.2582   -0.2582
   -0.2582   -0.2582    1.3150    1.3150   -0.2582   -0.2582    1.3150    1.3150
```

Table 36: View factor matrix F in a rectangular cavity – 8 segments mesh

```
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
```

Closure property:

```
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
```

```
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   -0.1369   -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
```

Table 37: Rectangular cavity – 8 segments – 2 Gauss points

→ b) Street section

For the street section analyses, we use the parameters Gi = 14 when radiation is present and *G*i =15 for the linear transient analyses. (Matlab© function *cad_gin.m*, *Table 71*).

```
[(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
```

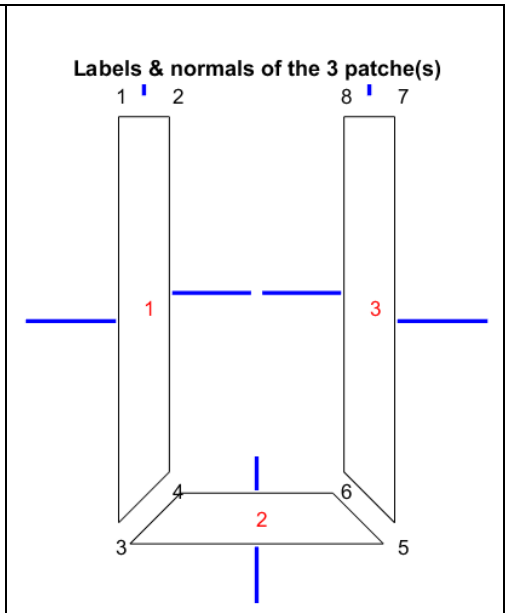


Figure 56: Data – Gi = 14, Gi = 15: Street section, aspect ratio dx/dy = 3/7

After the cavity, we analyze how radiative exchanges behave in an open space like a street section. We define a domain with 8 vertices, 3 patches and 10 interfaces. This domain has an area of 20 m^2 and a perimeter of 42 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 variable *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 inserted two nodes per interface (*line 10*). The domain is represented at the right of *Figure 56* 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 38*.

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

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

```
23          f(i)   = (((xn(i0)-xc(np))/r0(i)-(xn(i1)-xc(np))/r1(i))*...
24              ts(nuc,1)-((yn(i0)-yc(np))/r0(i)-(yn(i1)-yc(np))/r1(i))*...
25              ts(nuc,2))*ts(nuc,3)/2;
26      end
27      for I      = 1:n;f((nuc-1)*n+i) = 0.;end    % View factor of anal. face
28      F (np,:)   = f;
29  end
30  end
```

Table 38: Matlab© function geo_stf.m – view factor matrix – street section

A simple instruction is used to display the view factor matrices both for the rectangular cavity (*Figure 55*) using the Matlab© function *geo_vfc.m* of *Table 27* and the street section (*Figure 56*) using the Matlab© function *geo_stf.m* of *Table 38*. 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.

```
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 39: 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 40: Street section (6 segments): view factor matrix and sky view factor vector

```
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 41: Street section (9 segments): view factor matrix and sky view factor vector

→ 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 57*), 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 58*), the vertical wall is always seeing half the ground, while the horizontal one is seeing more than half the ground.

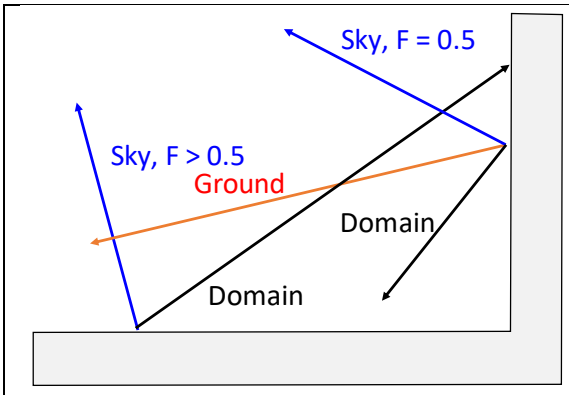


Figure 57: Vertical wall above horizontal edge

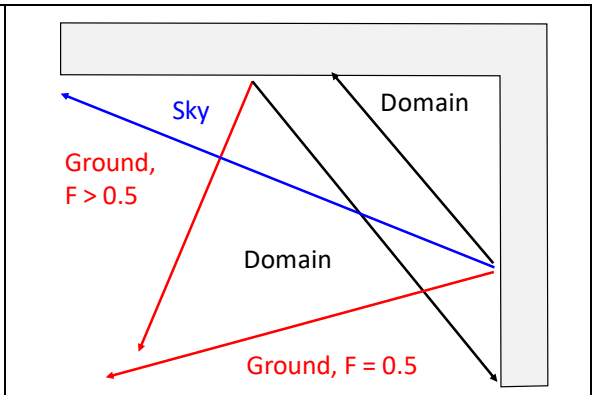


Figure 58: Vertical wall below horizontal edge

→ d) Thermal bridge

In this section we define the simple situation of a thermal bridge (*Figure 59*)

For the thermal bridge analyses, we use the parameters *Gi* = ?? when radiation is present and *Gi* =16 for the linear transient analyses. (Matlab© function *cad_gin.m*, *Table 71*).

```
[(1:npv)'  xyz_cao]
```

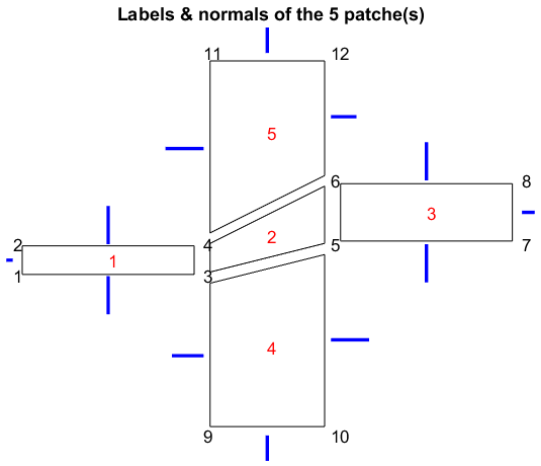
1	-2	5
2	-2	6
3	4	5
4	4	6
5	8	6
6	8	8
7	14	6
8	14	8
9	4	0
10	8	0
11	4	12
12	8	12

```
[(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 59: Data – Gi = 16: Thermal bridge

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

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

```
    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 42: Matlab© function geo_baf.m – view factor matrix – wall with balcony

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 43: 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 43* & *Figure 60*), with the consequence that the corresponding lines and columns of matrix *F* are also equal to zero.

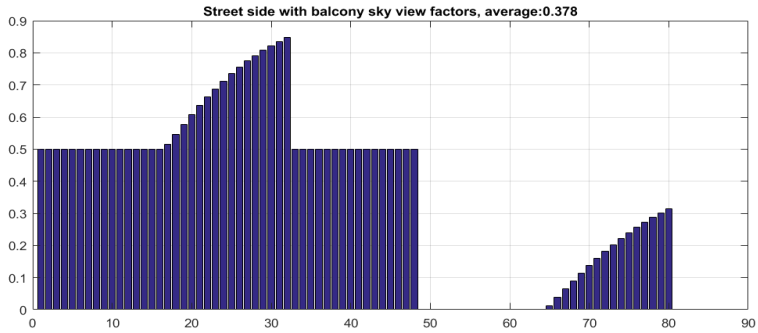


Figure 60: Fsky in a street with balcony – 16 segments / patch side

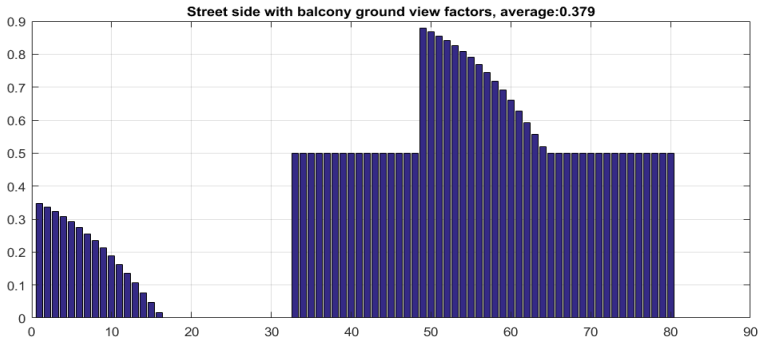


Figure 61: Fgr in a street with balcony – 16 segments / patch side

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 44*. 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 44*, by using the instructions *disp (sum(M)); disp(sum(M'))*.

Radiosity matrix [*M*] as a function of the view factor matrix [*F*]:

```
re = 1; F = geo_vfs(2,[1 1],4); M = eye(size(F,1))-re*F
```

1.0000	0	-0.0840	-0.1160	-0.1787	-0.2425	-0.1023	-0.2764
0	1.0000	-0.2764	-0.1023	-0.2425	-0.1787	-0.1160	-0.0840
-0.1023	-0.2764	1.0000	0	-0.0840	-0.1160	-0.1787	-0.2425
-0.1160	-0.0840	0	1.0000	-0.2764	-0.1023	-0.2425	-0.1787
-0.1787	-0.2425	-0.1023	-0.2764	1.0000	0	-0.0840	-0.1160
-0.2425	-0.1787	-0.1160	-0.0840	0	1.0000	-0.2764	-0.1023
-0.0840	-0.1160	-0.1787	-0.2425	-0.1023	-0.2764	1.0000	0
-0.2764	-0.1023	-0.2425	-0.1787	-0.1160	-0.0840	0	1.0000

```
disp (sum(M))
```

```
   1.0e-15 *
   -0.0555   -0.0833   -0.1110   -0.0555    0.0278   -0.0555         0         0
```

Table 44: 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 39*, *Table 40*, *Table 41*).

Street view factor matrix `F = geo_stf(2,[3 7])`

0	0	0.0192	0.0466	0.1822	0.5039
0	0	0.1204	0.1277	0.5039	0.1822
0.0515	0.3952	0	0	0.2296	0.1174
0.1174	0.2296	0	0	0.3952	0.0515
0.1822	0.5039	0.1277	0.1204	0	0
0.5039	0.1822	0.0466	0.0192	0	0

`F = geo_stf(1,[3 7])`

0	0.1204	0.7593
0.3952	0	0.3952
0.7593	0.1204	0

Table 45: Street section: view factor matrices

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:

Matlab©: `re = 0.;M = eye(size(F,1)) - re*F`

1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
0	0	0	0	0	1

1	0	0
0	1	0
0	0	1

Table 46: Street section: radiosity matrices, ρ = 0

Matlab©: `re=1;M = eye(size(F,1)) - re*F`

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

1.0000	-0.1204	-0.7593
-0.3952	1.0000	-0.3952
-0.7593	-0.1204	1.0000

Table 47: Street section: radiosity matrices, ρ = 1

The sky view factor uni-column matrix *Fsky* is obtained as the complement to unity of the view factor matrix *Fs* (*line 146* in *Fiammetta.m*):

Matlab©: Fsky = 1-sum(Fs,2)'
Fsky' = 0.2481 0.0659 0.2063 0.2063 0.0659 0.2481 For 1 segment per side, *Fsky'* = 0.1204 0.2095 0.1204 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(['Sky view factor SVF in a street section, max, min, mean in %: ', num2str([max(SVF) min(SVF) mean(SVF)],2)],'fontsize', 20) ylabel('%', 'fontsize', 20)

Table 48: Street section: sky view factor – uni-column matrix Fsky

Its distribution along the street walls (*Figure 62*) is computed with the Matlab© instruction of *Table 48*.

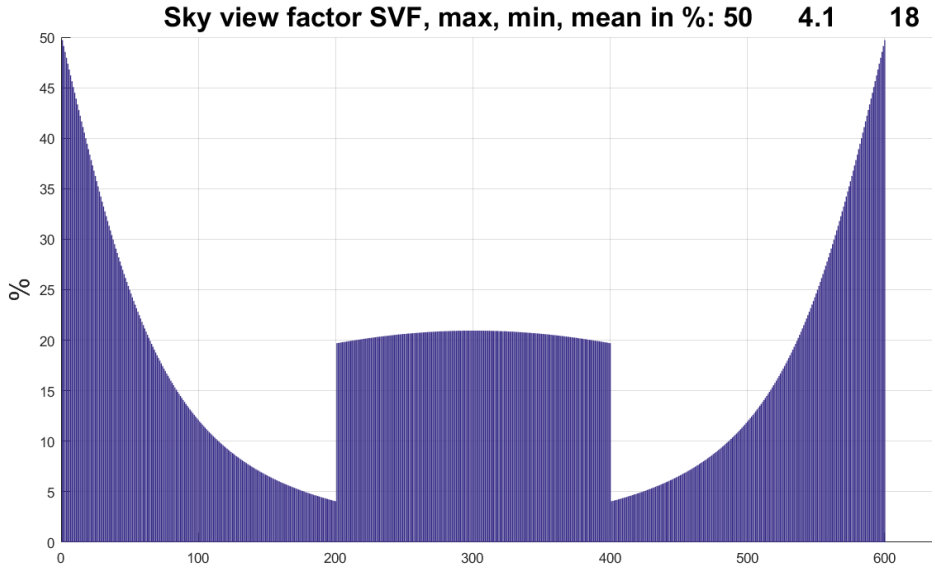


Figure 62: Sky View Factors in the street section – 200 radiative segments per side

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 %.

When we have one segment per side, the view factor matrix of the street segments with respect to themselves yet computed in *Table 45* are shown with an extra column giving the sky view factor in *Table 49* (left side). The closure is satisfied for each line. This matrix is a sub-matrix of the transposed view factor matrix (right side) of a rectangle where lines 2, 3, 4 are conserved and column 1 shifted in the fourth position.

Fs=geo_stf(1,[3 7]);disp([Fs 1-sum(Fs,2)])				geo_vfs (1, [3 7],4)			
0	0.1204	0.7593	0.1204	0	0.1204	0.2095	0.1204
0.3952	0	0.3952	0.2095	0.3952	0	0.3952	0.7593
0.7593	0.1204	0	0.1204	0.2095	0.1204	0	0.1204
				0.3952	0.7593	0.3952	0

Table 49: Street section: full view factor matrix including segments and sky

6. Tutorial VI: Radiative heat exchanges

In the simple situation where the domain edge is either horizontal with only sky above it 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 we do not matter them.

6.1 A simple convex domains

In this situation, we simply introduce elements on the radiative boundary (*Table 75*). On a pure horizontal edge, the sky view factor is equal to 1 and the ground view factor to zero. On a vertical wall, both are equal to 0.5.

To test this situation, we first propose a problem with virtual convective nodes on both vertical sides of the domain. We observe that the difference of temperature between both sides is equal to 4 *K*, while the differences of temperature between the walls and the virtual nodes is 8 *K*. We compare this solution shown in *Figure 63* to the problem including a radiative wall on the left and a convective one on the right.

```
Fiammetta_20211111
Standard rect. 2 squares, Gi: 7
L    4, Me, Di, Ne    : 3  7  0
L    5, Co nnr nvn    : 3  0  2
L    6, rc, ra, cs    : 0  0  0
L    7, CAD interf.   : 7
L    9, N. pa., vert.: 2  6
L   11, num. nod side: 9
L   13, num elem side: 10
L   23, Domain area  : 2 m2
L   29, Num. elements: 200
L   31, Num. of DOF  : 233
L   62, Domain perim.: 6 m
L   64, Thickness    : 0.1 m
L   65, Conduction k : 1 W/(mK)
L   66, DT isotherms : 1 K
L   72, Fixed DOF    : 232  233
L   76, Convection h : 18 W/(m2K)
c. 03, dim. K Co cs : 233    3    0
c. 04, N.virt c.nod.: 2
t. 04, Initial temp.: 280 K
t. 20, nit  N. iter.: 720
t. 21, Time step    : 3600 s
t. 23, Analyzed per.: 720 h, 30 days
t. 21, Spec. capac. : 1000 J/(kg.K)
t. 22, Spec. mass   : 2500 kg.m-3
t. 40, sum(sum(C))  : 0.5 MJ/K
t. 41, area*th*ro*Cp: 0.5 MJ/K
t. 45, Imposed Heat : 0 W/m-2
t  67, iteration    : 180
t  67, iteration    : 360
t  67, iteration    : 540
t  67, iteration    : 720
t. 89, Tmean - Tini : 10 K
t. 90, DT* sumsum(C): 5.02 MJ
t. 92, Min obs. temp: 280 K
t. 93, Max obs. temp: 300 K
t.102, Tot. reaction: 5 MJ
L 231, End Me=3. CPU: 2.3 sec.
```

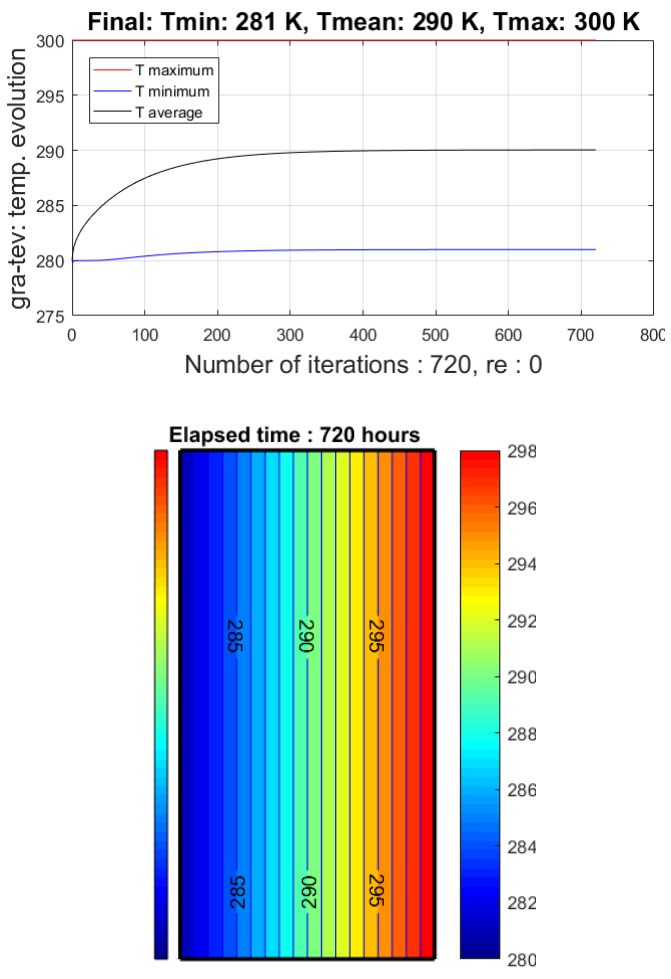


Figure 63: Rectangle with two convective walls

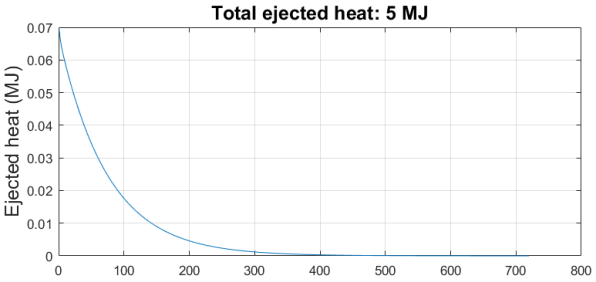


Figure 64: Evolution of the outgoing heat plotted at line 91 of fem_til.m (Table 62)

The convergence is obtained after about 100 iterations.

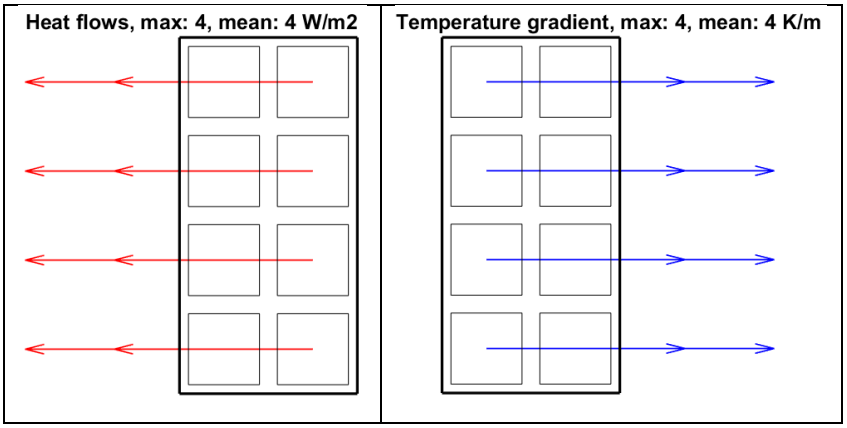
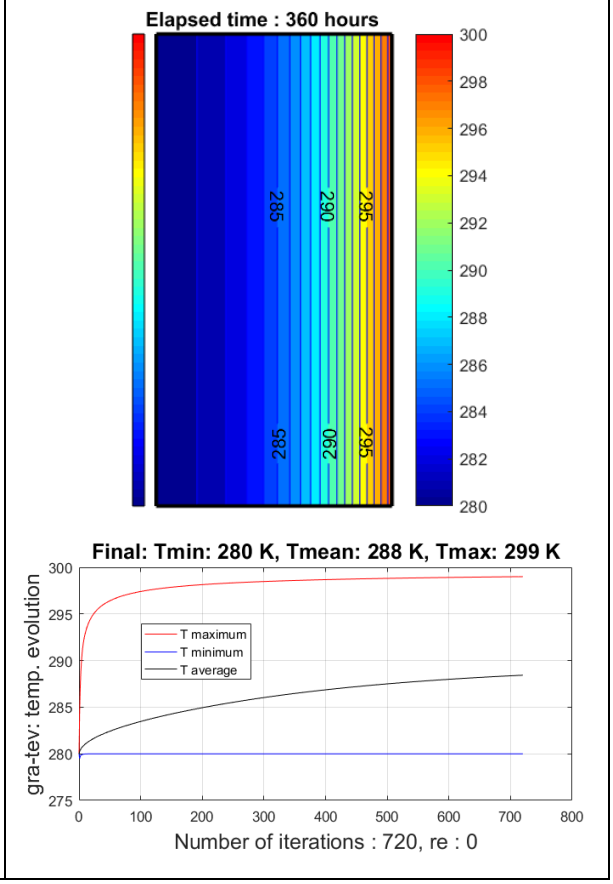


Figure 65: Domain with constant temperature gradient and heat flow

This solution is compared to the problem with convection on the right wall and radiation on the left one.

```
Fiammetta_20211111
Standard rect. 2 squares, Gi: 10
L   4, Me, Di, Ne    : 4  9  0
L   5, Co nnr nvn    : 4  1  1
L   6, rc, ra, cs    : 1  0  4
L   7, CAD interf.   : 7
L   9, N. pa., vert.: 2  6
L  11, num. nod side: 15
L  13, num elem side: 16
L  23, Domain area   : 2 m2
L  26, Num. elements: 512
L  28, Num. of DOF   : 563
L  59, Domain perim.: 6 m
L  61, Thickness     : 0.1 m
L  62, Conduction k  : 0.2 W/(mK)
L  63, DT isotherms  : 1 K
L  65, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD 53, Fixed nodes   : 562  563
LD 54, Fix. temper.  : 280  300 K
L  69, Fixed DOF     : 562  563
L  76, Convection h  : 5 W/(m2K)
c. 03, Numb. var. dK: 563
c. 04, N.virt c.nod.: 1
L  96, Anis. index   : 0
L 125, Sky temper.   : 280 K
L 130, Rad. edg. len: 1  1 m
L 131, Radiativ area: 0.2 m2
L 132, Refl. coeff.  : 0.5
L 233, Num. fix. DOF: 562  563
L 234, T. fix. nod.  : 280  300 K
t. 04, Initial temp.: 280 K
t. 05, Fixed DOF     : 562  563
t. 08, Time step     : 3600 s
t. 10, Analyzed per.: 720 h, 30 days
t. 16, Spec. capac.  : 1000 J/(kg.K)
t. 17, Spec. mass    : 2500 kg.m-3
t. 28, sum(sum(C))   : 0.5 MJ/K
```



```
t. 29, area*th*ro*Cp: 0.5 MJ/K
t. 32, Impos. Heat  : 0 W/m-2
t. 38, mcr nv       : 33   1
t. 45, N. rad. edges: 2
t. 46, rad seg leng.: 1   1 m
t. 47, N. rad. elem.: 32
t. 48, rad sid leng.: 0.0625 … 0.0625    m
t 120, Iteration    : 360 gt: 280 1  300 K
t 120, Iteration    : 720 gt: 280 1  300 K
t.142, Tmean - Tini : 8.43 K
t.143, Stored heat  : 4.22 MJ
t.145, Min it+1 temp: 280 K
t.146, Max it+1 temp: 299 K
t.152, Ejected heat : 0.202 MJ
L 240, Diss in solid: 7.54 WK
L 241, Total dissip.: 8.03 WK
L 242, End Me=4. CPU: 5.69 sec.
```

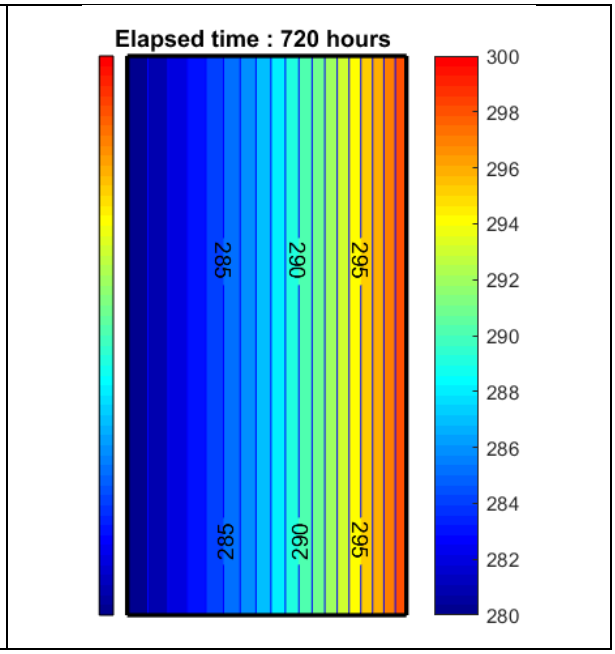


Figure 66: Rectangle with one radiative wall (left) and one convective wall (right)

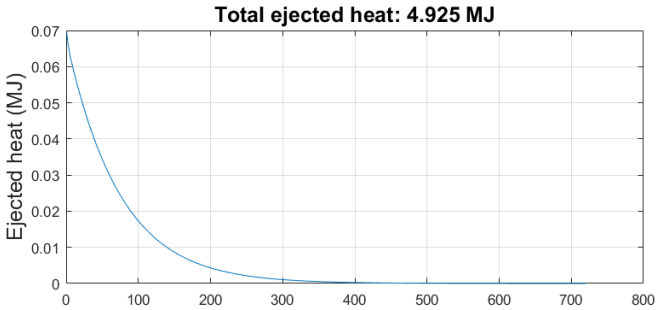


Figure 67: Evolution of the outgoing heat, see line 125 in fem_tir.m (Table 63)

Using the Matlab notations, we verify that the convective (*fem_Kcv.m*, *Table 6*) and radiative (*fem_Kcr.m*, *Table 75*) matrices are both multiple of:

```
SB    = 5.6704e-8; th= 1; SBt = SB*th;tcan = ones(17,1)*280;
fem_Kcr(xyz,lc(I,😊),SBt,tcan)/((280^2 + 280^2)*280*SBt)-
fem_Kcv(xyz,lc, 20*th) % The difference is a nil matrix.
```

```
    0.3333    0.1667   -0.5000
    0.1667    0.3333   -0.5000
   -0.5000   -0.5000    1.0000
```

Table 50: Convective and radiative matrices for boundary segments

The difference is a nil matrix.

6.2 Square cavity

The second test is the square cavity. The description of the cavity boundary is obtained by calculating the list *lcont* of the cavity boundary nodes ordered from bottom left vertex and following the border, cavity area left. 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 26*). This description is using the matrices *bor* and *pbo* computed in *cad_mes.m* (*Table 15*).

The radiosity equations allow computing the radiative contribution of the conductivity matrix on the boundaries of the closed cavity. In *lines 5* and following of the function *fem_caK.m*, *Table 51*, we observe the use of temperatures defined both on the element sides (variable *cam*) and on the nodes (variable *aaa*).

Matlab© function *fem_caK.m* quadrilateral cavity

```
function[Kr] = fem_caK(tca,re,SBt,Mpr,it,lcont,lon)
ndK          = size(tca,1);       % Size of the global conductivity matrix
Kr           = zeros(ndK,ndK);       % Init. of cavity conductivity matrix
mc           = size(lcont,1);                % Number of radiative nodes
cat          = tca(lcont);        % Unicolumn of cavity nodal temperatures
jt=1000;
if it ==jt;disp(['caK 1, Temperature n: ',num2str(cat')]);end
cam(1:mc-1,1)= (cat(1:mc-1,1)+cat(2:mc,1))/2;  % Cavity mid-segments temp.
cam(mc,1)    = (cat(mc,1)+cat(1,1))/2;% Unicol cavity midside temperatures
if it ==jt;disp(['caK 2, Temperature s: ',num2str(cam')]);end
N            = size(lcont,1);% Number of elements along the radiating zone
lel          = zeros(N,1);nci = N/4;% nci = numb. elements per cavity side
for i            = 1 : nci   % Compute lengths of N elements cavity border
    lel(i)       = lon(1); lel(i+nci)   = lon(2);
    lel(i+2*nci) = lon(3); lel(i+3*nci) = lon(4);
end
Les          = eye(N).*lel*(1-re);% Diag. mat. seg. lengths, 4 equal sides
if it ==jt;disp(['caK 3, Lengths x emi: ',num2str(lel(:,1)'*(1-re))]);end
aa           = Mpr*SBt*Les*cam.^3;  % Unicolumn of midside radiative terms
aaa(2:mc,1)  = (aa(1:mc-1,1)+aa(2:mc,1))/2; % Vector of nodal third powers
aaa(1,1)     = (aa(mc,1)+aa(1,1))/2;                  % of radiative terms
for i        = 1:mc;Kr(lcont(i),lcont(i)) = aaa(i);end% Diag. cond. matrix
if it ==jt;disp(['caK 4, Addit conduct: ',num2str(aaa'),' W/K']);end
end
```

Table 51: Matlab© function fem_caK.m – radiative K_r in a cavity

Input of *fem_caK.m*:

- 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*,
- list *lcont* of the *DOF* on the border of the cavity (computed in *fem_cca.m*),
- lengths *lon* of the elements of the radiating sides and

Output of *fem_caK.m*:

- contribution *Kr* to the global matrix of the radiative terms.

Matlab© function *fem_rsm.m* second member ina radiatve open section

```
function[Mn] = fem_rsm(tca,re,SBt,Mpr,it,Ms,lcont,lon,dK)   % Open section
mcr          = size(lcont,1);                % Number of radiative nodes
cat          = tca(lcont);          % Nodal temperatures of radiative nodes
si           = mcr-1; Mn = zeros(dK,1);
cam(1:si)    = (cat(1:si,1)+cat(2:mcr,1))/2;   % Mid-segments temperatures
aa           = Mpr(1:si,1:si)*eye(si).*lon*(1-re)*SBt*cam'.^4;% heat loads
aaa(1,1)     = aa(1)/2; aaa(mcr,1) = aa(si)/2;
for i        = 1:si-1;  aaa(i+1,1) = (aa(i,1)+aa(i+1,1))/2;end
Mss(1,1)     = Ms(1)/2; Mss(mcr,1) = Ms(si)/2;
for i        = 1:si-1;  Mss(i+1,1) = (Ms(i,1)+Ms(i+1,1))/2;end
for i        = 1:mcr; Mn(lcont(i)) = aaa(i) + Mss(i);end
if it == 1
 if mcr < 10;disp(['.rs 13, Iter. number: ',num2str(it')])            ;end
 if mcr < 10;disp(['.rs 14, cam (t-mid) : ',num2str(cam) ,' K'])      ;end
 if mcr < 10;disp(['.rs 15, heat mid aa : ',num2str(aa') ,' W'])      ;end
%  if mcr < 10;disp(['.rs 16, Mr mid-edges: ',num2str(Mr') ,' W'])    ;end
 if mcr < 10;disp(['.rs 17, aaa nodes   : ',num2str(aaa'),' W'])      ;end
 if mcr < 10;disp(['.rs 18, Mss nodes   : ',num2str(Mss'),' W'])      ;end
 if mcr < 10;disp(['.rs 19, Mn Tot nodes: ',num2str(Mn(lcont')),' W']);end
             disp(['.rs 20, sum(aaa)    : ',num2str(sum(aaa)  ),' W'])
             disp(['.rs 21, sum(Mss)    : ',num2str(sum(Mss)  ),' W'])
end
end
```

Table 52: Matlab© function fem_rsm.m - second member in a radiatve open section

Input of *fem_rsm.m*:

- 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 *fem_cca.m*),
- lengths *lon* of the elements of the radiating sides.
- *dK* is the dimension of the system of equations

Output of *fem_opK.m*:

- contribution *Mn* to the second member of heat equations.

a) Black body square cavity $\rho = 0$

```
Fiammetta_20211111
Standard cavity,  Gi: 11
L    4, Me, Di, Ne    : 5   0   1
L    5, Co nnr nvn    : 0   0   0
L    6, rc, ra, cs    : 1   1   1
L    7, CAD interf.   : 12
L    9, N. pa., vert.: 4   8
L   11, num. nod side: 7
L   13, num elem side: 8
L   23, Domain area   : 8 m2
L   29, Num. elements: 256
L   31, Num. of DOF   : 288
L   62, Domain perim.: 16 m
L   64, Thickness     : 0.1 m
L   65, Conduction k : 1 W/(mK)
L   66, DT isotherms : 1 K
L   68, St. Boltzmann: 5.6704e-08 W/(m2K4)
L   72, Fixed DOF     : 0
N   12, Loaded  area : 0.3 m2
N   31, Loaded nodes : 3   58 … 63   64    4
L   92, Numb. r.nodes: 32
L   94, Rad. vertices: 5   6   7   8
L 100, Anis. index   : 0
L 134, Rad. edg. len: 1   1   1   1 m
L 135, Radiativ area: 0.4 m2
L 136, Refl. coeff. : 0
L 140, sum(sum(M))   : 32
Lt 04, Initial temp.: 280 K
Lt 07, Time step     : 3600 sec
Lt 09, Analyzed per.: 720 h, 30 days
Lt 15, Spec. capac. : 1000 J/(kg.K)
Lt 16, Spec. mass    : 2500 kg.m-3
Lt 29, sum(sum(C))   : 2 MJ/K
Lt 35, Ampl inp heat: 50 W/m-2
Lt 79, iteration     : 360
Lt 79, iteration     : 720
Lt108, Tmean - Tini : 5.96 K
Lt109, Min obs. temp: 280 K
Lt110, Max obs. temp: 301 K
Lt111, Final temper.: 281 286 297
Lt113, Capac * Del T: 11.9 MJ
Lt119, Injected heat: 12.3 MJ
Lt120, Exact ih x 30: 12.4 MJ
L 272, End Me=5. CPU: 2.12 sec.
```

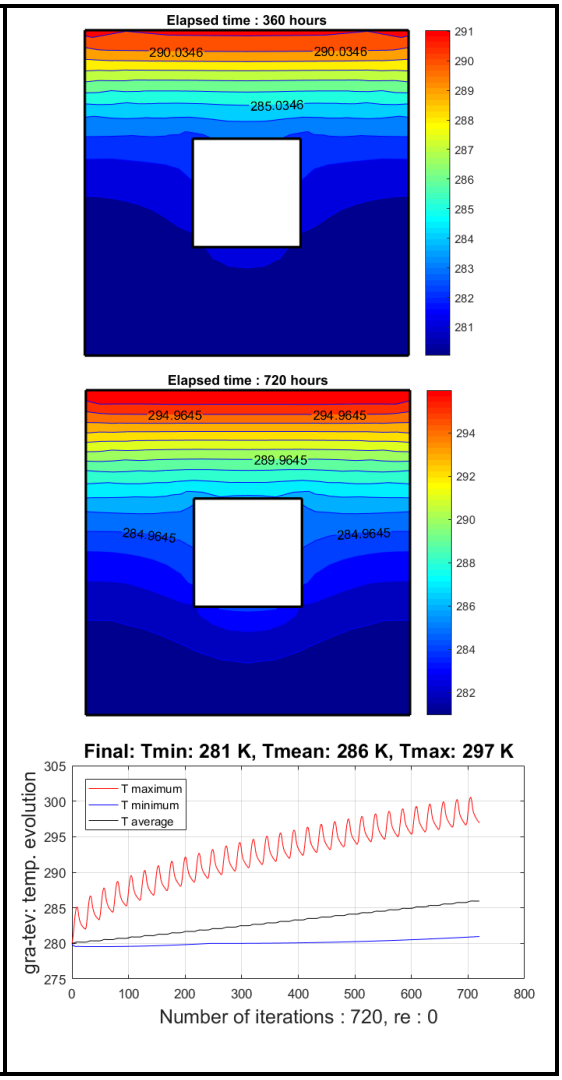


Figure 68: Isotherms, radiation, 256 elements, 1 month, $\rho = 0$

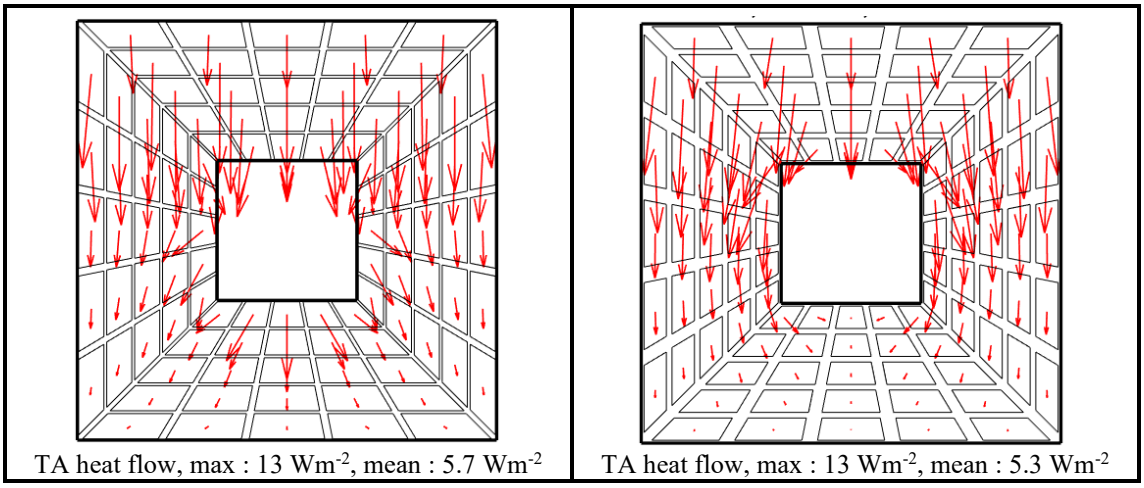


Figure 69: Element heat flow s after 30 days, left: black body, $\rho = 0$; right: mirror, $\rho = 1$

The generalized nodal heat flows along the boundary of the cavity are shown in *Figure 70*. The second member *gsm* of the conductivity system of equations where the radiative term have been removed are computed with the pure conductivity matrix [*Kk*]). The *Figure 70* uses the classical Matlab© bar function on the reduced set of second member limited to the *DDL* listed in *lcont*. The diagram shows the values of the nodal heat loads expressed in *W*. They are ordered from the left bottom vertex along the boundary, cavity left. In this example, the number of elements per side is equal to 15 with a total of 64 nodes after adding the four vertices.

```
gsm = Kk*tca; figure ('Position', [100 100 800 300]); bar (gsm(lcont)'); grid on
title (['Emittance: ', num2str(1-re)])
```

Kr T = εσT⁴ (W)

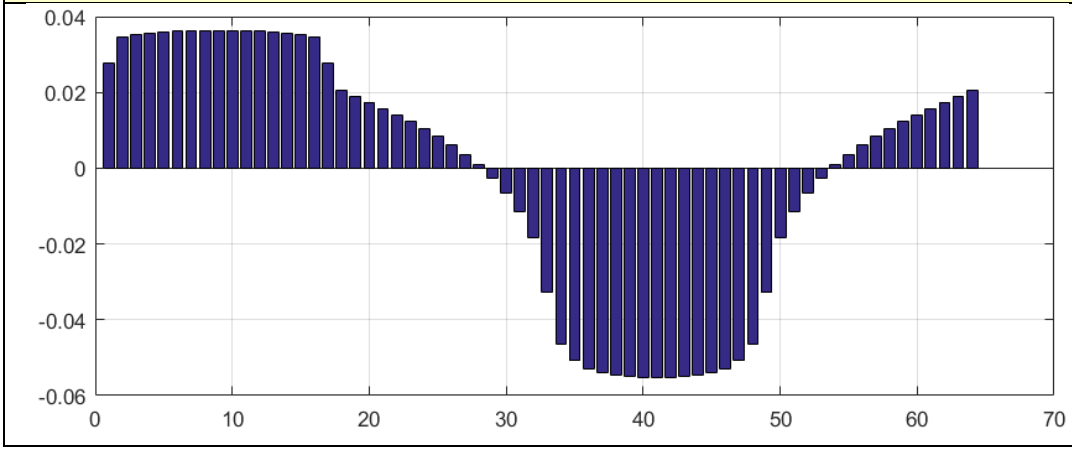


Figure 70: Nodal heat loads on the cavity boundary (16 elem. per side)

b) Gray body square cavity

```
Fiammetta_20211111
Standard cavity,  Gi: 11
L    4, Me, Di, Ne    : 5   0   1
L    5, Co nnr nvn    : 0   0   0
L    6, rc, ra, cs    : 1   1   1
L    7, CAD interf.   : 12
L    9, N. pa., vert.: 4   8
L   11, num. nod side: 7
L   13, num elem side: 8
L   23, Domain area   : 8 m2
L   29, Num. elements: 256
L   31, Num. of DOF   : 288
L   62, Domain perim.: 16 m
L   64, Thickness     : 0.1 m
L   65, Conduction k : 1 W/(mK)
L   66, DT isotherms : 1 K
L   68, St. Boltzmann: 5.6e-08 W/(m2K4)
L   72, Fixed DOF     : 0
N   12, Loaded  area : 0.3 m2
L   92, Numb. r.nodes: 32
L   94, Rad. vertices: 5   6   7   8
L 101, Anis. index   : 0
L 129, Sky temper.   : 400 K
L 134, Rad. edg. len: 1   1   1   1 m
L 135, Radiativ area: 0.4 m2
L 136, Refl. coeff. : 0.5
L 140, sum(sum(M))   : 16
Lt 04, Initial temp.: 280 K
Lt 07, Time step     : 3600 sec
Lt 09, Analyzed per.: 720 h, 30 days
Lt 15, Spec. capac. : 1000 J/(kg.K)
Lt 16, Spec. mass    : 2500 kg.m-3
Lt 29, sum(sum(C))   : 2 MJ/K
Lt 35, Ampl inp heat: 50 W/m-2
Lt 79, iteration     : 360
Lt 79, iteration     : 720
Lt108, Tmean - Tini : 5.95 K
Lt109, Min obs. temp: 280 K
Lt110, Max obs. temp: 301 K
Lt111, Final temper.: 281 286 297
Lt113, Capac * Del T: 11.9 MJ
Lt119, Injected heat: 12.3 MJ
Lt120, Exact ih x 30: 12.4 MJ
L 272, End Me=5. CPU: 2.35 sec.
```

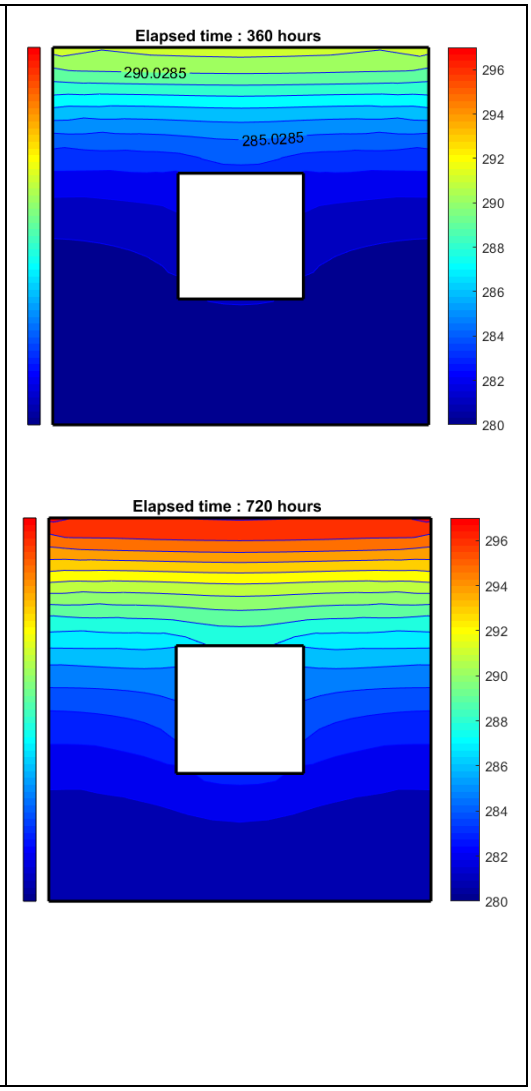


Figure 71: Isotherms, radiation, 800 elements, 1 month, $\rho = 0.5$

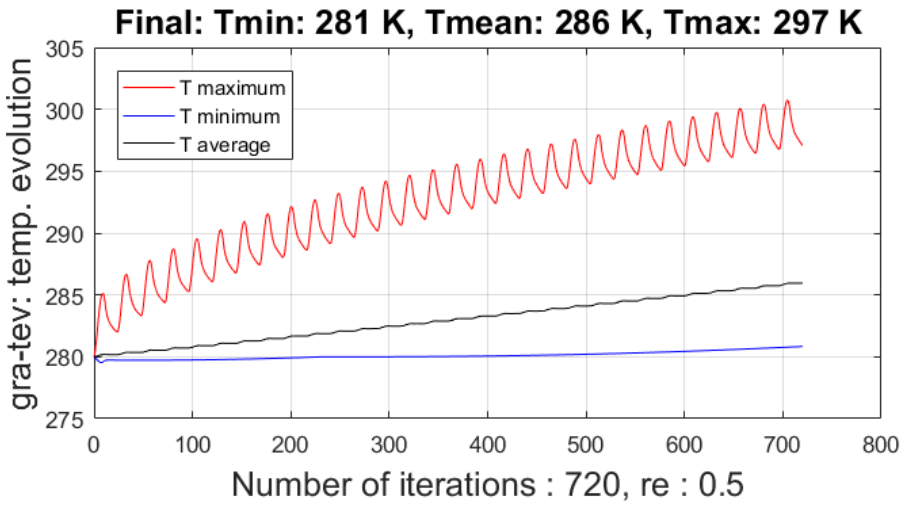


Figure 72: T evolution, radiative exchanges, 256 elements, 1 month, $\rho = 0.5$

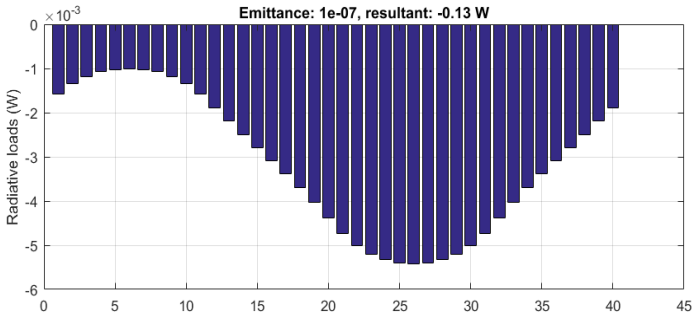


Figure 73: Generalized loads Kr $T = \varepsilon\sigma T^4$ (W), 40 nodes, $\rho = 1$, max: 510^{-3}

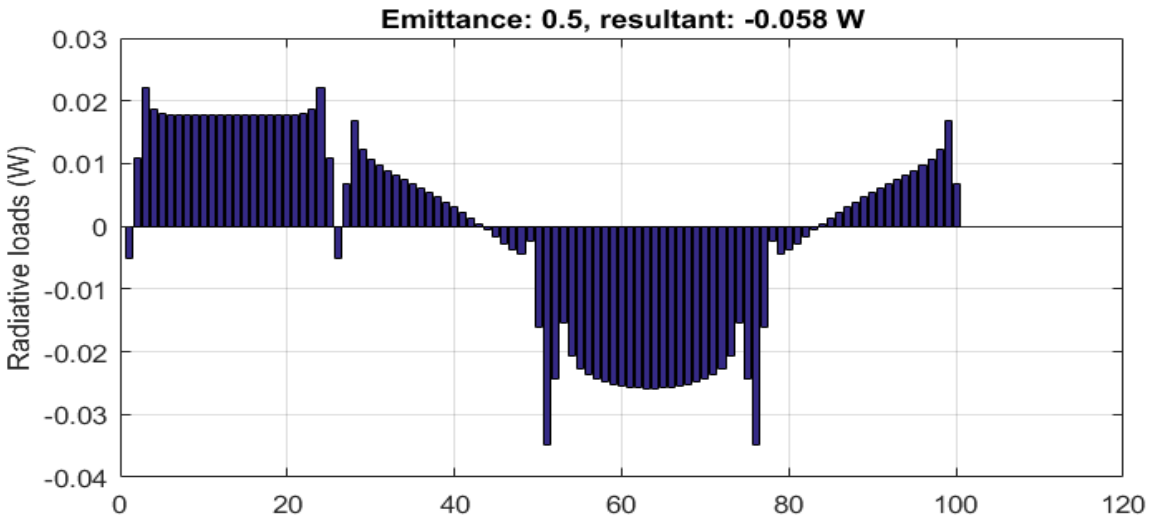
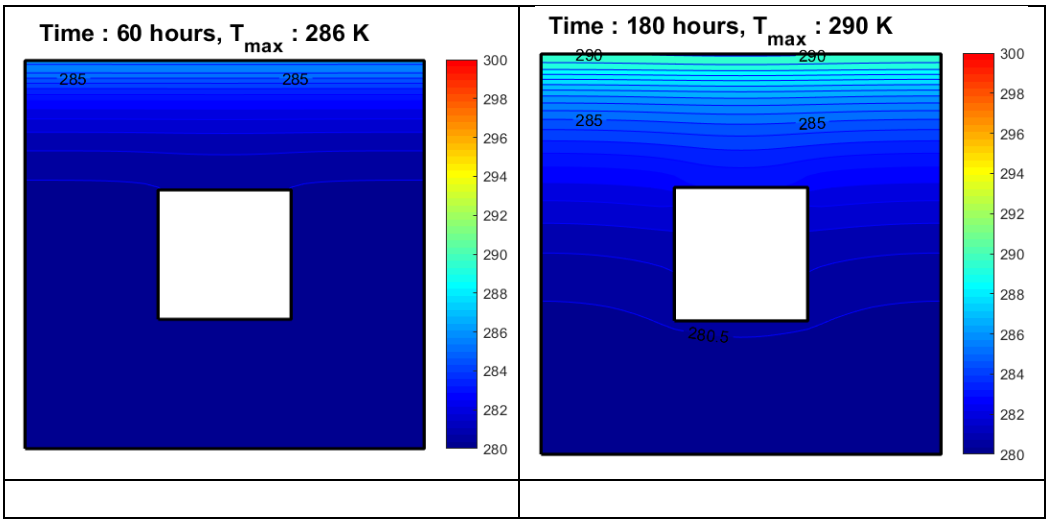


Figure 74: Generalized loads Kr $T = \varepsilon\sigma T^4$ (W), 100 cavity nodes



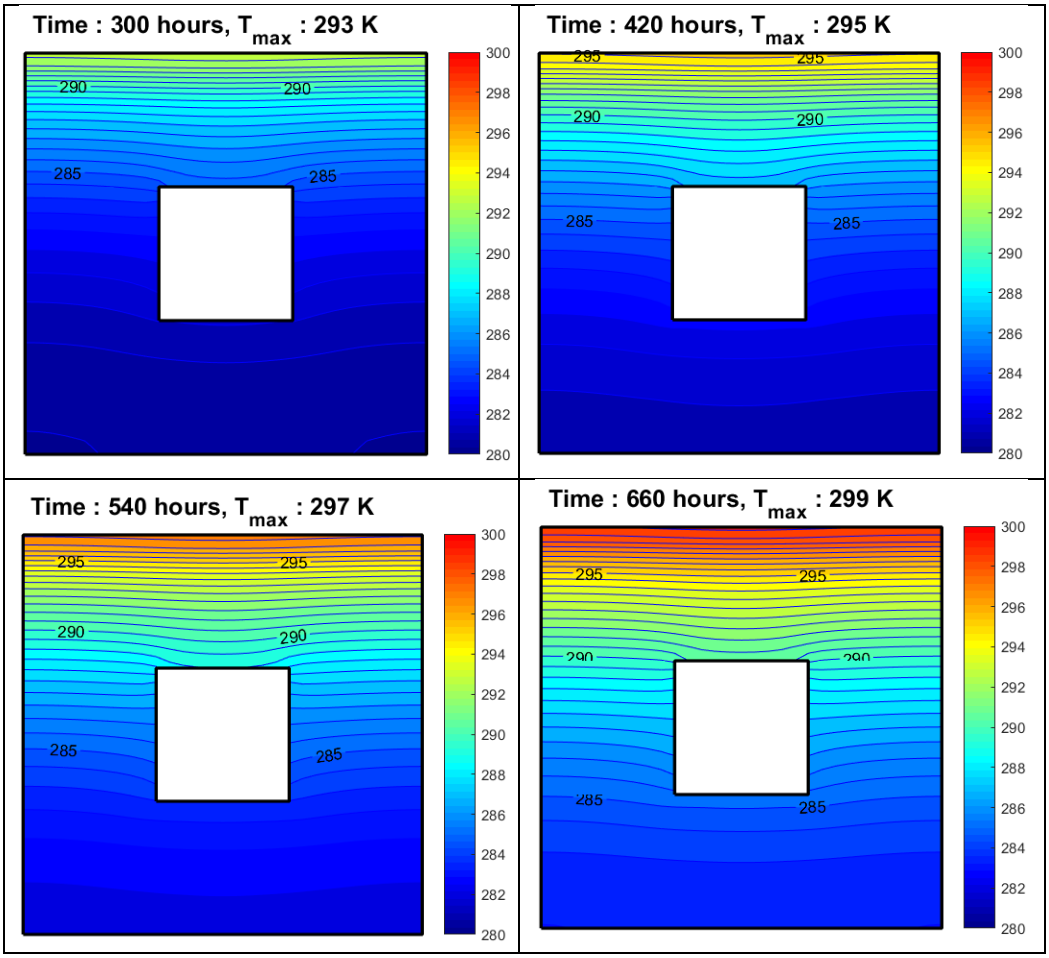
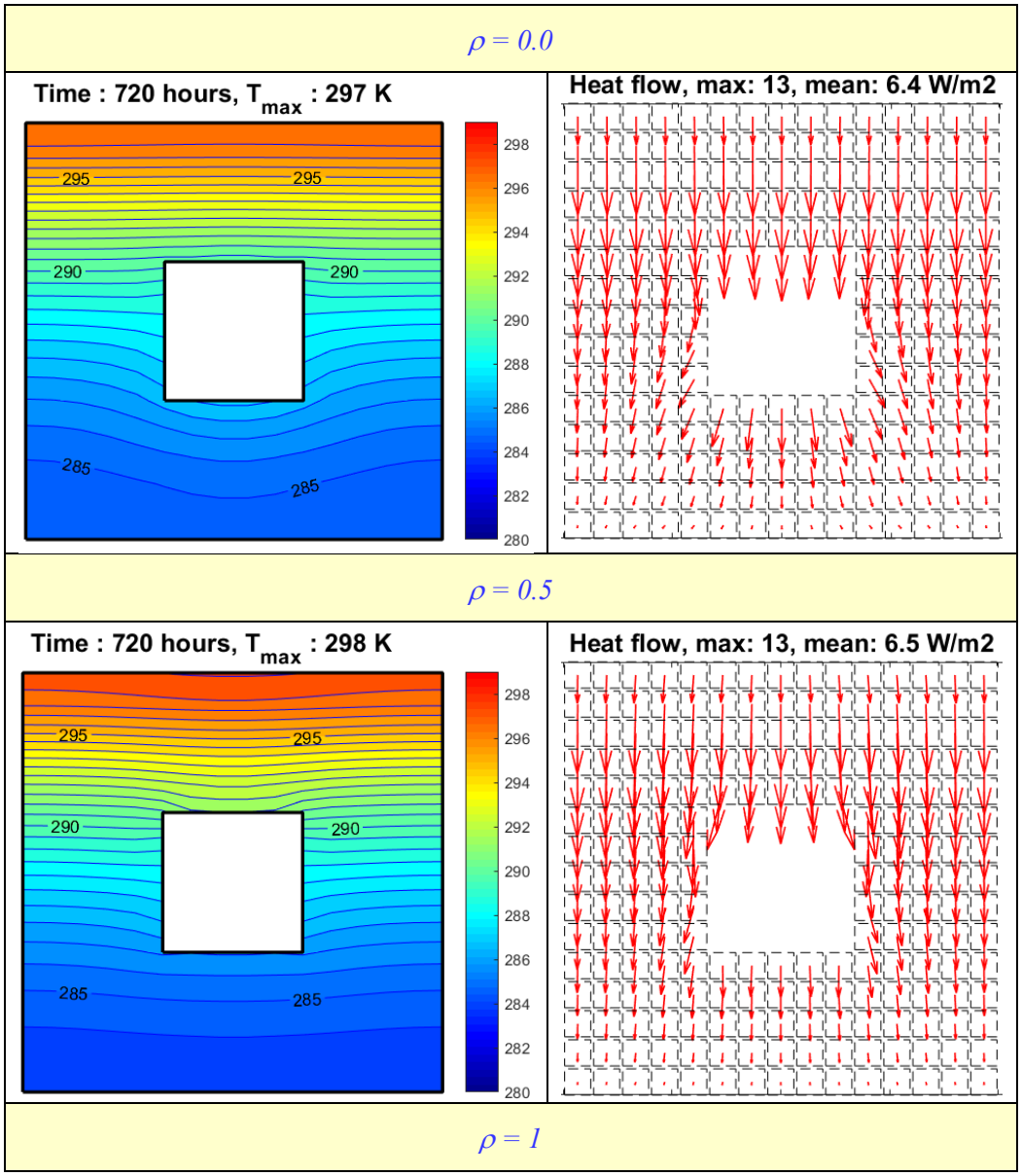


Figure 75: Isotherms, gray body, evolution after 60, 120, …, 660 hours, $\rho = 0.5$

c) Comparison of gray cavities

In *Figure 76*, tests are carried out with reflection coefficients equal to 0, 0.5 and 1, over a period of 30 days. The differences are well marked on the isothermal diagrams and even better on the representations of heat fluxes. Presented in the form of graphical animations, these results should help the user to better understand these physical phenomena in their four dimensions (space and time).



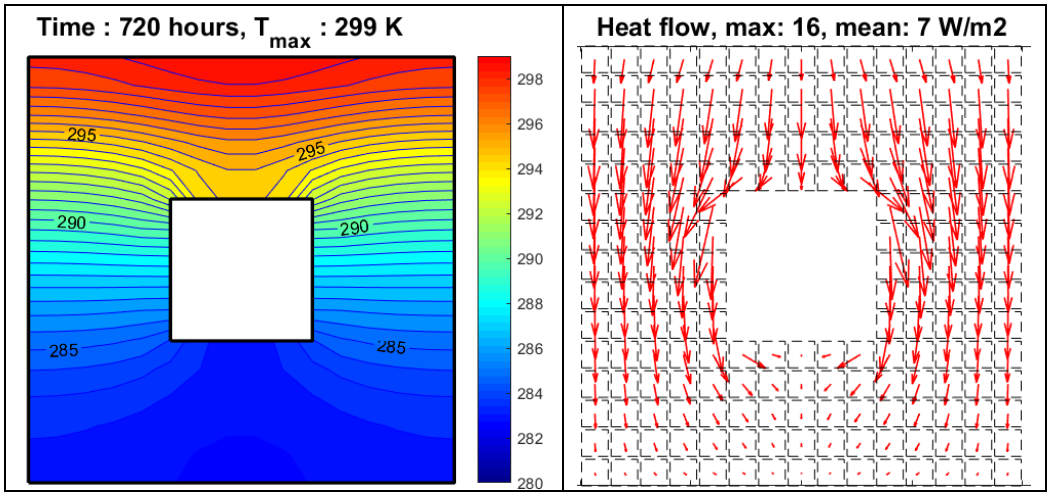


Figure 76: Isotherms and heat flows, black, gray body & mirror, 30 days

6.3 Rectangular cavity

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 $\varepsilon = 0$)

```
Fiammetta_20211111
Rectangular cavity,  Gi: 13
L   4, Me, Di, Ne    : 3  8  0
L   5, Co nnr nvn    : 0  0  0
L   6, rc, ra, cs    : 0  0  0
L   7, CAD interf.   : 12
L   9, N. pa., vert.: 4  8
L  11, num. nod side: 1
L  13, num elem side: 2
L  23, Domain area   : 14 m2
L  29, Num. elements: 16
L  31, Num. of DOF   : 24
L  62, Domain perim.: 28 m
L  64, Thickness     : 0.1 m
L  65, Conduction k : 10 W/(mK)
L  66, DT isotherms : 1 K
L  72, Fixed DOF     : 1  11   2
L 101, Anis. index   : 0
t. 04, Initial temp.: 300 K
t. 20, nit  N. iter.: 720
t. 21, Time step     : 3600 s
t. 23, Analyzed per.: 720 h, 30 days
t. 21, Spec. capac. : 1000 J/(kg.K)
t. 22, Spec. mass    : 2500 kg.m-3
t. 40, sum(sum(C))   : 3.5 MJ/K
t. 41, area*th*ro*Cp: 3.5 MJ/K
t. 45, Imposed Heat : 100 W/m-2
t  67, iteration     : 180
t  67, iteration     : 360
t  67, iteration     : 540
t  67, iteration     : 720
t. 80, Tmean - Tini : -19.3 K
t. 81, DT* sumsum(C): -67.5 MJ
t. 90, Heat inp.(81): 24.8 MJ
L 231, End Me=3. CPU: 1.94 sec.
```

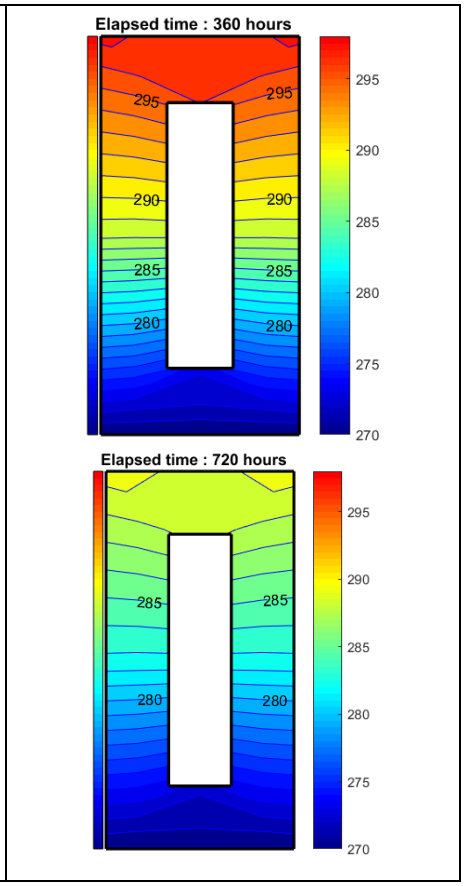


Figure 77: Adiabatic rectangular cavity

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 ($\rho = 1$), which means that the boundary is adiabatic. The results shown in *Figure 78* are the same as those of *Figure 77*.

```
Fiammetta_20211111
Rectangular cavity,  Gi: 12
L   4, Me, Di, Ne    : 5  8  0
L   5, Co nnr nvn    : 0  0  0
L   6, rc, ra, cs    : 1  0  1
L   7, CAD interf.   : 12
L   9, N. pa., vert.: 4  8
L  11, num. nod side: 1
L  13, num elem side: 2
L  23, Domain area   : 14 m2
L  29, Num. elements: 16
L  31, Num. of DOF   : 24
L  62, Domain perim.: 28 m
L  64, Thickness     : 0.1 m
L  65, Conduction k : 10 W/(mK)
L  66, DT isotherms : 1 K
L  68, St. Boltzmann: 5.6704e-08 W/(m2K4)
L  72, Fixed DOF     : 1  11   2
L  92, Numb. r.nodes: 8
L  93, Num. vertices: 5  6  7  8
L  94, R. nod. lcont: 5 9 6 15 7 18 8  19
L 101, Anis. index   : 0
L 134, Rad. edg. len: 1  4  1  4 m
L 135, Radiativ area: 1 m2
L 136, Refl. coeff. : 1
L 140, sum(sum(M))   : 1.0755e-16
Lt 04, Initial temp.: 300 K
Lt 07, Time step     : 3600 sec
Lt 09, Analyzed per.: 720 h, 30 days
Lt 15, Spec. capac. : 1000 J/(kg.K)
Lt 16, Spec. mass    : 2500 kg.m-3
Lt 29, sum(sum(C))   : 3.5 MJ/K
Lt 35, Ampl inp heat: 0 W/m-2
Lt 40, Loaded  area : 0.4 m2
Lt 79, iteration     : 360
Lt 79, iteration     : 720
Lt108, Tmean - Tini : -19.2 K
Lt109, Min obs. temp: 270 K
Lt110, Max obs. temp: 304 K
Lt111, Final temper.: 270 281 290
Lt113, Capac * Del T: -67.2 MJ
Lt119, Injected heat: 0 MJ
Lt122, ejected heat : -52.4 MJ
L 258, End Me=5. CPU: 1.29 sec.
```

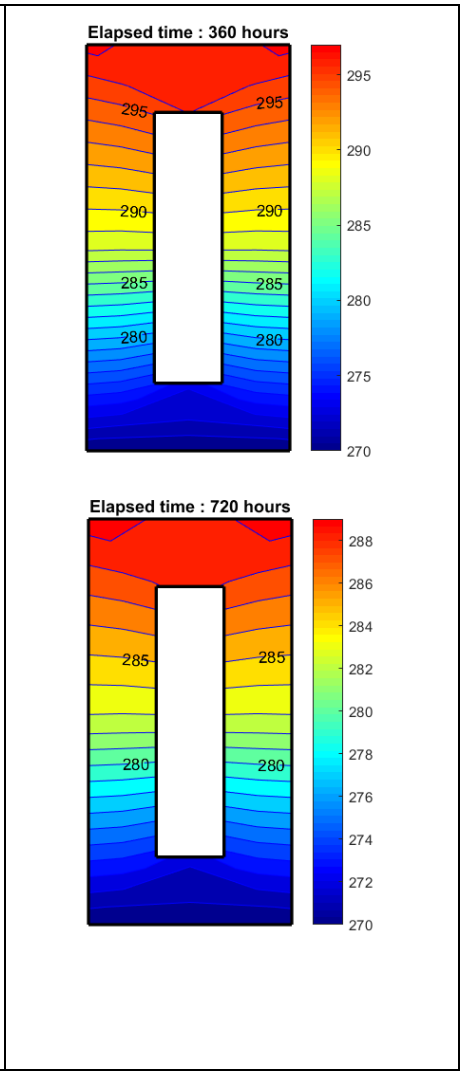


Figure 78: Cavity with adiabatic boundary $\varepsilon = 0$, $\rho = 1$

For coarse, medium and fine meshes with respectively 16, 256 and 1024 elements, we obtain the temperature evolutions shown in *Figure 79* and *Figure 80*. The error present in the first steps of the integration process disappears with the refinement of the mesh. With 16 elements per patch side, the error is equal to 0.85 *K* at iteration 1 and 0.1 *K* at iteration 6.

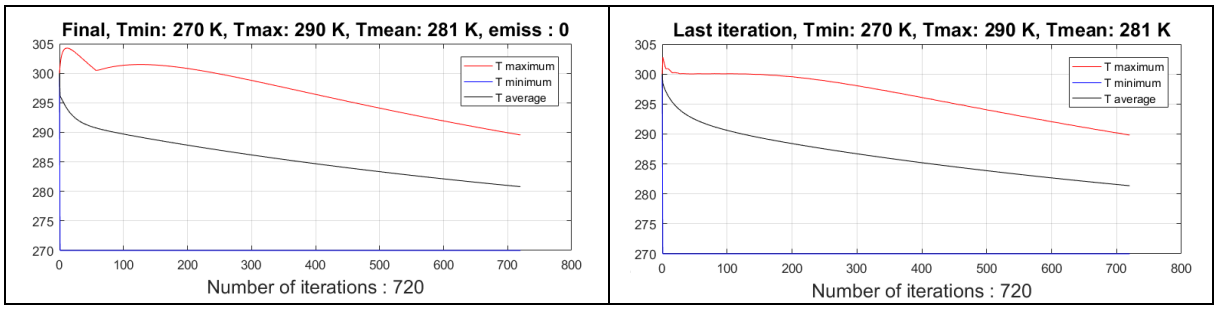


Figure 79: Temperature evolution, $\varepsilon = 0$, $\rho = 1$, 16 – 256 elements

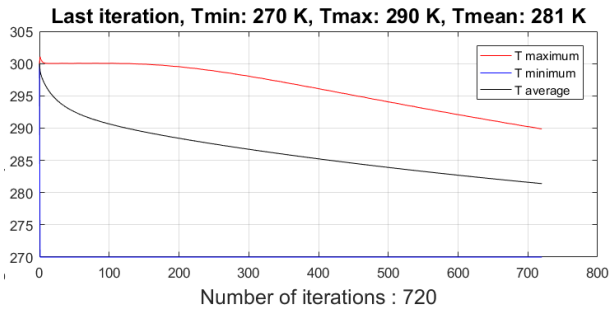


Figure 80: Temperature evolution, $\varepsilon = 0$, $\rho = 1$, 1024 elements

Introducing radiative exchanges inside the cavity does not change the result if the reflection coefficient is equal to 1, which means that the borders of the cavity are adiabatic. However, when we introduce some emissivity on the cavity boundary: $\rho = 0.9$, the behavior of the solution is changing drastically. If we define a reflection coefficient less than one, we observe that the symmetry is not perfectly achieved. It is due to the lack of symmetry of the view factor matrix (see prints of lines *L 151* to *L 153*). However, when the mesh is refined this lack of symmetry is decreasing.

```
Fiammetta_20211111
Rectangular cavity,  Gi: 12
L    4, Me, Di, Ne    : 5  8  0
L    5, Co nnr nvn    : 0  0  0
L    6, rc, ra, cs    : 1  0  1
L    7, CAD interf.   : 12
L    9, N. pa., vert.: 4  8
L   11, num. nod side: 1
L   13, num elem side: 2
L   23, Domain area   : 14 m2
L   29, Num. elements: 16
L   31, Num. of DOF   : 24
L   62, Domain perim.: 28 m
L   64, Thickness     : 0.1 m
L   65, Conduction k : 10 W/(mK)
L   66, DT isotherms : 1 K
L   68, St. Boltzmann: 5.6704e-08 W/(m2K4)
L   72, Fixed DOF     : 1  11   2
L   92, Numb. r.nodes: 8
L   93, Num. vertices: 5  6  7  8
L   94, R. nod. lcont: 5 9 6 15 7 18 8  19
L 101, Anis. index   : 0
L 134, Rad. edg. len: 1  4  1  4 m
L 135, Radiativ area: 1 m2
L 136, Refl. coeff. : 0.9
L 140, sum(sum(M))   : 0.8
Lt 04, Initial temp.: 300 K
Lt 07, Time step     : 3600 sec
Lt 09, Analyzed per.: 720 h, 30 days
Lt 15, Spec. capac. : 1000 J/(kg.K)
Lt 16, Spec. mass    : 2500 kg.m-3
Lt 29, sum(sum(C))   : 3.5 MJ/K
Lt 35, Ampl inp heat: 0 W/m-2
Lt 40, Loaded  area : 0.4 m2
Lt 79, iteration     : 360
Lt 79, iteration     : 720
Lt108, Tmean - Tini : -19.5 K
Lt109, Min obs. temp: 270 K
Lt110, Max obs. temp: 304 K
Lt111, Final temper.: 270 280 289
Lt113, Capac * Del T: -68.3 MJ
Lt119, Injected heat: 0 MJ
Lt122, ejected heat : -53.2 MJ
L 258, End Me=5. CPU: 1.45 sec.
```

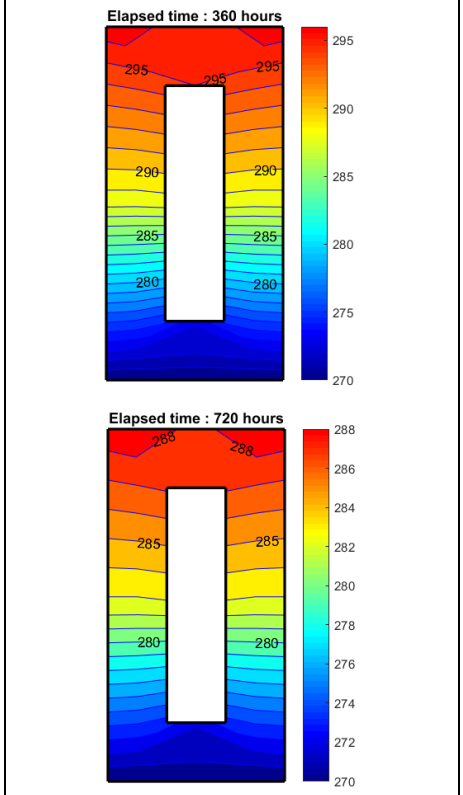


Figure 81: Cavity, $\varepsilon = .1$, $\rho = 0.9$

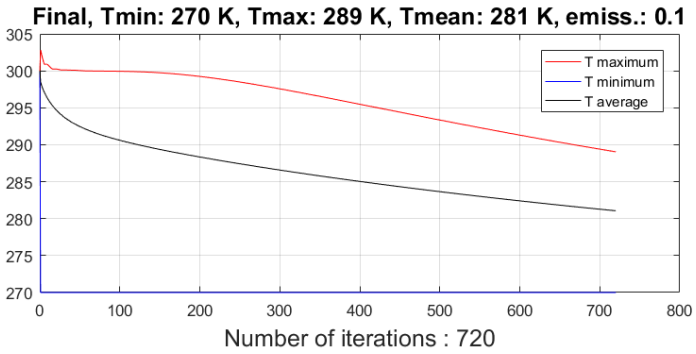


Figure 82: Temperature evolution, 256 elements, $\varepsilon = .1$, $\rho = 0.9$

6.4 Street section

6.4.1 Adiabatic 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). The evolution of three typical temperatures is given in *Figure 83*. At the end of the integration process of 720 hours, T_{min} = 280 K, T_{max} = 318 K and T_{mean} = 288 K.

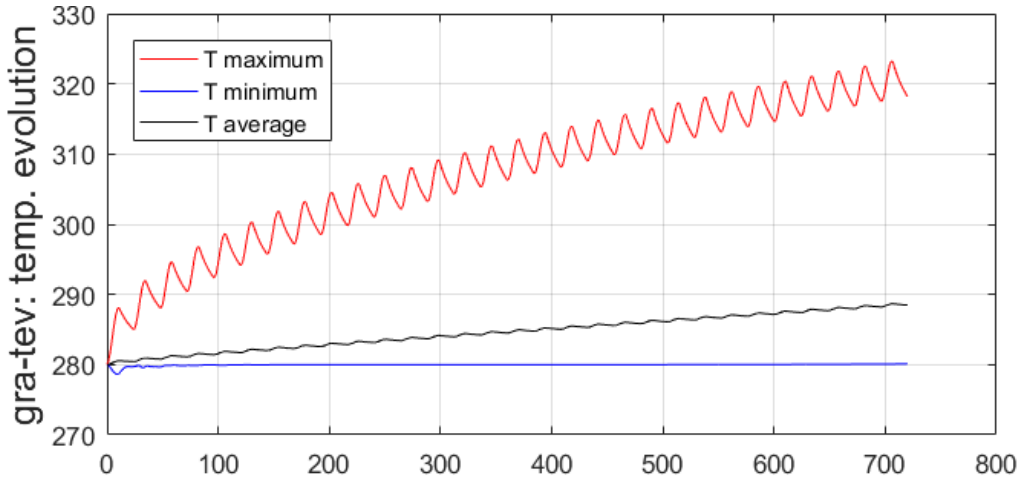
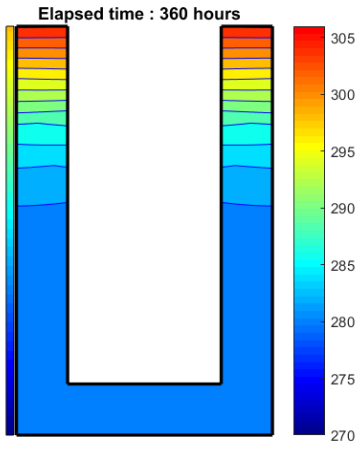


Figure 83: Street section – adiabatic walls - 225 DOF – 720 iterations

```
Fiammetta_20211111
Street section,  Gi: 15
L   4, Me, Di, Ne    : 3   0  11
L   5, Co nnr nvn    : 0   0   0
L   6, rc, ra, cs    : 0   0   0
L   7, CAD interf.   : 10
L   9, N. pa., vert.: 3   8
L  11, num. nod side: 7
L  13, num elem side: 8
L  23, Domain area   : 19 m2
L  29, Num. elements: 192
L  31, Num. of DOF   : 225
L  62, Domain perim.: 40 m
L  64, Thickness     : 0.1 m
L  65, Conduction k : 5 W/(mK)
L  66, DT isotherms : 2 K
L  72, Fixed DOF     : 0
N  27, N. heat flows: 18
L 101, Anis. index   : 0
t. 04, Initial temp.: 280 K
t. 20, nit  N. iter.: 720
t. 21, Time step     : 3600 s
t. 23, Analyzed per.: 720 h, 30 days
```



```
t. 21, Spec. capac. : 1000 J/(kg.K)
t. 22, Spec. mass   : 2500 kg.m-3
t. 40, sum(sum(C))  : 4.75 MJ/K
t. 41, area*th*ro*Cp: 4.75 MJ/K
t. 45, Imposed Heat : 100 W/m-2
t  67, iteration    : 360
t  67, iteration    : 720
t. 80, Tmean - Tini : 8.48 K
t. 81, DT* sumsum(C): 40.3 MJ
t. 83, Min obs. temp: 279 K
t. 84, Max obs. temp: 323.2 K
t. 90, Heat inp.(81): 41.3 MJ
L 231, End Me=3. CPU: 1.36 sec.
```

The color scales of the two drawings are the same thanks to enabling instructions 62 & 63 in Matlab© function *fem_til.m* (*Table 62*)

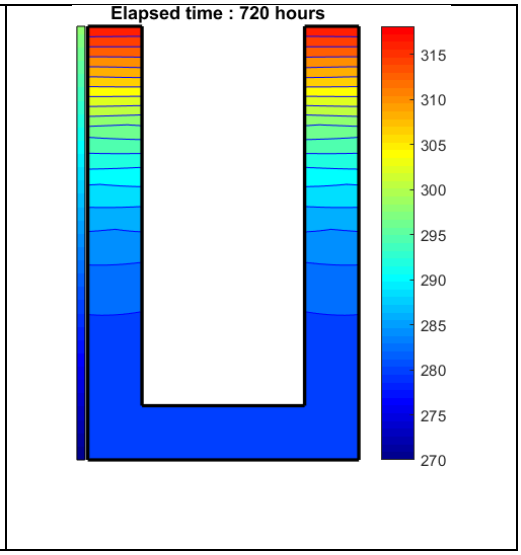


Figure 84: Adiabatic Street section: k = 5 $Wm^{-1}K^{-1}$, heat load density = 100 Wm^{-2}

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 oscillations of the maximum temperature is approximatively equal to 7 K (*Figure 85*).

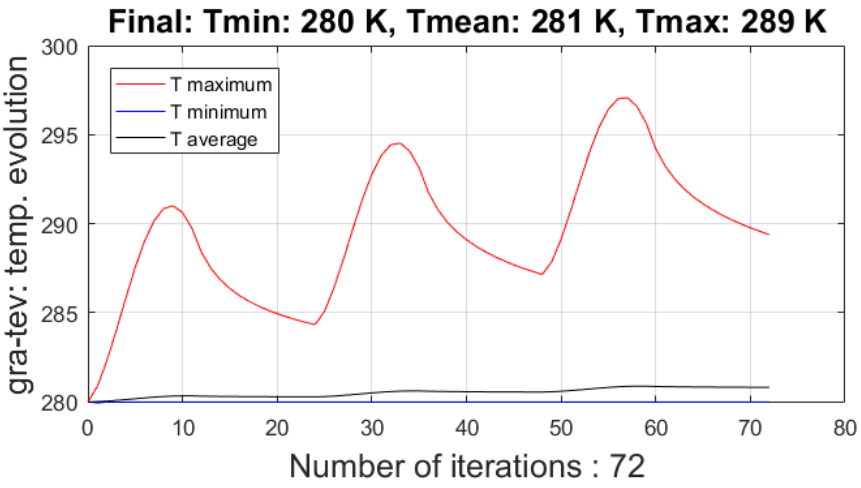


Figure 85: Temperature evolution, adiabatic walls, 3 days, 32 elements per side

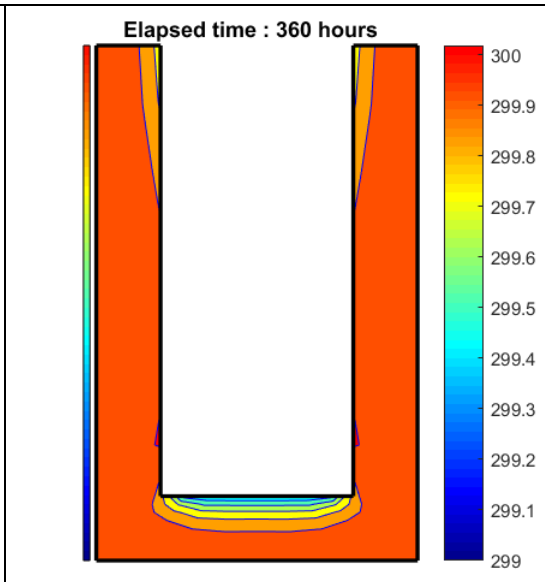
Adiabatic wall	Perfectly reflective wall ($\rho = 1$)
Fiammetta_20211111 Street section, Gi: 14 L 4, Me, Di, Ne : 5 0 11 L 5, Co nnr nvn : 0 0 0 L 6, rc, ra, cs : 0 0 0 L 7, CAD interf. : 10 L 9, N. pa., vert.: 3 8 L 11, num. nod side: 15 L 13, num elem si de: 16 L 23, Domain area : 19 m2 L 29, Num. elements: 768 L 31, Num. of DOF : 833 L 62, Domain perim.: 40 m L 64, Thickness : 0.1 m	Fiammetta_20211111 Street section, Gi: 14 L 4, Me, Di, Ne : 5 0 11 L 5, Co nnr nvn : 0 0 0 L 6, rc, ra, cs : 1 0 2 L 7, CAD interf. : 10 L 9, N. pa., vert.: 3 8 L 11, num. nod side: 15 L 13, num elem side: 16 L 23, Domain area : 19 m2 L 29, Num. elements: 768 L 31, Num. of DOF : 833 L 62, Domain perim.: 40 m L 64, Thickness : 0.1 m

L 65, Conduction k : 5 W/(mK) L 66, DT isotherms : 1 K L 72, Fixed DOF : 0 N 27, N. heat flows: 34 Lt 04, Initial temp.: 280 K Lt 07, Time step : 3600 sec Lt 09, Analyzed per.: 720 h, 30 days Lt 15, Spec. capac. : 1000 J/(kg.K) Lt 16, Spec. mass : 2500 kg.m-3 Lt 29, sum(sum(C)) : 4.75 MJ/K Lt 35, Ampl inp heat: 100 W/m-2 Lt 40, Loaded area : 0.2 m2 Lt 79, iteration : 360 Lt 79, iteration : 720 Lt108, Tmean - Tini : 3.18 K Lt109, Min obs. temp: 280 K Lt110, Max obs. temp: 298 K Lt111, Final temper.: 280 283 295 Lt113, Capac * Del T: 15.1 MJ Lt119, Injected heat: 16.4 MJ Lt120, Exact hd x 30: 16.5 MJ L 258, End Me=5. CPU: 9.82 sec.	L 65, Conduction k : 5 W/(mK) L 66, DT isotherms : 1 K L 68, St. Boltzmann: 5.6704e-08 W/(m2K4) L 72, Fixed DOF : 0 N 27, N. heat flows: 34 L 92, Numb. r.nodes: 49 L 101, Anis. index : 0 L 134, Rad. edg. len: 7 3 7 m L 135, Radiativ area: 1.7 m2 L 136, Refl. coeff. : 1 L 145, sum(sum(M)) : 8.7283 L 148, Sky temper. : 300 K L 152, Sum sky loads: 0 W/m2 Lt 04, Initial temp.: 280 K Lt 07, Time step : 3600 sec Lt 09, Analyzed per.: 720 h, 30 days Lt 15, Spec. capac. : 1000 J/(kg.K) Lt 16, Spec. mass : 2500 kg.m-3 Lt 29, sum(sum(C)) : 4.75 MJ/K Lt 35, Ampl inp heat: 100 W/m-2 Lt 40, Loaded area : 0.2 m2 Lt 48, Radiat. area : 1.7 m2 Lt 62, sum(Ms) : 0 W Lt 64, sum(Mn) : 0 W Lt 79, iteration : 360 Lt 79, iteration : 720 Lt108, Tmean - Tini : 3.18 K Lt109, Min obs. temp: 280 K Lt110, Max obs. temp: 298 K Lt111, Final temper.: 280 283 295 Lt113, Capac * Del T: 15.1 MJ Lt119, Injected heat: 16.4 MJ Lt120, Exact hd x 30: 16.5 MJ L 258, End Me=5. CPU: 10.4 sec.

Table 53: Street: comparison between adiabatic and perfectly reflective walls

In the example of *Table 53*, we observe that a perfectly reflective wall is identical to an adiabatic one. The last one needs less data and is easier to run because it does not use radiative exchanges parameters or methods.

6.4.2 Black body walls, $\rho = 0$

Fiammetta_20211111 Street section, Gi: 14 L 4, Me, Di, Ne : 5 0 11 L 5, Co nnr nvn : 0 0 0 L 6, rc, ra, cs : 1 0 2 L 7, CAD interf. : 10 L 9, N. pa., vert.: 3 8 L 11, num. nod side: 7 L 13, num elem side: 8 L 23, Domain area : 19 m2 L 29, Num. elements: 192 L 31, Num. of DOF : 225 L 62, Domain perim.: 40 m L 64, Thickness : 1 m L 65, Conduction k : 0.2 W/(mK) L 66, DT isotherms : 0.1 K L 68, St. Boltzmann: 5.6704e-08 W/(m2K4) L 72, Fixed DOF : 0 N 28, N. load. nod.: 18 L 92, Numb. r.nodes: 25 L 94, Rad. vertices: 2 4 6 8 L 101, Anis. index : 0 L 134, Rad. edg. len: 7 3 7 m L 135, Radiativ area: 17 m2 L 136, Refl. coeff. : 0	

```
L 145, sum(sum(M))   : 24
L 147, Sky temper.   : 300 K
L 154, Radiat. area : 17 m2
L 157, Sum sky loads: -6e+03 W/m2
Lt 04, Initial temp.: 300 K
Lt 07, Time step     : 3600 sec
Lt 09, Analyzed per.: 720 h, 30 days
Lt 15, Spec. capac. : 1000 J/(kg.K)
Lt 16, Spec. mass    : 2500 kg.m-3
Lt 29, sum(sum(C))   : 47.5 MJ/K
Lt 35, Ampl inp heat: 0 W/m-2
Lt 48, Radiat. area : 17 m2
Lt 62, sum(Ms)       : -6002 W
.rs 23, sum(aaa)     : 536.4935 W
.rs 24, sum(Mss)     : -6002.0056 W
Lt 64, sum(Mn)       : -5466 W
Lt 79, iteration     : 360
Lt 79, iteration     : 720
Lt108, Tmean - Tini : -0.121 K
Lt109, Min obs. temp: 299 K
Lt110, Max obs. temp: 300 K
Lt111, Final temper.: 299 300 300 K
Lt113, Capac * Del T: -5.75 MJ
Lt119, Injected heat: 0 MJ
L 272, End Me=5. CPU: 1.67 sec.
```

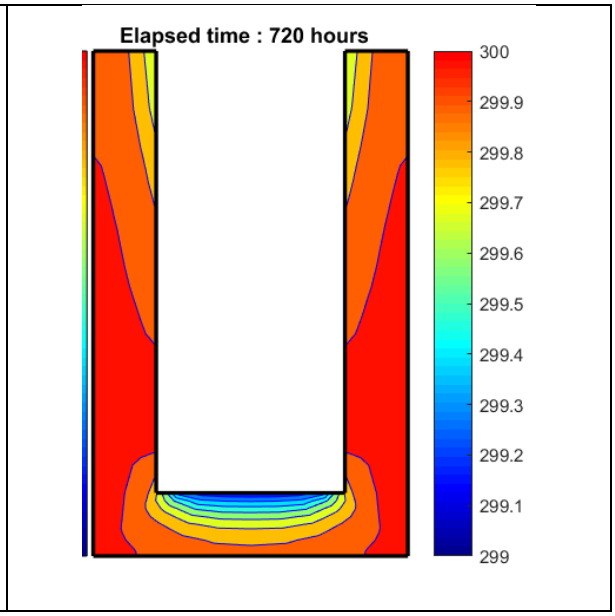


Table 54: Street: same temperature for sky and initial conditions, $\rho = 0$

For black bodies ($\rho = 0$), $[M] = [I]$ and equation (99) reduces to:

$$Q = \{[I]-[F]\}E(\tau)-\{[F_{sky}]\,E_{sky}+[F_{gr}]\,E_{gr}\} \tag{105}$$

We assume that *Ms* is a uni-column matrix containing the products of mid-side temperature to power 4 by the Stefan-Boltzmann constant, the emissivity of the edge and its area. Expressing also the sky and ground exitances as a function of the sky and/or ground temperatures it becomes:

$$Q = \{[I]-[F]\}Ms-\{[F_{sky}]\,\sigma T_{sky}^4+[F_{gr}]\,\sigma T_{gr}^4\} \tag{106}$$

If the temperature field is constant:

$$\tau_i^{mid-side} = T_{sky} = T_{gr} \tag{107}$$

```
M     = (I-re*Fs);if re == 0;M=I;end
Fsky  = 1-sum(Fs,2)';
(sum([Fs Fsky'],2))'=1 1 1 1 1 1
Ms    = SB*Lel(2)*th*(re*(I-Fs)*M^(-1)-I)*Fsky'*Tsky^4
Mpr   = (I-Fs)*M^(-1);
aa    = Mpr(1:si,1:si)*eye(si).*lon*(1-re)*SBt*cam'.^4;
if re == 0;Mpr=(I-Fs);end
if re == 0; Ms = (-SB*Lel(2)*th*Fsky'*Tsky^4)';end

if re == 0;aa= ((I-Fs)*lon'*SB*th*300.^4)';end
if re == 0;Ms= -SB*Lel(2)*th*((1-sum(Fs,2)')'*300^4)';end
sum(Ms)+sum(aa)
```

If we impose black body street walls and 280 *K* sky temperature, we observe in *Figure 87* that the heat is sucked mainly on the street walls. At the end of a cooling process of 72 hours, the temperatures are: T_{min72}=253 *K*, T_{max72}= 278 *K*, T_{mean72}= 267 *K*, (*bottom right*). The amplitudes of the oscillations due to the sinusoidal heat loads are decreasing with time.
From the pure geometric characteristics $[F_{sky}]$ and $[F_{gr}]$, we deduce the impact of the radiation emitted by the sky, which is proportional to its temperature. The basic instruction relates the

sky temperature to the emitted radiation. The emissivity of the sky is equal to 1. The fourth power of the sky temperature is multiplied by the Stefan-Boltzmann constant, by the sky view factor and by the areas of the elements.

esm =th lon'.* [F_{sky}] *SB*Tsky^4; (Watt)*

This result is defined on the boundary elements which are here boundary segments. It is transformed in nodal values with typical formulas like:

sn(i+1,1) =(esm(i,1) + esm(i+1,1))/ 2; (Watt)

This transformation satifies the relation:

sum(esm) = sum(es)

It corresponds to an imposed sky temperature given in L 148, Sky temperat.: 280 *K*. The graph of nodal heats is given in *Figure 86*. It is obtained with the Matlab© instruction:

```
figure('Position',[0 0 1500 700]);hold on;grid on;bar(sn);
title([' sum, max, min, mean sn: ',num2str([sum(sn) max(sn) min(sn) mean(sn)],3),'
W'], 'fontsize',20)
```

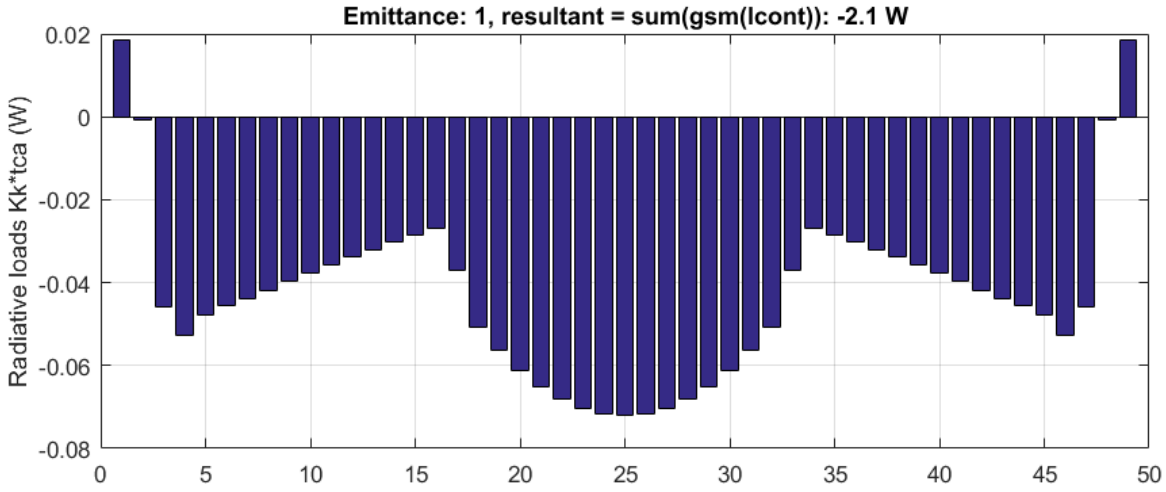


Figure 86: Nodal heat loads coming from sky

```
Fiammetta_20211111
Street section,  Gi: 14
L    4, Me, Di, Ne    : 5    0   11
L    5, Co nnr nvn    : 0   0   0
L    6, rc, ra, cs    : 1   0   2
L    7, CAD interf.   : 10
L    9, N. pa., vert.: 3   8
L   11, num. nod side: 15
L   13, num elem side: 16
L   23, Domain area   : 19 m2
L   29, Num. elements: 768
L   31, Num. of DOF   : 833
L   62, Domain perim.: 40 m
L   64, Thickness     : 0.1 m
L   65, Conduction k : 5 W/(mK)
L   66, DT isotherms : 0.2 K
L   68, St. Boltzmann: 5.6704e-08 W/(m2K4)
L   72, Fixed DOF     : 0
N   28, N. load. nod.: 34
L   92, Numb. r.nodes: 49
L   94, Rad. vertices: 2   4   6   8
L 101, Anis. index   : 0
L 134, Rad. edg. len: 7   3   7 m
L 135, Radiativ area: 1.7 m2
L 136, Refl. coeff. : 0
L 145, sum(sum(M))   : 48
L 147, Sky temper.   : 280 K
L 154, Radiat. area : 1.7 m2
L 157, Sum sky loads: -913 W/m2
Lt 04, Initial temp.: 280 K
Lt 07, Time step     : 3600 sec
Lt 09, Analyzed per.: 72 h, 3 days
Lt 15, Spec. capac. : 1000 J/(kg.K)
Lt 16, Spec. mass    : 2500 kg.m-3
Lt 29, sum(sum(C))   : 4.75 MJ/K
Lt 35, Ampl inp heat: 100 Wm-2
Lt 48, Radiat. area : 1.7 m2
Lt 62, sum(Ms)       : -912.6 W
Lt 64, sum(Mn)       : -871.5 W
Lt 79, iteration     : 36
Lt 79, iteration     : 72
Lt108, Tmean - Tini : 0.348 K
Lt109, Min obs. temp: 280 K
Lt110, Max obs. temp: 287 K
Lt111, Final temper.: 280 280 284 K
Lt113, Capac * Del T: 1.65 MJ
Lt119, Injected heat: 1.64 MJ
Lt120, Exact ih x 30: 1.65 MJ
L 272, End Me=5. CPU: 2.58 sec.
```

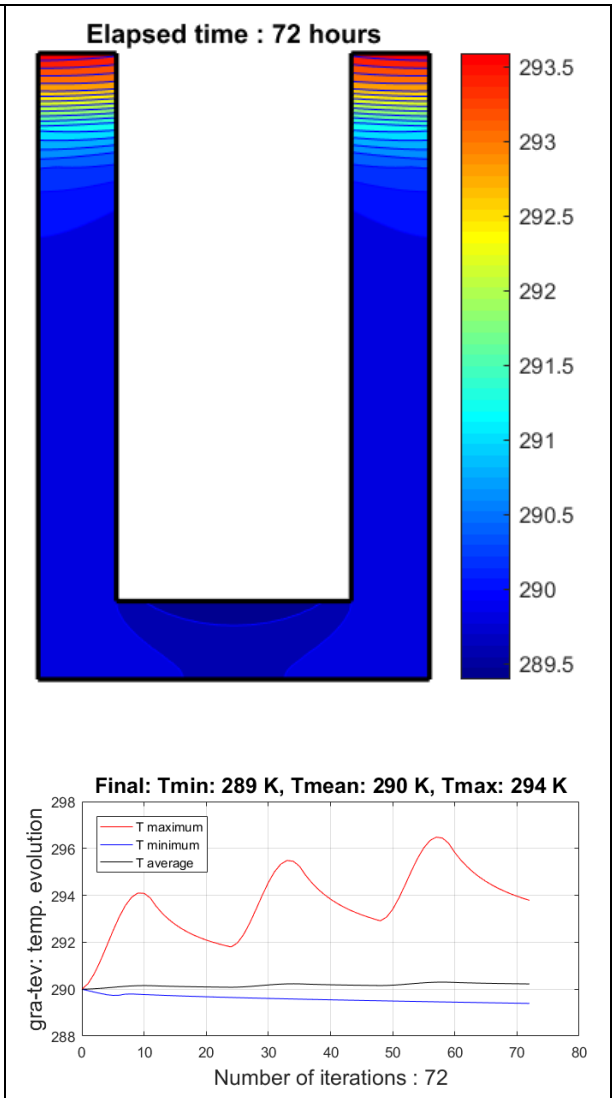


*Figure 87: Street section, **black body walls**, injected heat: 1.64 MJ, sky temperature = 280 K*

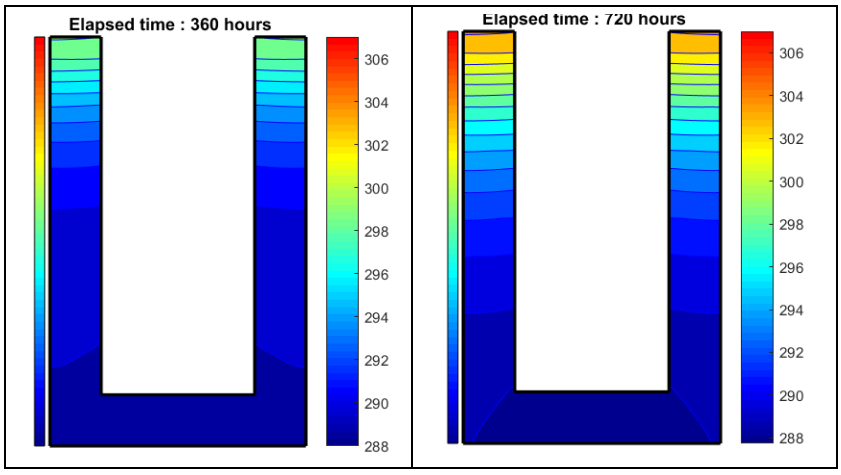


Figure 88: Same result as Figure 87, but 720 hours and same color bar, $\rho = 0$

6.5 Thermal bridge

6.5.1 Stationary heat flow - 2 convective virtual nodes

The following example relates to a domain involving convective boundaries on both sides of the main body *Figure 89*. As written under the left drawing of *Figure 89*, the convection coefficients may vary on the different patch sides of the domain (*Figure 59*).

```
                % Extreme right vertical side
50              lg =  [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
                % Convection on the right side
129             bd = [[car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];
130                   [car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)];
131                   [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
132                   [car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)];
133                   [car_cao(5,2) bor(pbo(5,2),5):bor(pbo(5,2),6) car_cao(5,3)]];
                % Convection on the left side
                bg = [[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)]];
                % Convection on both sides with 2 virtual nodes
147             bt = [[car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)];
148                   [car_cao(3,1) bor(pbo(3,1),5):bor(pbo(3,1),6) car_cao(3,2)];
149                   [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
150                   [car_cao(3,3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)];
151                   [car_cao(5,2) bor(pbo(5,2),5):bor(pbo(5,2),6) car_cao(5,3)];
152                   [car_cao(5,4) bor(pbo(5,4),5):bor(pbo(5,4),6) car_cao(5,1)];
153                   [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)];
154                   [car_cao(1,4) bor(pbo(1,4),5):bor(pbo(1,4),6) car_cao(1,1)];
155                   [car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
156                   [car_cao(4,4) bor(pbo(4,4),5):bor(pbo(4,4),6) car_cao(4,1)]];
```

Table 55: Explicit definitions of convective edges

```
Fiammetta_20211111
Thermal bridge, Gi: 18
L    4, Me, Di, Ne    : 1   2   0
L    5, Co nnr nvn    : 5   0   2
L    6, rc, ra, cs    : 0   0   0
L    7, CAD interf.   : 16
L    9, N. pa., vert.: 5   12
L   11, num. nod side: 15
L   13, num elem side: 16
L   23, Domain area   : 66 m2
L   29, Num. elements: 1280
L   31, Num. of DOF   : 1379
L   62, Domain perim.: 56 m
L   64, Thickness     : 0.1 m
L   65, Conduction k : 10 W/(mK)
L   66, DT isotherms : 1 K
LD 15, Fixed nodes   : 1378   1379
LD 16, Fix. temper. : 300   280 K
LD 91, N. fix. nodes: 2
L   72, Fixed DOF     : 1378   1379
L   79, Convection h : 20 W/(m2K)
c. 03, dim. K Co cs : 1379      5 0
c. 04, N.virt c.nod.: 2
L 101, Anis. index   : 0
L 188, Total dissip.: 80.6 WK
L 191, Dis. in solid: 71.1 WK
L 192, DT   in solid: 20 K
L 201, Fixed DOF     : 1378   1379
L 202, Imposed temp.: 300     280 K
L 203, React. flows : 8.06 -8.06 W
L 205, End Me=1, CPU: 0.8 sec.
```

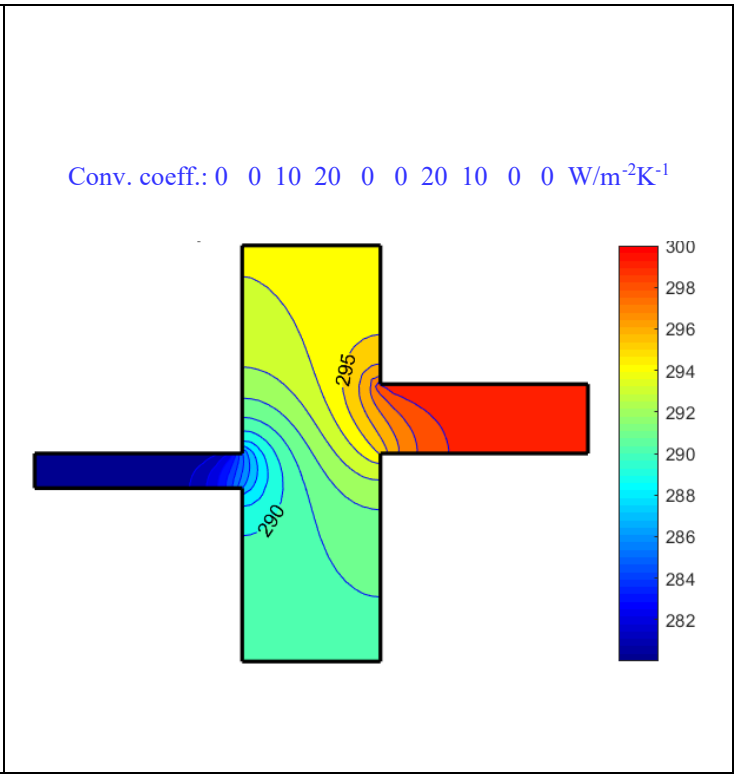


Figure 89: Thermal bridge with 2 convective virtual nodes

```
Fiammetta_20211111
Thermal bridge, Gi: 18
L   4, Me, Di, Ne    : 1   2   0
L   5, Co nnr nvn    : 5   0   2
L   6, rc, ra, cs    : 0   0   0
L   7, CAD interf.   : 16
L   9, N. pa., vert.: 5   12
L  11, num. nod side: 63
L  13, num elem side: 64
L  23, Domain area   : 66 m2
L  29, Num. elements: 20480
L  31, Num. of DOF   : 20867
L  62, Domain perim.: 56 m
L  64, Thickness     : 0.1 m
L  65, Conduction k : 10 W/(mK)
L  66, DT isotherms : 1 K
LD 15, Fixed nodes   : 20866  20867
LD 16, Fix. temper.  : 300   280 K
LD 91, N. fix. nodes: 2
L  72, Fixed DOF     : 20866  20867
L  79, Convection h : 20 W/(m2K)
c. 03, dim. K Co cs : 20867 5   0
c. 04, N.virt c.nod.: 2
L 101, Anis. index   : 0
L 188, Total dissip.: 82.6 WK
L 191, Dis. in solid: 77.2 WK
L 192, DT   in solid: 20 K
L 201, Fixed DOF     : 20866  20867
L 202, Imposed temp.: 300     280 K
L 203, React. flows : 8.26  -8.26 W
L 205, End Me=1, CPU: 123 sec.
```

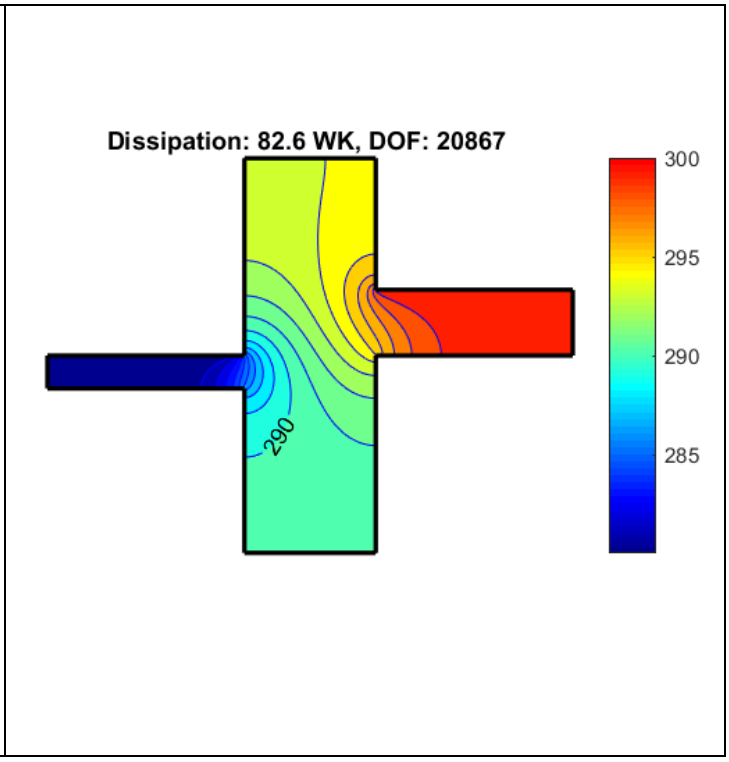


Figure 90: Thermal bridge, 2 convective virtual nodes, more than 20000 DOF

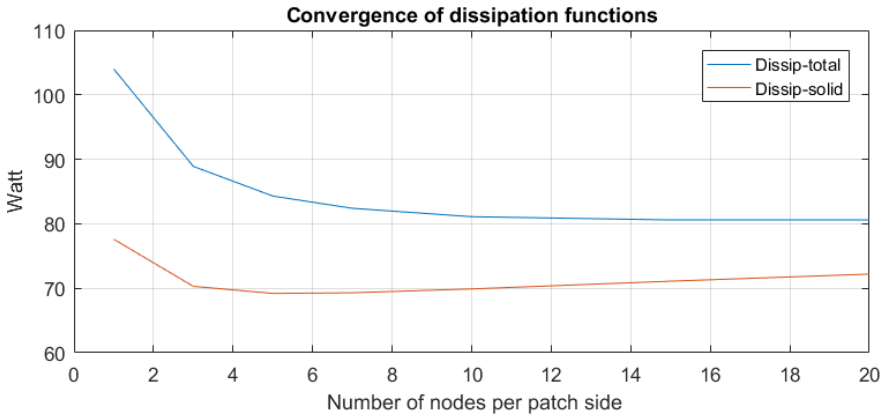


Figure 91: Two convection virtual nodes - convergence curves of dissipation functions

The convergence curves of two dissipation functions (*Figure 91*) are built with the sequence of Matlab© instructions of *Table 56*.

Matlab© instructions enabling the representation of convergence curves	
1	`mesh   = [1    3    5    7    10   15   20];`
2	`dissol = [104  88.9 84.3 82.4 81.1 80.6 80.6];`
3	`distot = [77.6 70.3 69.2 69.3 69.9 71.1 72.2];`
4	`figure('Position',[100 100 900 400]);`
5	`plot  (mesh,dissol);hold on;grid on;plot(mesh,distot);hold on`
6	`legend('Dissip-total','Dissip-solid');grid on`
7	`ylabel('Watt');xlabel(' Number of nodes per patch side');hold on`
8	`title ('Convergence of dissipation functions')`

Table 56: Instructions to draw convergence curves of dissipation functions

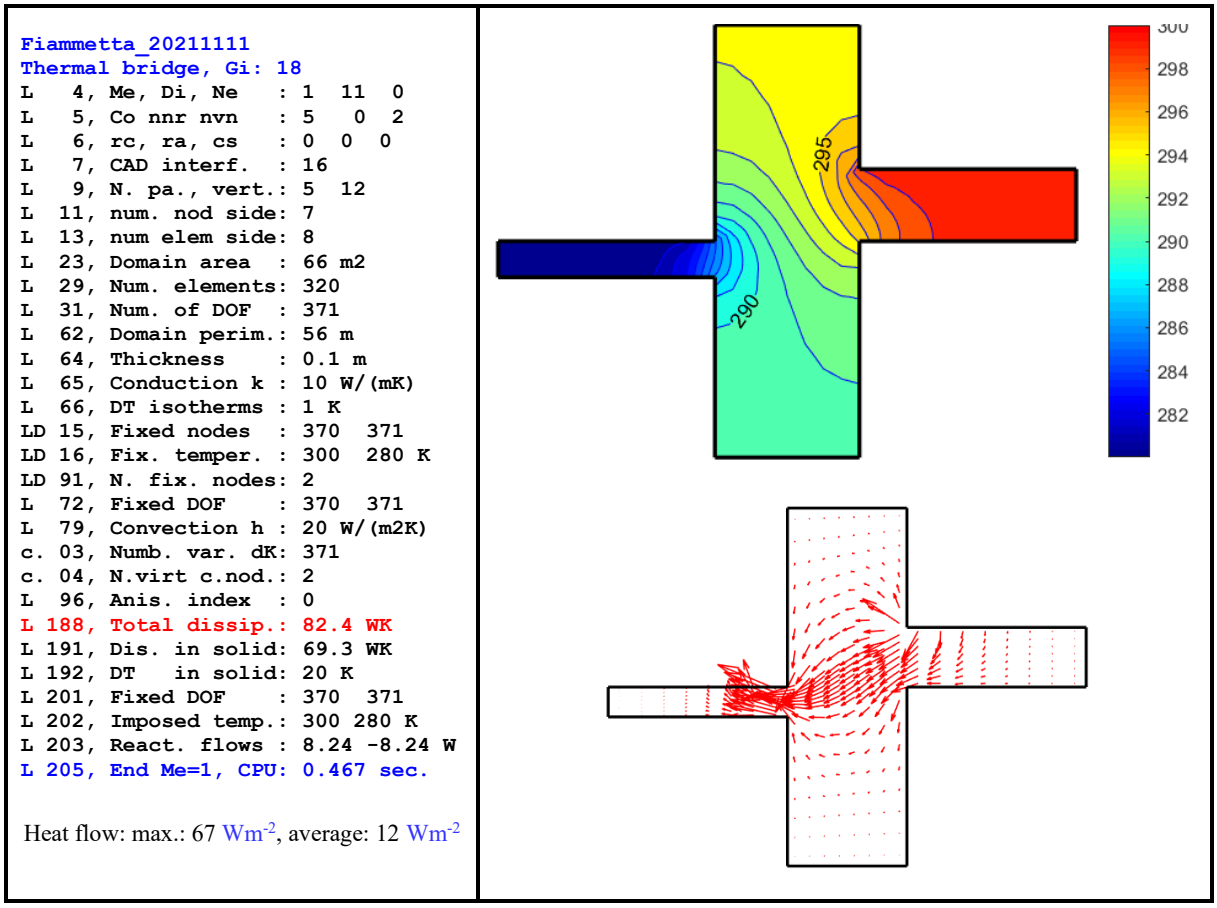


Figure 92: Thermal bridge with 2 convective virtual nodes

With 32 elements on each patch side, the reactive flows on the virtual convective nodes remain close to the results of the relatively coarse mesh, they are equal to ± 8.24 *W*. The dissipation in the solid was equal to 69.3 *WK*. With 5317 *DOF*, the total dissipation is 81.2 WK, in the solid it is equal to 74 *WK*. With 64 elements per side, we reach 20869 *DOF*, the total dissipation = 82.6 *WK*, the dissipation in the solid = 77.2 *WK*, the reactive flows = ± 8.26 *W*, The *CPU* = 135 seconds.

In the test of *Figure 92*, we have 8 elements per patch side. The instruction: "`disp([B(dK+1) gh(dK+1); B(dK+2) gh(dK+2)])`" where *B* is the vector of unknowns and *gh*, the second member, is giving the result:

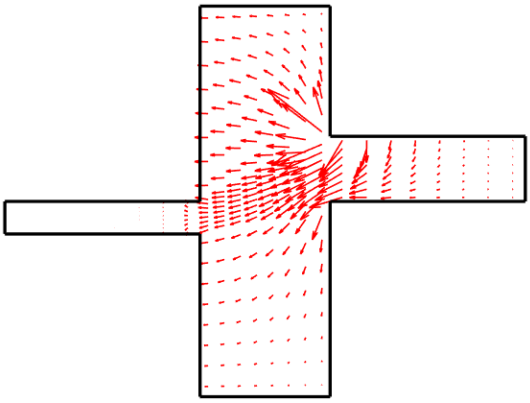
```
-8.2359   300.0000
 8.2359   280.0000
```

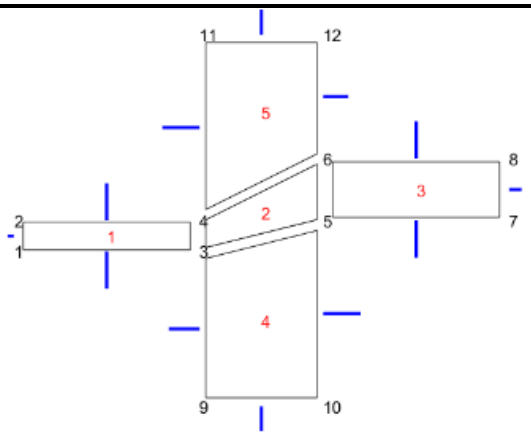
The instruction using the dot or scalar product: `-dot (B, gh)/2`, provides the total dissipation = 82.359 *WK*. Due to the difference of 20 *K* between both convective virtual nodes, the dissipation is equal to 10 *K* multiplied by the heat flows reactions equal to 8.24 *W*.

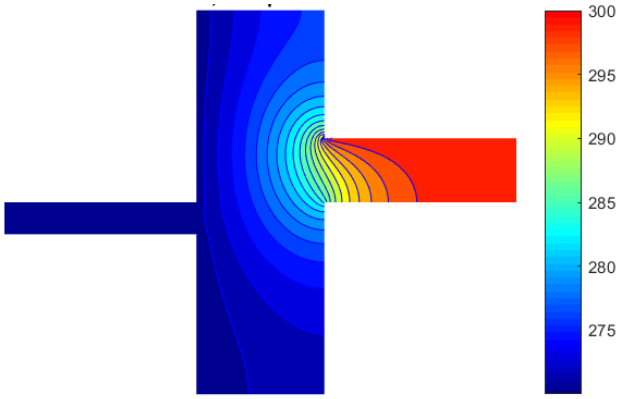
6.5.2 Stationary heat flow - 1 convective & 1 radiative virtual node

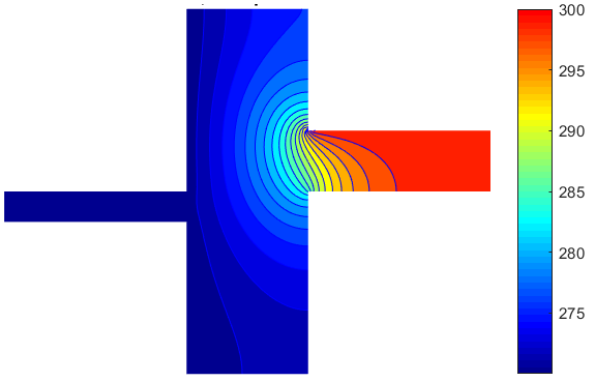
In the example of *Figure 93*, the right side of the domain is still submitted to convection, but the left one is submitted to radiation. In this example, we use an adaptation of the convective elements to generate radiative ones. Radiation is acting from the radiative boundary to a reference virtual node at imposed temperature of 270 *K*.

```
Fiammetta_20211111
Thermal bridge, Gi: 16
L   4, Me, Di, Ne    : 2  2  0
L   5, Co nnr nvn    : 5  1  1
L   6, rc, ra, cs    : 1  0  3
L   7, CAD interf.   : 16
L   9, N. pa., vert.: 5  12
L  11, num. nod side: 31
L  13, num elem side: 32
L  23, Domain area   : 66 m2
L  29, Num. elements: 5120
L  31, Num. of DOF   : 5315
L  62, Domain perim.: 56 m
L  64, Thickness     : 0.1 m
L  65, Conduction k : 10 W/(mK)
L  66, DT isotherms : 1 K
L  68, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD 15, Fixed nodes  : 5314  5315
LD 16, Fix. temper. : 270  300 K
LD 91, N. fix. nodes: 2
L  69, Fixed DOF    : 5314  5315
L  76, Convection h : 20 W/(m2K)
c. 03, Numb. var. dK: 5315
c. 04, N.virt c.nod.: 1
L  89, Numb. r.nodes: 161
L 100, Anis. index  : 0
L 133, Rad. edg. len: 6  6  1  6  5 m
L 134, Radiativ area: 2.4 m2
L 135, Refl. coeff. : 0.5
Ls  2, list f. nodes: 5314  5315
Ls  3, Fix. temper. : 270  300 K
Ls  4, N. fixed nod.: 2
Ls  5, N. rad elem. : 160
LS 17, Iteration N. : 1 / 2
Ls 32, Dissipation  : 794 WK
Ls 33, React. flows : 26.5     -26.5 W
Ls 43, max - min tca: 30.1 K
LS 17, Iteration N. : 2 / 2
Ls 32, Dissipation  : 801 WK
Ls 33, React. flows : 26.7     -26.7 W
Ls 43, max - min tca: 30.1 K
L 215, End Me=2, CPU: 5.67 sec.
```









Heat flow max: 180 Wm^{-2} mean: 27 Wm^{-2}

*Figure 93: Thermal bridge, mixed b. c. 1 radiative v. node (**ρ = .5**), 1 convective v. node*

The product of the reaction flow of 26.7 *W* (*Ls 33, Figure 93*) by the difference of temperature 30 *K* (*Ls 43, Figure 93*) is equal to the dissipation: 801 *WK* (*Ls 32, Figure 93*).
The use of conductive-radiative elements (*Table 75*) equivalent to conductive-convective elements (*Table 6*) is acceptable but not very effective because the heat exchange is concentrated on a single virtual node. To take into account the interactions between radiative elements, it is necessary to associate them. This operation is related to the view factors computation.

6.5.3 Transient heat exchanges

a. Two convective virtual nodes

```
Fiammetta_20211111
Thermal bridge, Gi: 20
L    4, Me, Di, Ne    : 3   7   0
L    5, Co nnr nvn    : 5   0   2
L    6, rc, ra, cs    : 0   0   0
L    7, CAD interf.   : 16
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   64, Thickness     : 0.1 m
L   65, Conduction k : 10 W/(mK)
L   66, DT isotherms : 1 K
LD 91, N. fix. nodes: 2
L   72, Fixed DOF     : 370   371
L   79, Convection h : 20 W/(m2K)
c. 03, dim. K Co cs : 371     5     0
c. 04, N.virt c.nod.: 2
L 100, Anis. index   : 0
t. 04, Initial temp.: 280 K
t. 20, nit  N. iter.: 720
t. 21, Time step     : 3600 s
t. 23, Analyzed per.: 720 h, 30 days
t. 21, Spec. capac. : 1000 J/(kg.K)
t. 22, Spec. mass    : 2500 kg.m-3
t. 40, sum(sum(C))   : 16.5 MJ/K
t. 41, area*th*ro*Cp: 16.5 MJ/K
t. 45, Imposed Heat : 0 W/m-2
t   67, iteration     : 360
t   67, iteration     : 720
t. 80, Tmean - Tini : 6.57 K
t. 81, DT* sumsum(C): 108 MJ
t. 83, Min obs. temp: 278 K
t. 84, Max obs. temp: 300.1 K
t. 93, Tot. reaction: 105 MJ
L 231, End Me=3. CPU: 6.82 sec.


c. 82, Conv. coeff. :
0 0 10 20 0 0 20 10 0 0 W/(m2K)
```

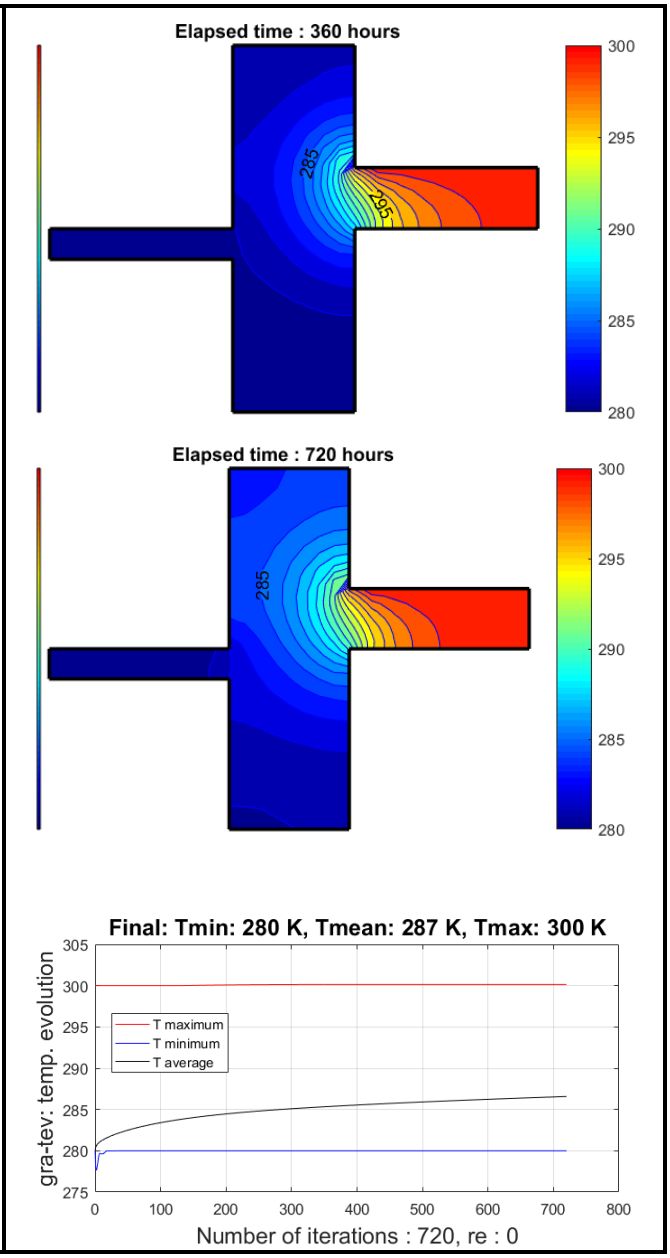
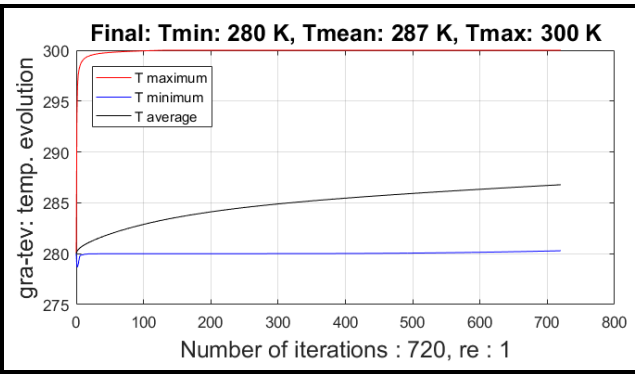


Figure 94: Thermal bridge, transient analysis, 2 convective virtual nodes

b. One convective and one radiative virtual node

```
Fiammetta_20211111
Thermal bridge, Gi: 21
L    4, Me, Di, Ne    : 4   2   0
L    5, Co nnr nvn    : 5   1   1
L    6, rc, ra, cs    : 1   0   3
L    7, CAD interf.   : 16
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   64, Thickness     : 0.1 m
L   65, Conduction k : 10 W/(mK)
```



```
L  66, DT isotherms : 1 K
L  68, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD 15, Fixed nodes  : 370  371
LD 16, Fix. temper. : 280  300 K
LD 91, N. fix. nodes: 2
L  72, Fixed DOF    : 370  371
L  79, Convection h : 20 W/(m2K)
c. 03, dim. K Co cs : 371
c. 04, N.virt c.nod.: 1
L  89, Numb. r.nodes: 41
L  96, Anis. index  : 0
L 129, Sky temper.  : 280 K
L 133, Rad. edg. len: 6  6  1  6  5 m
L 134, Radiativ area: 2.4 m2
L 132, Refl. coeff. : 1
L 231, Num. fix. DOF: 370  371
L 232, T. fix. nod. : 280  300 K
t. 04, Initial temp.: 280 K
t. 05, Fixed DOF    : 370  371
t. 08, Time step    : 3600 s
t. 10, Analyzed per.: 720 h, 30 days
t. 16, Spec. capac. : 1000 J/(kg.K)
t. 17, Spec. mass   : 2500 kg.m-3
t. 27, sum(sum(C))  : 16.5 MJ/K
t. 28, area*th*ro*Cp: 16.5 MJ/K
t. 31, Impos. Heat  : 0 W/m-2
t. 38, mcr nv       : 40   1
t. 43, N. rad. edges: 5
t. 44, rad seg leng.: 6  6  1  6  5 m
t. 46, N. rad. elem.: 40
t.151, Tmean - Tini : 5.63 K
t.152, Stored heat  : 92.8 MJ
t.154, Min it+1 temp: 280 K
t.155, Max it+1 temp: 300 K
t.164, Ejected heat : 0.202 MJ
L 239, Diss in solid: 172 WK
L 240, Total dissip.: 186 WK
L 241, End Me=4. CPU: 5.89 sec.
```

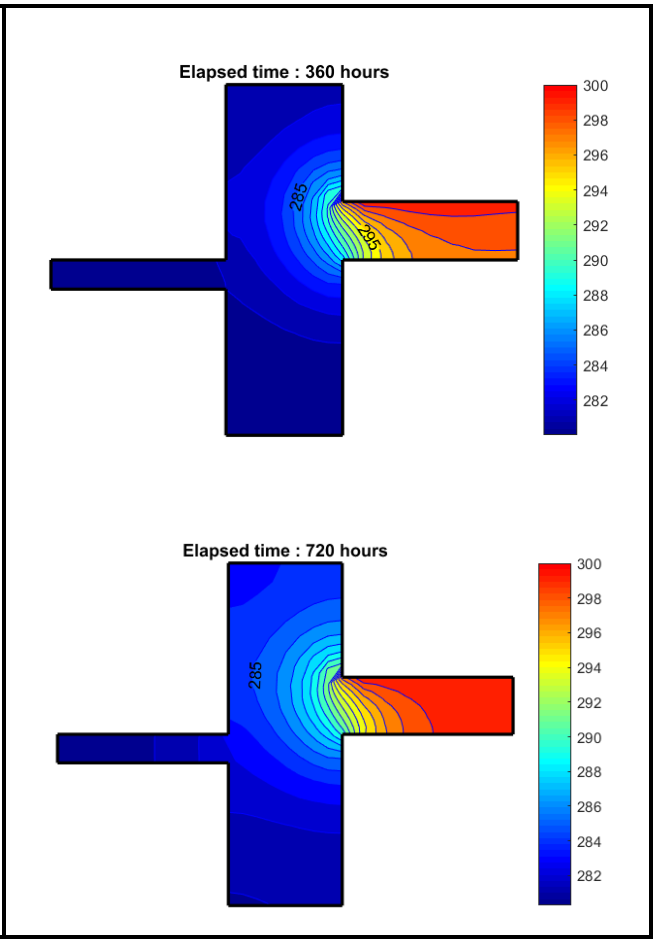


Figure 95: Thermal bridge, transient analysis, 1 convective & 1 radiative v. node, $\rho = 1$

```
Fiammetta_20211111
Thermal bridge, Gi: 21
L   4, Me, Di, Ne   : 4  2  0
L   5, Co nnr nvn   : 5  1  1
L   6, rc, ra, cs   : 1  0  3
L   7, CAD interf.  : 16
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  64, Thickness    : 0.1 m
L  65, Conduction k : 10 W/(mK)
L  66, DT isotherms : 1 K
L  68, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD 15, Fixed nodes  : 370  371
LD 16, Fix. temper. : 280  300 K
LD 99, N. fix. nodes: 2
L  72, Fixed DOF    : 370  371
L  79, Convection h : 20 W/(m2K)
c. 03, Numb. var. dK: 371
c. 04, N.virt c.nod.: 1
L  92, Numb. r.nodes: 41
L 100, Anis. index  : 0
L 129, Sky temper.  : 280 K
L 134, Rad. edg. len: 6  6  1  6  5 m
L 135, Radiativ area: 2.4 m2
L 136, Refl. coeff. : 1
L 159, Radiat. area : 2.4 m2



L 231, Num. fix. DOF: 370  371
```

```
Fiammetta_20211111
Thermal bridge, Gi: 22
L   4, Me, Di, Ne   : 4  12   0
L   5, Co nnr nvn   : 5  0  1
L   6, rc, ra, cs   : 1  1  3
L   7, CAD interf.  : 16
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  26, Num. elements: 320
L  28, Num. of DOF  : 370
L  59, Domain perim.: 56 m
L  61, Thickness    : 0.1 m
L  62, Conduction k : 10 W/(mK)
L  63, DT isotherms : 1 K
L  65, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD 15, Fixed nodes  : 370
LD 16, Fix. temper. : 300 K
LD 99, N. fix. nodes: 1
L  69, Fixed DOF    : 370
L  76, Convection h : 20 W/(m2K)
c. 03, Numb. var. dK: 370
c. 04, N.virt c.nod.: 1
L  89, Numb. r.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 132, Refl. coeff. : 1
L 158, numb. col. F : 42
L 160, determinant M: 0.69345
L 164, Radiat. area : 2.4 m2
L 166, Sum 2d member: 7.01e-15 W
L 231, Num. fix. DOF: 370
```

```
L 232, T. fix. nod. : 280  300 K
t. 04, Initial temp.: 280 K
t. 05, Fixed DOF    : 370  371
t. 08, Time step    : 3600 s
t. 10, Analyzed per.: 720 h, 30 days
t. 16, Spec. capac. : 1000 J/(kg.K)
t. 17, Spec. mass   : 2500 kg.m-3
t. 27, sum(sum(C))  : 16.5 MJ/K
t. 28, area*th*ro*Cp: 16.5 MJ/K
t. 31, Impos. Heat  : 0 W/m-2
t. 38, mcr nv       : 40  1
t. 44, N. rad. edges: 5
t. 45, rad seg leng.: 6  6  1  6  5
t. 46, N. rad. elem.: 40


t.151, Tmean - Tini : 6.78 K
t.152, Stored heat  : 112 MJ
t.154, Min it+1 temp: 280.3 K
t.155, Max it+1 temp: 300 K
t.164, Ejected heat : 0.202 MJ
L 239, Diss in solid: 129 WK
L 240, Total dissip.: 140 WK
L 241, End Me=4. CPU: 3.74 sec.
```

```
L 232, T. fix. nod. : 300 K
t. 04, Initial temp.: 280 K
t. 05, Fixed DOF    : 370
t. 08, Time step    : 3600 s
t. 10, Analyzed per.: 720 h, 30 days
t. 16, Spec. capac. : 1000 J/(kg.K)
t. 17, Spec. mass   : 2500 kg.m-3
t. 27, sum(sum(C))  : 16.5 MJ/K
t. 28, area*th*ro*Cp: 16.5 MJ/K
t. 31, Impos. Heat  : 0 W/m-2
t. 38, mcr nv       : 40  42
t. 44, N. rad. edges: 5
t. 45, rad seg leng.: 6  6  1  6  5 m
t. 46, N. rad. elem.: 40
t. 66, sum(gr)      : 326 W
t. 80, sum(Mn)      : 7.01e-15 W
t.151, Tmean - Tini : 6.78 K
t.152, Stored heat  : 112 MJ
t.154, Min it+1 temp: 280.3 K
t.155, Max it+1 temp: 300 K
t.164, Ejected heat : 0.202 MJ
L 239, Diss in solid: 129 WK
L 240, Total dissip.: 140 WK
L 241, End Me=4. CPU: 4.78 sec.
```

Figure 96: Thermal bridge comparison, radiative virtual node – radiative edge, $\rho = 1$

For radiative exchange handled with a virtual radiative node, the result is independent of the reflection coefficient ρ more or less in the range $0 \le \rho \le .9$.

```
Fiammetta_20211111
Thermal bridge, Gi: 21
L   4, Me, Di, Ne   : 4  2  0
L   5, Co nnr nvn   : 5  1  1
L   6, rc, ra, cs   : 1  0  3
L   7, CAD interf.  : 16
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  64, Thickness    : 0.1 m
L  65, Conduction k : 10 W/(mK)
L  66, DT isotherms : 1 K
L  68, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD 15, Fixed nodes  : 370  371
LD 16, Fix. temper. : 280  300 K
LD 99, N. fix. nodes: 2
L  72, Fixed DOF    : 370  371
L  79, Convection h : 20 W/(m2K)
c. 03, Numb. var. dK: 371
c. 04, N.virt c.nod.: 1
L  92, Numb. r.nodes: 41
L 100, Anis. index  : 0
L 129, Sky temper.  : 280 K
L 134, Rad. edg. len: 6  6  1  6  5 m
L 135, Radiativ area: 2.4 m2
L 132, Refl. coeff. : .5
L 159, Radiat. area : 2.4 m2



L 231, Num. fix. DOF: 370  371
L 232, T. fix. nod. : 280  300 K
t. 04, Initial temp.: 280 K
t. 05, Fixed DOF    : 370  371
t. 08, Time step    : 3600 s
t. 10, Analyzed per.: 720 h, 30 days
t. 16, Spec. capac. : 1000 J/(kg.K)
t. 17, Spec. mass   : 2500 kg.m-3
t. 27, sum(sum(C))  : 16.5 MJ/K
t. 28, area*th*ro*Cp: 16.5 MJ/K
t. 31, Impos. Heat  : 0 W/m-2
```

```
Fiammetta_20211111
Thermal bridge, Gi: 22
L   4, Me, Di, Ne   : 4  12   0
L   5, Co nnr nvn   : 5  0  1
L   6, rc, ra, cs   : 1  1  3
L   7, CAD interf.  : 16
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  26, Num. elements: 320
L  28, Num. of DOF  : 370
L  59, Domain perim.: 56 m
L  61, Thickness    : 0.1 m
L  62, Conduction k : 10 W/(mK)
L  63, DT isotherms : 1 K
L  65, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD 15, Fixed nodes  : 370
LD 16, Fix. temper. : 300 K
LD 99, N. fix. nodes: 1
L  69, Fixed DOF    : 370
L  76, Convection h : 20 W/(m2K)
c. 03, Numb. var. dK: 370
c. 04, N.virt c.nod.: 1
L  89, Numb. r.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 132, Refl. coeff. : .5
L 159, Radiat. area : 2.4 m2
L 161, numb. col. F : 42
L 163, determinant M: 0.9161
L 167, Sum 2d member: -348 W
L 233, Num. fix. DOF: 370
L 234, T. fix. nod. : 300 K
t. 04, Initial temp.: 280 K
t. 05, Fixed DOF    : 370
t. 08, Time step    : 3600 s
t. 10, Analyzed per.: 720 h, 30 days
t. 16, Spec. capac. : 1000 J/(kg.K)
t. 17, Spec. mass   : 2500 kg.m-3
t. 27, sum(sum(C))  : 16.5 MJ/K
t. 28, area*th*ro*Cp: 16.5 MJ/K
t. 31, Impos. Heat  : 0 W/m-2
```

```
t. 38, mcr nv       : 40  1
t. 44, N. rad. edges: 5
t. 45, rad seg leng.: 6  6  1  6  5
t. 46, N. rad. elem.: 40


t.151, Tmean - Tini : 5.63 K
t.152, Stored heat  : 92.9 MJ
t.154, Min it+1 temp: 280 K
t.155, Max it+1 temp: 300 K
t.164, Ejected heat : 0.202 MJ
L 239, Diss in solid: 172 WK
L 240, Total dissip.: 185 WK
L 241, End Me=4. CPU: 3.84 sec.
```

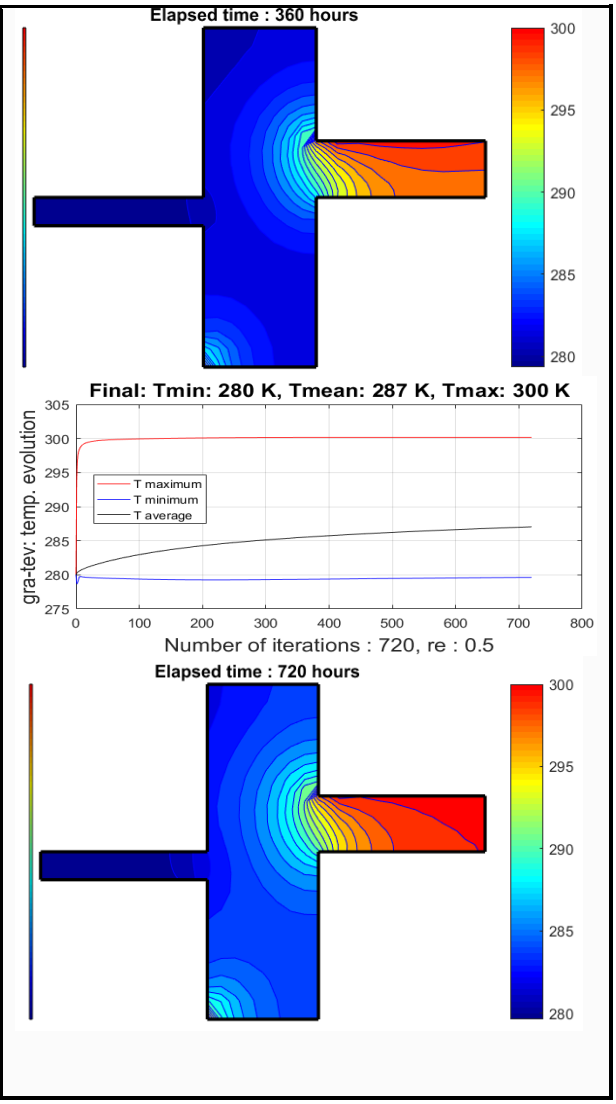
```
t. 38, mcr nv       : 40  42
t. 44, N. rad. edges: 5
t. 45, rad seg leng.: 6  6  1  6  5 m
t. 46, N. rad. elem.: 40
t. 66, sum(gr)      : 131 W
t. 80, sum(Mn)      : -348 W
t.142, Tmean - Tini : 7.04 K
t.143, Stored heat  : 116 MJ
t.145, Min it+1 temp: 279.6 K
t.146, Max it+1 temp: 300 K
t.152, Ejected heat : 0.202 MJ
L 240, Diss in solid: 162 WK
L 241, Total dissip.: 174 WK
L 242, End Me=4. CPU: 6.12 sec.
```

Figure 97: Thermal bridge comparison, radiative virtual node – radiative edge, $\rho = .5$

c. One convective node and one radiative edge

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 in this example. The variable *ra* select 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 warranty of a temperature everywhere greater or equal to 280 *K*, is obtained only if we use a reflection coefficient greater or equal to 0,9999. Even if the results seem correct, this condition is unbelievable.

```
Fiammetta_20211111
Thermal bridge, Gi: 22
L   4, Me, Di, Ne   : 4  12   0
L   5, Co nnr nvn   : 5  0  1
L   6, rc, ra, cs   : 1  1  3
L   7, CAD interf.  : 16
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  26, Num. elements: 320
L  28, Num. of DOF  : 370
L  59, Domain perim.: 56 m
L  61, Thickness    : 0.1 m
L  62, Conduction k : 10 W/(mK)
L  63, DT isotherms : 1 K
L  65, St. Boltzmann: 5.6704e-08 W/(m2K4)
LD 15, Fixed nodes  : 370
LD 16, Fix. temper. : 300 K
LD 99, N. fix. nodes: 1
L  69, Fixed DOF    : 370
L  76, Convection h : 20 W/(m2K)
c. 03, Numb. var. dK: 370
c. 04, N.virt c.nod.: 1
L  89, Numb. r.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 132, Refl. coeff. : 0.5
L 159, Radiat. area : 2.4 m2
L 161, numb. col. F : 42
L 163, determinant M: 0.9161
L 167, Sum 2d member: -348 W
L 233, Num. fix. DOF: 370
L 234, T. fix. nod.  : 300 K
t. 04, Initial temp.: 280 K
t. 05, Fixed DOF    : 370
t. 08, Time step    : 3600 s
t. 10, Analyzed per.: 720 h, 30 days
t. 16, Spec. capac. : 1000 J/(kg.K)
t. 17, Spec. mass   : 2500 kg.m-3
t. 28, sum(sum(C))  : 16.5 MJ/K
t. 29, area*th*ro*Cp: 16.5 MJ/K
t. 31, Impos. Heat  : 0 W/m-2
```



```
t. 38, mcr nv       : 40  42
t. 45, N. rad. edges: 5
t. 46, rad seg leng.: 6  6  1  6  5 m
t. 47, N. rad. elem.: 40
t. 63, sum(gr)      : 131 W
t. 75, sum(Mn)      : -348 W
t.142, Tmean - Tini : 7.04 K
t.143, Stored heat  : 116 MJ
t.145, Min it+1 temp: 279.6 K
t.146, Max it+1 temp: 300 K
t.152, Ejected heat : 0.202 MJ
L 240, Diss in solid: 162 WK
L 241, Total dissip.: 174 WK
L 242, End Me=4. CPU: 6.12 sec.
```

Figure 98: Thermal bridge, transient analysis, 1 convective node & 1 radiative side

Matlab© function *fem_Kra.m*

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

Table 57: Matlab© function fem_Kra.m

7. Conclusion

In the transient problems, the non-linearity due to radiation does not seem to give any particular problem. 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 finished 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 visualization of the results at different time steps by isotherms and by heat flow vectors for relatively coarse meshes is very effective and allows understanding the physics of heat exchanges by conduction, convection and radiation.

8. Miscellaneous

8.1 Main procedure: *Fiammetta.m*

Simplified scheme of the solution method used in the procedure *Fiammetta_20211111.m* (*Table 58*).

Line	*14→ 23*:	Computation of the domain area		
Line	*27→ 32*:	Geometric input data		
Line	*34→ 40*:	Nodes and mesh generation	*cad_mes.m*	(*Table 15*)
Line	*41→ 61*:	Computing the outwards normal	*gra_mnl.m*	(*Table 66*)
Line	*65*:	End of geometric data processing		

Line	*69→ 72*:	Dirichlet boundary conditions		*cad_Dir.m*	(*Table 17*)
Line	*73→ 74*:	Neumann boundary conditions		*cad_Neu.m*	(*Table 18*)
Line	*77→ 82*:	Topology of the convection elem.		*cad_con.m*	(*Table 19*)
Line	*84→ 96*:	Identification of the radiating nodes		*cad_ban.m*	(*Table 26*)
Line	*97→ 117*:	Assemble cond. & conv. el. matrices			
Line	*119→ 173*:	Radiative heat exchanges	cavity	*geo_vfs.m*	(*Table 27*)
			street	*geo_stf.m*	(*Table 38*)
			balcony	*geo_baf.m*	(*Table 42*)

cs = 1, 2, 3:	View factor and radiosity matrix	*137 → 173*		
Me = 1:	linear permanent heat transfer	*175 → 214*		
Me = 2:	Nonlinear permanent heat transfer	*215 → 223*	*fem_snl.m*	(*Table 76*)
Me = 3:	Transient linear heat transfer	*224 → 230*	*fem_til.m*	(*Table 62*)
Me = 4:	Nonlinear transient heat transfer	*231 → 244*	*fem_tir.m*	(*Table 63*)
Me = 5:	Cavity, VF matrix radiat exchanges	*245 → 275*	*fem_tit.m*	(*Table 60*)

Line 274: End of the procedure

Procedure *Fiammetta_20211111.m*

```
1   CPU = tic; deb = 0 ;Ai = 0;fa=1000 ;F=0;lon=0.;
2   Gi  =10;[xyz_cao,car_cao,nbo,Me,Di,Ne,Co,nvn,nnr] = cad_gin(Gi);
3   rc  = 1;ra = 0;cs = 4;      % rc = 1: rad. exch. computed, ra = 1: VF used
4   disp(['L   4, Me, Di, Ne   : ',num2str([Me  Di  Ne])])
5   disp(['L   5, Co nnr nvn   : ',num2str([Co nnr nvn])])
6   disp(['L   6, rc, ra, cs   : ',num2str([rc  ra  cs])])
7   disp(['L   7, CAD interf.  : ',num2str(nbo)])
8   np   = size(car_cao,1);npv  = size(xyz_cao,1);
9   disp(['L   9, N. pa., vert.: ',num2str([np npv])])
10  nni  =15;if nni<1;nni=1;end     % Number nodes inserted on patches borders
11  disp(['L  11, num. nod side: ',num2str(nni)])
12  nci  = nni+1;mc=nci*4;if cs==2; mc = mc-nni; end % cs = 2 street 3 patches
13  disp(['L  13, num elem side: ',num2str(nci)])
14  area = 0;
15  for i    = 1:np           % Sum of patch areas = domain area: lines 13-21
16      area = area + norm(...
17          cross([xyz_cao(car_cao(i,3),:)-xyz_cao(car_cao(i,1),:) 0],...
18                [xyz_cao(car_cao(i,4),:)-xyz_cao(car_cao(i,1),:) 0]));
19      area = area + norm(...
20          cross([xyz_cao(car_cao(i,2),:)-xyz_cao(car_cao(i,1),:) 0],...
21                [xyz_cao(car_cao(i,3),:)-xyz_cao(car_cao(i,1),:) 0]));
22  end
23  area = area/2; disp(['L  23, Domain area  : ',num2str(area),' m2'])
24  if deb==1 ;disp( 'L  25, Call function: ..... cad_mes .....');end
25  [xyt,lK,bor,pbo] = cad_mes(xyz_cao,car_cao,nni,nbo);     % Mesh generation
26  nel  = size(lK,1) ; disp(['L  26, Num. elements: ',num2str(nel)])
27  no   = size(xyt,1);dK = no + nvn + nnr;
28                     disp(['L  28, Num. of DOF  : ',num2str(dK)])
29  for i    = 1:nel         % Enforce anticlockwize ordering of element nodes
30      n    = cross([xyt(lK(i,3),:)-xyt(lK(i,1),:) 0],...
31                [xyt(lK(i,4),:)-xyt(lK(i,1),:) 0]);
32      if n(3) < 0;iq = lK(i,2);lK(i,2) = lK(i,4);lK(i,4) = iq;end
33  end
34  bov   = zeros(nbo,npv+2); bov(:,1:2)=bor(:,1:2); pe = 0.;% Outward normals
35  if deb==1;disp('L  39, Call function: ..... gra_mnl .....');end
36  gra_mnl(xyz_cao,car_cao,[0 0 0]);axis equal;  % Drawing normals: + line 56
37  axis off;title(['Labels & normals of the ',num2str(np),' patche(s)'])
38  for i = 1 : nbo % ........................ Loop on the nin CAD interfaces
39      if bor(i,4)==0% interface i must be a single one to be on the boundary
40          ne = cross([xyt(bor(i,2),:)-xyt(bor(i,1),:) 0],[0 0 1]);
41          nn = ne / norm(ne);
42          mi = [(xyt(bor(i,6),:) + xyt(bor(i,5),:))/2 0];         % Mid edge
43          li = norm(xyt(bor(i,2),:) - xyt(bor(i,1),:));        % Edge length
44          pe = pe + li;             % Update the perimeter of the CAD domain
45          po = (mi + nn*li/5 );                    % Normal vector: mi - po
46          plot([mi(1) po(1)],[mi(2) po(2)],'b','LineWidth',2);hold on
47          for j     = 1:npv     % Loop on npv vertices not in patch bor(i,3)
48              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  59, Domain perim.: ',num2str(pe),' m'])    % End outward normals
xyz  = zeros(dK,3); xyz(1:no,1:2) = xyt;     % Coordinates expressed in 3D
th   =.1 ; disp(['L  61, Thickness    : ',num2str(th),' m'])
k    =.2; disp(['L  62, Conduction k : ',num2str(k),' W/(mK)'])
pai  =  1 ; disp(['L  63, DT isotherms : ',num2str(pai), ' K'])
SB   =  5.6704e-8;                        % Stefan-Boltzmann constant Wm-2K-4
if rc > 0 ;disp(['L  65, St. Boltzmann: ',num2str(SB),' W/(m2K4)']);end
if deb    == 1;disp( 'L  67, 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 nf     < 10 ;disp(['L  69, Fixed DOF    : ',num2str(lfi)]);end
if deb    == 1; disp( 'L  71, Call function: ..... cad_Neu .....');end
[gh,bos]  =  cad_Neu(Ne,dK,car_cao,pbo,bor,th,xyz_cao);% Neumann b.c.
if deb    == 1;disp('L  75, Call function: ..... cad_mnl .....');end
if nel < 101;gra_mnl(xyz,lK,[0 0 0]);axis equal,axis off;hold on;end
if Co > 0
    vc=zeros(nvn+nnr,2);xyz(no+(1:nvn+nnr),1:2)=vc;   % Virtual nodes def.
    h= 5;disp(['L  76, Convection h : ',num2str(h),' W/(m2K)'])
    if deb == 1;disp( 'L  78, Call function: ..... cad_con ....c');end
    [lc,he] = cad_con(car_cao,bor,pbo,h,dK,nci,deb,nvn,cs,Co);% mcr=0;
end % End option Co > 0. Some sides are linked to virtual convective nodes
lcont = zeros(nbo*nci,1);
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 cs ~=4           % If cs = 5, both vertical sides are convective
    if deb==1; disp( 'L  86, Call function: ..... cad_ban .....');end
    [lcont,lv] = cad_ban(car_cao,bor,pbo,nni,cs);
%     mcr = size(lcont,1)+1;     % For open space like a street or balcony
    mcr = size(lcont,1);                              % Closed cavities
%     if mcr > 0; disp(['L  89, Numb. r.nodes: ',num2str(mcr)])
%         if mcr < 12;disp(['L  90, R. nod. lcont: ',num2str(lcont')]);end
%         if size(lv,2) > 1;disp(['L  91, Rad. vertices: ',num2str(lv)]);end
%     end
end
%  ..................... Assembling the nel element conductivity matrices
Kk      = zeros(dK,dK);                                    % Initializations
disp(['L  96, 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 100, Call function: ..... fem_Kco .....');end
for n   = 1:nel            % Compute global K matrix, loop on the elements
    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 109, 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
Ms        = zeros(nbo*nci,1);
if rc     == 0                          % Radiative exhanges are not present
    re = 0; Lel(1) = 0; ns = 0;              % Init. edge elements lenghts
else          % View factor and radiosity matrices for radiative transfers
    if cs == 1;lcc = [lv' ;lv(1)];ns = 4;end   % lcc & ns: quadril. cavity
    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 == 5;lcc = [lv' ;lv(1)];ns = 4;end   % lcc & ns: quadril. cavity
    Lel   = zeros(ns,1);    % Compute the lengths of th ns radiative edges
    Tsky =280;disp(['L 125, Sky temper.  : ',num2str(Tsky),' K'])
```

```
126      for i = 1:ns
127          Lel(i)= sqrt((xyz_cao(lcc(i+1),1)-xyz_cao(lcc(i),1))^2 + ...
128              (xyz_cao(lcc(i+1),2)-xyz_cao(lcc(i),2))^2);
129      end;
130      disp(['L 130, Rad. edg. len: ',num2str(Lel(1:ns)'),' m'])
131      disp(['L 131, Radiativ area: ',num2str(sum(Lel)*th),' m2'])
132      re =0.5 ;disp(['L 132, Refl. coeff. : ',num2str(re)])
133      if cs == 1                              % Quadrilateral cavity view factors
134          if deb==1;disp('L 155, Call function: ..... geo_vfs .....');end
135          Fs = geo_vfc(nci,Lel,ns);I=eye(size(Fs,1));
136          M =(I-re*Fs);disp(['L 136, det    (M))  : ',num2str(det(M))])
137      end
138      if cs == 2                                % Street section view factors
139          if deb ==1;disp('L 140, Call function: .... . geo_stf .....');end
140          Fs      = geo_stf(nci,Lel(2:3));I=eye(size(Fs,1));% Street VF mat.
141          M =(I-re*Fs);disp(['L 141, sum(sum(M))  : ',num2str(sum(sum(M)))])
142          Fsky = 1-sum(Fs,2)'; % Col. vect. of sky view fact.
143          if nni ==1;disp(['L 146, Fs=1-sum(F,2): ',num2str(Fsky,3)]);end
144          if re == 1; Ms = zeros(size(Fs,1),1)              ;else
145      nne             = (mcr-1)/3;  % nne = numb segm per street side
146      lon             = zeros(1,nne);
147      for i           = 1:nne  % Computing the element lengths on the 3 sides
148          lon(i)      = Lel(1)/(nne); lon(mcr-i)  = Lel(3)/(nne);
149          lon(i+nne) = Lel(2)/(nne);
150      end;       disp(['L 154, Radiat. area : ',num2str(sum(lon)*th),' m2'])
151    Ms =(re*(I-Fs)*M^(-1)-I)*SB*Lel(2)*th*Fsky'*Tsky^4;%Ms=-SB*Fsky'*Tsky^4;
152         if mcr < 10;disp(['L 156, Ms mid-edges : ',num2str(Ms') ,' W']);end
153          end ;disp(['L 157, Sum sky loads: ',num2str(sum(Ms),3),' W/m2'])
154      end
155      if cs == 3                                % Thermal bridge view factors
156          if deb==1;disp('L 156, Call function: ..... geo_baf .....');end
157          Ftot  = 0    ;nco = size(Lel,1);k = 0;lon = zeros(1,nco*nci);
158          for i = 1:nco;for j = 1:nci;k=k+1;lon(k) = Lel(i)/nci;end;end;
159          disp(['L 159, Radiat. area : ',num2str(sum(lon)*th),' m2'])
160          if ra == 1;F = geo_baf(nci,xyz_cao);
161              nv = size(F,2);disp(['L 161, numb. col. F : ',num2str(nv)])
162              Fs    = F(1:nv-2,1:nv-2);I = eye(size(Fs,1));M = (I-re*Fs);
163              disp(['L 163, determinant M: ',num2str(det(M))])
164              Fgr   = F(1:nv-2,nv-1); Fsky=F(1:nv-2,nv);
165              nne   = size(Fs,1);k=0;%lon = zeros(1,nne
166              Ms    =(re*(I-Fs)*M^(-1)-I)*SB*th*(Fsky+Fgr).*Tsky^4.*lon';
167              disp(['L 167, Sum 2d member: ',num2str(sum(Ms),3),' W'])
168              if nni == 1
169                  disp(['L 169, Ms  Eq. 104  : ',num2str(Ms')])
170                  disp(['L 170, unicol Fgr   : ',num2str(Fgr' ,3)]);
171                  disp(['L 171, unicol Fsky  : ',num2str(Fsky',3)])
172                  disp(['L 172, sum(F,2)     : ',num2str(sum(F,2)',3)])
173              end
174          end
175      end
176  end
177  if Me == 1 % ..... Solution of linear steady state heat transfer problems
178      nf    = size(lfi,2);  N  = zeros(nf,dK);% L. con. fix.%gh=zeros(nf,1);
179      for i = 1:nf;N(i,lfi(i)) = 1;gh(dK+i) = fT(i); end;
180      A     = [K N';N zeros(nf,nf)];B = A\gh;tca = B(1:dK);
181      figure;gt =[min(tca) pai max(tca)];
182      if deb ==1;disp(['L 173, DT isoth gt  : ',num2str(gt),' K']);end
183      nc3    = (nci)^2;  % nc3 = numb elem / patch, same for all the patches
184      if deb ==1;disp( 'L 175, Call function: ..... gra_ipa .....');end
185      for i = 1 : np                                  % Loop on CAD patches
186          gra_ipa(nci,nci,lK((i-1)*nc3+1:nel,:),tca,xyz,gt);hold on
187      end
188      axis equal;colorbar;axis off
189      title (['Dissipation: ',num2str(.5*tca'*K*tca,3),' WK, DOF: ',...
190              num2str(dK),' '])
191      for i = 1:nbo                         % Drawing the border of the domain
192          if bor(i,4)==0
193              plot([xyz(bor(i,1),1) xyz(bor(i,2),1)], ...
194                      [xyz(bor(i,1),2) xyz(bor(i,2),2)],'k','LineWidth',2)
195          end
196      end                                     % End drawing of the isotherms
197      disp(['L 188, Total dissip.: ',num2str(tca'*K*tca/2,3),' WK'])
198  %     disp(['L 220, -dot(B,gh)/2 : ',num2str(-dot(B,gh)/2,3),' WK'])
199      dissol = .5*tca(1:no)'*Kk(1:no,1:no)*tca(1:no);
200      disp(['L 191, Dis. in solid: ',num2str(dissol,3),' WK'])
201      disp(['L 192, DT   in solid: ',num2str(max(tca(1:no))-...
202              min(tca(1:no)),3),' K'])
```

```
203     if deb==1;disp(['L 197, tca mi.ma.av.: ',...
204             num2str([min(tca) max(tca) mean(tca)],3),' K']);end
205 %     if nvn > 2;disp(['L 226, DT. convect  : ',...
206 %             num2str(max(tca(lic))-min(tca(lic)),3),' K'])
207 %     end                       % End standard linear heat transfer process
208     reac = K*tca;
209     if nf < 6
210         disp(['L 201, Fixed DOF    : ',num2str(lfi)])
211         disp(['L 202, Imposed temp.: ',num2str( tca(lfi)',3),' K'])
212         disp(['L 203, React. flows : ',num2str(reac(lfi)',3),' W'])
213     end
214     disp(['L 205, End Me=1, CPU: ',num2str(toc(CPU),3),' sec. '])
215     if deb ==1;disp('L 168-207, End stationary linear');end
216 end
217 if Me == 2 % Non linear stat. rad. heat transfer - conv. like virtual node
218     if nnr > 0              % nnr is the number of radiative virtual nodes
219         if deb ==1;disp('L 217, Call function: ..... fem_snl .....');end
220         tca=fem_snl(K,gh,lfi,fT,xyz,lK,lcont,nnr,nvn,nci,np,SB*(1-re));
221         disp(['L 215, End Me=2, CPU: ',num2str(toc(CPU),3),' sec. '])
222     end
223     if deb ==1;disp('L 208-215, End non lin virt rad node.');end
224 end
225 if Me == 3  % ........ Solution of transient linear heat transfer problems
226     if deb==1;disp ('L 224, Call function: ..... fem_til .....');end
227     [tca]=fem_til(np,xyz,lK,dK,nci,deb,rc,cs,re,area,th,pai,fT,lfi,gh,...
228         nbo,bor,K,Gi);
229     disp(['L 231, End Me=3. CPU: ',num2str(toc(CPU)  ,3),' sec. '])
230     if deb ==1;disp('L 216-222, End linear transient');end
231 end
232 if Me == 4  % Solution of non linear transient rad. heat transfer problems
233     disp(['L 233, Num. fix. DOF: ',num2str(lfi)])
234     disp(['L 234, T. fix. nod. : ',num2str(fT),' K'])
235     if deb==1,disp('L 234, Call function: ..... fem_tir .....');end
236 %     Ftot = zeros(size(lcont,1)-1,size(lcont,1)+1);
237     [tca] = fem_tir(np,xyz,lK,dK,nci,deb,rc,ra,cs,area,th,xyz_cao,...
238     gh,lfi,fT,Tsky,nbo,bor,K,Lel,re,SB*th*(1-re),lcont,Ms,pai,F,lon);
239     rea = Kk*tca;reac = K*tca;
240     disp(['L 240, Diss in solid: ',num2str(rea' *tca/2,3),' WK'])
241     disp(['L 241, Total dissip.: ',num2str(reac'*tca/2,3),' WK'])
242     disp(['L 242, End Me=4. CPU: ',num2str(toc(CPU)  ,3),' sec. '])
243 %     test=SB*280^4*th*3;
244     if deb ==1;disp('L 232-245, End non lin radiat. trans. H.T.');end
245 end %End of Fiammetta for non linear transient rad. heat transfer problems
246 if Me == 5        % Cavity with view factor matrix and radiative exchanges
247     Mpr = zeros(nbo*nci,nbo*nci);% if cs~=2;Ms = zeros(nbo*nci,1);end
248     if rc > 0
249         if re ==1
250             Mpr = eye(size(Fs,1),size(Fs,1));
251         else      % if re = 0 : black body, Mpr+Fs = I
252             Mpr =(eye(size(Fs,1),size(Fs,1))-Fs)*M^(-1);
253         end
254     end
255 %     if nni < 4
256 %         disp(['L 242, Line clos. Fs: ',num2str(sum(Fs ))])
257 %         disp(['L 243, Col. clos. Fs: ',num2str(sum(Fs,2)')])
258 %         disp(['L 244, sum(sum(Fs)) : ',num2str(sum(sum(Fs)))])
259 %         disp(['L 245, Line clos. M : ',num2str(sum(M ))])
260 %         disp(['L 246, Col. clos. M : ',num2str(sum(M,2)')])
261 %     end
262     if deb==1;disp('L 248, Call function: ..... fem_tit .....');end
263     [tca]  = fem_tit(np,xyz,lK,dK,nci,deb,rc,cs,nel,area,bos,...
264         gh,fT,lfi,nbo,bor,Kk,mcr,Lel,SB,re,Mpr,Ms,lcont,pai,th);
265     if rc > 0
266         gsm = Kk*tca;
267         figure ( 'Position', [100 100 800 300]); bar(gsm(lcont)');grid on
268         title  (['Emittance: ', num2str((1-re),2),...
269         ', resultant = sum(gsm(lcont)): ',...
270         num2str(sum(gsm(lcont)),2),' W']);hold on
271         ylabel ( 'Radiative loads Kk*tca (W)')
271         disp(['L 271, Resultant fl.: ',num2str(sum(gsm(lcont)),2),' W'])
273     end
274     disp(['L 272, End Me=5. CPU: ',num2str(toc(CPU),3),' sec. '])
275     if deb ==1;disp('L 244-274, End cavity, view factors, rad exch.');end
276 end
```

Table 58: Matlab© procedure Fiammetta_20211111.m

	Line	Occ.	Name	Meaning of the variables used in the procedure *Fiammetta.m*
	1	19	deb	Flag to control the display of functions calls
	1	3	Neu	Flag to control the von Neumann boundary conditions
	1	6	pr	Uni-line matrix to control the procedure Fiammetta
	1	3	rae	Flag indicating the use of conductive radiative elements
1	4	7	nbo	Input data: Number of CAD patches interfaces
2	4	13	cs	View factor matrix flag: 1 = cavity, 2 = street, 3 = thermal bridge
3	7	6	np	Number of CAD patches
4	8	3	npv	Number of CAD vertices
5	8	9	area	Area of the solid domain m^2, computed in *lines 9 - 18*
6	22	9	nni	Number of nodes per side of CAD patches
7	23	11	nci	Number of elements per side of CAD patches
8	24	8	nnv	Number of virtual convection nodes
9	25	5	nnr	Number of virtual radiative nodes
10	25	6	rc	Flag indicating presence of radiative exchange (1 = yes, 0 = no)
12	29	6	th	Thickness, *m*
13	30	2	k	Conductivity coefficient $Wm^{-1}K^{-1}$
15	31	4	SB	Stefan-Boltzmann constant: 5.6704 10^{-8} $Wm^{-2}K^{-4}$
19	33	4	pai	Temperature interval in isotherms drawing *K*
16	34	16	lK	Localization matrix of conductive elements (dimension: *nel* x 4)
17	36	7	nel	Number of conductive elements
	37	14	no	Number of nodes of the mesh
20	38	10	ndK	Number of *DOF*
18	44	4	pe	Perimeter of the domain, *m*
	70	11	xyz	Matrix of the nodal coordinates expressed in 3D, *m*
	73	10	lfi	Uni-column matrix - list of fixed nodes (Dirichlet), (*cad_Dir.m*)
	73	4	fT	Uni-column matrix of fixed nodal temperatures, (*cad_Dir.m*)
	73	2	nfi	Number of fixations (dim of *fT* & lfi), (*cad_Dir.m*)
	75	5	gh	Weight functions of patch side loads (see *lines 117 - l 26*)
29	80	3	h	Reference convection coefficient $Wm^{-2}K^{-1}$
30	82	3	he	Uni-line matrix of the element convection coefficients
31	82	6	lc	Localizations of convective elements defined in *cad_con.m*
23	85	5	mcr	Number of radiative nodes, size(lcont,1)
21	88	6	lcont	Uni-column matrix of *mcr* radiative cavity, street, balcony nodes
22	88	5	lv	Uni-column matrix of cavity, street, balcony vertices
33	93	6	Kk	Global conductivity matrix of meshed solid domain WK^{-1}

	94	2	co	Uni-line matrix of the *nel* products of thickness by cond. coeff. *k*
34	97	2	Kel	Element conductive matrix WK^{-1} (*fem_Kco.m*)
35	102	8	K	Global conductivity matrix WK^{-1} (solid part + …)
46	106	2	Kec	Element "conductive – convective" matrix WK^{-1} (*fem_Kcv.m*)
36	112	6	re	Coefficient of reflexion (adimensional)
37	113	5	ns	Number of edges of the radiative part of the boundary
38	113	7	lcc	Sequence of cavity CAD vertices + first repeated if cavity
39	121	6	Lel	Uni-column matrix of the *ns* radiative edges' lengths *m*
42	127	7	Fs	View factor matrix for radiative element from mesh border
43	132	6	Fsky	Uni-column matrix of sky view factors for street & balcony
	137	5	F	Thermal bridge view factor matrix including mesh, sky, ground
	137	9	nf	Size of view factor matrix for thermal bridge
	164	3	Fgr	Uni-column matrix of ground view factors
	139	2	Ftot	[*Fs Fsky Fgr*]
44	136	9	M	Radiosity matrix (adimensional)
	146	2	Mm1	M^{-1}
45	147	2	M_pr	$(I - Fs)\ M^{-1}$ matrix - flow out of a segment (adim.)
	180	26	tca	Uni-column of the *dK DOF* or nodal temperatures
41	181	3	gt	Uni-line matrix for isolines definition in *gra_ipa.m*
	171	3	Kr	Conductivity global matrix due to radiative exchanges

Table 59: Variables used in the procedure Fiammetta.m

Solution of the system.

To allow introducing fixations anywhere in the mesh, it is convenient to group all the fixed *DOF* at the end of the conductivity matrix.
A simple method to do it is:

Examine all the *DOF* from the first one present in the *K* matrix.
If a *DOF* is fixed, swap it with the last free *DOF* of the matrix. This operation needs:

1. a swapping of the lines,
2. a swapping of the columns
3. a swapping of the heat loads,

After this operation, the system is easily solved, using the matrix decomposition:

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} 0 \\ ? \end{bmatrix} \qquad (108)$$

When the system is solved, it is necessary to swap back the temperature vector.
Lines 32 - 33 : Output. The last lines of the procedure are devoted to the output, basically temperature levels drawing by using the Matlab© standard function *fill* function (*Figure 6*). Another solution is obtained in *Figure 8* with the Matlab© function *gra_ipa.m* of *Table 69* or the function *gra_ist.m* given in *Table 3*.

To solve a steady state heat transfer problem, we have to fix at least one node to take out the singularity of the problem (this action is setting the level of temperature). To obtain a non-uniform temperature field, at least another node has to be fixed at a different temperature or at least one second member term has to be introduced on any node different from the fixed one. All the nodes are concerned with this rule.
With both imposed temperatures (T_2, in red) and prescribed heat fluxes (F_1, in red), the system to be solved is characterized by 4 submatrices. The unknown variables are the vectors [T_1] and [F_2] (in blue)

$$[K]=\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \quad ; \quad [K]\begin{bmatrix} T_1 \\ T_2 \end{bmatrix}=\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \tag{109}$$

The system to be solved is:

$$[T_1]=[K_{11}]^{-1}\{[F_1]-[K_{12}][T_2]\} \tag{110}$$

Finally, the fluxes are calculated in the following operation:

$$[F_2]=[K_{21} \quad K_{22}]\begin{bmatrix} T_1 \\ T_2 \end{bmatrix}=[K_{21} \quad K_{22}]\begin{bmatrix} [K_{11}]^{-1}\{[F_1]-[K_{12}][T_2]\} \\ T_2 \end{bmatrix} \tag{111}$$

We proceed with a very simple case with imposed temperatures on the lower face (base) and a constant flow on the left vertical edge (*Figure 9*). The mesh has 20 x 40 elements. As the upper and right faces are adiabatic, the isotherms are orthogonal to them. The base temperature is fixed to 270 *K*. The total flow on the left side is equal to 40 *W*.
In computing the second member of the system of equations, there is ambiguity for the node of the lower left corner that receives a flux of 20/*ny W*, but is fixed. The dissipation energy is obtained by pre- and post- multiplying the global conduction matrix by the temperature vector:

$$\frac{1}{2}[T]^T[K][T] \tag{112}$$

When the mesh is refined, this energy is converging to its exact value of 651 *WK* (in this example, the convergence is yet achieved with the 20 x 40 mesh [Beckers & Beckers 2015]).

8.2 Transient heat transfer problem

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

```
function[tca]  = ...
    fem_tit(np,xyz,lK,dK,nci,deb,rc,cs,nel,area,bos,gh,...
    fT,lfi,nbo,bor,Kk,mcr,Lel,SB,re,Mpr,Ms,lcont,pai,th)
tmi   = 280;disp(['Lt 04, Initial temp.: ',num2str(tmi) ,' K'])
tcan  = ones(dK,1)*tmi;                                  % Initial conditions
nit   = 72 ;dth=1;dtd=nit/2;          % Numb. iter. & delta tau per it. (h)
dti   = dth*3600;disp(['Lt 07, Time step    : ',num2str(dti),' sec'])
tt    = dti*nit;                                  % Analyzed period in seconds
       disp(['Lt 09, 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(['Lt 15, Spec. capac. : ',num2str(Cp),' J/(kg.K)']);
ro    = 2500 ;disp(['Lt 16, 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('Lt 19, Call function: ..... fem_Cae .....');end
for n     = 1:nel      % Glob. capacity mat. assemb., loop on nel elements
    Cae   = fem_Cae(xyz,lK(n,:))*th*Cp*ro;       % 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 capacity comp. from mat. data
disp(['Lt 29, sum(sum(C))  : ',num2str(sum(sum(C))*1e-6), ' MJ/K'])
% disp(['Lt 30, area*th*ro*Cp: ',num2str(cap), ' MJ/K']);
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(['Lt 35, Ampl inp heat: ',num2str(ih), ' Wm-2'])
% gt      = [tmi pai tma];                % gt = [min(tcan) pai max(tcan)]
nc2   = nci*nci;                          % Number of nodes 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
lon     = ones(1,4*nci)/nci;            % Valid only for the square cavity
% if rc*cs == 1;sn = 0;end                                        % Cavity
if rc*cs == 2                  % Radiation in the street section (3 sides)
    lon            = ones(1,3*nci)/nci;
    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
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)
if mcr <10;disp(['Lt 54, lon          : ',num2str(lon)        ,' m']);end
           disp(['Lt 55, Radiat. area : ',num2str(sum(lon)*th),' m2'])

% ........................................... Loop on the time ierations
Mn      = zeros(dK,1);
for it  = 1:nit                                % Loop on time iterations step
    if rc    == 1        % Add radiative matrix Kr to conductive matrix Kk
      if cs ==1;[Kr] = fem_caK(tcan,re,SB*th*(1-re),Mpr,it,lcont,Lel/nci);
        K(lcont,lcont) = Kk(lcont,lcont) + Kr(lcont,lcont); end
        if cs > 1
            if it==1
           if deb==1;disp('Lt 61, Call function: ..... fem_rsm .....');end
            disp(['Lt 62, sum(Ms)      : ',num2str(sum(Ms),4),' W'])
            end
            [Mn] = fem_rsm(tcan,re,SB*th,Mpr,it,Ms,lcont,lon,dK);
     if it==1;disp(['Lt 64, sum(Mn)      : ',num2str(sum(Mn),4),' W']);end
        end
    end    % Injected heat = weights * periodic function f(t)* load * area
```

```
    g           = gh * f(it) * ih * bos;                    % Injected heat
    if rc > 0;g(lcont) = g(lcont);end               % Heat coming from sky
    ga(it+1)    = sum(g)*dti;     % Heat input at step it (for statistics)
    ip          = ip+1; if lfi(1)== 0;nfi=0;else;nfi=size(lfi,2);end
    if nfi      == 0                              % Domain without fixations
        tca     = (C + dti*K)\(C*tcan + dti*g + Mn);      % Equation 68 ??
    else                                    % Domain including some fixations
        if it   == 1; tcan(lfi) = fT; end
        tca     = fem_tra(K,C,dti,g,lfi,tcan);
        go      = K * tca;                      % Second member of the system
        gou(it) = sum(go(lfi))*dti;        % Outgoing heat at iteration it
    end                                          % tcav (it+1) = tca (ndK,1);
    tsmax(it+1) = max  (tca);
    tsmin(it+1) = min  (tca);
    tmoy (it+1) = mean (tca); %(tsmin(it+1)+tsmax(it+1))/2;
    tcan        = tca;
    if ip == dtd       % In this iteration, isotherms drawing is generated
        ip = 0; gt = [min(tca) pai max(tca)];
        disp(['Lt 79, iteration    : ',num2str(it)])
        if deb==1;disp ('Lt 80, Call function: ..... gra_ipa .....');end;
        figure
%         a=-.2;b=-.1;ha=max(xyz(:,2));ti=299;ta=300;%..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(['Lt 88 : ',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(['Lt108, Tmean - Tini : ',num2str(ddt,3),' K'])
    disp(['Lt109, Min obs. temp: ',num2str(min(tsmin),3),' K'])
    disp(['Lt110, Max obs. temp: ',num2str(max(tsmax),3),' K'])
    disp(['Lt111, Final temper.: ',num2str([tsmin(nit+1) tmoy(nit+1)...
        tsmax(nit+1)],3)])
    disp(['Lt113, Capac * Del T: ',num2str(ah ,3),' MJ'])
if deb==1;disp( 'Lt114, Call function: ..... gra_tev .....');end
    gra_tev(nit,tt,re,tsmax,tsmin,tmoy)          % Drawing temp. evolution
if deb == 1;disp( 'Lt116, Call function: ..... gra_hie .....');end
% gra_hie(nit,tt,ga)                          % Drawing heat input evolution
hdm    = tt*ih/pi*bos*1e-6; % Explicit exact injected heat Equ. (80) pp 49
    disp(['Lt119, Injected heat: ',num2str(sum(ga)/1.e06,3),' MJ'])
if ih ~= 0;disp(['Lt120, Exact ih x 30: ',num2str(hdm,3),' MJ']);end
if nfi > 0       % Results shown & displayed only in presence of fixations
    disp(['Lt122, ejected heat : ',num2str(sum(gou)*1.e-6,3),' MJ'])
    figure('Position',[100 100 700 300]);
    plot(gou(1:it));grid on;hold on
    title(['Total ejected heat: ',num2str(sum(gou),4),' J'],...
            'fontsize',15);hold on
    ylabel('Ejected heat (J)','fontsize',15);hold on
end
end
```

Table 60: Matlab© function fem_tit.m – Cavity, VF matrices, radiative transfers

	Line	Occ.	Name	Meaning of the variables used in the function *fem_tit.m*
In	2	2	np	Number of patches
In	2	7	xyz	Matrix of the 3D nodal coordinates expressed in, *m*
In	2	7	lK	Localization matrix of conductive elements (dimension: *nel* x 4)
In	2	8	ndK	Number of *DOF*
In	2	9	nci	Number of elements per patch edge

In	2	4	deb	Flag enabling the display of function call
In	2	4	rc	Flag indicating presence of radiative exchanges (1 = yes, 0 = no)
In	2	8	cs	Flag for view factor matrix: 1 = cavity, 2 = street, 3 = balcony
In	2	2	nel	Number of conductive elements
In	2	3	area	Area of the solid domain m^2
In	2	10	th	Thickness, m
In	2	5	xyz_cao	((Num. patch vert.) x 2) matrix of coordinates of patches vertices
In	2	2	gh	Uni-column matrix of flow input distribution (*cad_Neu.m*)
In	3	3	nfi	Number of fixed nodes
In	3	9	fT	If nfi > 0, uni-column matrix of fixed nodal temperatures K
In	3	5	lfi	List of fixed nodes on a side (Dirichlet), (*cad_Dir.m*)
In	3	2	pai	Temperature interval in isotherms drawing K
In	3	2	nbo	Number of *CAD* patches interfaces
In	3	6	bor	matrices *bor* and *pbo* computed in *cad_mes.m* (*Table 15*).
In	3	2	no	Number of mesh nodes
In	3	2	Kk	Global conductivity matrix of meshed solid domain WK^{-1}
In	3	9	mcr	Number of radiative nodes, *size (lcont,1)*
In	3	5	Lel	Uni-column matrix of the cavity or street section edges lengths m
In	3	2	Fsky	Uni-column matrix of sky view factors from street section elements
In	3	2	Fgr	Uni-column matrix of ground view factors from street elements
In	3	5	SB	Stefan-Boltzmann constant: 5.6704 10^{-8} $Wm^{-2}K^{-4}$
In	3	3	re	Coefficient of reflexion (adimensional)
In	3	3	M_pr	$(I - F)\ M^{-1}$ matrix for flow outcoming from a segment (adim.)
In	3	7	lcont	List of radiative boundary nodes (*cad_ban.m*)
1	4	7	tmi	Initial temperature (scalar expressed in K)
2	5	8	tcan	Nodal temperatures of the solid before starting the iterations
3	6	10	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 temperature in solid at each time step K
16	28	4	tsmax	Uni-line matrix, highest temperature in solid at each time step K
17	29	4	ga	Uni-line matrix, heat input at each step sum(g)*dti, expressed 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	62	3	Kr	Radiative matrix added to the conductive one
28	64	4	K	Global conductivity matrix

Table 61: Variables used in the function fem_tit.m

Function *fem_til.m* - solution of linear transient equations

```
function [tca]= ...
    fem_til(np,xyz,lK,dK,nci,deb,rc,cs,re,area,th,pai,fT,lfi,gh,nbo,...
    bor,Kk,Gi)
tmi   = 280;tma=300;nel=size(lK,1);
disp(['t. 05, Initial temp.: ',num2str(tmi),' K'])
if Gi == 17           % In the example of figure 41 - 42, nni must be even
    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                                 %     nfi = 0;tcan = ones(dK,1)*tmi;
else                                   % Initial and fixed temperatures
    tcan  = ones(dK,1)*tmi; nfi = size(lfi,1); tcan(lfi) = fT;
end
if dK < 5; disp(['t. 18, Initial temp.: ',num2str(tcan'),' K']);end
nit   =720;dth=1;dtd=nit/2;     % Numb. iter. & delta tau per it. (hours)
           disp(['t. 20, nit  N. iter.: ',num2str(nit)])
dti   = dth*3600;disp(['t. 21, Time step    : ',num2str(dti),' s'])
tt    = dti*nit;                            % Analyzed period in seconds
       disp(['t. 23, 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(['t. 29, Spec. capac. : ',num2str(Cp),' J/(kg.K)']);
ro    = 2500 ;disp(['t. 30, Spec. mass   : ',num2str(ro),' kg.m-3'])
C     = zeros(dK,dK);       % Initialization of the global capacity matrix
if deb==1;disp('t. 32, Call function: ..... fem_Cae .....');end
for n     = 1:nel         % 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(['t. 40, sum(sum(C))  : ',num2str(sum(sum(C))*1e-6), ' MJ/K'])
disp(['t. 41, area*th*ro*Cp: ',num2str(cap), ' MJ/K']);
tsmax = ones(1,nit+1)*tma ;         tsmin = ones(1,nit+1)*tmi;
tmoy  = ones(1,nit+1)*tmi ; tcav  = tmoy;gou = zeros(1,nit+1);
ih   =  50;% 4.04008;%                          % Imposed heat load in Wm-2
disp(['t. 45, Imposed Heat : ',num2str(ih), ' W/m-2'])
bos     = th*(max(xyz(:,1))-min(xyz(:,1)));% Cross section area upper edge
g       = zeros(dK,1);
if rc*cs== 1;   disp( 't. 40, Call function: ..... fem_caK .....');end
K       = Kk; ip=0; % Conduct. matrix related to solid and convective part
for it  = 1:nit % ............................. Loop on the time ierations
    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
        tca     = fem_tra(K,C,dti,g,lfi,tcan);
        go      = K  * tca;                  % Second member of the system
        gou(it) = sum(go(lfi))*dti;            % Reactions at iteration it
    end
    tcav (it+1) = tca (dK,1);
    tsmax(it+1) = max (tca(1:dK-size(lfi,1)));
    tsmin(it+1) = min (tca(1:dK-size(lfi,1)));
    tmoy (it+1) = mean(tca(1:dK-size(lfi,1)));
    tcan        = tca;
    if ip == dtd       % In this iteration, isotherms drawing is generated
        ip = 0; gt=[min(tca) pai max(tca)];
        disp(['t  67, iteration    : ',num2str(it)])
        if deb==1;disp ('t  68, 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(['Lt 76 : ',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(['t. 89, Tmean - Tini : ',num2str(ddt,3),' K'])
      disp(['t. 90, DT* sumsum(C): ',num2str(ah ,3),' MJ'])
if ddt > .1
    disp(['t. 92, Min obs. temp: ',num2str(ceil(min(tsmin)),4),' K'])
    disp(['t. 93, Max obs. temp: ',num2str(max(tsmax),4),' K'])
    if deb==1;disp( 't.110, Call function: ..... gra_tev .....');end
    gra_tev(nit,tt,re,tsmax,tsmin,tmoy);   % tcav(1) = -1;% Temper. evol.
end
eih    = tt*ih/pi*bos*1e-6;                    % Equation (81) of tutorial
%           disp(['t. 96, S&G Heat in  : ',num2str(ga*nit,3),' MJ'])
if ih ~= 0;disp(['t. 90, Heat inp.(81): ',num2str(eih,3),' MJ']);end
if nfi > 0       % Results shown & displayed only in presence of fixations
    if sum(gou) > 0
        disp(['t.102, 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: ',num2str(sum(gou)/1.e6,3),...
            ' MJ'],'fontsize',15);hold on
    end
end
end
```

Table 62: Matlab© function fem_til.m – solution of linear transient equations

Function *fem_tir.m* – solution of non linear transient equations

```
function[tca]  = ...
    fem_tir(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(['t. 04, Initial temp.: ',num2str(tmi) ,' K'])
            disp(['t. 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(['t. 08, Time step    : ',num2str(dti),' s'])
tt    = dti*nit;                                % Analyzed period in seconds
      disp(['t. 10, 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(['t. 16, Spec. capac. : ',num2str(Cp),' J/(kg.K)']);
ro    = 2500 ;disp(['t. 17, Spec. mass   : ',num2str(ro),' kg.m-3'])
C     = zeros(dK-1,dK-1);   % Initialization of the global capacity matrix
       if deb==1;disp('t. 19, 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(['t. 28, sum(sum(C))  : ',num2str(sum(sum(C))*1e-6), ' MJ/K'])
disp(['t. 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(['t. 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(['t. 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(['t. 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(['t. 45, N. rad. edges: ',num2str(nu)]);
    disp(['t. 46, rad seg leng.: ',num2str(Lel'),' m']);% fT(2)=280;nfi=2;
    disp(['t. 47, N. rad. elem.: ',num2str(size(lon,2))])
    disp(['t. 48, rad sid leng.: ',num2str(lon),' m'])
    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(['t. 63, sum(gr)      : ',num2str(sum(gr),3),' W'])
        if deb==1
            disp( 't. 65, Call function: ..... fem_Kra .....')
            if size(Qs,2) < 15
                disp(['t. 67, gr           : ',num2str(gr(lcont),3),' W'])
                disp(['t. 68, lcont        : ',num2str(lcont')])
                disp(['t. 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(['t. 75, sum(Mn)       : ',num2str(sum(Mn),3),' W'])
    end
end
ip = 0.;
if deb ==1;disp( 't. 95, Call function: ..... fem_Kra .....');end
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(['t 120, Iteration    : ',num2str(it),' gt: ',num2str(gt),' K'])
        if deb==1;disp ('t 127, 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(['t.139 : ',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(['t.142, Tmean - Tini : ',num2str(ddt,3),' K'])
      disp(['t.143, Stored heat  : ',num2str(ah ,3),' MJ'])
if ddt > .1
    disp(['t.145, Min it+1 temp: ',num2str(tsmin(it+1),4),' K'])
    disp(['t.146, Max it+1 temp: ',num2str(tsmax(it+1),3),' K'])
    if deb==1;disp( 't.148, Call function: ..... gra_tev .....');end
    gra_tev(nit,tt,re,tsmax,tsmin,tmoy);           % Temperature evolution
end
if nfi > 0       % Results shown & displayed only in presence of fixations
    if sum(gou) > 0
```

```
153             disp(['t.152, Ejected heat : ',num2str(sum(gou)*1.e-6,3),' MJ'])
154         end
155     end
156     end
```

Table 63: Matlab© function fem_tir.m – solution of non linear transient equations

8.3 Postprocessing functions

After running the procedure *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 66*)

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

2. Temperature gradient element by element (see *Table 67*)

```
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 67*), 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 68*)

```
xy=xyz;T=tca;
figure              % Drawing the element heat flows in the meshed domain
gra_ahf(xy ,lK,T,co); gra_mel(xy ,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)]);hold on
end                   % End drawing element heat flows and domain border
```

4. Node and element labels (see *Table 66*)

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

5. Displaying nodal temperatures (see *Table 65*)

```
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 65*)

```
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 69*)

```
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 70*)

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

10. Drawing isotherms after running *Fiammetta_v01.m*

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

11. Drawing the boundaries of the domain after running *Fiammetta_v01.m*

```
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 70*)

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

Table 64: Postprocessing functions

The postprocessing functions of *Table 64* are listed in *Table 65* to *Table 70*.

Function Matlab© *gra_mel.m* - drawing a shrunk mesh

```
function [] = gra_mel(xyz,lK,dn,sh)              % Drawing the shrunk mesh
% dn = 1 : conductive elements, dn = 2 : convective elements
% 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
        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);
    if dn == 1;plot((1-sh)*ce(1)+sh*X,(1-sh)*ce(2)+sh*Y,  'k');hold on;end
    if dn == 2;plot((1-sh)*ce(1)+sh*X,(1-sh)*ce(2)+sh*Y,'--k');hold on;end
    if dn == 1;text(ce(1),ce(2),num2str(j),'Color','r');hold on;end
%     if dn == 2;text(ce(1),ce(2),num2str(j),'Color','m');hold on;end
end
end
```

Table 65: 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)        % Display node and elements labels
figure;gra_mel(xyz,lK,1,.9)    % 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));hold on;end
end
```

Table 66: Matlab© function gra_mnl.m - displaying node & element labels

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);
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;
    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;... % Jacobian 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 gradient at 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;
gm    = [max(sqrt(u.*u+v.*v)) mean(sqrt(u.*u+v.*v))];% grad: max & average
title(['Temperature gradient, max: ',num2str(gm(1),2),', mean: ',...
    num2str(gm(2),2),' K/m'],'fontsize',15);axis off;hold on
disp(['Temp. grad. maximum : ', num2str(gm(1),3),', mean: ',...
    num2str(gm(2),3),' K/m'])
end
```

Table 67: Matlab© function gra_atg.m - visualization of temperature gradient arrows

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

```
function [] = 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)
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(['h  23, Max heat flow: ', num2str(gm(1),2),', mean: ',...
    num2str(mean(sqrt(u.*u+v.*v)),2),' W/m2'])
end
```

Table 68: Matlab© function gra_ahf.m - visualization of heat flow arrows

Matlab© function *gra_ipa.m* - drawing isotherm lines in a meshed 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 69: Matlab© function gra_ipa.m - drawing isotherms in a meshed 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 70: Matlab© function gra_lin.m - visualization of the levels of a scalar function

8.4 Illustrative input data

In this section, we collect representative input data used in the examples. These data are organized in section including an identification number *Gi* and the name of the concerned problem.

CAD data inserted at the start of *Fiammetta.m* with the function *cad_gin.m*

```
function[xyz_cao,car_cao,nbo,Me,Di,Ne,Co,nvn,nnr] = cad_gin(Gi)
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;
    disp([ 'Standard cavity,  Gi: ',num2str(Gi)])
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;
    disp([ 'Standard cavity,  Gi: ',num2str(Gi)])
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)])
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;
    disp([ 'Standard cavity,  Gi: ',num2str(Gi)])
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)])
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([ 'Standard trapezoidal shape, Gi:  ',num2str(Gi)])
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(['Standard rect. 2 squares, Gi: ',num2str(Gi)])
end
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)])
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)])
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 = 9;Ne = 0;Co = 4;nvn = 1;nnr = 1;
    disp(['Standard rect. 2 squares, Gi: ',num2str(Gi)])
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;
```

```
76      Me = 5;Di = 0;Ne = 1;Co = 0;nvn = 0;nnr = 0;
77      disp([ 'Standard cavity,  Gi: ',num2str(Gi)])
78  end
79  if Gi == 12
80  % ....... 12. Rectangular cavity ......................................
81      xyz_cao   = [0 0;3 0;3 6;0 6;1 1;2 1;2 5;1 5];
82      car_cao   = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
83      Me = 5;Di = 8;Ne = 0;Co = 0;nvn = 0;nnr = 0;
84      disp([ 'Rectangular cavity,  Gi: ',num2str(Gi)])
85      % cs : 0 = noth., 1 = cavity: 2 = str.
86  end
87  if Gi == 13
88  % ....... 12. Rectangular cavity ......................................
89      xyz_cao   = [0 0;3 0;3 6;0 6;1 1;2 1;2 5;1 5];
90      car_cao   = [6 5 1 2;6 2 3 7;7 3 4 8;1 5 8 4];nbo = 12;
91      Me = 3;Di = 8;Ne = 0;Co = 0;nvn = 0;nnr = 0;
92      disp([ 'Rectangular cavity,  Gi: ',num2str(Gi)])
93  end
94  if Gi == 14
95  % ....... 11. Street section       ....................................
96      xyz_cao  = [0 8;1 8;0 0;1 1;5 0;4 1;5 8;4 8];nbo = 10;
97      car_cao  = [2 1 3 4;4 3 5 6; 6 5 7 8];
98      Me = 5;Di = 0;Ne = 11;Co = 0;nvn = 0;nnr = 0;
99  disp(['Street section,  Gi: ',num2str(Gi)])
100 end
101 if Gi == 15
102 % ....... 11. Street section       ....................................
103     xyz_cao  = [0 8;1 8;0 0;1 1;5 0;4 1;5 8;4 8];nbo = 10;
104     car_cao  = [2 1 3 4;4 3 5 6; 6 5 7 8];
105     Me = 3;Di = 0;Ne = 11;Co = 0;nvn = 0;nnr = 0;
106 disp(['Street section,  Gi: ',num2str(Gi)])
107 end
108 if Gi == 16
109 % ....... 16. Thermal bridge ......................................
110     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];
111     car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
112     Me = 2;Di = 2;Ne = 0;Co = 5;nvn = 1;nnr = 1;
113     disp([ 'Thermal bridge, Gi: ',num2str(Gi)])
114 end
115 if Gi == 17
116 % ...... 17. Horizontal rectangle ....................................
117     ha = 1;xyz_cao = [0 0;2 0;2 ha;0 ha];
118            car_cao = [1 2 3 4 ];nbo=4;
119     Me = 3;Di = 0;Ne = 0;Co = 0;nvn = 0;nnr = 0;
120     disp([ 'Horizon. 1 rect., Gi: ',num2str(Gi)])
121 end
122 if Gi == 18
123 % ....... 18. Thermal bridge ......................................
124     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];
125     car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
126     Me = 1;Di = 11;Ne = 0;Co = 5;nvn = 2;nnr = 0;
127     disp([ 'Thermal bridge, Gi: ',num2str(Gi)])
128 end
129 if Gi == 19
130 % ....... 19. Thermal bridge ......................................
131     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];
132     car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
133     Me = 2;Di =10;Ne = 0;Co = 5;nvn = 1;nnr = 2;
134     disp([ 'Thermal bridge, Gi: ',num2str(Gi)])
135 end
136 if Gi == 20
137 % ....... 19. Thermal bridge ......................................
138     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];
139     car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
140     Me = 3;Di =7;Ne = 0;Co = 5;nvn = 2;nnr = 0;
141     disp([ 'Thermal bridge, Gi: ',num2str(Gi)])
142 end
143 if Gi == 21
144 % ....... 16. Thermal bridge ......................................
145     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];
146     car_cao = [1 3 4 2;3 5 6 4;5 7 8 6;9 10 5 3;4 6 12 11];nbo = 16;
147     Me = 4;Di = 2;Ne = 0;Co = 5;nvn = 1;nnr = 1;
148     disp([ 'Thermal bridge, Gi: ',num2str(Gi)])
149 end
150 if Gi == 22
151 % ....... 16. Thermal bridge ......................................
152     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 = 5;nvn = 1;nnr = 0;
    disp([ 'Thermal bridge, Gi: ',num2str(Gi)])
end
end
```

Table 71: Matlab© function cad_gin.m: input data - definition of domains

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;
end
if Di == 1
    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;
end
if Di == 2
    fT =[280 300];lfi=[no+1 no+2];nf=2;   % Fixation of two virtual nodes
    disp(['LD 15, Fixed nodes  : ',num2str(lfi)])
    disp(['LD 16, Fix. temper. : ',num2str(fT),' K']);
end
if Di == 3
    nnc =  5;    % nnc = the 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];
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
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)];
    nf = size(lfi,2);
    if nf > 2 ;fT = [ones(1,nf/2)*300 ones(1,nf/2)*310 ];end
end
if Di == 7
    fT =[300 280];lfi=[no+1 no+2];nf=2;        % Fix. of both virt. nodes
end
if Di == 8
    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)*300;  % Dirichlet boundary conditions
    disp(['LD 50, N. fix. nodes: ',num2str(nf)])
    disp(['LD 51, av. imp. temp: ',num2str(mean(fT)),' K'])
end
if Di == 9
    fT =[280 300];lfi=[no+1 no+2];nf=2;           % Fix. of 2 virt. nodes
    disp(['LD 53, Fixed nodes  : ',num2str(lfi)])
    disp(['LD 54, Fix. temper. : ',num2str(fT),' K']);
end
if Di == 10
    fT =[270 270 300];lfi=[no+1 no+2 no+3];nf=3;   % Fix. of 3 virt. nodes
    disp(['LD 58, Fixed nodes  : ',num2str(lfi)])
    disp(['LD 59, Fix. temper. : ',num2str(fT),' K']);
end
if Di == 11
    fT =[300 280];lfi=[no+1 no+2];nf=2;    % Fixation of two virtual nodes
    disp(['LD 63, Fixed nodes  : ',num2str(lfi)])
    disp(['LD 64, Fix. temper. : ',num2str(fT),' K']);
end
```

```
68  if Di == 12
69      fT =300;lfi=no+1 ;nf=1;                % Fixation of one virtual nodes
70      disp(['LD 68, Fixed nodes  : ',num2str(lfi)])
71      disp(['LD 69, Fix. temper. : ',num2str(fT),' K']);
72  end
73  if Di == 13
74      fT =[300 270];lfi=[no+1 no+2];nf=2;    % Fixation of two virtual nodes
75      disp(['LD 73, Fixed nodes  : ',num2str(lfi)])
76      disp(['LD 74, Fix. temper. : ',num2str(fT),' K']);
77  end
78  % if nfi < 7;disp(['L  73, fix. top side: ',num2str(lfi)]);end
79  % bc = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
80  %       [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
81  %       [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
82  %       [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)]];
83  % disp(['L  78, n.fix. cavity: ',num2str(size(bc))])
84  % fT      = zeros(1,no); lfi = zeros(1,no);
85  if Di > 13
86  if nvn == 0                    % lfi = uni-line matrix, fT = uni-line matrix
87      lfi = [[car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)]';
88              [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)]']';
89      nf = size(lfi,2)/2;fT=[ones(nf,1)*270 ;ones(nf,1)*300];
90  %     mfi = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
91  end
92  if nvn == 1
93  %     lfi = [car_cao(1,3) bor(pbo(1,3),5):bor(pbo(1,3),6) car_cao(1,4)]';
94  %     nfi=size(lfi,2);fT=ones(1,nfi)*280;
95  % % DOF of the 1. Standard cavity
96  % bc = [[car_cao(1,1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
97  %       [car_cao(2,4) bor(pbo(2,4),5):bor(pbo(2,4),6) car_cao(2,1)];
98  %       [car_cao(3,4) bor(pbo(3,4),5):bor(pbo(3,4),6) car_cao(3,1)];
99  %       [car_cao(4,2) bor(pbo(4,2),5):bor(pbo(4,2),6) car_cao(4,3)]];
100
101 end
102 % if nvn == 2               % Dirichlet boundary conditions for convection
103 %     fT =[270 300];lfi=[no+1 no+2];         % Fixation of 2 virtual nodes
104 % end
105 if nvn == 3
106     fT=[270 285 300];lfi=[no+1 no+2 no+3];   % Fixation of 3 virtual nodes
107 end
108 nfi = size(lfi,2);disp(['LD106, N. fix. nodes: ',num2str(nfi)])
109 if nfi > 0
110     if nfi < 15
111         disp(['LD109, Numb. fix. N.: ',num2str(nf)])
112         disp(['LD110, Fixed nodes  : ',num2str(lfi)])
113         disp(['LD111, Fix. temper. : ',num2str(fT),' K']);
114     end
115 end
116 end
117 end
```

Table 72: Dirichlet boundary conditions, function cad_Dir.m

Neumann boundary conditions inserted with the function *cad_Neu.m*

```
1   function [gh,bos] = cad_Neu(Ne,dK,car_cao,pbo,bor,th,xyz_cao)
2   %..................................... Neumann boundary conditions
3   if Ne == 1
4       gh = zeros(dK,1);                      % second member initialization
5       lg = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
6       nh = size(lg,2);np=nh-1;g(1)=1/(2*np);g(nh)=g(1);g(2:nh-1)=1/np;
7       gh(lg(1:nh)) = g(1:nh)/sum(g);
8       bos = (xyz_cao(2,1)-xyz_cao(1,1))*th;
9       disp(['N   9, Loaded  area : ',num2str(bos),' m2'])
10  end
11  if Ne == 11
12      gh = zeros(dK,1);
13      lg  = [car_cao(3, 3) bor(pbo(3,3),5):bor(pbo(3,3),6) car_cao(3,4)...
14             car_cao(1, 1) bor(pbo(1,1),5):bor(pbo(1,1),6) car_cao(1,2)];
15      nh  = size(lg,2)/2;                    % Loaded nodes lg for CAD model 11
16  %     lg = [car_cao(3,2) bor(pbo(3,2),5):bor(pbo(3,2),6) car_cao(3,3)];
17  %     nh = size(lg,2);
18      for i             = 1:nh;   gh(lg(1,i),1) = 2; end % Weights right side
19      gh(lg(1,1),1)     = 1   ; gh(lg(1,nh),1)   = 1;
20      gh(lg(1:nh))      = gh(lg(1:nh))/sum(gh(lg(1:nh)));
21      for i             = nh+1:2*nh;gh(lg(1,i),1) = 2; end% Weights left side
22      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;
    bos = (xyz_cao(2,1)-xyz_cao(1,1)+xyz_cao(7,1)-xyz_cao(8,1))*th;
    disp(['N  26, N. load. nod.: ',num2str(size(lg,2))])
end
if Ne == 0
    gh  = zeros(dK,1);bos=0;   % 2d member initialization always necessary
else
    if nh < 10
        disp(['N  32, Loaded nodes : ',num2str(lg)])
        disp(['N  33, Loads weights: ',num2str(gh(lg)')])
    end
end
end
```

Table 73: Data - Neumann boundary conditions, function cad_Neu.m

Convection boundary conditions to be inserted in *Fiammetta.m*

```
if nnv == 1 %.......................... Convection towards 1 virtual nodes
    % Convection on the right side
    bd = [[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)]];
    lc = zeros(nci,3);nco = 0;     % Generation of the convective elements
    C(ndK,ndK)        = 1;            % Capacity of the fluid virtual node
    for     ic        = 1:size(bd,1)              % Loop on the conv edges
        for ie        = 1:nci         % Loop on the nci elements of a side
            nco       = nco+1;
            lc(nco,1) = bd(ic,ie);
            lc(nco,2) = bd(ic,ie+1);
            lc(nco,3) = ndK;
        end
    end
    disp(['L 144, Num. conv el.: ',num2str(nco)])           % Stored in lc
end     %.......................... End convection towards 1 virtual nodes

if nnv == 2 %.............. Convection on both sides with 2 virtual nodes
    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)]];
    nec  = (nci)*size(bt,1);          % Taking into account the conv elem.
    ncc  = size(bt,1);he = ones(1,nec)*h;
    hv   = [0 0 h/2 h 0 0 h h/2 0 0]; % hv = ones(1,ncc)*h;%
           disp(['L 160, Conv. c. p.s.: ',num2str(hv),'  W/(m2K)'])
    lc = zeros(nec,3);nco = 0;     % Generation of the convective elements
    C(ndK,ndK)        = 1;            % Capacity of the fluid virtual node
    for     ic        = 1:size(bt,1)              % Loop on the conv edges
        for ie        = 1:nci         % Loop on the nci elements of a side
            nco       = nco+1;
            lc(nco,1) = bt(ic,ie);
            lc(nco,2) = bt(ic,ie+1);
            lc(nco,3) = ndK;
            he (1,nco)= hv(ic);
        end
    end
    for i = 1:nec/2
        lc(i,3)=ndK-1;lc(nec/2+i,3) = ndK;
    end
    disp(['L 149, Num. conv el.: ',num2str(nco)])           % Stored in lc
end  % End option nnv = 2: 2 sides of the domain linked to 2 virtual nodes
if nnv      > 2 %..................... Convection towards nnv virtual nodes
    lc      = zeros(nec*nnv,3);
    caoet   = [car_cao car_cao(:,1)];  % patch + first node (N: 1 2 3 4 1)
    for ic  = 1 : nnv    % Vector nc contains the list of convective sides
        nc  = [2 4]; % Left & right sides are convective % nc  = [1 2 3 4]
        li  = [caoet(1,nc(1,ic)) bor(pbo(1,nc(ic)),5):...
               bor(pbo(1,nc(ic)),6) caoet(1,nc(ic)+1)];
```

```
165          nec = size(li,2)-1;
166          disp(['L 135,Num. conv. elem.: ',num2str(nec)])
167          for i=1:nec
168              lc(i+(ic-1)*nec,1) = li(i);
169              lc(i+(ic-1)*nec,2) = li(i+1);
170              lc(i+(ic-1)*nec,3) = no+ic;
171          end
172          lic((ic-1)*nec+1:ic*nec)=li(1:nec); % lic = list conv. sides nodes
173      end
174      if size(lic,2)< 12;disp(['L 167, Conv. nodes   : ',num2str(lic)]);end
175  %     lic  = [car_cao(2,2) bor(pbo(2,2),5):bor(pbo(2,2),6) car_cao(2,3)];
176      nnc = nnv*(nci);     % nec = number of nodes connected for convection
177      disp(['L 150, Num. conv. elem.: ',num2str(nnc)])
178  end
```

Table 74: Data - Convection boundary conditions: nnv = 1, 2, nnv > 2

8.6 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 (1.43) and (1.58) depend on the size *L* of these elements.

As a convective element, a radiative element may have any orientation. Its length is *L*, its thickness *e*, and the Stefan-Boltzmann coefficient 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 75*) computes the radiation matrices of an element as pseudo convection matrices. The third argument of the function is the product of the Stefan-Boltzmann constant and the thickness. The temperatures are stored in the vector *tca* (argument 4). The *xyz* matrix contains the node coordinates and *lc* is the localization matrix of the element.

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

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

Table 75: Matlab© function fem_Kcr.m, radiative - conductive element matrix

Because radiative boundary conditions lead to a nonlinear system of equation, a new Matlab© function *fem_snl.m* is needed to solve the system (*Table 76*).

Matlab© function *fem_snl.m* - solution of the nonlinear radiative system

```
1  function [tca] = fem_snl(Kk,gh,lfi,fT,xyz,lK,lcont,nnr,nnv,nci,np,SB)
2                          disp(['Ls  2, list f. nodes: ',num2str(lfi)])
3                          disp(['Ls  3, Fix. temper. : ',num2str(fT),' K'])
4  nfn   = size(lfi,2);    disp(['Ls  4, N. fixed nod.: ',num2str(nfn)])
5  ner   = size(lcont,1)-1;disp(['Ls  5, N. rad elem. : ',num2str(ner)])
6  dK    = size(Kk,1);                  % Kk is the pure conductivity matrix
7  no    = dK - nnr - nnv;       % Number of unknowns of the system to solved
8  % lic      = zeros(nnr,round(sqrt(no)));
9  % for ii=1:nnr          % Loop on the sides involving radiation conditions
10 %    j         = ii+2;if j==5;j=1;end
11 %    lic(ii,:) = [car_cao(1,ii+1) bor(pbo(1,ii+1),5):bor(pbo(1,ii+1),6)...
12 %        car_cao(1,j)];     % List of the DOF of the irradiated patch side
13 % end
14 lc       = zeros(ner,3);   % Localization matrix of the radiative elements
15 K        = Kk;                                % Pure conductivity matrix
16 tcant    = [ones(1,no)*fT(1) fT];            % Initial temperature field
17 for iter = 1 : 2 ;disp(['LS 17, Iteration N. : ',num2str(iter),' / 2'])
18     for n = 1:nnr              % Generation of cond. rad. element matrices
19         for i       = 1:ner
20             lc(i,1) = lcont(i,1); lc(i,2) = lcont(i+1,1); lc(i,3) = no+n;
21             Ker     = fem_Kcr(xyz,lc(i,:),SB,tcant); % Elem cond rad matr.
22             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
    nf    = size(lfi,2); N = zeros(nf,dK);   % Linear const. 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(['Ls 32, Dissipation  : ',num2str(-B'*gh,3),' WK'])
    disp(['Ls 33, React. flows : ',num2str(B(dK+1:dK+nfn)',3),' W'])
    gt    = [min(tca),round((max(tca)-min(tca))/2)/10,max(tca)];
    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,xyz,gt);
        colorbar;hold on
    end
    title(['Iteration: ',num2str(iter),', dissipation: ',...
        num2str(tca'*K*tca,3),' WK']);axis off
    disp(['Ls 42, Dissipation  : ',num2str(tca'*K*tca,3),' WK'])
    disp(['Ls 43, 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 76: Matlab© function fem_snl.m - solution of the nonlinear radiative system

The *fem_snl.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.

8.7 List of Matlab© procedures

Fiammetta_20210929.m	*Table 58*
Fiammetta_33_20210830.m	*Table 1*
cad_Dir.m	*Table 17*
cad_Neu.m	*Table 18*
cad_ban.m	*Table 26*
cad_bou.m	*Table 9*
cad_con.m	*Table 19*
cad_edg.m	*Table 16*
cad_gin.m	*Table 71*
cad_mes.m	*Table 15*
fem_Cae.m	*Table 20*
fem_Kco.m	*Table 11*
fem_Kcr.m	*Table 75*
fem_Kcv.m	*Table 6*
fem_Kra.m	*Table 57*
fem_caK.m	*Table 51*
fem_rsm.m	*Table 52*
fem_snl.m	*Table 76*
fem_til.m	*Table 62*
fem_tir.m	*Table 63*
fem_tit.m	*Table 60*
fem_tra.m	*Table 25*
geo_baf.m	*Table 42*
geo_stf.m	*Table 38*
geo_vfr.m	*Table 29*
geo_vfc.m	*Table 27*
gra_Hfl.m	
gra_Tgr.m	

gra_ahf.m	*Table 68*
gra_atg.m	*Table 67*
gra_cob.m	*Table 2*
gra_hie.m	*Table 23*
gra_ipa.m	*Table 69*
gra_ist.m	*Table 3*
gra_lin.m	*Table 70*
gra_mel.m	*Table 65*
gra_mnl.m	*Table 66*
gra_tev.m	*Table 22*
mat_cok.m	*Table 8*

Table 77: Matlab© procedures and functions related to Fiammetta

8.8 Exercises proposed in 2020

8.8.1 Steady State Conduction

Exercise 1: Playing with boundary conditions

Impose fluxes on the four sides, and a temperature on a single node. What happens if the fluxes are balanced (what comes in from the left and above is equal, respectively, to what comes 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.

8.8.2 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.

8.8.3 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.

8.8.4 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.

8.8.5 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?

8.9 Exercises proposed in 2021

8.9.1 Steady state conduction

Exercise 1: Nodes, elements and Dirichlet boundary conditions

Using an explicit definition of the nodal coordinates, the mesh and the fixations, formulate a problem on a domain composed of more than 5 elements.

8.9.2 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.

8.9.3 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.

8.9.4 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.

8.9.5 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 B. Masset L. & Beckers P., "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 2013] Beckers Benoit, "Taking Advantage of Low Radiative Coupling in 3D Urban Models", Eurographics Workshop on Urban Data Modelling and Visualisation, 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 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/downloads/Beckers%2020191213%20-%20Tutorials%20Heat%20Transfer%20Lectures%20in%20Montevideo

[Beckers 2020a] Benoit Beckers, "Angle solide et Facteur de vue", *Architecture et Physique Urbaine, ISA BTP, Université de Pau et des Pays de l'Adour,* 2016, (première version novembre 2011, mises à jour en 2016 et en **2020** dans "Beckers 20200815 - Angle solide et Facteur de vue", 14 pages, 8 tableaux, 12 figures)

[Beckers 2020b] Beckers Benoit, "Rayonnement dans une cavité", 18 pages, Rapport interne, **2020**0815.

[Beckers 2020c] Beckers Benoit, "Rayonnement dans un espace fermé", 48 pages, Rapport interne , 20200815. Beckers 20201226 Radiosité en 2D dans une enceinte fermée, **2020**

[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 non linear 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 Analyser 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 W., 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", inSolar 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