Five Lectures on Finite Element Method Applied to Heat Transfer

– Tutorials –

Benoit Beckers
November 2018
Urban Physics Joint Laboratory
Université de Pau et des Pays de l'Adour (France)

The five 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 last lecture describes the isoparametric elements, which make it possible to generalize to any geometry what has been described here on the simplest shape: a rectangle meshed by squares.

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, in particular, 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 only very short programs, written in Matlab©, which are progressively developed as conduction, convection, radiation and transient regime are introduced. Each step is accompanied by an exercise that will allow the reader becoming familiar with the main notions of the lectures.

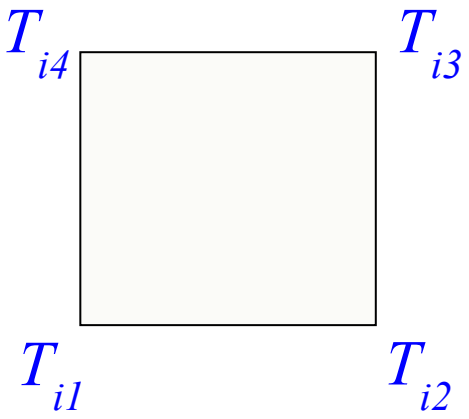
Tutorial I: Conductive Heat Transfer

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 the vertices four nodal temperatures T_{ij}.

$$\tau_i = \sum_{j=1}^{4} T_{ij} f_{ij}(x,y) \tag{1.1}$$

This definition allows computing the conduction matrix of a square (*Figure 1*). This matrix does not depend on the size of the square; it only depends on the conductivity coefficient k and the

thickness *e*. In the same *Figure 1*, we also show 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 all the finite elements of the domain are added to build the discretized global functional $I(T)$:

$$< I(T) = \sum_{i=1}^{nel} \left(\int_{\Omega j} \frac{1}{2} k_i \ (grad \sum_{j=1}^{4} T_{ij} f_{ij})^T . grad \sum_{j=1}^{4} T_{ij} f_{ij} \ d\Omega_i + \int_{S_{2i}} \bar{q}_n \ \sum_{j=1}^{4} T_{ij} f_{ij} \ dS_i \right) > \qquad (1.2)$$

After introducing the polynomial trial functions given in (1.1), we can write (1.2) in matrix form:

$$< I(T) = \sum_{i=1}^{nel} [T_i]^T [K_i][T_i] + [T_i]^T [F_i] > \qquad (1.3)$$

The last term of (1.3) $[F_i]$ is the vector of heat loads. In the next step, we have to express the continuity of the temperature field across the whole domain. For this purpose, at each interface between two elements, it must be stated that the nodal temperatures are identical, which means that the nodal temperatures of the elements sharing a same node are the same. In the mesh of *Figure 2*, the second node of element 3, the first of element 4, the third of element 5 and the fourth of element 6 are assigned to the global node 8. For each element, we can then write the relation between local $[T_i]$ and global nodes $[T]$.

$$[T_i] = [L_i][T] \qquad (1.4)$$

For instance, the matrix $[L_4]$ of the element number 4 is:

$$L_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (1.5)$$

To assemble the conductivity matrix of element 4 into the global one, we need to perform the following product:

$$[L_4]^T [K_4][L_4] \tag{1.6}$$

The coefficients of the conductivity matrix of the element number 4 are located at positions 8, 9, 6 and 5 of the global conductivity matrix.

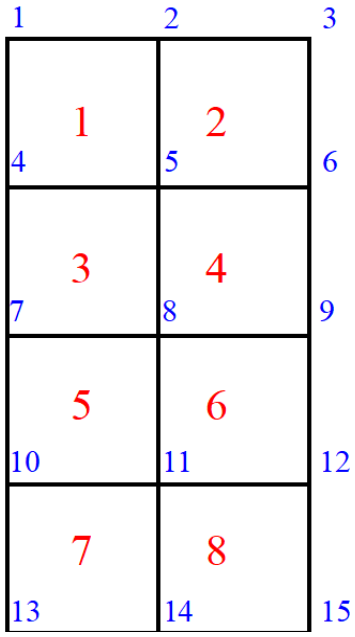


Figure 2: Numbering and sequence of nodes and elements

As for the element number 4 of the domain, the nodal temperature vectors of the elements are linked to the global vector of the domain temperatures by localization matrices $[L_i]$. We can therefore perform the summation, ensuring the continuity of the field by identification with the nodes of the domain.

$$I(T) = \sum_{i=1}^{nel} \left([T]^T [L_i]^T [K_i][L_i][T] + [T]^T [L_i]^T [F_i] \right) \tag{1.7}$$

The next step is to get the vector $[T]$ out of the sum in (1.7).

$$I(T) = [T]^T \left(\sum_{i=1}^{nel} \left([L_i]^T [K_i][L_i][T] + [L_i]^T [F_i] \right) \right) \tag{1.8}$$

By canceling the first derivatives of this quadratic function with respect to the parameters $[T]$, we obtain the linear system:

$$\begin{aligned} &\sum_{i=1}^{nel} \left([L_i]^T [K_i][L_i] \right) [T] = -\sum_{j=1}^{nel} [L_i]^T [F_i] \\ &with \ [K] = \sum_{i=1}^{nel} [L_i]^T [K_i](L_i) \quad \text{and} \quad [F] = -\sum_{i=1}^{nel} [L_i]^T [F_i] \end{aligned} \tag{1.9}$$

The matrix $[K]$ is the global conductivity matrix.

$$[K][T] = [F] \tag{1.10}$$

Procedures used to solve conduction heat transfers

To perform tests on conduction, we use the procedure of *Table 1*. The finite element model is restricted to a rectangle only composed of squares, two times more in the y direction (*ny*) than in the x direction (*nx*). The thickness *th* can be specified. The temperatures are always expressed in Kelvin (*K*). The conductivity coefficients are specified in the Matlab© function *conde.m*. It is possible to define any distribution of conductivities, but one per element in the vector *co* (*nel* components with *nel*, the number of elements).

To test the procedure, we impose temperatures on the top and bottom horizontal sides. If the difference of temperatures is equal to 50 *K*, the quantities of incoming heat on the top side and the outgoing one in the base are identical and given by:

$$k\frac{\Delta T}{\Delta y} = k\frac{50}{2} = 25\ W \tag{1.11}$$

Matlab procedure *pp_ conduction.m*

```
nx    = 20;ny = nx*2;nel = nx*ny;no = (nx+1)*(ny+1);th = 1;          % Mesh
co    = conde(nx,ny);
Kel   = th/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];    % element K
lK    = loca(nx,ny);K = zeros(no,no);
for n = 1:nel;for i=1:4;for j=1:4   % Assembling nel conduct. matrices Kel
          K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+co(n)*Kel(i,j);end;end;end
% % Init ftv
% tb    = 270; tt = tb+50;gap=3.;
% nb    = nx+1;nu =no-2*nb;na = nb*2;                                % Imp. T
% % nb    = max(1,round(nx/2));if nb>(nx+1);nb=nx+1;end;nu=no-2*nb;
% K21   = K(nb+1:nu+nb , 1     : nb);K22 = K(nb+1:nu+nb , nb+1    :nu+nb);
% K23   = K(nb+1:nu+nb , nu+nb+1 : nu+nb*2);ar=ones(nb,1);na=nb*2;
% tca   = [ar*tt;K22\(-K23*ar*tb-K21*ar*tt);ar*tb];   % Sol. of the system
% % End ftv
% Init ihf
sm    = zeros(no-nx-1,1);gap=1;qw=25;na=nx+1;
for i = nx+2:nx+1:no-2*nx-1;sm(i)=qw/ny;end;sm(1)=qw/(2*ny);tb = 273;
tca   = [K(1:no-nx-1,1:no-nx-1)\(sm-K(1:no-nx-1,no-nx:no)*tb*...
    ones(nx+1,1));tb*ones(nx+1,1)];
% End ihf
grisb (nx,ny,tca,gap);axis off;                          % Output 1: isot
figure('Position',[10 50 1200 500]);per=(1:(nx+ny)*2)';   % Output 2: hfpe
gperi = cageco(nx,ny,K*tca);bar(gperi,'k');grid on;
title(['Bottom flow: ',num2str(sum(gperi(1:nx+1)),'%0.3g'),...
    ' W, input flow: '  ,num2str(sum(gperi(nx+2:size(gperi,1))),...
    '%0.3g'),' W'],'fontsize',15)
disp(['Base temperature    : ',num2str(tb,'%0.3g'),' K']) % Output 3: disp
disp(['Max  temperature    : ',num2str(max(tca),'%0.3g'),' K'])
disp(['Mesh size           : ',num2str(nx),' x ',num2str(nx*2)])
disp(['Fix. nod. 2 hor. fa.: ',num2str(na)])
disp(['Diss tcaT*(K*tca)/2 : ',num2str(tca'*(K*tca)/2,'%0.3g'),' WK'])
```

Table 1: Matlab© procedure pp_conduction.m for conduction problems

The procedure of *Table 1* is providing three outputs: two figures and displayed results concerning input and/or output data. They are grouped in the *Figure 3*. The bar diagram is showing the nodal heat quantities. To compute them, we move along the border in the counterclockwise direction. So, we start with the inferior side, from left to right, after, with the right vertical side, the horizontal top side, from right to left and the left vertical side from top to bottom. On the

horizontal sides, each inner node links two element edges but the outer ones connect only one. At the extremities of the side, 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 (1.11): 25 *W* divided by the number of elements 25/*nx W* for inner nodes and half for the others: 25/(2*nx*) *W*.

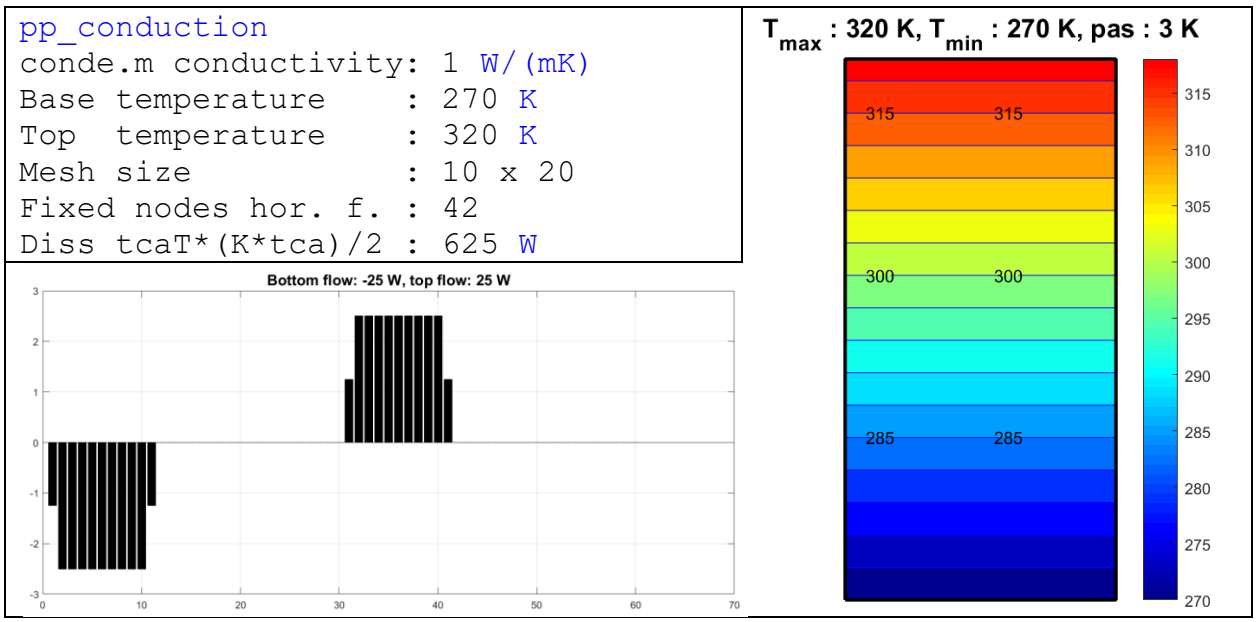


Figure 3: Example with imposed temperatures producing a vertical gradient

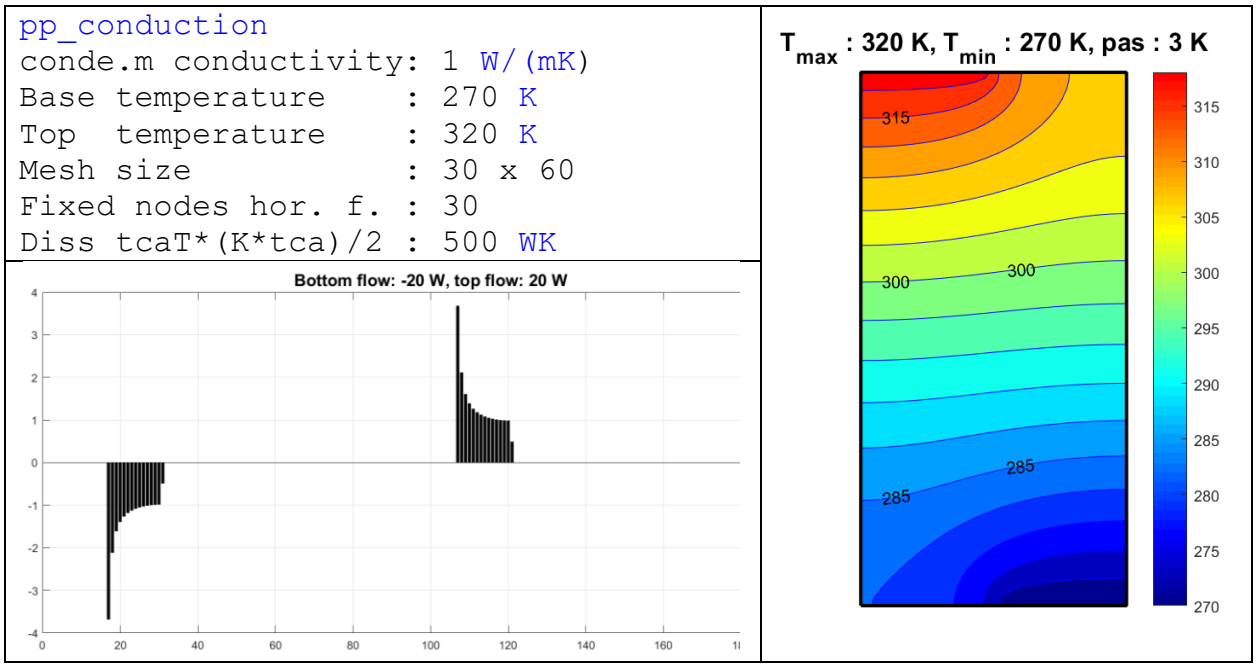


Figure 4: Imposed temperatures on a part of the horizontal faces

We can exchange the following lines in the procedure (*line 8*) to modify the distribution of prescribed temperatures on the horizontal faces. With the second version, we obtain the *Figure 4*.

```
% nl     = nx+1;nu =no-2*nl;                                          % Imp. T
nl     = max(1,round(nx/2));if nl>(nx+1);nl=nx+1;end;nu=no-2*nl; % Imp. T
```

Input data are specified at *lines 1* and *2* of the procedure of *Table 1*: top (*tt*) and bottom (*tb*) temperatures, thickness (*th*) of the domain and size of the mesh (*nx*). The number *(n1)* of nodes with imposed temperatures zone is given at *lines 8* or *9* (one being effective the other, set as comment).

Additional comments about the procedures

1. Principal procedure: *pp_Base_conduction.m*

Line 1 : Data input

The variable *nx* defines the number of elements in the horizontal direction while *ny* is the number of elements in the vertical direction. The other items concern the computation of the number of elements: *nel* and number of nodes: *no*. The variables *th* corresponds to the thickness.

Line 2 : Conductivity coefficients in the elements (function *conde.m, Table 4*)

For a mesh of *nx* x *ny* elements (arguments of the function), we define the vector *co* (output of the function) which contains the values of the coefficients of isotropic conductivity of all the elements. In a non-homogeneous medium, the conductivities may vary from one element to another. The conductivity acts as coefficient in the assembly of the global matrix (*line 6*). The conductivities of the elements are stored in the vector *co* of dimension *nel*.

Line 3 : Conductivity matrix of a square element (*Figure 1*)

The matrix coefficients are written in compact form (a single line). The matrix *Kel* is independent of the position of the element. It can thus be defined either in local or in global coordinates.

Line 4 : compute the localization matrix (function *loca.m*, *Table 5*)

The element nodes sequence shown in ***Figure 1*** is always the same and must be respected during assembling**.** The four nodes of each element are located in the domain mesh. For the first element, line 1 of the localization matrix, we have then the global nodes: 4, 5, 2 and 1, etc...
The formulation using a localization matrix provides a more efficient method than (1.9) and exhibits better computational performances. In Matlab© notation, the localization matrix of the 2 x 4 mesh of *Figure 2*, is:

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

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

Lines 5 – 6 : Global conductivity matrix assembling

The external loop is performed on the elements and the two internal ones 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 *loca.m*. Moreover, the coefficients of the matrices *Kel* are multiplied by the element conductivity coefficient *co (n)* (see *line 2*).

Line 8 - 9 : Data for fixed temperatures

In the proposed examples of *Figure 3* & *Figure 4*, we fix some temperatures on the horizontal sides, starting from opposite corners: left on the top, right in the bottom. The number *nb* of fixed temperatures is less or equal to the number of nodes on a horizontal line: *nx + 1* (checked in the procedure). One of these lines has to be disabled by putting it as a comment. The situation exhibited in *Table 1* corresponds to the example of *Figure 3*.

Lines 11 – 13 : Solution of the system

The solution of a problem including only imposed temperatures is performed as follows: the fixed temperatures are split into two sets of dimensions *nb*, the first in the beginning of the global matrix and the second at the end. The final size of the matrix to be inverted is *nu*. As a consequence, the global matrix is divided into 9 sub matrices. The *nb* imposed temperatures are stored in the vector $[T_3]$ in the bottom and in $[T_1]$ on the top.

$$[K]=\begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \tag{1.12}$$

$$[T_2]=[K_{22}]^{-1}\left(-[K_{23}][T_3]-[K_{21}][T_1]\right) \tag{1.13}$$

Lines 15 – 20 : Sequence corresponding to imposed heat flows

This sequence seen as a comment is presently disabled. It will be enabled in the next tutorial while *lines 7* to *15* will be put as comments.

Lines 21 – 31: Output

The last lines of the procedure are devoted to the output, successively: isotherms (*line 21*, function *grisb.m*), visualization of the nodal heat flows (*lines 22 – 276*, function *cageco.m*) and a summary of data (*lines 27* to *31*).

2. Collection of functions: *cageco.m, conde.m, loca.m, grisb.m, br56.m*

Matlab function *cageco.m*

```
1  function [gperi] = cageco(nx,ny,tca)
2  my        = ny + 1;   % Nodal values along the boundary without repetition
3  ii        = 0;gperi = zeros(2*(ny+nx),1);
4  for i     = ny*(nx+1)+1 : my*(nx+1)
5      ii    = ii+1;gperi(ii) = tca(i);
6  end
```

```
for i      = my*(nx+1)-(nx+1):-(nx+1):nx+1
    ii     = ii+1;gperi(ii) = tca(i);
end
for i      = nx: -1 :1
    ii     = ii+1;gperi(ii) = tca(i);
end
for i      = nx+2:nx+1 :(ny-1)*(nx+1)+1
    ii     = ii+1;gperi(ii) = tca(i);
end
end
```

Table 3: Matlab© function cageco.m for selecting quantity along the border of the domain

Matlab© function *conde.m*

```
function [co] = conde(nx,ny)      % Treatment of non uniform conductivity
k   = 1 ;                                                    % W/(mK)
nel = nx*ny;                                % Number of elements of the mesh
co  = ones(nel,1)*k;                                % Constant conductivity
disp(['conde.m uniform k   : ',num2str(k,'%0.3g'),' W/(mK)'])
end
Se2 % function [co] = conde(nx,ny) % Treatment of non uniform conductivity Hor
    % k   = 1;                                                  % W/(m K)
    % nel = nx*ny;           % Number of element computed from mesh definition
    % fa  = 1000 ;      % Ratio between the 2 conductivities, if 1, k is a cst
    % co  = ones(nel,1)*k;
    % if nx>1;co(nx*ny/2+1:nx*ny/2+nx)  = k*fa;    % 2d k on horizontal band 1
    % if nx>2;co(nx*(ny/2-1)+1:nx*ny/2) = k*fa;end % 2d k on horizontal band 2
    % disp(['Conductivity coeff. : ',num2str(k,'%0.3g'),' W/(m K)'])
    % disp(['Main & bridge cond. : ',num2str([co(1) co(nx*ny/2+1)]),' W/(m K)'])
    % disp(['Rel. strip thickness: ',num2str(2/ny,'%0.3g')])
    % % figure;plot(co(1:nx:nx*(ny-1)+1)')
    % end
Se3 % function [co] = conde(nx,ny) % Treatment of non uniform conductivity Ver
    % k   = 1;                                                  % W/(m K)
    % nel = nx*ny;           % Number of element computed from mesh definition
    % fa  = 0.1;       % Ratio between the 2 conductivities, if 1, k is a cst
    % co  = ones(nel,1)*k;                          % nx must be even ad > 3
    % co(nx/2:nx:nx*ny-nx/2+1)    = k*fa;      % Second k on horizontal band 1
    % co(nx/2+1:nx:nx*ny-nx/2+2)  = k*fa;      % Second k on horizontal band 1
    % disp(['Conductivity coeff. : ',num2str(k,   '%0.3g'),' W/(m K)'])
    % disp(['Cond. coeff. x fact.: ',num2str(k*fa,'%0.3g'),' W/(m K)'])
    % disp(['Rel. strip thickness: ',num2str(2/nx,'%0.3g')])
    % end
Se4 % function [co] = conde(nx,ny)           % Random non uniform conductivity
    % k   = 1;                                                  % W/(mK)
    % nel = nx*ny;                          % Number of elements of the mesh
    % co  = k*(ones(nel,1)+rand(nel,1)*999);         % Random conductivity > 1
    % disp(['conde.m rand k aver : ',num2str(mean(co),'%0.3g'),' W/(mK)'])
    % end
```

Table 4: Matlab© function conde.m for defining non homogeneous conductivity

The above function is subdivided into four sequences. The first one is enabled, the three others being disabled by using the "comment" command of Matlab©. To switch from one sequence to another one, it is necessary to modify the status of the first one into "comment" and to remove the "comment" status of the second one. They correspond to non homogeneous materials. In the second one, a horizontal strip is introduced, in the third one, a vertical strip. The fourth one corresponds to the introduction of random conductivities varying between one and one thousand.

Matlab© function *loca.m*

```
function [lK]=loca(nx,ny)              % Localization matrix with fixed base
nel    = nx*ny;
nf     = nx+1;
lK     = zeros(nel,4);    % Elements are numbered left - right, top - bottom
for j = 1:ny                 % Nodes are numbered left - right, top - bottom
    for i                  = 1:nx
        lK((j-1)*nx+i,1) = j*(nx+1)          + i;
        lK((j-1)*nx+i,2) = lK((j-1)*nx+i,1) + 1;
        lK((j-1)*nx+i,3) = lK((j-1)*nx+i,1) - nx;
        lK((j-1)*nx+i,4) = lK((j-1)*nx+i,1) - nf;
    end
end
end
```

Table 5: Matlab© function loca.m for computing the localization matrix

Matlab function *grisb.m*

```
function [] = grisb(nx,ny,tca,gap)
figure('Position',[1 1 600 512]);
my = ny+1;no=(nx+1)*(ny+1);
B          = ones(my,nx+1)*tca(1);x = zeros(my,nx+1);y = zeros(my,nx+1);
for j      = 1 : nx+1;for i = 1 : ny;x(i,j) = j-1; y(i,j) = my-i;end;end;
ii         = 0;
for i      = 1:ny;for j = 1:nx+1;ii = ii+1; B(i,j) = tca(ii);end;end
x(my,:)    = x(ny,:);y(:,1)     = y(:,2);B(my,:)    = tca(ii+1:no );
br56;colormap(br56);                                  % Color map definition
[CS,H]       = contourf(x,y,B,(0.:gap:max(tca)),'b');hold on;axis equal
   clabel(CS,H,[275 280 285 290 295 300 305 310 315 320]);
plot  ([0 nx nx 0 0],[0 0 ny ny 0],'k','LineWidth',2);hold on;axis equal
   title (['T_m_a_x : ',num2str(round(max(tca))),' K, T_m_i_n : ',...
num2str(round(min(tca))),' K, pas : ',num2str(gap),' K'],'fontsize',15);
end
```

Table 6: Matlab© function grisb.m for drawing isotherms

Matlab function *br56.m*

```
function [bbr] = br56
bbr=[     0          0     0.5625
          0          0     0.6250
          0          0     0.6875
          0          0     0.7500
          0          0     0.8125
          0          0     0.8750
          0          0     0.9375
          0          0     1.0000
          0     0.0625     1.0000
          0     0.1250     1.0000
          0     0.1875     1.0000
          0     0.2500     1.0000
          0     0.3125     1.0000
          0     0.3750     1.0000
          0     0.4375     1.0000
          0     0.5000     1.0000
          0     0.5625     1.0000
          0     0.6250     1.0000
          0     0.6875     1.0000
          0     0.7500     1.0000
          0     0.8125     1.0000
          0     0.8750     1.0000
```

```
    24          0    0.9375    1.0000
    25          0    1.0000    1.0000
    26     0.0625    1.0000    0.9375
    27     0.1250    1.0000    0.8750
    28     0.1875    1.0000    0.8125
    29     0.2500    1.0000    0.7500
    30     0.3125    1.0000    0.6875
    31     0.3750    1.0000    0.6250
    32     0.4375    1.0000    0.5625
    33     0.5000    1.0000    0.5000
    34     0.5625    1.0000    0.4375
    35     0.6250    1.0000    0.3750
    36     0.6875    1.0000    0.3125
    37     0.7500    1.0000    0.2500
    38     0.8125    1.0000    0.1875
    39     0.8750    1.0000    0.1250
    40     0.9375    1.0000    0.0625
    41     1.0000    1.0000         0
    42     1.0000    0.9375         0
    43     1.0000    0.8750         0
    44     1.0000    0.8125         0
    45     1.0000    0.7500         0
    46     1.0000    0.6875         0
    47     1.0000    0.6250         0
    48     1.0000    0.5625         0
    49     1.0000    0.5000         0
    50     1.0000    0.4375         0
    51     1.0000    0.3750         0
    52     1.0000    0.3125         0
    53     1.0000    0.2500         0
    54     1.0000    0.1875         0
    55     1.0000    0.1250         0
    56     1.0000    0.0625         0
    57     1.0000         0         0];
    58 end
```

Table 7: Matlab© function br56.m for defining a color bar

Exercise

Using the Matlab© procedure and the functions presented in *Table 1* to *Table 7*, it is proposed to examine the effects of a modification of the conductivity coefficients. Let us try, for instance, to introduce a thermal bridge by increasing the conductivity along a vertical or an horizontal strip. This modification has to be performed by modifying the function *conde.m*. The elements are numbered from left to right and from top to bottom (*Figure 2*).

Tutorial II: Convective Heat Transfer

Prescribed heat fluxes

Before examining the convection problem, let us go back to the heat conduction problem involving prescribed heat fluxes. We use the functional presented in the first lecture.

$$< I(\tau) = \int_{\Omega} \frac{1}{2} k \ (grad\tau)^T . grad\tau \ d\Omega + \int_{S_2} \bar{q}_n \ \tau dS \ > \quad minimum \tag{1.14}$$

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:

$$\int_{S_{2edge}} \bar{q}_n \tau dS_{el} = g_1 T_1 + g_2 T_2 \tag{1.15}$$

If we express the edge temperature in term of edge weight functions

$$\tau_{edge} = T_1 \left(1 - \frac{x}{l}\right) + T_2 \frac{x}{l} \tag{1.16}$$

We can write the discretized functional:

$$\int_{S_{2edge}} \bar{q}_n \tau dS_{el} = \int_{S_{2edge}} \bar{q}_n \left(T_1 \left(1 - \frac{x}{l}\right) + T_2 \frac{x}{l} \right) dS_{el} \tag{1.17}$$

In matrix form, we have:

$$\text{With : } [T] = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \text{ we have : } \int_{S_{2edge}} \bar{q}_n \tau dS_{el} = \int_{S_{2edge}} \bar{q}_n \begin{bmatrix} \left(1 - \frac{x}{l}\right) & \frac{x}{l} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} dS_{el} \tag{1.18}$$

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

$$\int_{S_{2edge}} \bar{q}_n \tau dS_{el} = \int_{S_{2edge}} \bar{q}_n \begin{bmatrix} \left(1 - \frac{x}{l}\right) & \frac{x}{l} \end{bmatrix} dS_{el} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \tag{1.19}$$

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

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix}^T = \int_{S_{2edge}} \bar{q}_n \begin{bmatrix} \left(1 - \frac{x}{l}\right) & \frac{x}{l} \end{bmatrix} dS_{el} \tag{1.20}$$

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:

$$F_1 = \bar{q} \frac{el}{2} \quad ; \quad F_2 = \bar{q} \frac{el}{2} \tag{1.21}$$

We treat a very simple case with imposed temperatures on the lower face (base) and a constant flow on the left vertical edge. 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 273 *K*. The total flow on the left side is equal to 25 *W*. This example is obtained using the procedure of *Table 1* in which the *lines 7* to *14* are disabled and the *lines 15* to 20 enabled.
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 0.3125 *W*, but is fixed. This flow is therefore removed from the balance in the above graph.

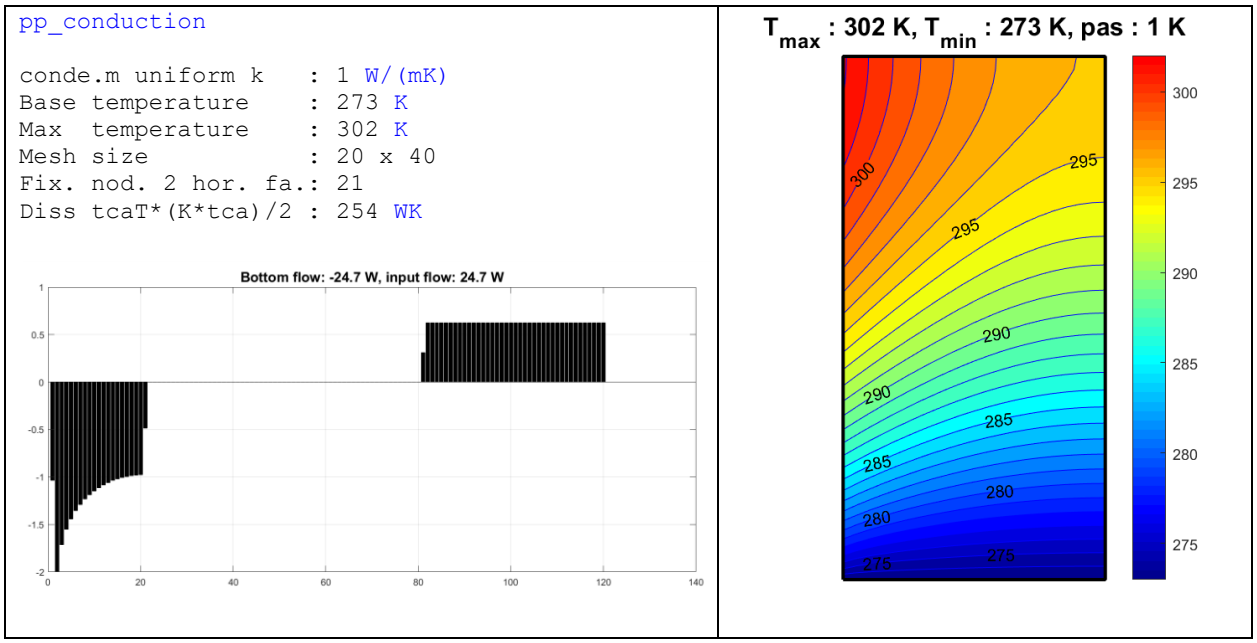


Figure 5: Isocurves for the problem of prescribed heat flows

Convection

We can replace the computation of the partial differential equations of the convective heat transfer problem by the expression of the stationnarity conditions of the functional:

$$< I(\tau) = \int_{\Omega} \frac{1}{2} k \ (grad\tau)^T . grad\tau \ d\Omega + \frac{1}{2} \int_{S_3} h\left(\tau - \tau_f\right)^2 dS + \int_{S_2} \bar{q}_n \ \tau dS \ > \quad minimum \qquad (1.22)$$

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.

$$\text{Functional at element level: } I_{el} = \frac{1}{2} \int_{S_3} h\left(\tau - \tau_f\right)^2 dS \qquad (1.23)$$

In (1.22) and (1.23), h represents the convection coefficient. The fluid temperature τ_f is assumed to be uniform. We study an element side which is a line segment of length L. We discretize the edge temperature as follows:

$$\tau = T_0(1 - \frac{x}{L}) + T_1 \frac{x}{L} \qquad (1.24)$$

Replacing in the functional, we obtain:

$$
\begin{aligned}
I_{el} &= \frac{1}{2}\int_0^L h\left(\begin{bmatrix} 1-\frac{x}{L} & \frac{x}{L}\end{bmatrix}[T]-t_f\right)^2 dx \\
&= \frac{1}{2}h\int_0^L \left\{[T]^T\begin{bmatrix} 1-\frac{x}{L} \\ \frac{x}{L}\end{bmatrix}\begin{bmatrix} 1-\frac{x}{L} & \frac{x}{L}\end{bmatrix}[T]-2\begin{bmatrix} 1-\frac{x}{L} & \frac{x}{L}\end{bmatrix}[T]t_f+t_f^2\right\}dx
\end{aligned}
\tag{1.25}
$$

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:

$$
T_{el} = \begin{bmatrix} T_0 & T_1 & T_f\end{bmatrix}^T \tag{1.26}
$$

Replacing in the first term of (1.25), we obtain:

$$
I_{el} = \frac{1}{2}hT_f^T\left(\int_0^L \begin{bmatrix} 1-\frac{x}{L} \\ \frac{x}{L} \\ -1\end{bmatrix}\begin{bmatrix} 1-\frac{x}{L} & \frac{x}{L} & -1\end{bmatrix}dx\right)T_f \tag{1.27}
$$

Developping the expression:

$$
I_{el} = \frac{1}{2}hT_f^T\int_0^L \begin{bmatrix} \left(1-\frac{x}{L}\right)^2 & \left(1-\frac{x}{L}\right)\frac{x}{L} & -\left(1-\frac{x}{L}\right) \\ \left(1-\frac{x}{L}\right)\frac{x}{L} & \left(\frac{x}{L}\right)^2 & -\frac{x}{L} \\ -\left(1-\frac{x}{L}\right) & -\frac{x}{L} & 1\end{bmatrix}dx\ T_f \tag{1.28}
$$

After integrating and including the thickness e to ensure the coherence of units, we transform the functional (1.25) into an **algebric function** of the nodal temperatures:

$$
I_{el}^{al} = \frac{1}{12}h\ eL\ T_f^T\begin{bmatrix} 2 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & 6\end{bmatrix}T_f \tag{1.29}
$$

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

$$K_h = h\,\frac{eL}{6}\begin{bmatrix} 2 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & 6 \end{bmatrix} \tag{1.30}$$

As well as the pure conduction matrix, this matrix is also 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 definite positive.

To solve a problem including conduction and convection, we have to compute two conductivity matrices. We call the first one K_k and the second one K_h. Later, the two matrices have to be added. The second one carries additional degrees of freedom corresponding to the virtual convective nodes.

$$K = K_k + K_h \tag{1.31}$$

The convection virtual nodes may be free or fixed.

Procedures used to solve convection heat transfers

Procedure Matlab © *pp_visopa_convection.m* for convection

```
hh   = 25;tb = 273;    nfc = 3;ta = [0;0; 290; 303];pge = [1;2;.1; 8];
% hh   = 18;tb = 273;    nfc = 2;ta = [300;280]    ;pge = [1;2;.1;1 ];
th   = pge(3); gap=1;
nx   = pge(4);ny = 2*nx;nel = nx*ny;no = (nx+1)*(ny+1);%nf = nx+1;  % Mesh
       disp('======================')
       disp(['Boundary cond. type : ',num2str(nfc)])
       disp(['Domain dim. w, h, t : ',num2str(pge(1:3,1)'),' m'])
       disp(['Mesh dimension      : ',num2str(nx),' x ',num2str(ny)])
       disp(['Impos. virt. nod T. : ',num2str(ta(1:size(ta,1))'),' K'])
       disp(['Impos. base Temp.   : ',num2str(tb),' K'])
       disp(['Convection coeff.   : ',num2str(hh,2),' W/(m2K)'])
tst  = tic;          % Beginning the analysis, initialisation of the timer
co   = conde(nx,ny);                    % Element conducrivity coefficients
xyz  = Nxyz(nx,ny);                         % Geometry ; nodal coordinates
[K ] = Coco(nx,ny,xyz,hh*th,nfc);             % Computing the convection K
lK   = loca(nx,ny);                                % Localization matrix
for n    = 1:nel                              % Loop on the nel elements
    Kel  = th*co(n)*Kelu(xyz,lK(n,:));       % Element conductivity matrix
    for i= 1:4              % Assembling the nel conductivity matrices Kel
        for j=1:4;K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+Kel(i,j);end;end;
end;
if nfc == 2                                                % 1. Adiabatic base
    gco  =  K(1:no,no+1:no+size(K,1)-no)*ta;
    tca  = -K(1:no,1:no)\gco;
    Hflo(nx,ny,xyz,lK,tca,co);mgra(nx,ny,xyz,lK,tca);
    grisc(nx,ny,tca(1:no),xyz,gap);axis off            % Drawing isotherms
    sm   = (K*[tca;ta(1:size(K,1)-no)])';% Compute second member of system
    qxg  = hh*(ta(2)-min(tca));
    qxc  = co(1)*(min(tca)-max(tca))/pge(1);
    qxd  = hh*(max(tca)-ta(1));
    t1   = (co(1)*ta(1)+(co(1)+hh*pge(1))*ta(2))/(2*co(1)+hh*pge(1));
    t2   = ta(2)+ta(1)-t1;
    hvn  = K(no+1:no+size(ta,1),:)*[tca;ta];
```

```
34      disp(['Heat on virt. nodes : ',num2str(hvn'),' W'])
35      disp(['H convl cond. convr : ',num2str([qxg qxc qxd]),' Wm-2'])
36      disp(['Surf. T. right left : ',num2str([max(tca) min(tca)]),' K'])
37  end
38  if nfc == 3    % 2. One free and two imposed temperatures of virtual nodes
39      if ta(2)==0.
40          K11 = K(1:no-nx-1,1:no-nx-1);
41          K12 = K(1:no-nx-1,no-nx:no);
42          K13 = K(1:no-nx-1,no+1);
43          K14 = K(1:no-nx-1,no+2:no+3);
44          K31 = K(no+1,1:no-nx-1);
45          K32 = K(no+1,no-nx:no);
46          K33 = K(no+1,no+1);
47          K34 = K(no+1,no+2:no+3);
48          T2  = ones(nx+1,1)*tb;
49          T4  = [ta(3); ta(4)];
50          tcb = [K11 K13;K31 K33]\(-[K12 K14;K32 K34]*[T2;T4]);
51          tca = [tcb(1:no-nx-1); ones(nx+1,1)*tb; tcb(size(tcb,1)); T4];
52          grisc(nx,ny,tca(1:no),xyz,gap);axis off   % Drawing  the isotherms
53          Hflo(nx,ny,xyz,lK,[tca;ones(nx+1,1)*tb],co);
54          disp(['Virtual nodes temp. : ',...
55              num2str(tca(no+1:size(K,1))',3),' K'])
56          disp(['Mesh min max Temp.  : ',num2str([min(tca(1:no)) ...
57                 max(tca(1:no))],3),' K'])
58      else
59          gco  = K(1:no-nx-1,no-nx:size(K,1))*[ones(nx+1,1)*tb;ta(2:4)];
60          disp(['Imposed temperatures: ',num2str([tb;ta(2:4)]',3),' K'])
61          tca  = -K(1:no-nx-1,1:no-nx-1)\gco;
62          grisc(nx,ny,[tca;ones(nx+1,1)*tb],xyz,gap);axis off % Drawing iso.
63  %         Hflo(nx,ny,xyz,lK,[tca;ones(nx+1,1)*tb]);
64          sm   =(K*[tca;ones(nx+1,1)*tb; ta(2:4)])';          % second member
65          rea  = sum(sm(size(sm,2)-2-nx-1:size(sm,2)-3)); % Heat flow bottom
66          disp(['Global heat balance : ',num2str([rea sm(size(sm,2)-2:...
67              size(sm,2))]),' W'])
68        disp(['Mesh min max Temp.  : ',num2str([min(tca) max(tca)],4),' K'])
69      end
70  end
71  disp(['K size (incl. conv.): ',num2str(size(K))])
72  disp(['Cpu                 : ',num2str(toc(tst),'%0.3g'),' sec.'])
```

Table 8: Matlab© procedure pp_visopa_convection.m

Lines 1 – 4 : Data input

Variable *hh* gives the convection coefficient for heat exchanges with exterior. Vector *pge* gives the size of the analyzed domain and the dimension of the mesh. We can work out from it the thickness *th=pge(3)* and the number of elements in the horizontal direction *nx=pge(4).*The variable *ny=nx*2* is the number of elements in the vertical direction. The following items concern the computation: *nel* is the number of elements, *no*, the number of nodes.

Lines 5 – 11 : Display some data and results

Line 13 : Conductivity coefficients in the elements (function *conde.m*)

Line 14 : Matrix of nodal coordinates (function *Nxyz.m*)

Line 15 – 16 : Compute the conduction-convection matrix in the function *Coco.m*.

$$K_{convection} = h \frac{eL}{6} \begin{bmatrix} 2 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & 6 \end{bmatrix} \quad (1.32)$$

The element is a vertical or a horizontal one. Its length is L (m), its thickness is e (m) and the convection coefficient is h ($Wm^{-2}K^{-1}$). The nodal sequence starts with the two real ones pertaining to the mesh and ends with the virtual one related to convection or radiation.

Function Matlab© *CoKr.m* compute the conduction matrix for convection

```
1   function[K] = CoKr(nx,ny,ep,he,hh,nfc)
2   disp(['CoKr.m conv. coeff. : ',num2str(hh),' Wm-2K-1'])
3   Kelc  = [2 1 -3;1 2 -3;-3 -3 6]*hh*ep*he/ny/6;        % Elem. conv. matrix
4   if nfc ==2
5   Ntca  = (nx+1)*(ny+1)+nfc;                            % Mesh nx x ny elements
6   ntv2  = [ Ntca-1 Ntca];        % Numbering of the convective virtual nodes
7   lc    = locc2(nx,ny,ntv2);     % Local. of the convection matrices 2 sides
8   disp(['Virtual conv. nodes : ',num2str(ntv2)])
9   end
10  if nfc ==3
11  Ntca  = (nx+1)*(ny+1)+nfc;                          % Mesh nx x ny elements
12  ntv3  = [Ntca-2 Ntca-1 Ntca];% Numbering of the convective virtual nodes
13  lc    = locc(nx,ny,ntv3);    % Local. of the convection matrices 3 sides
14  disp(['Virtual conv. nodes : ',num2str(ntv3)])
15  end
16  if nfc ==4
17  Ntca  = (nx+1)*(ny+1)+nfc;                          % Mesh nx x ny elements
18  ntv4  = [Ntca-3 Ntca-2 Ntca-1 Ntca];  % Number. convective virtual nodes
19  lc    = locc4(nx,ny,ntv4);   % Local. of the convection matrices 4 sides
20  disp(['Virtual conv. nodes : ',num2str(ntv4)])
21  end
22  K     = zeros(Ntca,Ntca);% Dimension of K including fixed DOF & add. nodes
23  for n = 1:size(lc,1);for i=1:3;for j=1:3  % Assembling conv. matrices Kelc
24            K(lc(n,i),lc(n,j))=K(lc(n,i),lc(n,j))+Kelc(i,j);end;end;end
25  end
```

Table 9: Matlab© function CoKr.m for the computation of the convective conduction matrices

The function *CoKr.m* allows computing the convection matrices of an *nx* x *ny* mesh (arguments 1 and 2 of the function) of a domain of thickness given by argument 3 and height by argument 4. The convection coefficient is given by argument 5. Argument 6 is giving the number of faces where convection elements are present.
A new version of *CoKr.m* is given in *Coco.m* (*Table 10*). It allows computing the convection matrices along boundary segments given by the two ends.

Function Matlab© *Coco.m* compute the conduction matrix for convection

```
1   function[K] = Coco(nx,ny,xyz,hh,nfc)
2   Kelc  = [2 1 -3;1 2 -3;-3 -3 6]*hh/6;          % Element convection matrix
3   if nfc ==2              % Virtual convection nodes on the 2 vertical sides
4   Ntca  = (nx+1)*(ny+1)+nfc;                          % Mesh nx x ny elements
5   ntv2  = [ Ntca-1 Ntca];        % Numbering of the convective virtual nodes
6   lc    = locc2(nx,ny,ntv2);     % Local. of the convection matrices 2 sides
7   disp(['Virtual conv. nodes : ',num2str(ntv2)])
8   end
9   if     nfc ==3                      % Virtual convection nodes on 3 sides
```

```
Ntca  = (nx+1)*(ny+1)+3;                              % Mesh nx x ny elements
ntv3  = [Ntca-2 Ntca-1 Ntca];  % Numbering of the convective virtual nodes
lc    = locc(nx,ny,ntv3);      % Local. of the convection matrices 3 sides
disp(['Virtual conv. nodes : ',num2str(ntv3)])
end
if nfc ==4                              % Virtual convection nodes on 4 sides
Ntca  = (nx+1)*(ny+1)+nfc;                            % Mesh nx x ny elements
ntv4  = [Ntca-3 Ntca-2 Ntca-1 Ntca];    % Number. convective virtual nodes
lc    = locc4(nx,ny,ntv4);     % Local. of the convection matrices 4 sides
disp(['Virtual conv. nodes : ',num2str(ntv4)])
end
K     = zeros(Ntca,Ntca);%Dimension of K including fixed DOF & virt. nodes
for n = 1:size(lc,1)
    Ls= sqrt((xyz(lc(n,1),1)-xyz(lc(n,2),1))^2+...
        (xyz(lc(n,1),2)-xyz(lc(n,2),2))^2);
    for i=1:3
        for j=1:3                          % Assembling conv. matrices Kelc
            K(lc(n,i),lc(n,j))=K(lc(n,i),lc(n,j))+Kelc(i,j)*Ls;
        end;
    end;
end
end
```

Table 10: Matlab© function Coco.m Convective conduction matrices along boundary segments

The Matlab© functions *locc2.m, locc.m, locc4.m* allow computing the localizations of the convection matrices on two, three or four sides. The function *locc2.m* performs the computation of the localization matrix of the convective elements on the vertical sides of the domain (shown in blue, *Figure 6*). The process starts with the right side, then, the left side. The computation is performed only if the identification number of the virtual node is given in the array *ntv* (third argument of the procedure) .

Function Matlab© *locc2.m* compute the localization matrix of the convective elements

```
function [lc]=locc2(nx,ny,ntv) % Localization vectors for conv. on 2 sides
no               = (nx+1)*(ny+1);lc = zeros(2*ny,3);ii = 0;
if ntv(1) > 0                                                  % Right side
    for i        = 1:ny
        lc(i,1)  = (nx+1)*i;lc(i,2) = lc(i,1)+nx+1;lc(i,3) = no+1;
    end;                                  ii = ii + ny;
end
if ntv(2) > 0                                                   % Left side
    for i        = 1:nx+1:(nx+1)*ny;    ii = ii + 1;
        lc(ii,1) = i; lc(ii,2)=lc(ii,1)+nx+1 ;lc(ii,3) = no+2;
    end
end
disp(['Number of conv. el. : ',num2str(ii)])
end
```

Table 11: Matlab© function locc2.m to compute the localization of convective elements

3	6	16
6	9	16
9	12	16
12	15	16
1	4	17
4	7	17
7	10	17
10	13	17
1	2	18
2	3	18

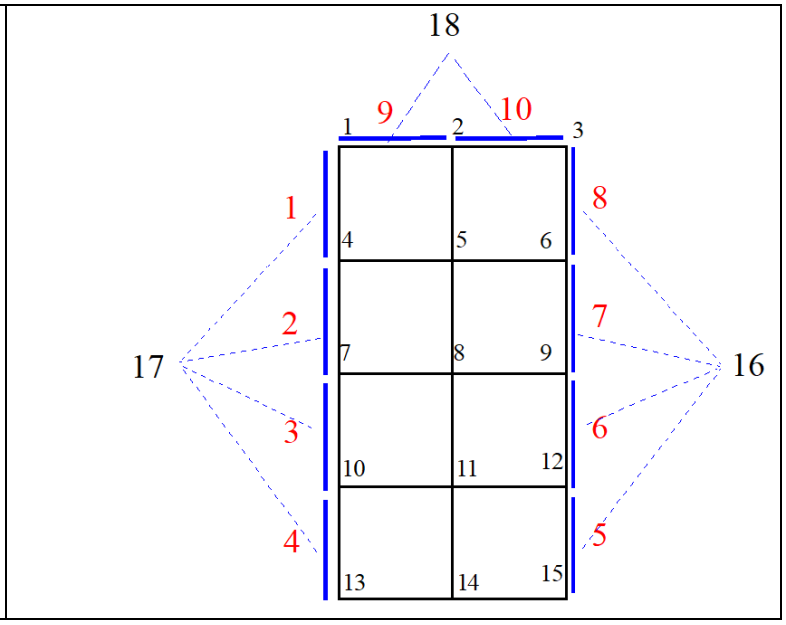


Figure 6: Localization matrix of the convective elements on 3 sides of a 2 x 4 mesh

Line 17 : Compute the localization matrix (function *loca.m*)

Line 18 : Conductivity matrix of a square element

Lines 19 – 22: Global conductivity matrix assembling

Lines 23 – 38: Solution of the system in the case of 2 convective end 2 adiabatic edges

The system is solved as follows: the imposed temperatures of the virtual nodes are transformed in a second member of the system of equations (see *line 24*) before solving the system (see *line 25*). *Lines 26 & 27* are devoted to the graphical output: isotherms (*Figure 7*), heat flows and temperature gradients. *Lines 28* to *37* are concerned with some statistics and variables showing the agreement of the solution with the analytical results.

Function Matlab© *locc.m* compute the localization matrix of the convective elements

```
1   function [lc]=locc(nx,ny,ntv)  % Localization vectors for conv. on 3 sides
2   no              = (nx+1)*(ny+1);lc = zeros(2*ny+nx,3);ii = 0;
3   if ntv(1) > 0                                                % Right side
4       for i       = 1:ny
5           lc(i,1)  = (nx+1)*i;lc(i,2) = lc(i,1)+nx+1;lc(i,3) = no+1;
6       end;                                  ii = ii + ny;
7   end
8   if ntv(2) > 0                                                 % Left side
9       for i       = 1:nx+1:(nx+1)*ny;     ii = ii + 1;
10          lc(ii,1) = i; lc(ii,2)=lc(ii,1)+nx+1 ;lc(ii,3) = no+2;
11      end
12  end
13  if ntv(3) > 0                                                  % Top side
14      for i       = 1:nx;                 ii = ii + 1;
15          lc(ii,1) = i;lc(ii,2) = lc(ii,1)+1;lc(ii,3) = no+3;
16      end
17  end;
```

```
% disp(['Number of conv. el. : ',num2str(ii)])
end
```

Table 12: Matlab© function locc.m for the localization of convective elements (3 sides)

Function Matlab© *locc4.m* compute the localization matrix of the convective elements

```
function [lc]=locc4(nx,ny,ntv) % Localization vectors for conv. on 4 sides
no              = (nx+1)*(ny+1);lc = zeros(2*(ny+nx),3);ii = 0;
if ntv(1) > 0                                                  % Right side
    for i       = 1:ny
        lc(i,1)  = (nx+1)*i;lc(i,2) = lc(i,1)+nx+1;lc(i,3) = no+1;
    end;                                  ii = ii + ny;
end
if ntv(2) > 0                                                   % Left side
    for i       = 1:nx+1:(nx+1)*ny;    ii = ii + 1;
        lc(ii,1) = i; lc(ii,2)=lc(ii,1)+nx+1 ;lc(ii,3) = no+2;
    end
end
if ntv(3) > 0                                                    % Top side
    for i       = 1:nx;                ii = ii + 1;
        lc(ii,1) = i;lc(ii,2) = lc(ii,1)+1;lc(ii,3) = no+3;
    end
end;
if ntv(4) > 0                                                 % Bottom side
    for i       = 1:nx;                ii = ii + 1;
        lc(ii,1) = no-nx-1+i;lc(ii,2) = lc(ii,1)+1;lc(ii,3) = no+4;
    end
end;
disp(['N. of convective el.: ',num2str(ii)])
end
```

Table 13: Matlab© function locc4.m for the localization of convective elements (4 sides)

Lines 39– 72 : Solution of other problems.

Two convective and two adiabatic faces

```
pp_convection
Boundary cond. type : 2
Domain dim. w, h, t : 1 2 0.1 m
Mesh dimension      : 1 x 2
Impos. virt. nod T. : 300  280 K
Impos. base Temp.   : 273 K
G. convection coeff.: 18 W/(m2K)
conde.m uniform k   : 1 W/(mK)
Number of conv. el. : 4
Virtual conv. nodes : 7  8
Convection coeff.   : 18 W/(m2K)
Heat flow, max      : 18, mean: 18 W/m2
- gradT, max        : 18, mean: 18 K/m
Heat on virt. nodes : 3.6        -3.6 W
H convl cond. convr : -18 -18 -18 Wm-2
Surf. T. right left : 299 281 K
Size of full K      : 8  8
Cpu                 : 1.98 sec.
```

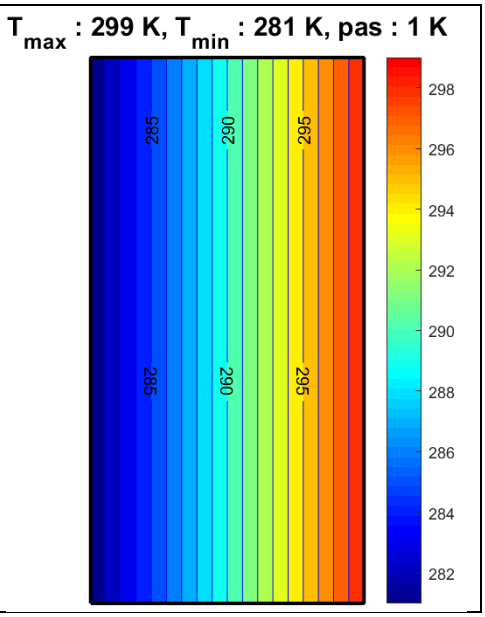


Figure 7: The isocurves are orthogonal to the adiabatic boundaries

This example deals with a very simple problem: we want to evaluate the temperature field in a wall submitted to convective heat transfers on both vertical sides. The horizontal sides are adiabatic.
The result 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]. The variables t_i 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 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	$t_2 < t_3$

$$q_x = -eh(t_1 - t_0) = -ek\frac{(t_2 - t_1)}{w} = eh(t_2 - t_3) \tag{1.33}$$

From the two convective terms, we deduce:

$$t_0 - t_1 = t_2 - t_3 \quad \rightarrow \quad t_2 = t_0 + t_3 - t_1 \tag{1.34}$$

It means that the difference between both virtual convective nodes and their corresponding surface temperatures are the same. We now express that the convective flux is the same as the internal conductive one.

$$\begin{aligned} t_1 - (t_0 + t_3 - t_1) &= \frac{wh}{k}(t_0 + t_3 - t_1 - t_3) \\ 2t_1 - t_0 - t_3 &= \frac{wh}{k}(t_0 - t_1) \\ t_1(2 + \frac{wh}{k}) &= t_3 + (1 + \frac{wh}{k})t_0 \\ t_1 &= \frac{kt_3 + (k + wh)t_0}{2k + wh} \end{aligned} \tag{1.35}$$

Because the finite element model is able to represent the exact solution, this analytical solution is obtained for any mesh.

Exercise

In the situation 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 variable $z = w\ h\ /\ k$ and to compare with the finite element model result. Convection on 3 faces, fourth one fixed, one virtual node free
We start splitting the system to be solved into 4 categories. The second members are denoted S_i.

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \\ K_{41} & K_{42} & K_{43} & K_{44} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix} \tag{1.36}$$

We assume that T_1 and T_3 are the unknowns to be computed. T_1 corresponds to internal nodes and T_3 to the unknown virtual node.

$$\begin{bmatrix} K_{11} & K_{13} \\ K_{31} & K_{33} \end{bmatrix} \begin{bmatrix} T_1 \\ T_3 \end{bmatrix} = - \begin{bmatrix} K_{12} & K_{14} \\ K_{32} & K_{34} \end{bmatrix} \begin{bmatrix} T_2 \\ T_4 \end{bmatrix} + \begin{bmatrix} S_1 \\ S_3 \end{bmatrix}$$
$$\begin{bmatrix} T_2 \\ T_4 \end{bmatrix} = \begin{bmatrix} K_{11} & K_{13} \\ K_{31} & K_{33} \end{bmatrix}^{-1} \left(\begin{bmatrix} S_1 \\ S_3 \end{bmatrix} - \begin{bmatrix} K_{11} & K_{13} \\ K_{31} & K_{33} \end{bmatrix} \begin{bmatrix} T_2 \\ T_4 \end{bmatrix} \right) \quad (1.37)$$

```
pp_convection
pp_visopa_convection

Boundary cond. type : 3
Domain dim. w, h, t : 1 2 0.1 m
Mesh dimension      : 50 x 100
Impos. virt. nod T. : 0  0 290  303 K
Impos. base Temp.   : 273 K
Convection coeff.   : 25 W/(m2K)
conde.m uniform k   : 2 W/(mK)
Locc.m N conv. elem.: 250
Virtual conv. nodes : 5152  5153  5154
Convection coeff.   : 25 W/(m2K)
Heat flow, max      : 451 mean: 33 W/m2
Virtual nodes temp. : 287 290 303 K
Mesh min max Temp.  : 273 301 K
Size of convect. K  : 5154  5154
Cpu                 : 7.06 sec.
```

Left virtual node (tca(no+1))
computed temperature: 286.6 *K*

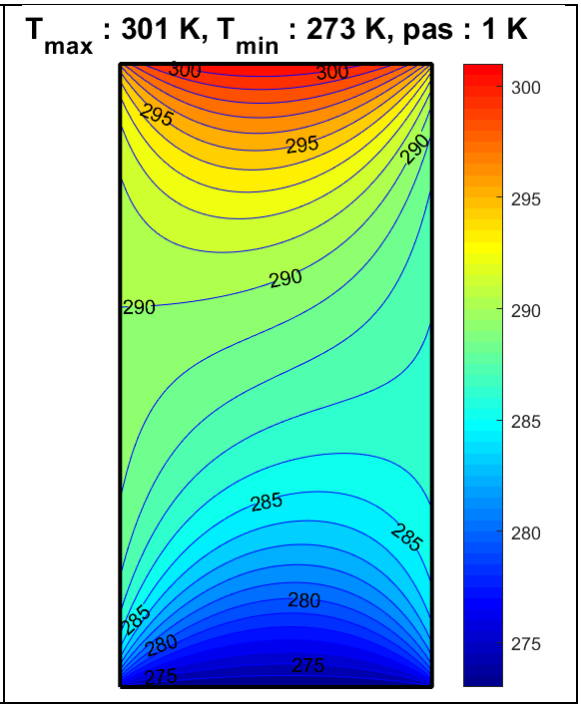


Figure 8: Isocurves for 1 free and 2 imposed temperatures of virtual nodes

To let free the left virtual node of the previous problem, we use the new sequence of sentences corresponding to the option *nfc = 3*
In this case, the left virtual node is reaching the temperature of 286.6 K (to get this information, enter: tca (no+1) after running the procedure). Note that the above sequence is reproducing exactly and with the same notations the final line of the development (1.37).

Exercise

It is proposed to modify the boundary conditions of the application presented in *Figure 8* in order to obtain more or less the same temperatures on both horizontal sides and obtain a temperature gradient mainly oriented from right to left. At the end of the simulation, we can display the temperature of the left virtual node. (Indication: use the same temperature for the top virtual node and the base of the rectangular domain).

Tutorial III: Radiative Heat Transfer

Method used for the solution of radiation heat transfers

Thermal exchanges by infrared radiation emission are very important. Stefan-Boltzmann's law establishes that irradiance or radiated power per unit area of a blackbody and per unit of time is proportional to the fourth power of the body's thermodynamic temperature, with the Stefan-Boltzmann coefficient $\sigma = 5.6704\ 10^{-8}\ Wm^{-2}K^{-4}$

$$Q = \sigma\left(T_k^4 - T_r^4\right) \tag{1.38}$$

In this expression, T_k represents the surface temperature of the body and T_r the temperature of the outside element. By developing this expression we obtain:

$$Q = \sigma\left(T_k^2 + T_r^2\right)\left(T_k + T_r\right)\left(T_k - T_r\right) \tag{1.39}$$

For a radiation element, the "conductivity" matrix is calculated exactly as for convection, but the convection coefficient *h* is replaced by the coefficient:

$$\sigma\left(T_k^2 + T_r^2\right)\left(T_k + T_r\right) \tag{1.40}$$

In this expression, the term $\sigma = 5.6704\ 10^{-8}\ Wm^{-2}K^{-4}$ represents the Stefan-Boltzmann coefficient. T_k represents the surface temperature of the body and T_r the temperature of the scene element seen from the point where the heat exchange has to be evaluated.

For a radiative element, the "conductivity" matrix is calculated exactly as for convection.

$$K_r = \frac{eL\ \sigma\left(T_k^2 + T_r^2\right)\left(T_k + T_r\right)}{6}\begin{bmatrix} 2 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & 6 \end{bmatrix} \tag{1.41}$$

In this expression, some temperatures must be evaluated in the same loop as the computation one. Thus, the above coefficient has to be obtained by successive approximations during the iterations.

Procedures used for radiative heat transfers

Procedure Matlab © *pp_ radiation.m* for radiation

```
tb   = 273;pge=[10;20;1;20];pf=[0;12.5;0];ts=[286;290;303];SB = 5.6704e-8;
qs   = pf(1)*pge(1)*pge(2);qw=pf(2)*pge(2)*pge(3);qe=pf(3)*pge(2)*pge(3);
nx   = pge(4);ny = nx*2;nel = nx*ny;no = (nx+1)*(ny+1);              % Mesh
disp('********************')% First iteration ============================
co   = conde(nx,ny);
disp(['Stefan-Boltzmann    : ',num2str(SB,'%0.3g'),' Wm-2K-4'])
disp(['Mesh dimension      : ',num2str(nx),' x ',num2str(ny)])
disp(['Base temperature    : ',num2str(tb,3),' K'])
disp(['Virtual nodes temp. : ',num2str(ts'),' K'])
```

```
10  Kel  = pge(3)/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];   % elem. K
11  tca  = ones(no,1)*min(ts);
12  lK   = loca(nx,ny);          % Computing the localization matrix (nel x 4)
13  nit  = 2;
14  [K ] = ItKr (tca,ts,nx,ny,pge(3),pge(2),SB);
15  for n=1:nel;for i=1:4;for j=1:4     % Assembling nel conduct. matrices Kel
16            K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+co(n)*Kel(i,j);end;end;end
17  gco  =  K(1:no-nx-1,no-nx:no+size(K,1)-no)*[ones(nx+1,1)*tb; ...
18      ts(1:size(K,1)-no,1)];
19  tca  = -K(1:no-nx-1,1:no-nx-1)\gco;
20  gap=1;grisb(nx,ny,[tca ;ones(nx+1,1)*tb],gap);axis off% Drawing isotherms
21  tmi1   = min(tca(1:no-nx-1));
22  sm     = (K*[tca;ones(nx+1,1)*tb;ts(1:size(K,1)-no)])'; % syst. sec. memb.
23  hvn1   = sum(sm(size(sm,2)-2:size(sm,2)));
24  % Additional iterations ================================================
25  if nit > 1
26  [hvn ,tmi]=radit(tca,nx,ny,tb,ts,pge(3),pge(2),lK,Kel,co,nit,tmi1,hvn1,SB);
27  end
28  disp(['1st min. int. temp. : ',num2str(tmi(1:min(5,nit))',5),' K'])
29  disp(['1st it. conv. flows : ',num2str(hvn(1:min(5,nit))',5),' W'])
```

Table 14: Matlab© procedure pp_radiation.m

The procedure for the solution of a thermal radiation problem with radiative boundary conditions given in *Table 14* has the following characteritics.

Lines 1 – 3 : Data input
Line 4 : Definition of element conductivities (function *conde.m*)
Line 5 : Conductivity matrix of a square element
Line 6 : Initialization
Line 7 : Localization matrix (function *loca.m*)
Line 8 : Number of iterations required to test the convergence
Line 9 : Starting the first iteration
Line 10 : Conductivity-radiation matrix, function *ItKr.m*

The element can be vertical or horizontal. Its length is L, its thickness is e and the Stefan-Boltzmann coefficient is σ ($Wm^{-2}K^{-4}$). The global nodes sequence starts with the real ones pertaining to the mesh and finishes with the virtual ones. This function computes the radiation "convection" matrices of a mesh *nx* x *ny* (arguments 3 and 4 of the function) for a domain of dimensions given by arguments 3 and 4. The Stefan-Boltzmann constant is the last argument of the function. The temperatures of the domain are stored in the vector *tca* (argument 1) while the temperatures of the virtual nodes are stored in the vector *ts* (argument 2).

Function Matlab© *ItKr.m* for radiation

```
1   function[Kr]=ItKr(tca,ts,nx,ny,ep,he,SB)
2   no     = (nx+1)*(ny+1);Ntca = no+3;
3   Kelc   = [2 1 -3;1 2 -3;-3 -3 6]/6;             % Element convection matrix
4   Kr     = zeros(Ntca,Ntca);ntv=no+1:Ntca;% ntv: numbers of the virtual nodes
5   lc     = locc(nx,ny,ntv);        % Localizations of the convection matrices
6   crdm   = 0;iel=0;
7   for i                = nx+1 : nx+1 : no-nx-1        % Right face radiation
8       T1               = (tca(i)+tca(i+nx+1))/2;     % Edge mean temperature
9       crd              = SB*(T1^2+ts(1)^2)*(T1+ts(1))*ep*he/ny;
10      crdm             = crdm+crd;iel=iel+1;
11      for ii=1:3;for j = 1:3;Kr(lc(iel,ii),lc(iel,j)) = ...
```

```
12                  Kr(lc(iel,ii),lc(iel,j))+Kelc(ii,j)*crd;end;end
13  end
14  crd=crdm/iel;crgm     = 0;iel=0;         % Mean right convection coefficient
15  for i                 = 1 :nx+1 :(ny+1)*nx            % Left face radiation
16      T1                = (tca(i)+tca(i+nx+1))/2;      % Edge mean temperature
17      crg               = SB*(T1^2+ts(2)^2)*(T1+ts(2))*ep*he/ny;
18      crgm              = crgm+crg;iel=iel+1;
19      for ii=1:3;for j = 1:3;Kr(lc(iel+ny,ii),lc(iel+ny,j)) = ...
20                  Kr(lc(iel+ny,ii),lc(iel+ny,j))+Kelc(ii,j)*crg;end;end
21  end
22  crg = crgm/iel;crsm   = 0;iel = 0;       % Mean left convection coefficient
23  for i                 = 1:nx                            % Top face radiation
24      T1                = (tca(i)+tca(i+1))/2;         % Edge mean temperature
25      crs               = SB*(T1^2+ts(3)^2)*(T1+ts(3))*ep*he/ny;
26      crsm              = crsm+crs;iel = iel+1;
27      for ii=1:3;for j = 1:3;Kr(lc(iel+ny*2,ii),lc(iel+ny*2,j)) = ...
28                  Kr(lc(iel+ny*2,ii),lc(iel+ny*2,j))+Kelc(ii,j)*crs;end;end
29  end
30  crs = crsm/iel;                           % Mean top convection coefficient
31  disp(['ItKr.m rig left top : ',num2str([crd crg crs])]);
32  end
```

Table 15: Matlab© function ItKr.m

Lines 11 – 12: Assembling the global conductivity matrix
Lines 13 – 15: Solution of the system
Line 16 : Drawing the isotherms in the function *grisb.m*
Lines 17 – 19: Initialization of statistics variables
Line 21 : Starting the additional iterations
Line 22 : Additional iterations in the function *radit.m*

Function Matlab© *radit.m* for radiation

```
 1  function [hvn,tmi]=radit (tca,nx,ny,tb,ts,p3,p2,lK,Kel,co,nit,tmi1,hvn1,SB)
 2  nel=nx*ny;no=(nx+1)*(ny+1);hvn=ones(nit,1)*hvn1;tmi=ones(nit,1)*tmi1;
 3  for it      = 2 : nit
 4      [K ]    = ItKr ([tca;ones(nx+1,1)*tb;ts],ts,nx,ny,p3,p2,SB);
 5      for n   = 1:nel;for i=1:4;for j=1:4 % Assembling nel cond. matrices Kel
 6          K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+co(n)*Kel(i,j);end;end;end
 7      gco     =  K(1:no-nx-1,no-nx:no+size(K,1)-no)*[ones(nx+1,1)*tb; ...
 8                 ts(1:size(K,1)-no,1)];
 9      tca     = -K(1:no-nx-1,1:no-nx-1)\gco;
10      gap     = 1;grisb(nx,ny,[tca ;ones(nx+1,1)*tb],gap);axis off % Draw iso
11      tmi(it)= min(tca(1:no-nx-1));
12      sm      = (K*[tca;ones(nx+1,1)*tb;ts(1:size(K,1)-no)])';% second member
13      hvn(it)= sum(sm(size(sm,2)-2:size(sm,2)));
14  end
15  % Statistics ==========================================================
16  if nit > 10
17      figure('Position',[10 50 800 400]);plot(tmi(1:nit));gr id on;hold on
18      ylabel('Minimum temperature (K) ');xlabel('Iteration number ')
19      axis  ([1 nit min(tmi)*.95 max(tmi)/.95])
20      title (['Sky temperatures: ',num2str(ts'),' K'])
21      figure('Position',[10 50 800 400]);plot(hvn(1:nit));grid on;hold on
22      ylabel('Heat flowing to radiation virtual nodes (W) ');
23      xlabel('Iteration number ');axis([1 nit min(hvn)*1.01 min(hvn)*.95])
24      title (['Sky temperatures: ',num2str(ts'),' K'])
25  end
26  end
```

Table 16: Matlab© function radit.m

This function performs iterations to take into account the non linearity of the "conduction" – radiation matrix. It also gives some statistics about the convergence of the iterative loops if the number of stipulated iterations is greater than ten.
The updating of the matrix is performed at *line 4* with the function *ItKr.m*. In this example, the loading is limited to prescribed temperatures and is computed at *lines 7 & 8*. All the iterations provide an isotherm drawing.

Example of radiative transfer

We start with an example where the boundary conditions consist of radiation on three faces and fixed temperatures on the fourth one. The isotherms are shown for the first (left) and the second iteration (right). The pseudo convective coefficients computed according to (1.40) are changing with iteration. They can be different on the three faces. The results are given in *Table 17*

```
Matlab© procedure pp_radiation
**********************************
conde.m uniform k    : 1 W/(mK)
Stefan-Boltzmann     : 5.67e-08 Wm-2K-4
Mesh dimension       : 20 x 40
Base temperature     : 273 K
Virtual nodes temp. : 290  290  303 K
Pseudo h, ri, le, to: 2.77 2.77 2.96 Wm-2K-1
Pseudo h, ri, le, to: 2.76 2.76 3.14 Wm-2K-1
Convective flows     : 102.5 100.78 W
Base temperature     : 273 K
Virtual nodes temp. : 290  290  303 K
T maxi inside domai : 302.573 K
```

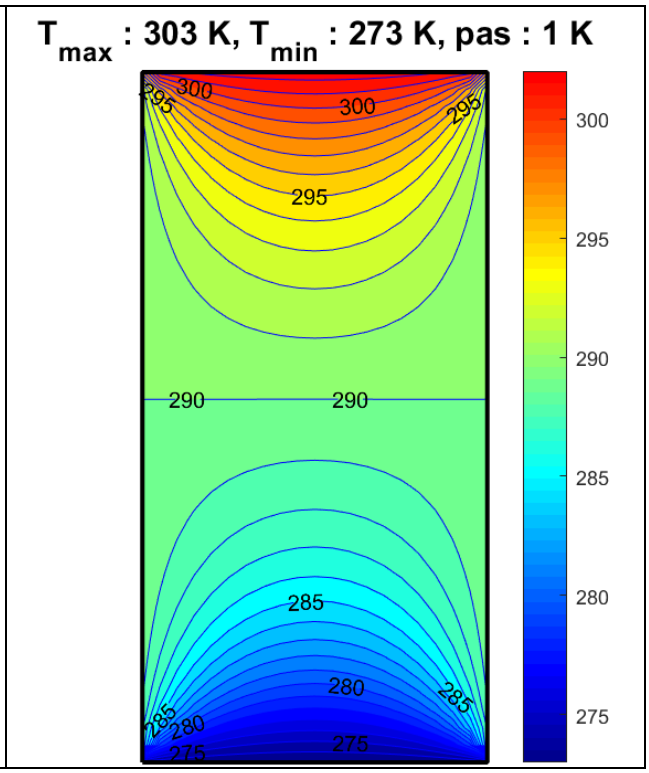


Table 17: Two iterations of a radiative heat transfer

The difference between iterations is insignificant. Consequently, only one isotherm diagram is presented.

Exercise

We propose to use the same boundary conditions as in the application presented in ***Figure 8***, but the convection conditions are transformed in radiation ones.

Tutorial IV: Transient Heat Transfer

To carry out the transient studies, new physical quantities, such as the density of the material and its heat capacity, are introduced.

The mass 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, its conjugate is the temperature.

The thermal diffusivity α of a material, expressed in m^2s^{-1}, represents its tendency to facilitate the diffusion of heat (a "good" thermal diffusivity in construction corresponds to a low value and a "bad" thermal diffusivity corresponds to a high one).

$$\alpha = \frac{k}{\rho c_p} \tag{1.42}$$

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

$$\left([C]+\theta\ \Delta t\ [K]\right)\left[T^{n+1}\right]=\left([C]-(1-\theta)\ \Delta t\ [K]\right)\left[T^{n}\right]+\Delta t\ \left(\theta\left[f^{n+1}\right]+(1-\theta)\left[f^{n}\right]\right) \tag{1.43}$$

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

$$\left([C]+\Delta t\ [K]\right)\left[T^{n+1}\right]=[C]\left[T^{n}\right]+\Delta t\ \left[f^{n+1}\right] \tag{1.44}$$

The vector $[T]$ can be divided into two parts:

1. the unknown temperatures T_{11} and
2. the fixed and therefore constant temperatures T_{1f}. So, we rewrite:

$$\left(\begin{bmatrix} C_{11} & C_{1f}\end{bmatrix}+\Delta t\ \begin{bmatrix} K_{11} & K_{1f}\end{bmatrix}\right)\begin{bmatrix} T_{11}^{n+1} \\ T_{1f}\end{bmatrix}=\begin{bmatrix} C_{11} & C_{1f}\end{bmatrix}\begin{bmatrix} T_{11}^{n} \\ T_{1f}\end{bmatrix}+\Delta t\ \left[f^{n+1}\right] \tag{1.45}$$

The superscripts n or $n+1$ indicate the iteration. Developing for the lines corresponding to the unknowns:

$$[C_{11}]\left[T_{11}^{n+1}\right]+\Delta t\ \begin{bmatrix} K_{11} & K_{1f}\end{bmatrix}\begin{bmatrix} T_{11}^{n+1} \\ T_{1f}\end{bmatrix}=[C_{11}]\left[T_{11}^{n}\right]+\Delta t\ \left[f^{n+1}\right] \tag{1.46}$$

$$\left([C_{11}]+\Delta t\ [K_{11}]\right)\left[T_{11}^{n+1}\right]\ =[C_{11}]\left[T_{11}^{n}\right]-\Delta t\ \left[K_{1f}\right]\left[T_{1f}\right]+\Delta t\ \left[f^{n+1}\right] \tag{1.47}$$

If there are no loads or fixations, we can remove the indices and we obtain the very simple equation:

$$\left(\left[C\right]+\Delta t\ \left[K\right]\right)\left[T^{n+1}\right]\ =\left[C\right]\left[T^{n}\right] \tag{1.48}$$

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

$$\tau = T_1\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right)+T_2\frac{x}{a}\left(1-\frac{y}{b}\right)+T_3\frac{x}{a}\frac{y}{b}+T_4\left(1-\frac{x}{a}\right)\frac{y}{b} \tag{1.49}$$

$$\begin{aligned} \tau &= [F][T] \\ [F] &= \left[\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right)\quad \frac{x}{a}\left(1-\frac{y}{b}\right)\quad \frac{x}{a}\frac{y}{b}\quad \left(1-\frac{x}{a}\right)\frac{y}{b}\right] \\ [T]^T &= [T_1\quad T_2\quad T_3\quad T_4] \end{aligned} \tag{1.50}$$

$$[C]=\int_V \rho c_p [F]^T [F]\,dV \tag{1.51}$$

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.

$$\begin{aligned} C_{33} &= e\int_0^a\left(\int_0^b \rho c_p \frac{x^2y^2}{a^2b^2}dy\right)dx \\ &= \rho e c_p\int_0^a \frac{b^3x^2}{3a^2b^2}dx = \rho e c_p \frac{b}{3}\int_0^a \frac{x^2}{a^2}dx = \rho e c_p \frac{ab}{9} = \frac{\rho V c_p}{9} \end{aligned} \tag{1.52}$$

It can be verified in the expression that the sum of its terms is equal to 36, and, therefore, that, concentrated in 1 point, the capacity would be equal to $\rho\, V\, c_p$. The matrix C is expressed in JK^{-1}.

$$[C]=\frac{\rho V c_p}{36}\begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix} \tag{1.53}$$

Procedures used for transient heat transfers

Procedure Matlab© pp_ transient.m for transient heat transfer

```
1  nx  = 20 ; ni =50 ; pe =100; dt=pe*3600; gap=.5;            % Input data
2  pte = [290;290;303;% Sky temperature                    East, West, Top
3          280;        % Initial temperature in transient analysis
4          280;        % Wall base temperature
```

```
      300;300;300]; % temperatures of atmosphere :         East, West, Top
pa6= 2      ;  disp(['Control param.  pa6 : ',num2str(pa6)])
hh = 25     ;  disp(['Convection coeff. h : ',num2str(hh),' W/(m2K)'])
Cp = 1000   ;  disp(['Specific heat       : ',num2str(Cp),' J.m-3.K-1']);
ro = 2500   ;  disp(['Specific mass       : ',num2str(ro),' kg.m-3'])
tb = pte(5) ;  disp(['Base temperature    : ',num2str(tb),' K'])
ti = pte(4) ;  disp(['Initial temperature : ',num2str(ti),' K'])
wi = 1;he=2;ep=0.1;                % Width, height & thickness of the wall
ny = nx*2;my = ny+1;nel = nx*ny;no = (nx+1)*(ny+1);      % Mesh definition
co = conde(nx,ny);
disp(['Domain dimensions   : ',num2str(wi),' x ',num2str(ep),' x ',...
    num2str(he),' m'])
scr= 0;scs=1;scl=2;t3=3;     % N. of add. nodes for convection & radiation
if pa6 == 2;t3=4;tad(1:4)=[pte(6:8)' pte(6)];
    disp(['Air temperatures    : ',num2str(tad),' K'])
else
    tf = pte(6:8)';disp(['Air temperatures    : ',num2str(tf),' K'])
end   % pa6 = 2: 4 virtual convection temp.
if pa6 == 1;t3=0;              end
ii = 0;
if scr >0;tad(ii+1:ii+3)=ts;ii=ii+3;end                % Sky temperatures
if scs >0;ii=ii+1;tad(ii)=pte(6); end       % Fluid temperature of the top
if scl >0;tad(ii+1:ii+2)=pte(7:8);end    % Fluid temp. of vertical borders
disp (['Mesh size           : ',num2str(nx),' x ',num2str(ny)])
disp (['N. virtual nodes t3 : ',num2str(t3)])
if pa6 == 0;nf=nx+1;else;nf = 0;end    % Computing the number of fixations
disp (['Number of fixations : ',num2str(nf)])
Ntca = no + t3;                                  % Total number of nodes
tst  = tic;          % Beginning the analysis, initialisation of the timer
disp(['Diffusivity          : ',num2str(co(1)/(ro*Cp)),' m2/s'])
disp(['Total duration       : ',num2str(dt/3600),' h'])
disp(['Number of iterations: ',num2str(ni)])
C    = zeros(Ntca,Ntca);    % Initialization of the global capacity matrix
[K ] = CoKr34 (nx,ny,ep,he,hh,pa6);             % Global K with convection
lK   = loca (nx,ny);% Elem. local. matrix with fixation without convection
Vel  = ep*wi/nx*he/ny;
Cel  = Cp*ro*Vel/36*[4 2 1 2;2 4 2 1;1 2 4 2;2 1 2 4];      % El. capacity
Kel  = ep/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];       % elem. K
for n=1:nel
    for i=1:4
        for j=1:4        % Assembling nel conductivity & capacity matrices
          K(lK(n,i),lK(n,j)) = K(lK(n,i),lK(n,j)) + co(n)*Kel(i,j);
          C(lK(n,i),lK(n,j)) = C(lK(n,i),lK(n,j)) + Cel(i,j)      ;
        end;
    end;
end
disp(['Tot. cap.sum(sum(C)): ',num2str(sum(sum(C))),' J/K'])
if pa6 == 1
    tcan = ones(Ntca,1)*tb;         % T. field initialization 2 subdomains
    for i             = 1:ny+1
        for j = 1:(nx-1)/2+1; ii = (i-1)*(nx+1)+j; tcan(ii) = ti; end
    end
else
    tcan = ones(Ntca,1)*ti;             % T. field initialization - genera
end
if pa6 == 0                              % Standard situation : fixed base
    tcan(no-nx:no) = tb;
    if t3>0; tcan(Ntca-t3+1:Ntca,1)=tad(1:t3);end
else
    if t3 > 0; tcan(Ntca-t3+1:Ntca,1)=tad(1:t3);end
end
tca         = tcan;                     if size(tca,1) < 30;disp(tca');end
fnp1        = zeros(Ntca,1);
tsmax       = zeros(ni,1);tmoy = zeros(ni,1);tsmin = zeros(ni,1);
disp(['Stat. Ntca no nf t3 : ',num2str([Ntca no nf t3])])
for it      = 1:ni % Solution of the iterative system ********************
    if pa6 == 2
        Kif = K(1:no-nf,no-nf+1:Ntca);
        tca(1:no-nf)=(C(1:no-nf,1:no-nf)+dt/ni*K(1:no-nf,1:no-nf))...
            \(dt/ni*(-Kif*tcan(no-nf+1:Ntca))+C(1:no-nf,1:no-nf)*...
            tcan(1:no-nf));
```

```
76          tcan        = [tca(1:no)'  tcan(no+1:Ntca)'];
77      else
78          if nf > 0                                   % Fixations are present
79              Kif = K(1:no-nf,no-nf+1:Ntca);
80              tca(1:no-nf)=(C(1:no-nf,1:no-nf)+dt/ni*K(1:no-nf,1:no-nf))...
81              \(dt/ni*(-Kif*tcan(no-nf+1:Ntca))+C(1:no-nf,1:no-nf)*...
82              tcan(1:no-nf));
83              if nf > 0;tca(no-nf+1:no) = tb;end    % Prescribing base temp.
84          tcan            = [tca(1:no-nx-1)'  tcan(no-nx:Ntca)'];
85          else                                % No fixations : adiabatic boundary
86              tca     = (C+dt/ni*K)\(C*tcan);
87              tcan    = tca;
88          end
89      end
90  tsmax(it) = max(tca(1:no-nf)); tsmin(it)=min(tca(1:no-nf)); % Inter. nodes
91  tmoy(it)  = sum(tca(1:no-nf))/size(tca(1:no-nf),1); % Stat. interior nodes
92  end
93  grisb(nx,ny,tca)                    % Drawing the isotherms on the domain
94  grhi(ni,dt,[ti;tsmax],[ti;tsmin],[ti;tmoy])         % Time evolutions of T
95  disp(['Cpu                  : ',num2str(toc(tst),'%0.3g'),' sec.'])
```

Table 18: Matlab© procedure pp_transient.m

The parameter *pa6* allows controlling the procedure:

pa6 = 0 Standard situation. *nf* = *nx* + 1
pa6 = 1 No load, no fixation. *t3* = 0, *nf* = *0*
pa6 = 2 Four convective nodes. *t3* = 4, *nf* = *0*

Examples of transient heat transfers

A uniform temperature test is carried out on a domain of dimensions (1*m* x 0.1*m* x 2 *m*) whose walls are adiabatic. Half of the area is at 303 *K*, the other half at 293 *K*. The time evolution and the moment at which the temperature becomes uniform are examined. The homogenization process depends on the diffusivity.

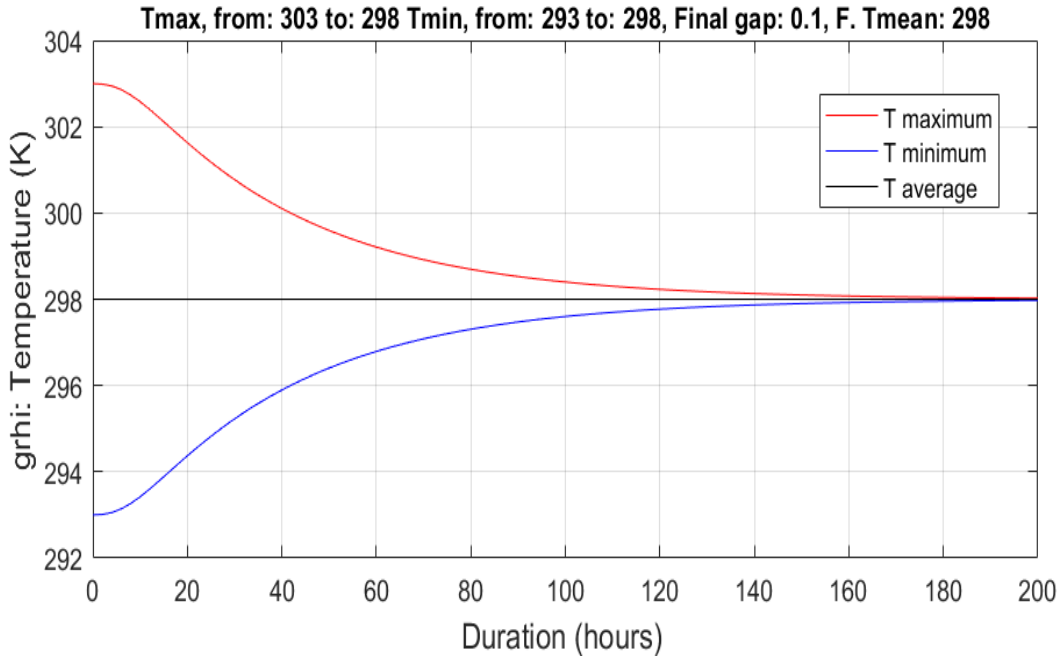


Figure 9: Convergence of the smoothing process

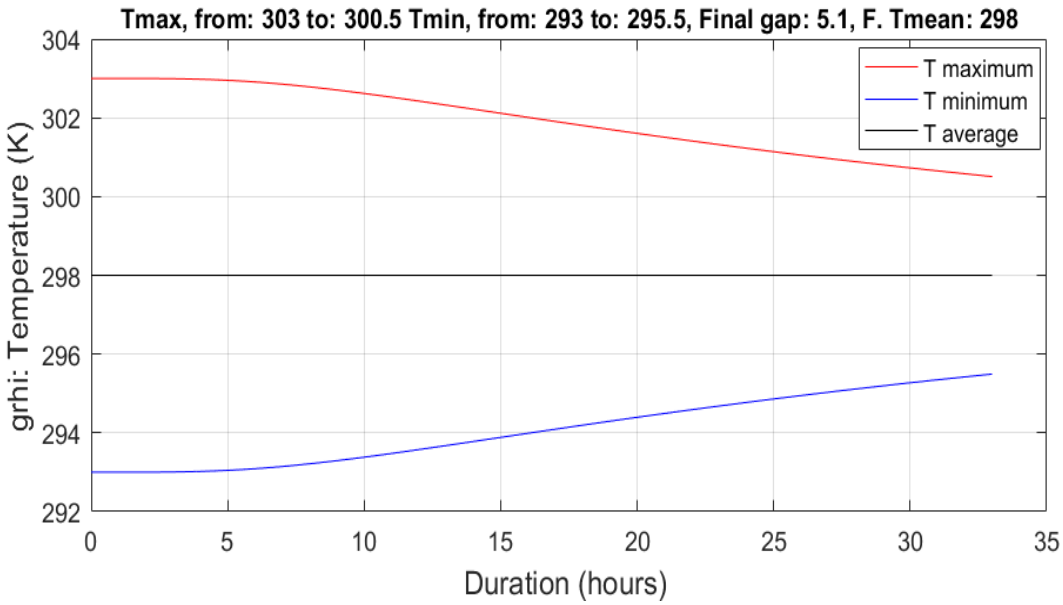


Figure 10: Beginning of the time evolution curve

Homogeneity of the temperature: after 33 hours, the gap is decreasing by half.

```
Control parameter  pa6   : 1
conde.m conductivity     : 2 W/(mK)
Specific heat            : 1000 J.m-3.K-1
Specific mass            : 2500 kg.m-3
Initial temperature      : 303 K
Wall base temperature    : 293 K
Domain dimensions        : 1 m x 0.1 m x 2 m
Mesh Dimension           : 21 x 42
Analyzed period          : 33. hours
Number of iterations     : 30
```

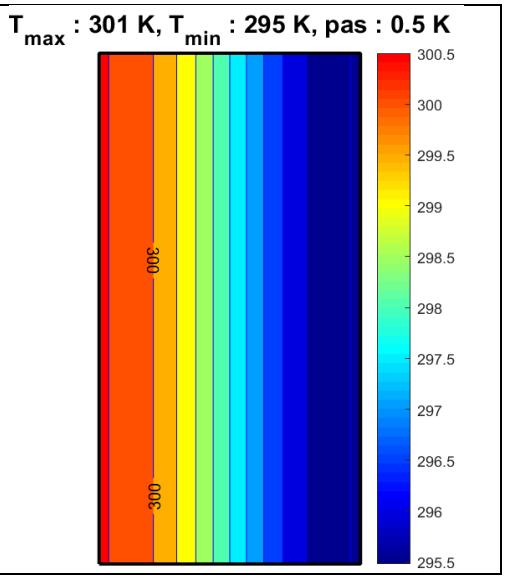


Figure 11: Evolution of the temperature field in a smoothing process

We follow with 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*, its mean value is 296 *K*.

```
Control param.  pa6 : 2
Convection coeff. h : 25 W/(m2K)
Specific heat       : 1000 J.m-3.K-1
Specific mass       : 2500 kg.m-3
Air temperatures    : 300  300  300 K
Initial temperature : 280 K
conde.m conductivity: 1 W/(mK)
Domain dimensions   : 1 x 0.1 x 2 m
Mesh size           : 20 x 40
N. virtual nodes t3 : 4
Number of fixations : 0
Domain heat capacity: 500000 J/K
Diffusivity         : 4e-07 m2/s
Total duration      : 100 h
Number of iterations: 50
N. of convective el.: 120
Virtual conv. nodes : 862  863  864  865
Stat. Ntca no nf t3 : 865  861    0  4
Cpu                 : 5.13 sec.
```

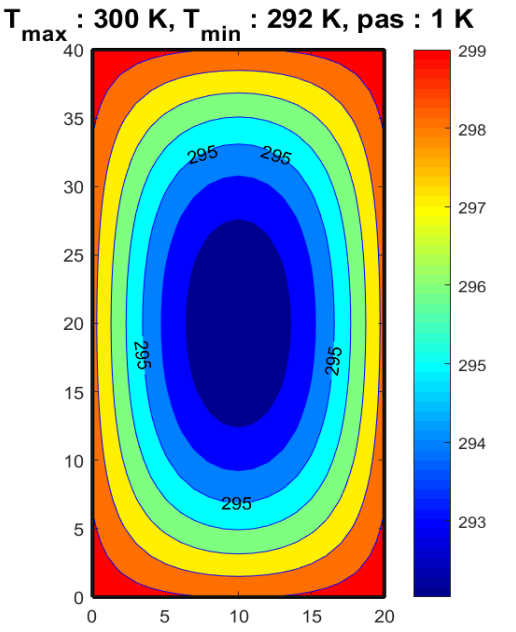


Figure 12: Evolution of the temperature field in a heating operation

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.

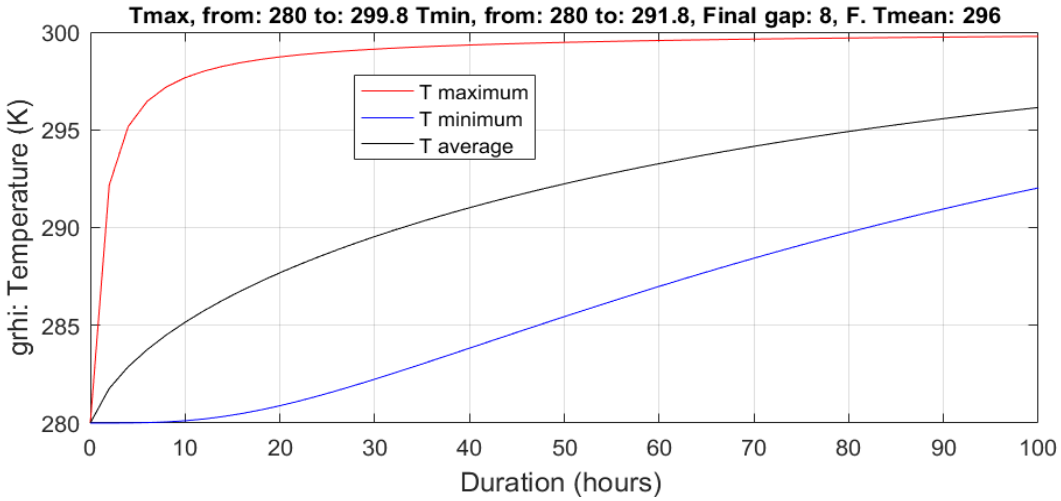


Figure 13: Evolution of max, min and mean temperatures in a heating operation

Using a 8 x 16 mesh, we can also represent the element heat flows using arrow whose length is proportional to their magnitude. This graphical output is performed in the function *Hflo.m* (*Table 21*) The elements are shown if there are no more than ten elements in the x direction.
The arrows are computed and drawed with or without the visualizaton of the shrink elements. The element are drawed with the Matlab© function *mesh.m* (*Table 22*).

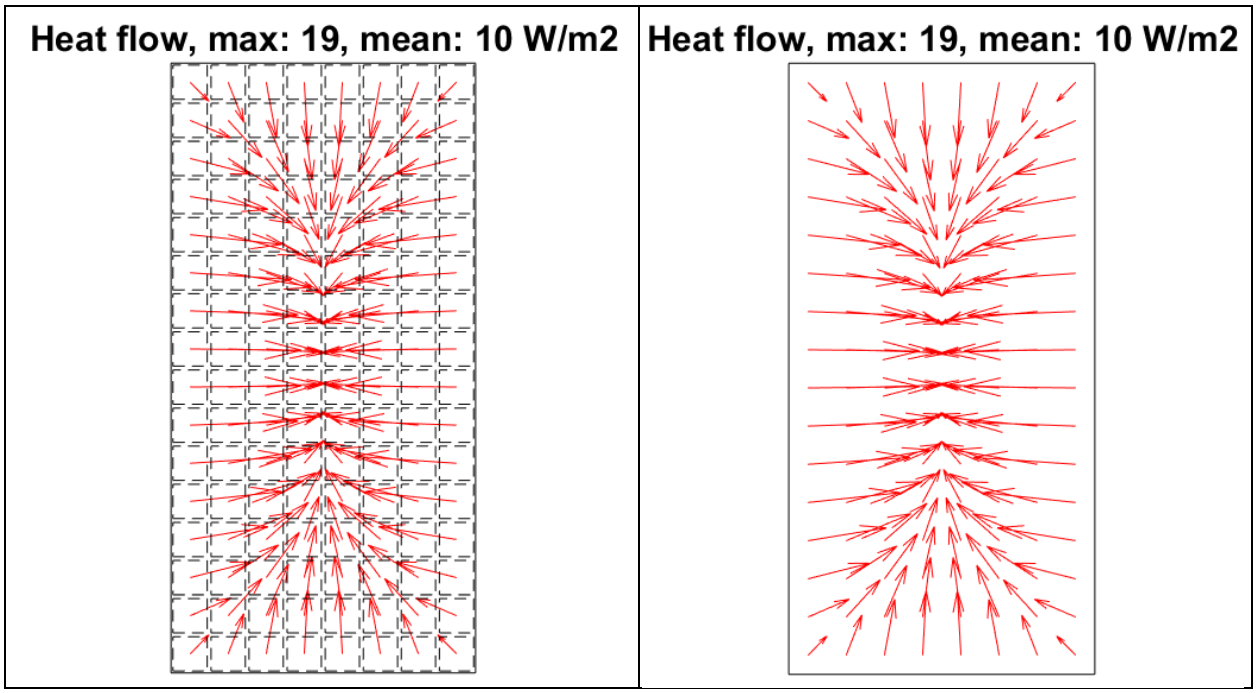


Figure 14: Heat flow represented with (left), or without (right) shrink elements

Function Matlab© *CoKr34.m* for convection in transient appnications

```
 1  function[K] = CoKr34(nx,ny,ep,he,hh,pa6)
 2  Kelc  = [2 1 -3;1 2 -3;-3 -3 6]*hh*ep*he/ny/6;        % Elem. conv. matrix
 3  if pa6 == 2
 4      Ntca  = (nx+1)*(ny+1)+4;                          % Mesh nx x ny elements
 5      ntv4  = [Ntca-3 Ntca-2 Ntca-1 Ntca];% Number. convective virtual nodes
 6      lc    = locc4(nx,ny,ntv4); % Local. of the convection matrices 4 sides
 7              disp(['Virtual conv. nodes : ',num2str(ntv4)])
 8  else
 9      Ntca  = (nx+1)*(ny+1)+3;                          % Mesh nx x ny elements
10      ntv3  = [Ntca-2 Ntca-1 Ntca];  % Numbering of convective virtual nodes
11      lc    = locc(nx,ny,ntv3);  % Local. of the convection matrices 3 sides
12              disp(['Virtual conv. nodes : ',num2str(ntv3)])
13  end
14  if nx*ny < 19; disp (lc); end
15  K     = zeros(Ntca,Ntca);% Dimension of K including fixed DOF & add. nodes
16  for n = 1:size(lc,1);for i=1:3;for j=1:3  % Assembling conv. matrices Kelc
17            K(lc(n,i),lc(n,j))=K(lc(n,i),lc(n,j))+Kelc(i,j);end;end;end
18  end
```

Table 19: Matlab© function CoKr34.m

Function Matlab© *grhi.m* for the drawing of evolution in transient appnications

```
 1  function [] = grhi(ni,dt,tsmax,tsmin,tmoy)      % Evolution of temperatures
 2  tem          = (0:ni)*dt/3600/ni;         % Time steps for the time graphics
 3  figure('Position',[100 100 700 300]);
 4  plot  (tem,tsmax','r');hold on;
 5  plot  (tem,tsmin','b');hold on;
 6  plot  (tem,tmoy  ,'k');hold on;grid on
 7  legend('T maximum ','T minimum ','T average')
 8  xlabel('Duration (hours) ','fontsize',12)
 9  ylabel('grhi: Temperature (K)','fontsize',12)
10  title (['Tmax, from: ',num2str(round(tsmax(1)*10)/10),' to: ',...
```

```
    num2str(round(tsmax(ni)*10)/10),' Tmin, from: ',...
    num2str(round(tsmin(1)*10)/10),' to: ',...
    num2str(round(tsmin(ni)*10)/10),', Final gap: ',...
    num2str(round((tsmax(ni)-tsmin(ni))*10)/10),' K, Tmean: ',...
    num2str(round(tmoy(ni)*10)/10),' K'],'fontsize',10);
end
```

Table 20: Matlab© function grhi.m

Matlab© function *Hflo.m:* drawing arrow in a quadrilateral element

```
function [xyz] = Hflo(nx,ny,xyz,lK,tca,co)
ii=0;X=zeros(nx*ny,1);Y=zeros(nx*ny,1);u=zeros(nx*ny,1);v=zeros(nx*ny,1);
xx=zeros(4,1);yy=zeros(4,1);te=zeros(4,1);
for i  = 1:nx
    for j  = 1:ny
        ii = ii+1;
        for k=1:4
            X(ii) = X(ii)+xyz(lK(ii,k),1)/4; % x coord of the elem. center
            Y(ii) = Y(ii)+xyz(lK(ii,k),2)/4; % y coord of the elem. center
            xx(k) = xyz(lK(ii,k),1);    % Compute 4 vertices x coordinates
            yy(k) = xyz(lK(ii,k),2);    % Compute 4 vertices y coordinates
            te(k) = tca(lK(ii,k));    % Compute 4 elem. nodal temperatures
        end
        Jacob = 1/4*[xx(2)+xx(3)-xx(1)-xx(4) yy(2)+yy(3)-yy(1)-yy(4);
                     xx(4)+xx(3)-xx(1)-xx(2) yy(4)+yy(3)-yy(1)-yy(2)];
        Jm1   = Jacob^(-1)*co(ii);
        u(ii) = -Jm1(1,:)/4*[-1 1 1 -1;-1 -1 1 1]*te;
        v(ii) = -Jm1(2,:)/4*[-1 1 1 -1;-1 -1 1 1]*te;
    end
end
gm = [max(sqrt(u.*u+v.*v)) mean(sqrt(u.*u+v.*v))];scale=2;
disp(['Elem. heat flow max : ', num2str(gm(1)),', mean: ',...
    num2str(gm(2)),' W/m2'])
figure;quiver(X,Y,u,v,scale,'r');hold on;
plot([xyz(ny*(nx+1)+1,1) xyz((nx+1)*(ny+1),1) xyz(nx+1,1) xyz(1,1) ...
    xyz(ny*(nx+1)+1,1)],[xyz(ny*(nx+1)+1,2) xyz((nx+1)*(ny+1),2)...
    xyz(nx+1,2) xyz(1,2) xyz(ny*(nx+1)+1,2)],'k');axis equal;hold on
if nx < 10;mesh(nx,ny,xyz,lK);end
title(['Heat flow, max: ', num2str(gm(1),2),', mean: ',num2str(gm(2),2),...
    ' W/m2'],'fontsize',15);axis off
end
```

Table 21: Matlab© function Hflo.m: Drawing arrow in a quadrilateral element

Function Matlab© *mesh.m* for the drawing of evolution in transient appnications

```
function []=mesh(nx,ny,xyz,lK)                % Drawing the shrinked mesh
nel    = nx*ny;X=zeros(5,1);Y=zeros(5,1);
sh     = 0.9;                          % Shrinking coefficient 0 < sh <= 1
for j  = 1:nel             % Nodes are numbered left - right, top - bottom
    ce = zeros(2,1);
    for i     = 1:4
        ce(1) = ce(1)+xyz(lK(j,i),1)/4;
        ce(2) = ce(2)+xyz(lK(j,i),2)/4;
        X(i)  = xyz(lK(j,i),1);
        Y(i)  = xyz(lK(j,i),2);
    end
    X(5)=X(1);Y(5)=Y(1);
    plot((1-sh)*ce(1)+sh*X,(1-sh)*ce(2)+sh*Y,'--k')
end
end
```

Table 22: Matlab© function mesh.m

Exercise

We propose to compute the cooling of a rectangular domain in a convective or radiative heat exchange process. As initial conditions, we impose a uniform temperature for the solid and identical conditions for the four sides of the domain in the convective or radiative exchange process.

Tutorial V: Isoparametric elements for heat transfer

This technique is based on the Coons patch developped in the frame of Computed Aided Design (CAD).

Procedures used for isoparametric elements

Function Matlab© *Kelu.m* for isoparamatric evaluation of the conductivity matrix

```
 1  function [K] = Kelu(xyz,lo)
 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  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;
11      dt   = ft * Q;
12      area = area          + sqrt(dot(cross(ds,dt),cross(ds,dt)))/4;
13      J    = [fs*Q(:,1) ft*Q(:,1);fs*Q(:,2) ft*Q(:,2)];
14      K=K+((J^(-1)*gra)'*J^(-1)*gra)*sqrt(dot(cross(ds,dt),cross(ds,dt)))/4;
15  end                                  % disp(['Patch area : ',num2str(area)])
16  end
```

Table 23: Matlab© function Kelu.m

This Matlab© function allows computing the conductivity matrix [*K*] (output of the function) of an isoparametric quadrilateral with a bilinear temperature field. To obtain the conductivity matrix, this result has to be multiplied by the conductivity coefficient *k* and the thickness *t*.
To compute a conductivity matrix, we use the matrix [*xyx*] of element coordinates (first argument of the function) and the geometric definition *lo* (second argument of the function *Kelu.m*) of the element, for instance, the positions of its four nodes in the coordinates matrix. A direct computation of the conduction matrix is given in *Table 24*, using explicit definitions of the coordinates and localization matrices.

Input	`xyz =[0 0 0;1 0 0;1 1 0;0 1 0];`
	`lo=[1 2 3 4];`
	`[K]=Kelu(xyz,lo)*6`	

Output	`Patch area : 1`	`K =  8  -2  -4  -2`
		`-2  8  -2  -4`
		`-4  -2  8  -2`

		-2	-4	-2	8

Table 24: Numerical integration of the conductivity matrix

To obtain the true conductivity matrix, it is necessary to multiply this result by $k\ e\ /\ 6$, where k is the conductivity coefficient and e the thickness.

The computation of conductivity matrix of isoparametric elements is now introduced in the procedure of *Table 1*. It provides the result of *Table 25*. As expected, when we run problems including rectangular domain, we obtain the same results as before. The following procedure exhibits the main characteristics of a finite element program (see the comments of lines 1 – 8).

Matlab procedure *pp_visopa_conduction.m* including isoparametric elements

```
1  nx    = 8;ny =nx*2;nel = nx*ny;no = (nx+1)*(ny+1);K=zeros(no,no);   % Mesh
2  disp('.......................')
3  co    = conde(nx,ny); th = 1;gap = 3;            % Material characteritics
4  xyz   = Nxyz (nx,ny);                        % Geometry ; nodal coordinates
5  lK    = loca(nx,ny);                       % Topology ; elements connections
6  for n = 1:nel                                    % Loop on the nel elements
7      Kel  = th*co(n)*Kelu(xyz,lK(n,:));     % Element conductivity matrices
8      for i=1:4;for j=1:4;K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+Kel(i,j);
9          end;end;            % Assembling the nel conductivity matrices Kel
10 end
11 % 1. Fixed temperaturess on part of the horizontal sides, lines 11 - 17
12 tb    = 270; tt = tb+50;                              % Boundary conditions
13 na    = max(1,round(nx/2));if na>(nx+1);na=nx+1;end;nu=no-2*na;% Prescr. T
14 % na    = nx+1;if na>(nx+1);na=nx+1;end;nu=no-2*na;% Prescr. T
15 K21   = K(na+1:nu+na , 1       : na);K22 = K(na+1:nu+na , na+1    : nu+na);
16 K23   = K(na+1:nu+na , nu+na+1 : nu+na*2);ar=ones(na,1);
17 tca   = [ar*tt;K22\(-K23*ar*tb-K21*ar*tt);ar*tb]; % Solution of the system
18 % % 2. Fixed temperaturess on the whole horizontal sides, lines 18 - 23
19 % tb    = .5; tt = .5;n1 = nx+1;nu=no-2*n1;
20 % disp(['Fix. nodes /hor side: ',num2str(n1)])
21 % K21   = K(n1+1:nu+n1 , 1       : n1);K22 = K(n1+1:nu+n1 , n1+1 : nu+n1);
22 % K23   = K(n1+1:nu+n1 , nu+n1+1 : nu+n1*2);ar=(nx:-1:0);
23 % tca   = [(ar*tt)';(K22\(-K23*(n1-1-ar)'*tb-K21*ar'*tt));((nx-ar)*tb)'];
24 grisc (nx,ny,tca,xyz,gap);axis off;                  % Output 1: isotherms
25 Hflo(nx,ny,xyz,lK,tca,co);%mgra(nx,ny,xyz,lK,tca);
26 xlabel(['nx : ',num2str(nx),' na : ',num2str(na)]);axis off;
27 figure('Position',[10 50 1200 500]);per=(1:(nx+ny)*2)'; % Output 2: Heat f
28 gperi = cageco(nx,ny,K*tca);bar(gperi,'k');grid on;
29 title(['Bottom flow: ',num2str(sum(gperi(1:nx+1)),'%0.3g'),...
30     ' W, top flow: '  ,num2str(sum(gperi(nx+ny+1:ny+2*nx+1)),'%0.3g'),...
31     ' W'],'fontsize',15)
32 disp(['Base temperature    : ',num2str(tb,'%0.3g'),' K']) % Output 3: Disp
33 disp(['Top  temperature    : ',num2str(tt,'%0.3g'),' K'])
34 disp(['Mesh size           : ',num2str(nx),' x ',num2str(ny)])
35 disp(['Fix. nod. 2 hor. fa.: ',num2str(na*2)])
36 disp(['Diss tcaT*(K*tca)/2 : ',num2str(tca'*(K*tca)/2,'%0.3g'),' WK'])
```

Table 25: Matlab procedure pp_visopa_conduction.m for isoparametric elements

This procedure needs the definition of nodes coordinates. It is performed in the Matlab© function *Nxyz.m*. The arguments correspond to the definition of the mesh. The definition of the domain geometry is included inside the function. In the *Table 26*, there is a first sequence of 13 lines corresponding to the function itself. It is followed by proposals for other shapes that can replace *line 3*. At the end there are four lines enabling to display the patch geometry.

Function Matlab© *Nxyz.m* for defining the node coordinates

```
function [xyz] = Nxyz(nx,ny)
ii = 0;nn=(nx+1)*(ny+1);xyz = zeros(nn,3);nc=2;% Nodes Coons patch nx x ny
P = [0 0 0; 2 0 0; 1.5 2 0; 0.5 2 0];               % Trapezoidal domain
for i = ny:-1:0
   for j = 0:nx
       t = i/ny; s = j/nx; ii = ii +1;
       for c=1:nc
           xyz(ii,c)= (1-s)*(1-t)*P(1,c)+ s*(1-t)    *P(2,c)+...
                       s*t       *P(3,c)+(1-s)*t     *P(4,c);
       end
   end
end
end

% P = [0 0 0; 1 0 0; 1 2 0; 0 2  0];          % Vertical rectangular domain
% P = [0 0 0; 4 0 0; 4 2 0; 0 2  0];        % Horizontal rectangular domain

% P = [0 0 0; 2 0 0; 1.5 1.5 0; 0.5 2  0; ];         % Quadrilateral domain
% P = [0 0 0; 2 0 40; 2 2 0; 0 2  40 ];nc=3;             % 3D square domain
% P = [0 0 0; 2 0 0; 2 2 0; 0 2  0 ];                  % Flat square domain
% P = [0 0 0; 10 0 0; 10 10 0; 0 10  0; ];            % Large square domain
% P = sqrt(2)*[0 -1 0; 1 0 0; 0 1 0; -1 0  0; ];%45° rotated square domain
% P = [.5 0 0; .5 0 0; 1 2 0; 0 2  0; ];% Triangular domain: M Ballesteros

% disp(['Coordinates  P1, P2 : ',...
%     num2str([P(1,1) P(1,2) P(2,1) P(2,2)],2),' m'])
% disp(['Coordinates  P3, P4 : ',...
%     num2str([P(3,1) P(3,2) P(4,1) P(4,2)],2),' m'])
```

Table 26: Matlab© function Nxyz.m

In the *lines 4* and *5* of the procedure of *Table 25*, the functions of *Table 26* and *Table 5* outline the usual basics of a finite element model: matrix [*xyz*] containing the nodal coordinates and matrix [*lK*] giving the element localizations. The typical isolines of a scalar component of the finite element solution are displayed in the *grisc.m* function (*Table 27*).

Function Matlab© *grisc.m* for isoparamatric evaluation of the conductivity matrix

```
function [] = grisc(nx,ny,z,xyz,gap)
figure('Position',[1 1 600 512]);
my = ny+1;no=(nx+1)*(ny+1);ii=0;jj=0;xx=zeros(my,nx+1);yy=zeros(my,nx+1);
tn          = ones(my,nx+1)*z(1);
for i       = 1:ny;for j = 1:nx+1;ii = ii+1; tn(i,j) = z(ii);end;end
for i       = 1:my
    for j   = 1:nx+1;jj=jj+1;xx(i,j)=xyz(jj,1);yy(i,j)=xyz(jj,2);end
end
tn(my,:)    = z(ii+1:no );
              br56;colormap(br56);                    % Color map definition
[CS,H]      = contourf(xx,yy,tn,(0.:gap:max(z)),'b');hold on;axis equal
              clabel(CS,H,[275 280 285 290 295 300 305 310 315 320]);
plot  ([xx(my,1) xx(my,nx+1) xx(1,nx+1) xx(1,1) xx(my,1)],...
       [yy(my,1) yy(my,nx+1) yy(1,nx+1) yy(1,1) yy(my,1)],'k',...
       'LineWidth',2);hold on;axis equal;colorbar
title (['T_m_a_x : ',num2str(round(max(z))),' K, T_m_i_n : ',...
num2str(round(min(z))),' K, pas : ',num2str(gap),' K'],'fontsize',15);
end
```

Table 27: Matlab© function grisc.m for the drawing of the isotherms

Problems dealing with other patch shapes.

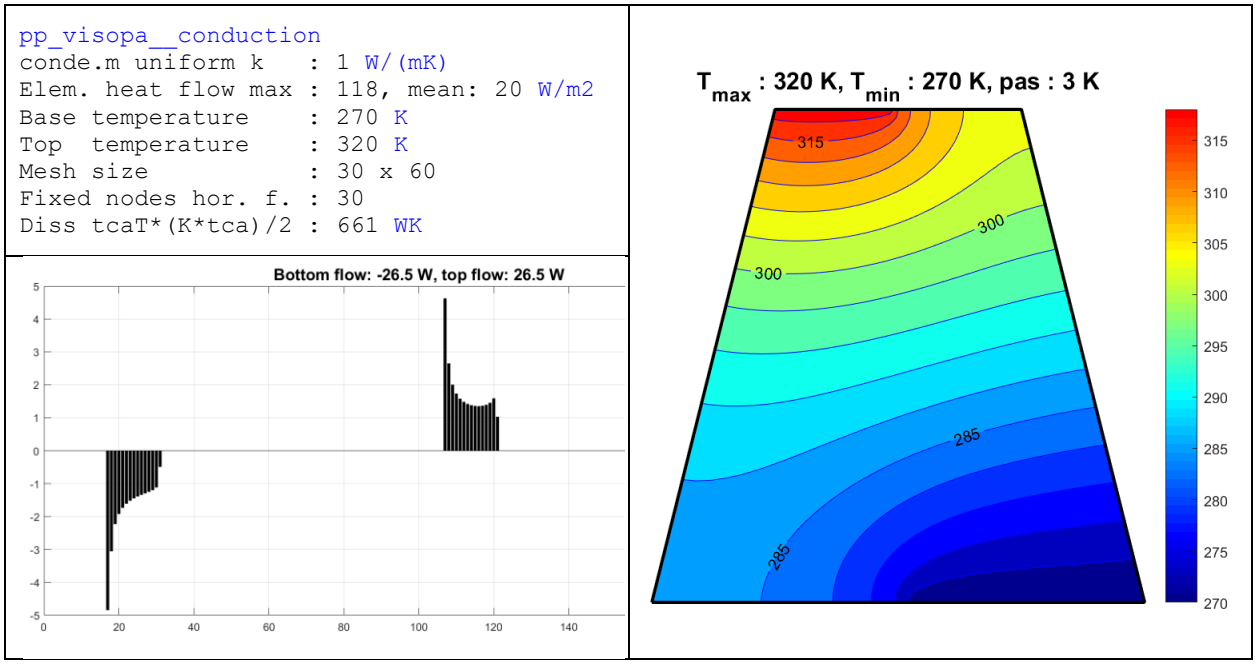


Figure 15: Imposed temperatures on a part of the horizontal faces (see Figure 4)

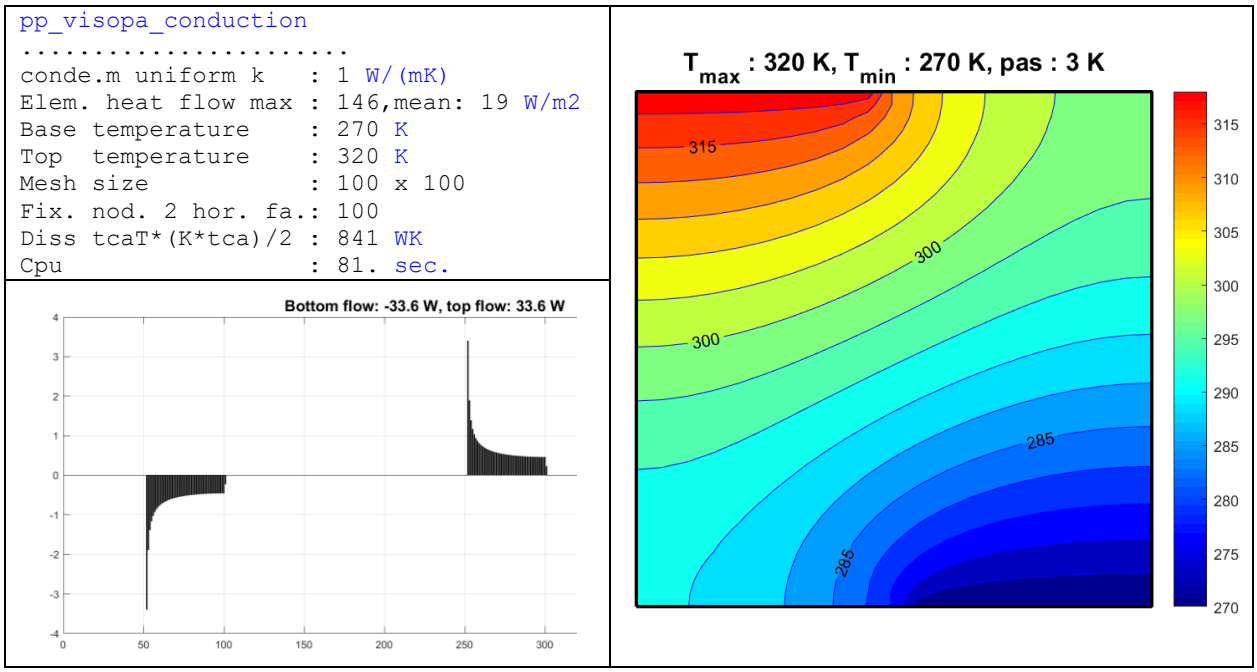


Figure 16: Imposed temperatures on a part of the horizontal faces (see Figure 4)

Name of the procedure	Aim	Matlab© functions	
pp_conduction.m ***Table 1***	Heat transfers in conduction with only prescribed teùperatures.	*cageco.m* *conde.m* *loca.m*	***Table 3*** ***Table 4*** ***Table 5***

(*pp_visopa_conduction.m* *Table 25*)		*grisb.m* *br56.m*	***Table 6*** ***Table 7***
pp_visopa_convection.m ***Table 8***	Heat transfers including convection and using isoparametric elements	*conde.m* *Nxyz.m* *Coco.m* *loca.m* *locc.m* *Kelu.m* *Hflo.m* *grisc.m* *br56.m* *locc2.m*	***Table 4*** ***Table 26*** ***Table 10*** ***Table 5*** ***Table 12*** ***Table 23*** ***Table 21*** ***Table 27*** ***Table 7*** ***Table 11***
pp_radiation.m ***Table 14***	Heat transfers including radiation	*conde.m* *loca.m* *locc.m* *ItKr.m* *grisb.m* *br56.m* *radit.m*	***Table 4*** ***Table 5*** *Table* ***12*** ***Table 15*** ***Table 6*** ***Table 7*** ***Table 16***
pp_ transient.m *Table 18* *pp_visopa_transient* (not available in this document because it contains the solution of the final exercice)	Transient heat transfers	*conde.m* *CoKr34.m* *loca.m* *grisb.m* *br56.m* *grhi.m* *locc4.m*	***Table 4*** ***Table 19*** ***Table 5*** ***Table 6*** ***Table 7*** ***Table 20*** *Table* ***13***

Table 28: Matlab© procedures and their linked functions

Exercise

We propose to compare a rectangular and a non rectangular shape on one of the previous problems.

Final exercise

We propose to compute the capacity matrix for an isoparametric element and to adapt the procedure of *Table 18*.

References

[Beckers & Beckers 2014] Beckers B., Beckers P., "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 P., Beckers B., "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 P., Beckers B., "A 33 line heat transfer finite element code", *Report Helio_010_en*, 2016. www.heliodon.net/heliodon/documents.html

[Beckers 2017] Beckers B., "Géométrie assistée par ordinateur", *Architecture et Physique Urbaine - ISA BTP Université de Pau et des Pays de l'Adour,* 2017
http://www.heliodon.net/downloads/Beckers_2017_10_15_GAO.pdf

[Coons 1967] Coons Steven A., "Surfacs for Computer-Aided Design of Space Forms", Project MAC-TR-41, Massachusetts Institue of Technology

[Courant 1943] Courant R. "Variational methods for solution of problems of equilibrium and vibrations", Bull. Amer. Math. Soc. 49 (1943), no. 1, 1—23

[Courant & Hilbert 1953] Courant R., Hilbert D., "Methods of Mathematical Physics", Volume 1, Library of Congress Catalog Card Number 53-7164, ISBN 0 470 17952 X, 1953

[Ergatoudis, Irons & Zienkiewicz 1968] Ergatoudis I., Irons B.M., Zienkiewicz O.C., "Curved, Isoparametric, "Quadrilateral" elements for finite element analysis", Int. J. Solids Structures. 1968, Vol. 4, pp. 31 to 42.

[Fish, Belytschko 2007] Fish J., Belytschko T., "A First Course In Finite Elements", (Wiley, 2007)

[Fraeijs de Veubeke *et al* 1972] Fraeijs de Veubeke B., Sander G., Beckers P., "Dual analysis by finite elements linear and non linear applications", AFFDL_TR_72_93, 1972

[Fraeijs de Veubeke & Hogge 1972] Fraeijs de Veubeke B., Hogge M., "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 B., Beckers P., Canales E., Galaz S., "Principios variacionales en conducción de calor", Informe del departamento de Ingeniería Mecánica de la Escuela de Ingeniería, Universidad de Concepción, Chile, 1977

[Irons 1966] Irons B., "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.")

[Lewis *et al* 2004] Lewis R.W., Nithiarasu P., Seetharamu K.N., "Fundamentals of the Finite Element Method for Heat and Fluid Flow", John Wiley & Sons Ltd, 2004, p. 356

[Sander & Beckers 1977] Sander G., Beckers P., "The influence of the choice of connectors in the finite element method", International Journal for Numerical Methods in Engineering, vol. 11, 1491-1505 (1977)

[Szabó & Babuska 1991] Szabó B., Babuska I., "Finite element analysis", John Wiley & sons, 1991

[Zienkiewicz 1971] Zienkiewicz, O.C., "The Finite Element Method in Engineering Science", McGraw-Hill. London, 1971

List of tables and figures

Contents