Tutorials of five lectures given in Montevideo on the 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 very short programs, written in Matlab©, which are progressively developed in conduction, convection, radiation and transient regime. Each step is complemented by an exercise that will allow the reader becoming familiar with the main features presented in the lectures.

Tutorial I: Conductive Heat Transfer

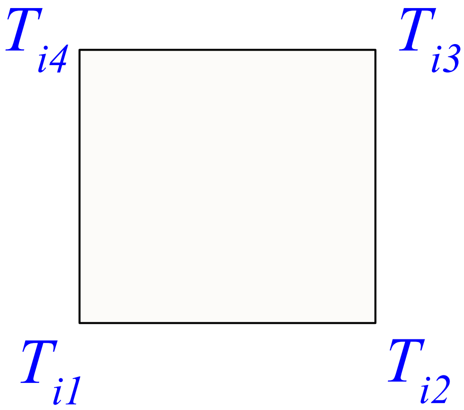
In each sub-domain or finite element numbered i, a Rayleigh-Ritz is applied. The temperature τ_i of element i is discretized by bilinear polynomial functions associated to j (four) nodal temperatures T_{ij} of the vertices.

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

Explicitely we have:

$$\tau_i = T_{i1}\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right) + T_{i2}\frac{x}{a}\left(1-\frac{y}{b}\right) + T_{i3}\frac{x}{a}\frac{y}{b} + T_{i4}\left(1-\frac{x}{a}\right)\frac{y}{b} \tag{1.2}$$

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. In *Figure 1*, we 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 the finite elements of the domain are added to build the discretized global functional $I\,(T)$:

$$< I(T) = \sum_{i=1}^{nel} (\int_{\Omega j} \frac{1}{2} k_i \ (grad \sum_{j=1}^{4} T_{ij} f_{ij})^T . \, grad \sum_{j=1}^{4} T_{ij} f_{ij} \ d\Omega_i + \int_{S_{2i}} \bar{q}_n \sum_{j=1}^{4} T_{ij} f_{ij} \ dS_i) > \qquad (1.3)$$

After introducing the polynomial trial functions given in (1.1), we can write (1.3) in matrix form:

$$< I(T) = \sum_{i=1}^{nel} [T_i]^T [K_i][T_i] + [T_i]^T [F_i] > \qquad (1.4)$$

The last term of (1.4) $[F_i]$ 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 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 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[1], we can then write the relation between local $[T_i]$ and global nodes $[T]$.

$$[T_i] = [L_i][T] \qquad (1.5)$$

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

$$L_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (1.6)$$

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

[1] This formulation facilitates the mathematical presentation of the localization process. However, in practice, a more efficient method oriented to progamming will be used.

$$[L_4]^T[K_4][L_4] \tag{1.7}$$

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.

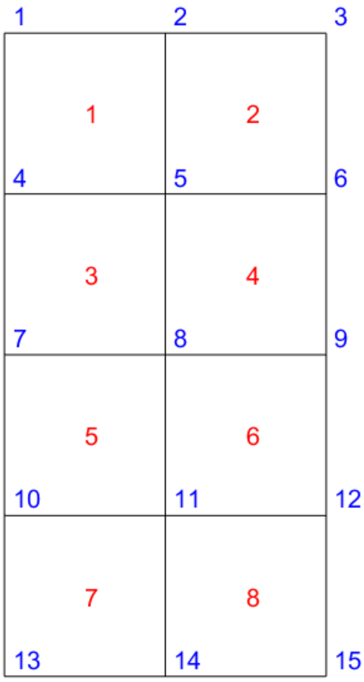


Figure 2: Numbering and sequence of nodes and elements

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.

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

The next step is to get the vector $[T]$ out of the sum in (1.8).

$$I(T)=[T]^T\left(\sum_{i=1}^{nel}\left([L_i]^T[K_i][L_i][T]+[L_i]^T[F_i]\right)\right) \tag{1.9}$$

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

$$\begin{gathered}\sum_{i=1}^{nel}\left([L_i]^T[K_i][L_i]\right)[T]=-\sum_{j=1}^{nel}[L_i]^T[F_i]\\ \text{with } [K]=\sum_{i=1}^{nel}[L_i]^T[K_i](L_i) \quad \text{and} \quad [F]=-\sum_{i=1}^{nel}[L_i]^T[F_i]\end{gathered} \tag{1.10}$$

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

$$[K]\ [T]=[F] \tag{1.11}$$

Temperatures gradients and heat flows

The temperature gradients are obtained by derivation of the element temperature field *Figure 1*:

$$\tau = T_1\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right)+T_2\frac{x}{a}\left(1-\frac{y}{b}\right)+T_3\frac{x}{a}\frac{y}{b}+T_4\left(1-\frac{x}{a}\right)\frac{y}{b} \tag{1.12}$$

It is easy to derive this polynomial expression:

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

$$\begin{bmatrix}\dfrac{\partial\tau}{\partial x}\\ \dfrac{\partial\tau}{\partial y}\end{bmatrix}=\frac{1}{ab}\begin{bmatrix}y-b & b-y & y & -y\\ x-a & -x & x & a-x\end{bmatrix}\begin{bmatrix}T_1\\T_2\\T_3\\T_4\end{bmatrix} \tag{1.13}$$

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

$$\begin{aligned}\frac{\partial\tau}{\partial x}&=\frac{1}{2a}\left((T_2+T_3)-(T_1+T_4)\right)\\ \frac{\partial\tau}{\partial y}&=\frac{1}{2b}\left((T_3+T_4)-(T_1+T_2)\right)\end{aligned} \tag{1.14}$$

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

$$\begin{bmatrix}q_x\\q_y\end{bmatrix}=-k\begin{bmatrix}\dfrac{\partial\tau}{\partial x}\\ \dfrac{\partial\tau}{\partial y}\end{bmatrix} \tag{1.15}$$

The heat flow (*Wm^{-2}*) are in the opposite direction of the gradients and proportional with the factor *k* in homogeneous isotropic mediums.

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 only 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:

$$k\frac{\Delta T}{\Delta y} = k\frac{50}{2} = 25\ W \tag{1.16}$$

Matlab procedure *pp_ conduction.m*

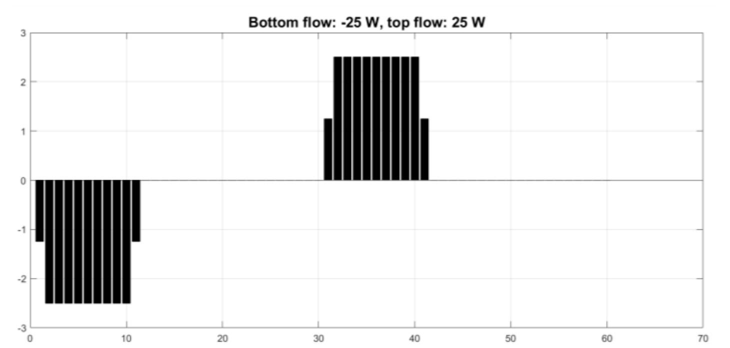
```
nx    = 50;ny = nx*2;nel = nx*ny;no = (nx+1)*(ny+1);th = 1;tini=tic;% Mesh
co    = conde(nx,ny);
Kel   = th/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];     % element K
lK    = loca(nx,ny);K = zeros(no,no);
for n = 1:nel;for i=1:4;for j=1:4    % Assembling nel conduct. matrices Kel
          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=1.;
% nb    = nx+1;nu =no-2*nb;                              % Imposed temperature
nb    = max(1,round(nx/5));if nb>(nx+1);nb=nx+1;end;nu=no-2*nb;
disp(['Fixed portion horiz : ',num2str(nb/nx,3)])
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
if nx <51;figure;Tg(nx,ny,lK,tca);hold on;end  % Drawing temperature grad.
if nx <51;figure;Hf(nx,ny,lK,tca,co);hold on;end       % Drawing heat flows
figure('Position',[10 50 1200 500]);per=(1:(nx+ny)*2)';    % Output 4: 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'])
disp(['Cpu                  : ',num2str(toc(tini),'%0.3g'),' sec.'])
```

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

The procedure of *Table 1* is providing three outputs: two figures and the 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 of the domain 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 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 (1.16): 25 W divided by the number of elements 25/nx W for inner nodes and half for the others: 25/(2nx) W.

```
pp_conduction
conde.m conductivity: 1 W/(mK)
Base temperature     : 270 K
Top  temperature     : 320 K
Mesh size            : 10 x 20
Fixed nodes hor. f.  : 22
Diss tcaT*(K*tca)/2  : 625 W
Cpu                  : 0.24 sec.
```



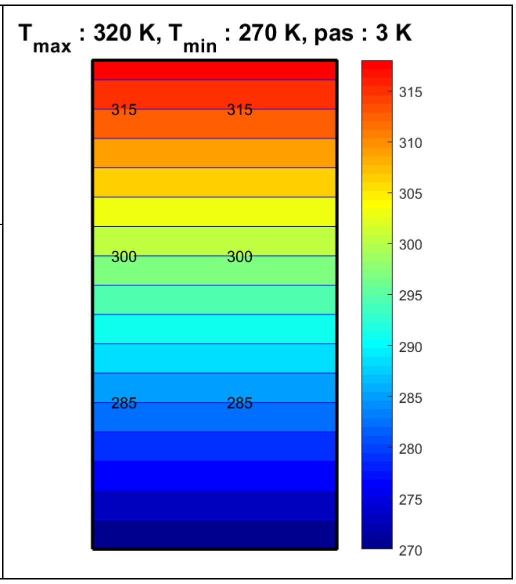


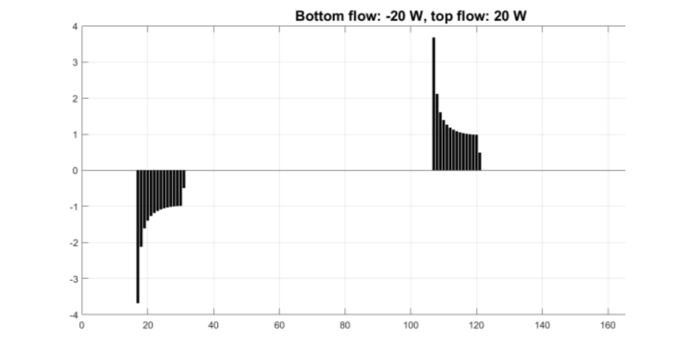
Figure 3: Example with imposed temperatures producing a vertical gradient

We can swap the status of *line 9 & line 10* in the procedure (*Table 1*) from “comment” to “enabeled” in order to modify the distribution of the prescribed temperatures on the horizontal faces. With *line 9* enabeled, we obtain the *Figure 3* while, with *line 10* enabeled, we obtain the *Figure 4* and the *Figure 5* for a very fine mesh involving about 13000 unknowns.

```
9   % nl     = nx+1;nu =no-2*nl;                                   % Imp. T
10  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, and the other set as comment).

```
pp_conduction
conde.m conductivity: 1 W/(mK)
Fixed portion horiz : 0.5
grisb.m, mesh dim.  : 30 x 60 m
Temperature gradient: 93.4, mean: 21.2 K/m
Heat flows          : 93.4, mean: 21.2 W/m2
Base temperature    : 270 K
Max  temperature    : 320 K
Mesh size           : 30 x 60
Fixed nodes hor. f. : 30
Diss tcaT*(K*tca)/2 : 500 WK
Cpu                 : 0.484 sec.
```



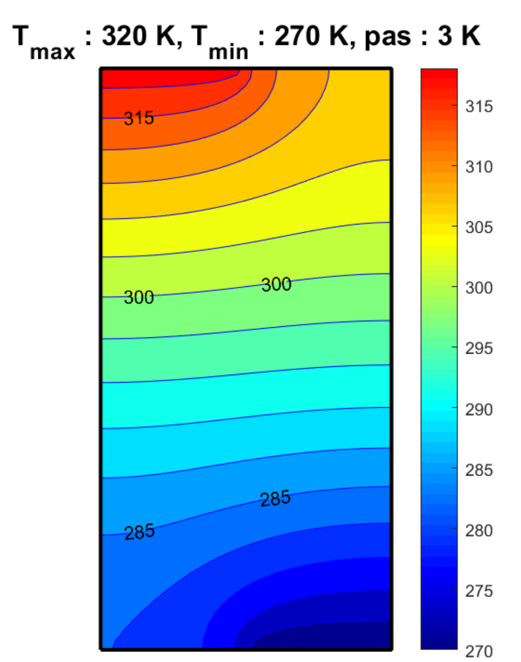
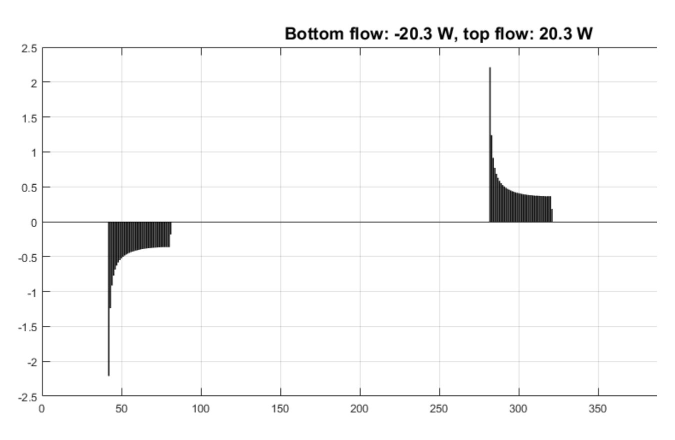


Figure 4: Imposed temperatures on a part of the horizontal faces (30 x 60 mesh)

```
pp_conduction
conde.m uniform k    : 1 W/(mK)
Fixed portion horiz  : 0.5
grisb.m, mesh dim.   : 80 x 160 m
Base temperature     : 270 K
Max  temperature     : 320 K
Mesh size            : 80 x 160
Fix. nod. 2 hor. fa.: 80
Diss tcaT*(K*tca)/2  : 508 WK
Cpu                  : 8.72 sec.
```



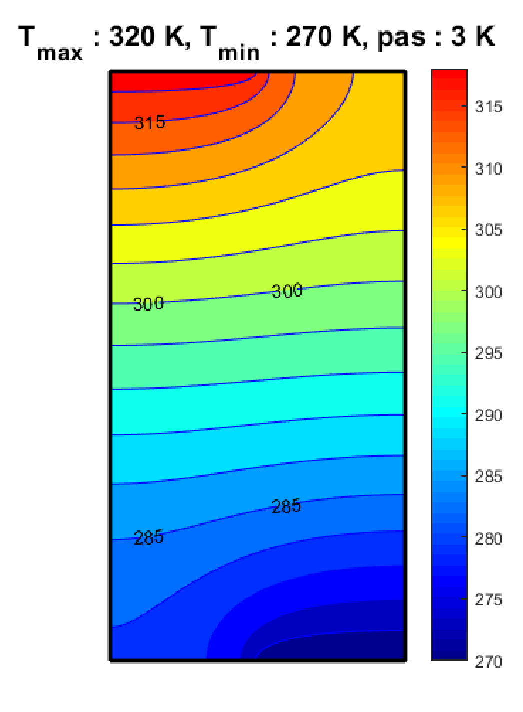


Figure 5: Imposed temperatures on a part of the horizontal faces (80 x 160 mesh)

Additional comments about the procedures

1. Principal procedure: *pp_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 3*)

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 4*)

The element nodes sequence shown in *Figure 1* is always the same and must be assumed 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 (1.10) and exhibits better computational performances. In Matlab© notation, the localization matrix of the 2 x 4 mesh of *Figure 2*, is:

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

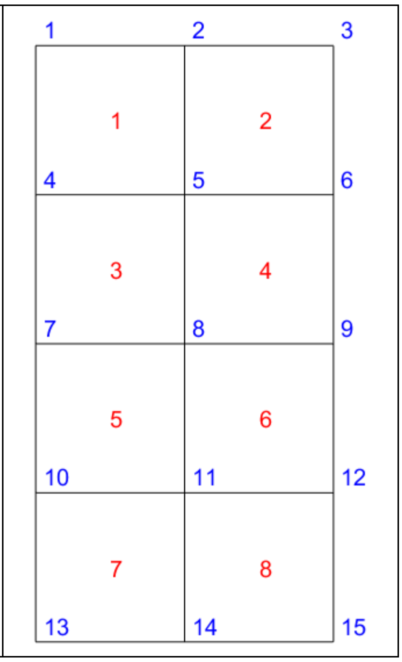


Table 2: Localization matrix for the 2 x 4 mesh of Figure 2

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 4Figure 3*.

Lines 10 – 14 : 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 $[T_3]$ for the bottom and in $[T_1]$ for the top.

$$[K]=\begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \quad ; \quad [K]\begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}=0 \tag{1.17}$$

$$[T_2]=[K_{22}]^{-1}\left(-[K_{23}][T_3]-[K_{21}][T_1]\right) \tag{1.18}$$

Lines 15 – 21 : 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 22 – 35: Output

The last lines of the procedure are devoted to the output, successively: isotherms (*line 22*, function *grisb.m*), temperature gradients isotherms (*line 23*, function *Tg.m*), heat flows (*line 24*, function *Hf.m*), visualization of the nodal heat flows (*lines 25 – 29*, function *cageco.m*) and a summary of data (*lines 30* to *35*).

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

Matlab© function *conde.m*

```
1    function [co] = conde(nx,ny)      % Treatment of non uniform conductivity
2    k   = 1 ;                                                         % W/(mK)
3    nel = nx*ny;                                 % Number of elements of the mesh
4    co  = ones(nel,1)*k;                                   % Constant conductivity
5    disp(['conde.m uniform k   : ',num2str(k,'%0.3g'),' W/(mK)'])
6    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  = 10 ;          % 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 2
     % 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 3: Matlab© function conde.m for defining non homogeneous conductivity

The above function is subdivided into four sequences. Here, the first one is enabled; the three others being disabled by using the "comment" command of Matlab©. To switch from one sequence to another, 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 4: Matlab© function loca.m for computing the localization matrix

The above function creates the localization matrix for *nx* x *ny* meshes defined in a vertical rectangle (1 m x 2 m). For the 2 x 4 mesh (*Figure 2*) the matrix is reproduced in *Table 2*. It is possible to enter directly a command like: lK=loca(2,4) in Matlab to generate this matrix.

Matlab© function *grisb.m* to draw isotherm lines

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

Table 5: Matlab© function grisb.m for drawing isotherms

To draw the isotherms it is not necessary to know the nodal coordinates. They are generated inside the function *grisb.m* (*line 5*), assuming that the size of each element is 1 x 1 m.

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
               0    0.9375    1.0000
               0    1.0000    1.0000
          0.0625    1.0000    0.9375
          0.1250    1.0000    0.8750
          0.1875    1.0000    0.8125
          0.2500    1.0000    0.7500
          0.3125    1.0000    0.6875
          0.3750    1.0000    0.6250
          0.4375    1.0000    0.5625
          0.5000    1.0000    0.5000
          0.5625    1.0000    0.4375
          0.6250    1.0000    0.3750
          0.6875    1.0000    0.3125
          0.7500    1.0000    0.2500
          0.8125    1.0000    0.1875
          0.8750    1.0000    0.1250
          0.9375    1.0000    0.0625
          1.0000    1.0000         0
          1.0000    0.9375         0
          1.0000    0.8750         0
          1.0000    0.8125         0
          1.0000    0.7500         0
          1.0000    0.6875         0
          1.0000    0.6250         0
          1.0000    0.5625         0
          1.0000    0.5000         0
          1.0000    0.4375         0
          1.0000    0.3750         0
          1.0000    0.3125         0
          1.0000    0.2500         0
          1.0000    0.1875         0
          1.0000    0.1250         0
          1.0000    0.0625         0
          1.0000         0         0];
end
```

Table 6: Matlab© function br56.m for defining a color bar

Matlab© function *Tg.m – temperature gradients*

```
function [] = Tg(nx,ny,lK,tca)
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);
nn = (nx+1)*(ny+1);xyz = zeros(nn,3);ii=0;
P  = [0 0 ; 1 0 ; 1 2 ; 0 2 ];                % Vertical rectangular domain
for i = ny:-1:0
   for j = 0:nx
       t = i/ny; s = j/nx; ii = ii +1;
       for c=1:2;xyz(ii,c) = s*(1-t)*P(2,c)+s*t*P(3,c)+(1-s)*t*P(4,c);end
   end
end
ii=0;
for i  = 1:nx                             % Loop on the nx columns of elemrnts
    for j  = 1:ny                         % Loop on the ny lines   of elemrnts
        ii = ii+1;                                 % Number of the element row
        for k=1:4                       % Loop on the 4 vertices of the element
            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); % xx contains the 4 vertices x coord.
            yy(k) = xyz(lK(ii,k),2); % yy contains the 4 vertices y coord.
            te(k) = tca(lK(ii,k));    % te contains the 4 vertices temperat
        end
        u(ii)     = -nx/2*[-1 1 1 -1]*te;     % x component of the gradient
        v(ii)     = -nx/2*[-1 -1 1 1]*te;     % y component of the gradient
```

```
        end
    end
    gm    = [max(sqrt(u.*u+v.*v)) mean(sqrt(u.*u+v.*v))];% grad: max & average
    scale = 2;
    disp(['Temperature gradient: ', num2str(gm(1),3),', mean: ',...
        num2str(gm(2),3),' K/m'])
    quiver(X,Y,u,v,scale,'b','LineWidth',1);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
    % mesh(nx,ny,xyz,lK);hold on;
    title(['Temp grad, max: ',num2str(gm(1),2),', mean: ',num2str(gm(2),2),...
        ' K/m'],'fontsize',15);axis off;hold on
end
```

Table 7: Matlab© function Tg.m for computing the temperature gradients

In the function *Tg.m*, the temperature gradient is computed in the center of the element and drawn as blue arrow oriented in the opposite direction to that of the gradient, its length beeing 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 (*nx* limited to 50 in the presented version of *pp_conduction.m*). The function is also displaying the maximum of the gradient modules and its average.

The function *Tg.m* is illustrated in *Figure 6* with and without the shrink mesh for a test equivalent to that of *Figure 4* or *Figure 5* but with a coarser mesh (4 x 8).

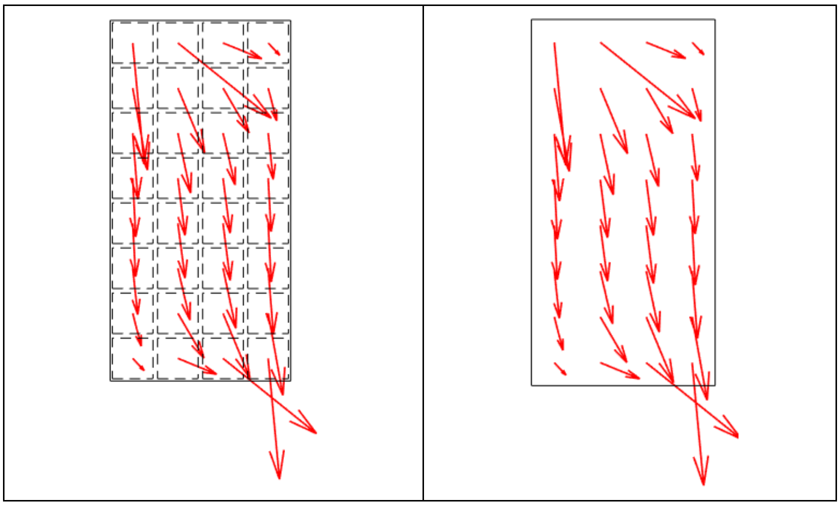


Figure 6: 2 x 4 mesh. Maximum temperature gradient : 36 K/m, average: 18 K/m

Matlab© function *Hf.m*

```
function [] = Hf(nx,ny,lK,tca,co)
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);
nn = (nx+1)*(ny+1);xyz = zeros(nn,3);ii=0;
P  = [0 0 ; 1 0 ; 1 2 ; 0 2 ];                    % Vertical rectangular domain
for i = ny:-1:0
   for j = 0:nx
       t = i/ny; s = j/nx; ii = ii +1;
       for c=1:2;xyz(ii,c) = s*(1-t)*P(2,c)+s*t*P(3,c)+(1-s)*t*P(4,c);end
   end
end
ii=0;
for i  = 1:nx                              % Loop on the nx columns of elements
    for j  = 1:ny                          % Loop on the ny lines   of elements
        ii = ii+1;                                  % Number of the element row
        for k=1:4                        % Loop on the 4 vertices of the element
            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); % xx contains the 4 vertices x coord.
                yy(k) = xyz(lK(ii,k),2); % yy contains the 4 vertices y coord.
                te(k) = tca(lK(ii,k));   % te contains the 4 vertices temperat
            end
            u(ii)      = -(nx*co(ii))/2*[-1 1 1 -1]*te;      % heat flow x comp.
            v(ii)      = -(nx*co(ii))/2*[-1 -1 1 1]*te;      % heat flow y comp.
        end
    end
    gm     = [max(sqrt(u.*u+v.*v)) mean(sqrt(u.*u+v.*v))];% grad: max & average
    scale = 2;
    disp(['Heat flows            : ', num2str(gm(1),3),', mean: ',...
        num2str(gm(2),3),' W/m2'])
    quiver(X,Y,u,v,scale,'r','LineWidth',1);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
    %     mesh(nx,ny,xyz,lK);hold on;
    title(['Heat flows, max: ',num2str(gm(1),2),', mean: ',num2str(gm(2),2),...
        ' W/m2'],'fontsize',15);axis off;hold on
    end
```

Table 8: Matlab© function Hf.m for computing the heat flows

Matlab© function *cageco.m*

```
function [gperi] = cageco(nx,ny,tca)
my          = ny + 1;    % Nodal values along the boundary without repetition
ii          = 0;gperi = zeros(2*(ny+nx),1);
for i       = ny*(nx+1)+1 : my*(nx+1)
    ii      = ii+1;gperi(ii) = tca(i);
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 9: Matlab© function cageco.m for selecting a nodal quantity along the boundary

This function allows drawing a nodal variable either the temperature either the second member of the system which corresponds to a heat quantity. The picture is generated with the Matlab© bar function (*line26* of the procedure *pp_conduction.m*, *Table 1*). An example of this kind of diagram concerning heat flow is shown in *Figure 4* or below, in *Figure 8*.

Exercise n°1: Conductivity coefficients

Using the Matlab© procedure and the functions presented in the tutorial, 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.

Function Matlab© *conde.m* for the definition of the conduction coefficients

```
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
```

Table 10: Matlab© function conde.m (uniform k)

```
pp_conduction

conde.m uniform k    : 1 W/(mK)
Fixed portion horiz  : 0.2
Base temperature     : 270 K
Top  temperature     : 320 K
Mesh size            : 50 x 100
                       (100 x 200)
Fix. nod. 2 hor. fa.: 20 (40)
Diss tcaT*(K*tca)/2 : 347 (352) WK
Degrees of freedom   : 5131 (20261)
Cpu                  : 1 (73.3) sec.
Bottom flow          : 13.9 (14.1) W
```

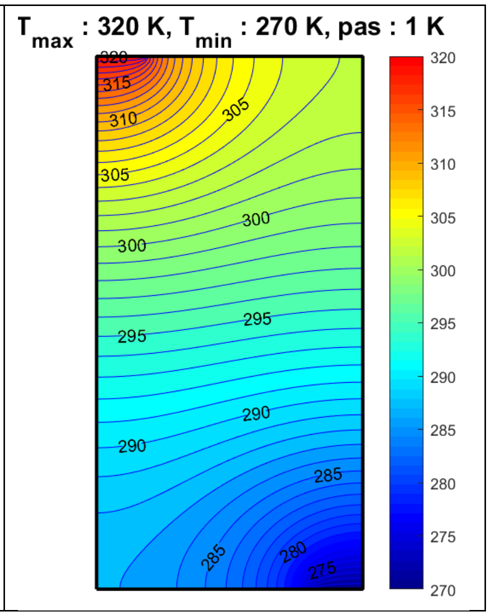


Figure 7: Isocurves for exercise 1 (uniform k, two meshes tested)

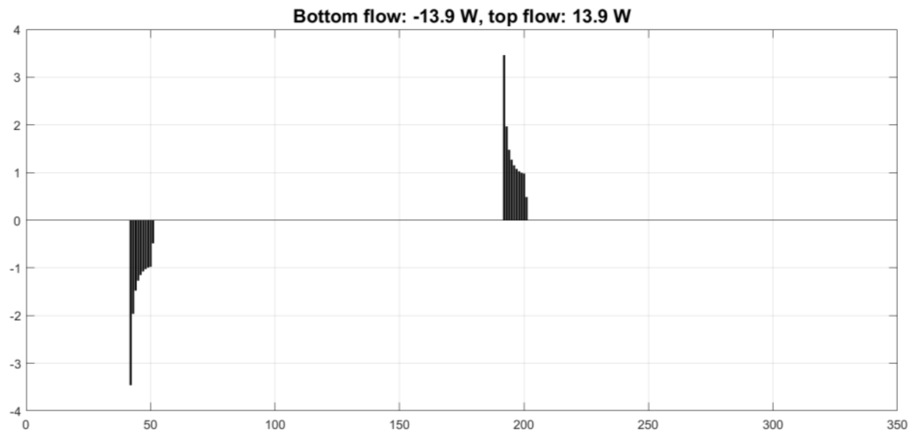


Figure 8: Heat input and output for exercise 1 (uniform k)

Function Matlab© *conde.m* for the definition of the conduction coefficients

```
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   = 10000;        % Ratio between the 2 conductivities, if 1, k is a cst
co   = ones(nel,1)*k;
         co(nx*nx+1:nx*nx+nx)    = k*fa;     % Second k on horizontal band 1
if nx>2;co(nx*(nx-1)+1:nx*nx) = k*fa;end % Second k on horizontal band 2
disp(['Conductivity coeff. : ',num2str(k,'%0.3g'),' W/(m K)'])
disp(['Main & bridge cond. : ',num2str([co(1) co(nx*nx+1)]),' W/(m K)'])
end
```

Table 11: Matlab© function conde.m (horizontal strip)

```
pp_conduction

Conductivity coeff.  : 1 W/(mK)
Main & bridge cond.  : 1  1000 W/(m K)
Rel. strip thickness: 0.02
Fixed portion horiz  : 0.2
grisb.m, mesh dim.   : 50 x 100 m
Heat flows           : 149, mean: 16 W/m2
Temperature gradient: 149, mean: 16 K/m
Base temperature     : 270 K
Top  temperature     : 320 K
Mesh size            : 50 x 100
Fix. nod. 2 hor. fa.: 20
Diss tcaT*(K*tca)/2 : 352 WK
Cpu                  : 1.22 sec.
```

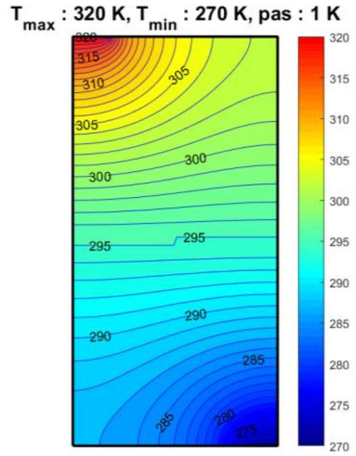


Figure 9: Isocurves for exercise 1 (horizontal strip)

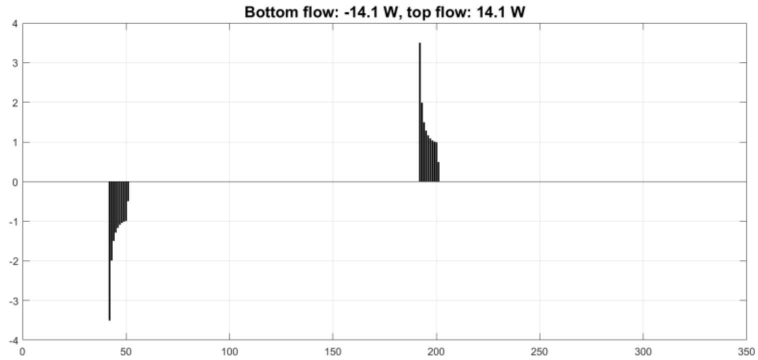


Figure 10: Heat input and output for exercise 1 (horizontal strip)

Function Matlab© *conde.m* for the definition of the conduction coefficients

```
function [co] = conde(nx,ny) % Treatment of non uniform conductivity Vert
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;                                  % 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)'])
end
```

Table 12: Matlab© function conde.m (vertical strip)

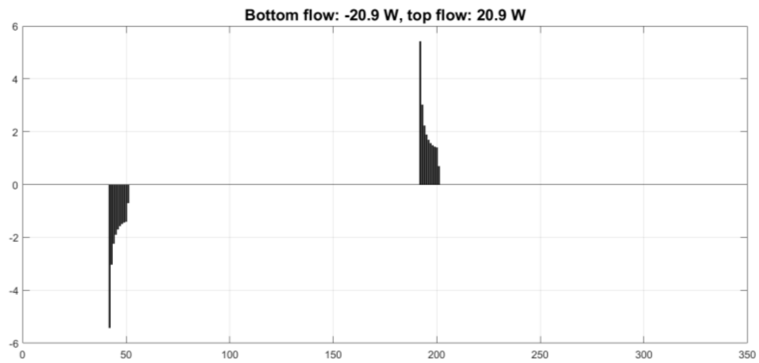


Figure 11: Heat input and output for exercise 1 (vertical strip)

```
pp_conduction

Conductivity coeff. : 1 W/(mK)
Cond. coeff. x fact.: 1000 W/(mK)
Rel. vert. strip th.: 0.04
Fixed portion horiz : 0.2
grisb.m, mesh dim.  : 50 x 100 m
Temperature gradient: 233, mean: 11.9 K/m
Heat flows          : 466, mean: 25.5 W/m2
Base temperature    : 270 K
Top  temperature    : 320 K
Mesh size           : 50 x 100
Fix. nod. 2 hor. fa.: 20
Diss tcaT*(K*tca)/2 : 522 WK
Fixed nodes hor. f. : 1.13 sec.
```

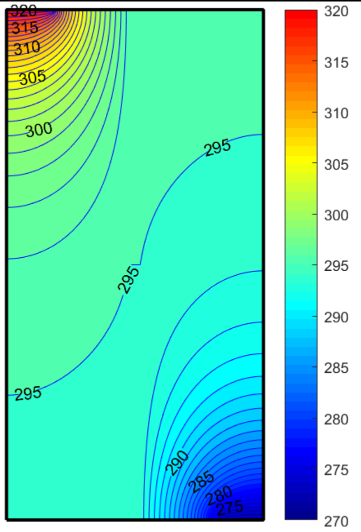


Figure 12: Isocurves for exercise 1 (vertical strip, mesh 50 x 100)

The heat flows on top and bottom horizontal sides are progressing from 13.9 W in the homogeneous case to 14.1 W in the case of horizontal strip and finally to 20.9 W in the case of vertical one.

```
pp_conduction

Conductivity coeff. : 1 W/(mK)
Cond. coeff. x fact.: 1000 W/(mK)
Rel. vert. strip th.: 0.125
Fixed portion horiz : 0.188
grisb.m, mesh dim.  : 16 x 32 m
Numb. fixed nodes   : 3
Heat flow, max      : 141, mean: 23 W/m2
- gradT, max        : 134, mean: 10 K/m
Base temperature    : 270 K
Top  temperature    : 320 K
Mesh size           : 16 x 32
Fix. nod. 2 hor. fa.: 6
Diss tcaT*(K*tca)/2 : 473 WK
Cpu                 : 0.397 sec.
```

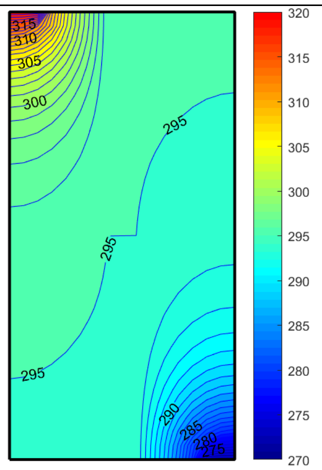
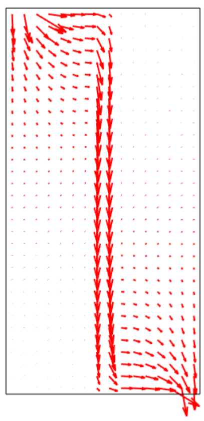
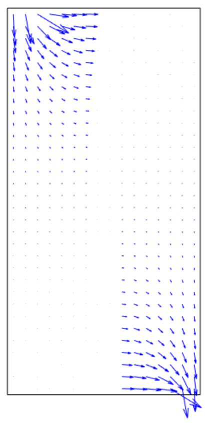


Figure 13: Isocurves for exercise 1 (vertical strip, mesh 16 x 32)



Heat flow: max.: 141 Wm^{-2}, mean: 23 Wm^{-2}



-grad τ: 134 Km^{-1}, average 10 Km^{-1}

Figure 14: Heat flows and temperature gradients for exercise 1 (vertical thermal bridge)

We have tested the function on a domain involving a vertical strip in which the ratio of conductivities is equal to 1000 in the example shown in *Figure 13*. In the *Figure 15*, we test the same mesh with a vertical strip of insulating material for which the conductivity is ten times smaller.

```
pp_conduction

Conductivity coeff. : 1 W/(m K)
Cond. coeff. x fact.: 0.1 W/(m K)
Rel. vert. strip th.: 0.125
Fixed portion horiz : 0.188
grisb.m, mesh dim.  : 16 x 32 m
Elem. heat flow max : 70, mean: 12 W/m2
- gradT, max        : 76., mean: 18 K/m
Base temperature    : 270 K
Top  temperature    : 320 K
Mesh size           : 16 x 32
Fix. nod. 2 hor. fa.: 6
Diss tcaT*(K*tca)/2 : 261 WK
Cpu                 : 0.326 sec.
```

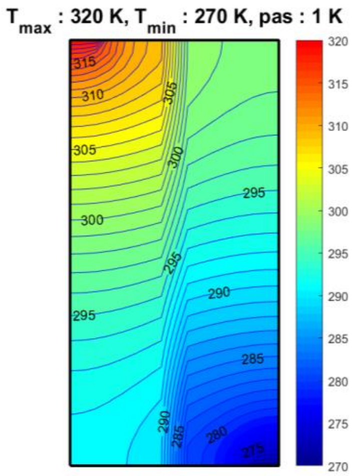
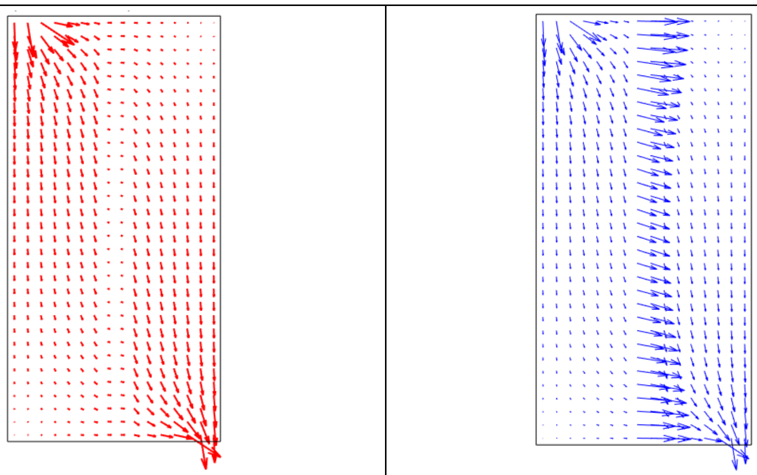


Figure 15: Isocurves for exercise 1 (vertical small conductivity strip)



Heat flow: max.: 70 Wm^{-2}, mean: 12 Wm^{-2}

-grad τ: 76 Km^{-1}, average 18 Km^{-1}

Figure 16: Heat flows and temperature gradients (vertical strip with small conductivity)

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.

$$< I(\tau) = \int_{\Omega} \frac{1}{2} k \ (grad\,\tau)^T . \, grad\,\tau \ d\Omega + \int_{S_2} \bar{q}_n \ \tau dS \ > \quad minimum \tag{1.19}$$

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:

$$\int_{S_{2edge}} \bar{q}_n \tau dS_{el} = g_1 T_1 + g_2 T_2 \tag{1.20}$$

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

$$\tau_{edge} = T_1\left(1-\frac{x}{l}\right) + T_2\frac{x}{l} \tag{1.21}$$

We can write the discretized functional:

$$\int_{S_{2edge}} \bar{q}_n \tau dS_{el} = \int_{S_{2edge}} \bar{q}_n \left(T_1\left(1-\frac{x}{l}\right) + T_2\frac{x}{l}\right) dS_{el} \tag{1.22}$$

In matrix form, we have:

$$\text{With : } [T] = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \text{ we have : } \int_{S_{2edge}} \bar{q}_n \tau dS_{el} = \int_{S_{2edge}} \bar{q}_n \begin{bmatrix} \left(1-\frac{x}{l}\right) & \frac{x}{l} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} dS_{el} \tag{1.23}$$

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

$$\int_{S_{2edge}} \bar{q}_n \tau dS_{el} = \int_{S_{2edge}} \bar{q}_n \begin{bmatrix} \left(1-\frac{x}{l}\right) & \frac{x}{l} \end{bmatrix} dS_{el} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \tag{1.24}$$

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

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix}^T = \int_{S_{2edge}} \bar{q}_n \begin{bmatrix} \left(1-\frac{x}{l}\right) & \frac{x}{l} \end{bmatrix} dS_{el} \tag{1.25}$$

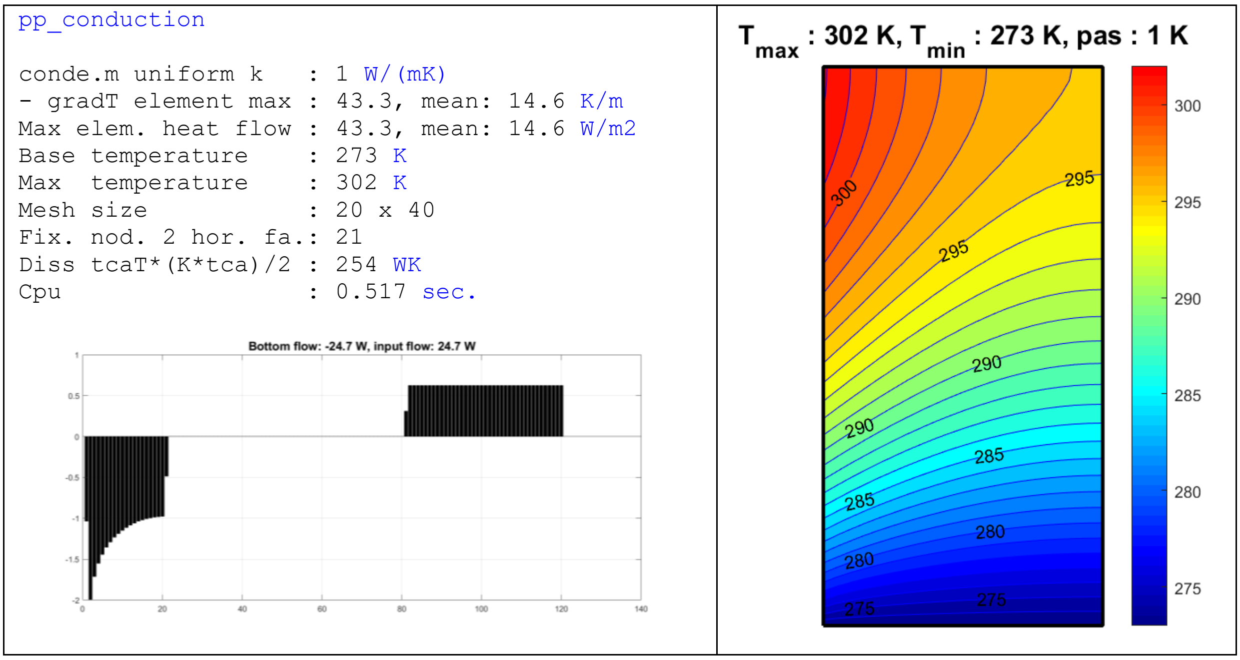


Figure 17: Isocurves for the problem of prescribed heat flows

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:

$$F_1 = \bar{q}\,\frac{el}{2} \quad ; \quad F_2 = \bar{q}\,\frac{el}{2} \tag{1.26}$$

With both imposed temperatures (T_3, in red) and prescribed heat fluxes (F_1, in red), the system that must be solved is characterizd by 9 submatrices. The unknown variables are the vectors [T_1], [T_2] and [F_3] (in blue)

$$[K] = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \quad ; \quad [K]\begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} F_1 \\ 0 \\ F_3 \end{bmatrix} \tag{1.27}$$

The system to be solved is:

$$\begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}^{-1} \left(\begin{bmatrix} F_1 \\ 0 \end{bmatrix} - \begin{bmatrix} K_{13} \\ K_{23} \end{bmatrix} [T_3] \right) \tag{1.28}$$

Finally the fluxes are calculated in the following operation:

$$[F_3] = [K_{31} \quad K_{32} \quad K_{33}] \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \tag{1.29}$$

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 17*). 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 *15* are disabled and the *lines 16* to 21 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.

The dissipation energy displayed in the last line of the output of *Figure 17* is obtained by pre- and post- multiplying the full conduction matrix by the temperature vector:

$$\frac{1}{2}[T_1 \quad T_2 \quad T_3] \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} \quad \text{or} \quad \frac{1}{2}[T]^T [K][T] \tag{1.30}$$

When the mesh is refined this energy is converging to its exact value (in this example the convergence is yet achieved with the 20 x 40 mesh.

Convection

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

$$< I(\tau) = \int_{\Omega} \frac{1}{2} k \ (grad\,\tau)^T . \ grad\,\tau \ d\Omega + \frac{1}{2} \int_{S_3} h\left(\tau - \tau_f\right)^2 dS + \int_{S_2} \bar{q}_n \ \tau dS \ > \quad minimum \tag{1.31}$$

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.

$$\text{Functional at element level: } I_{el} = \frac{1}{2} \int_{S_3} h\left(\tau - \tau_f\right)^2 dS \tag{1.32}$$

In (1.31) and (1.32), h represents the convection coefficient. The fluid temperature τ_f is assumed uniform. We study an element side which is a line segment of length L. We discretize the edge temperature as follows:

$$\tau = T_0 (1 - \frac{x}{L}) + T_1 \frac{x}{L} \tag{1.33}$$

In the previous expression the variable x varies between 0 and L. Replacing in the functional (1.28), we obtain:

$$\begin{aligned} I_{el} &= \frac{1}{2} \int_0^L h \left(\begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \end{bmatrix} - t_f \right)^2 dx \\ &= \frac{1}{2} h \int_0^L \left\{ [T]^T \begin{bmatrix} 1 - \frac{x}{L} \\ \frac{x}{L} \end{bmatrix} \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} [T] - 2 \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} \end{bmatrix} [T] t_f + t_f^2 \right\} dx \end{aligned} \tag{1.34}$$

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[2]:

$$T_{el} = \begin{bmatrix} T_0 & T_1 & T_f \end{bmatrix}^T \tag{1.35}$$

Replacing in the first term of (1.34), we obtain:

$$I_{el} = \frac{1}{2} h T_f^T (\int_0^L \begin{bmatrix} 1 - \frac{x}{L} \\ \frac{x}{L} \\ -1 \end{bmatrix} \begin{bmatrix} 1 - \frac{x}{L} & \frac{x}{L} & -1 \end{bmatrix} dx) \ T_f \tag{1.36}$$

[2] It is important to note that the virtual nodes are not related to a position, they do not have any coordinate. However to represent them as in *Figure 20*, we can give them an arbitrary position, only for the drawing.

Developping the expression:

$$I_{el} = \frac{1}{2} h T_f^T \int_0^L \begin{bmatrix} \left(1-\frac{x}{L}\right)^2 & \left(1-\frac{x}{L}\right)\frac{x}{L} & -\left(1-\frac{x}{L}\right) \\ \left(1-\frac{x}{L}\right)\frac{x}{L} & \left(\frac{x}{L}\right)^2 & -\frac{x}{L} \\ -\left(1-\frac{x}{L}\right) & -\frac{x}{L} & 1 \end{bmatrix} dx\ T_f \qquad (1.37)$$

After integrating and including the thickness *e* to ensure the coherence of units, we transform the functional (1.34) into an algebric function of the nodal temperatures:

$$I_{el}^{al} = \frac{1}{2} h\ e\ T_f^T \begin{bmatrix} \int_0^L \left(1-\frac{x}{L}\right)^2 dx & \int_0^L \left(1-\frac{x}{L}\right)\frac{x}{L} dx & -\int_0^L \left(1-\frac{x}{L}\right) dx \\ \int_0^L \left(1-\frac{x}{L}\right)\frac{x}{L} dx & \int_0^L \left(\frac{x}{L}\right)^2 dx & -\int_0^L \frac{x}{L} dx \\ -\int_0^L \left(1-\frac{x}{L}\right) dx & -\int_0^L \frac{x}{L} dx & \int_0^L dx \end{bmatrix} T_f \qquad (1.38)$$

$$\int_0^L \left(1-\frac{x}{L}\right)^2 dx = \int_0^L \left(1-2\frac{x}{L}+\frac{x^2}{L^2}\right) dx = L - L + \frac{L}{3} = \frac{L}{3} \qquad (1.39)$$

$$\int_0^L \left(1-\frac{x}{L}\right)\frac{x}{L} dx = \int_0^L \left(\frac{x}{L}-\frac{x^2}{L^2}\right) dx = \frac{L}{2} - \frac{L}{3} = \frac{L}{6} \qquad (1.40)$$

$$-\int_0^L \left(1-\frac{x}{L}\right) dx = -L + \frac{L}{2} = -\frac{L}{2} \qquad (1.41)$$

$$I_{el}^{al} = \frac{1}{2} h\ eL\ T_f^T \begin{bmatrix} \frac{1}{3} & \frac{1}{6} & -\frac{1}{2} \\ \frac{1}{6} & \frac{1}{3} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix} T_f = \frac{1}{2} T_f^T\ K_h\ T_f \qquad (1.42)$$

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

$$K_h = h\ \frac{eL}{6} \begin{bmatrix} 2 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & 6 \end{bmatrix} \qquad (1.43)$$

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, both matrices have to be added. The second one carries additional degrees of freedom corresponding to the virtual convective nodes.

$$K = K_k + K_h \tag{1.44}$$

The convection virtual nodes may be free or fixed.

Procedures used for the solution of convective heat transfers

Procedure Matlab© *pp_convection.m* for convection

```
hh   = 25;tb = 273;     nfc = 3;ta = [0;0; 290; 273];pge = [1;2;.1;10];
% hh   = 18;tb = 273;     nfc = 2;ta = [300;280]       ;pge = [1;2;.1;10 ];
th   = pge(3); gap=1;
nx   = pge(4);ny = nx*2;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(nx*2),' m'])
       disp(['Impos. virt. nod T. : ',num2str(ta(1:size(ta,1))'),' K'])
       disp(['Impos. base Temp.   : ',num2str(tb),' K'])
       disp(['Numb. convect. faces: ',num2str(nfc)])
       disp(['Convection coeffic. : ',num2str(hh,2),' W/(m2K)'])
tst  = tic;         % Beginning the analysis, initialisation of the timer
co   = conde(nx,ny);                      % Element conductivity coefficients
[K ] = CoKr(nx,ny,hh*th,nfc);                    % Computing the convection K
Bi   = hh/co(1);
disp(['Biot number hL/k     : ',num2str(Bi,2)])
% Biot hh*ar/(k*per)   :
lK   = loca(nx,ny);                                       % Localization matrix
Kel  = th/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];% Elem. K matrix
for n=1:nel                                        % Loop on the nel elements
    for i=1:4;for j=1:4      % Assembling the nel conductivity matrices Kel
          K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+co(n)*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;
    grisb(nx,ny,tca(1:no),gap);axis off  % Drawing isotherms on the domain
    if nx <51;figure;Tg(nx,ny,lK,tca);hold on;end  % Drawing temper. grad.
    if nx <51;figure;Hf(nx,ny,lK,tca,co);hold on;end  % Drawing heat flows
    sm    =  (K*[tca;ta])';                             % 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];
    disp(['Heat on virt. nodes : ',num2str(hvn'),' W'])
    disp(['H convl cond. convr : ',num2str([qxg qxc qxd]),' Wm-2'])
    disp(['Surf. T. right left : ',num2str([max(tca) min(tca)]),' K'])
end
if nfc == 3     % 2. One free and two imposed temperatures of virtual nodes
    if ta(2)==0.
        K11 = K(1:no-nx-1,1:no-nx-1);
        K12 = K(1:no-nx-1,no-nx:no);
```

```
46          K13 = K(1:no-nx-1,no+1);
47          K14 = K(1:no-nx-1,no+2:no+3);
48          K31 = K(no+1,1:no-nx-1);
49          K32 = K(no+1,no-nx:no);
50          K33 = K(no+1,no+1);
51          K34 = K(no+1,no+2:no+3);
52          T2  = ones(nx+1,1)*tb;
53          T4  = [ta(3); ta(4)];
54          tcb = [K11 K13;K31 K33]\(-[K12 K14;K32 K34]*[T2;T4]);
55          tca = [tcb(1:no-nx-1); ones(nx+1,1)*tb; tcb(size(tcb,1)); T4];
56          grisb(nx,ny,tca(1:no),gap);axis off% Drawing isoth. on the domain
57          if nx<50
58          figure;Tg(nx,ny,lK,[tca;ones(nx+1,1)*tb]);hold on
59          figure;Hf(nx,ny,lK,[tca;ones(nx+1,1)*tb],co);hold on
60          end
61       disp(['Virtual nodes temp. : ',num2str(tca(no+1:size(K,1))',3),' K'])
62       disp(['Mesh min max Temp.  : ',num2str([min(tca(1:no)) ...
63              max(tca(1:no))],3),' K'])
64      else
65          gco  = K(1:no-nx-1,no-nx:size(K,1))*[ones(nx+1,1)*tb;ta(2:4)];
66          disp(['Imposed temperatures: ',num2str([tb;ta(2:4)]',3),' K'])
67          tca  = -K(1:no-nx-1,1:no-nx-1)\gco;
68          grisb(nx,ny,[tca;ones(nx+1,1)*tb]);axis off% Drawing the isotherms
69          figure
70          Tg(nx,ny,xyz,lK,[tca;ones(nx+1,1)*tb]);
71          sm   =(K*[tca;ones(nx+1,1)*tb; ta(2:4)])';         % second member
72          rea  = sum(sm(size(sm,2)-2-nx-1:size(sm,2)-3)); % Heat flow bottom
73          disp(['Global heat balance : ',num2str([rea sm(size(sm,2)-2:...
74              size(sm,2))]),' W'])
75        disp(['Mesh min max Temp.  : ',num2str([min(tca) max(tca)],4),' K'])
76      end
77  end
78  disp(['Size of full K      : ',num2str(size(K))])
79  disp(['Cpu                 : ',num2str(toc(tst),'%0.3g'),' sec.'])
```

Table 13: Matlab© procedure pp _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 – 12 : Display some data and results
Line 14 : Conductivity coefficients in the elements (function *conde.m*)
Line 15 : Compute the conduction-convection matrix in the function *CoKr.m*. (1.39)

The convection 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. The function *CoKr.m* allows computing the convection matrices of a *nx* x *ny* mesh (arguments 1 and 2 of the function). The convection coefficient is given by argument 4. Argument 5 is giving the number of faces of the domain where convection elements are present (*Table 14*).

Line 16 : Compute the Biot number
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 – 39: Solution of the system in the case of two convective end two 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 19*), heat flows and temperature gradients. *Lines 29* to *37* are concerned with some statistics and variables showing the agreement of the solution with the analytical results.

Lines 40 - 77: Solution of other problems.

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

On the drawing, the nodes numbers are in blue, the conductive elements numbers in red and the convective elements numbers in magenta. The positions of the three virtual nodes 16, 17, 18, related to the convection are arbitrary.

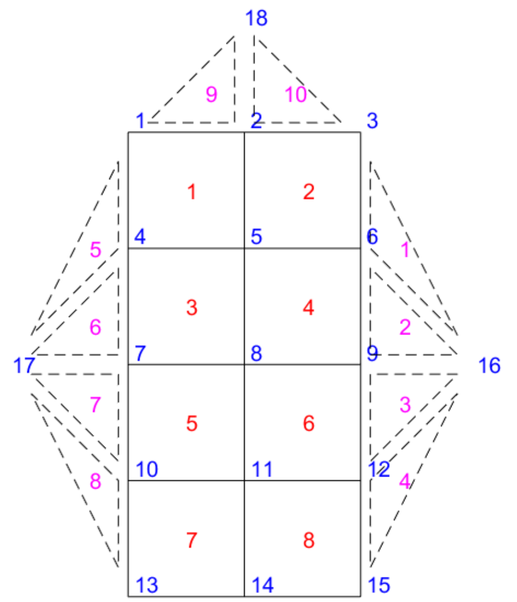


Figure 18: Localization matrix of the convective elements on 3 sides of a 2 x 4 mesh

Function Matlab© *CoKr.m* conductive convective matrix

```
function[K] = CoKr(nx,ny,le,nfc)
co     = le./[nx ny nx ny]';          % Convection coefficients on the 4 sides
% disp(['Co:le./[nx ny nx ny]: ',num2str(co'),' W/K'])
Kelc   = [2 1 -3;1 2 -3;-3 -3 6]*co(2)/3;        % Element convection matrix
if nfc ==1
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 ==2
Ntca   = (nx+1)*(ny+1)+nfc;                             % Mesh nx x ny elements
ntv2   = [ Ntca-1 Ntca ];       % Numbering of the convective virtual nodes
lc     = locc2(nx,ny,ntv2);     % Local. of the convection matrices 2 sides
disp(['Virtual conv. nodes : ',num2str(ntv2)])
end
if nfc ==3
Ntca   = (nx+1)*(ny+1)+nfc;                             % Mesh nx x ny elements
ntv3   = [Ntca-2 Ntca-1 Ntca 0];% 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
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
ii     = 0;for i=1:size(lc,1);ii=ii+1;if lc(ii,1)>0;dim=ii;end;end
K      = zeros(Ntca,Ntca);% Dimension of K including fixed DOF & add. nodes
for n = 1:dim;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);end;end;end
end
```

Table 14: Matlab© function CoKr.m for the conductivity matrix used for convection

Exercise n°2: One free convective virtual node

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 adimensional variable $\beta = w\ h\ /\ k$ (w is the width of the domain, h and k respectively the convection and conduction coefficient) and to compare with the finite element model result. Convection is present on 3 faces, the fourth face is fixed, and one of the three virtual nodes is free.

In this kind of application, it is convenient to introduce the adimensional Biot number β, which describes the relation between convection and conduction. It depends on the material properties h and k and on a characteristic length L.

$$\beta = \frac{hL}{k} \tag{1.45}$$

In this application, there is only one available characteristic length L which corresponds to the width (horizontal dimension) of the mesh .

Comparison in the case of 2 convective and 2 adiabatic faces

```
pp_ convection
Boundary cond. type : 2
Domain dim. w, h, t : 1 2 0.1 m
Mesh dimension      : 1 x 2 m
Impos. virt. nod T. : 300  270 K
Impos. base Temp.   : 273 K
Numb. convect. faces: 2
Convection coeffic. : 1 W/(m2K)
conde.m uniform k   : 1 W/(mK)
Number of conv. el. : 4
Virtual conv. nodes : 7  8
Biot number hL/k    : 1
- gradT element max : 10, mean: 10 K/m
Elem. heat flow max : 10, mean: 10 W/m2
Heat on virt. nodes : 2 -2 W
H convl cond. convr : -10 -10 -10 Wm-2
Surf. T. right left : 290 280 K
Size of full K      : 8  8
Cpu                 : 0.256 sec.
```

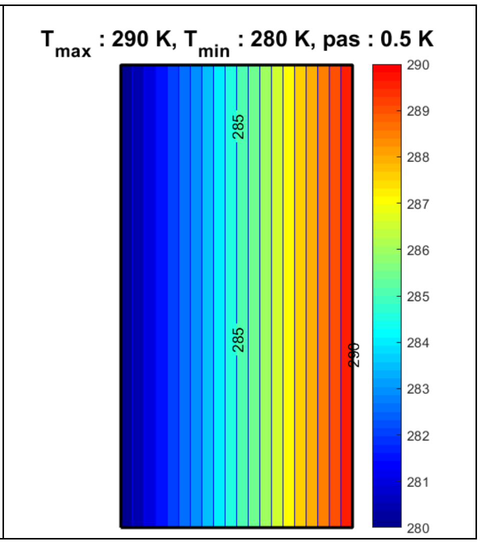


Figure 19: The isocurves are orthogonal to both adiabatic boundaries. β = 1

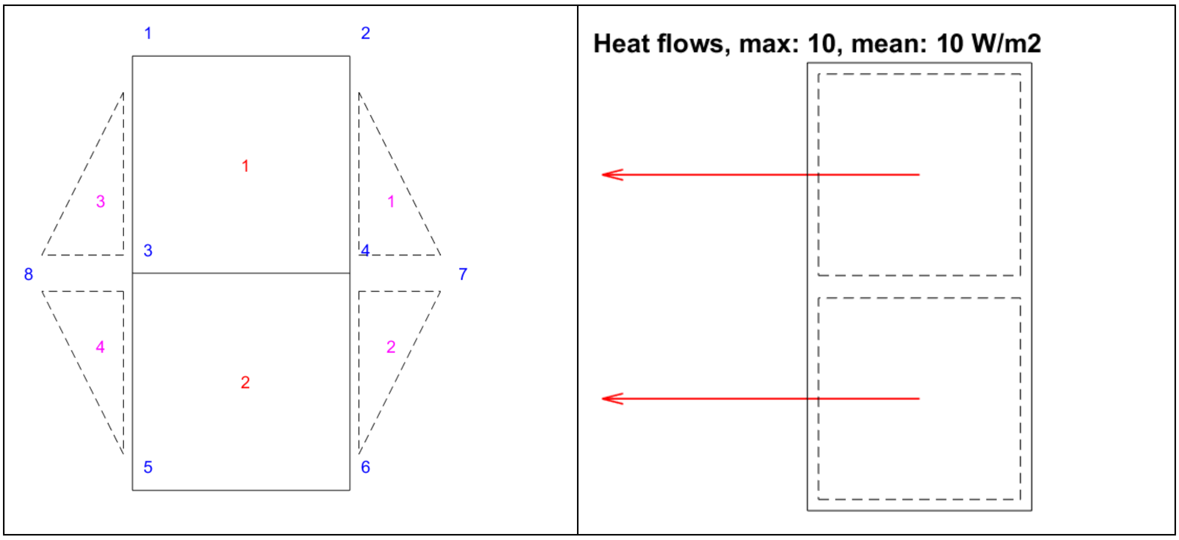


Figure 20: Nodes and elements numbering: heat flows with convective boundary conditions

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	$t_2 < t_3$

$$q_x = eh(t_0 - t_1) = ek\frac{(t_1 - t_2)}{w} = eh(t_2 - t_3) \tag{1.46}$$

The parameter w can be used as the length L in the adimensional Biot number definition

$$\beta = \frac{hw}{k}, \qquad \chi\left(t_0 - t_1\right) = t_1 - t_2 = \beta\left(t_2 - t_3\right) \tag{1.47}$$

From the first relation, we deduce:

$$t_1 = \frac{\beta\, t_0 + t_2}{1+\beta} \tag{1.48}$$

Now, we develop the second one:

$$\frac{\beta\, t_0 + t_2}{1+\beta} - t_2 = \beta\, t_2 - \beta\, t_3 \tag{1.49}$$

We obtain:

$$t_2 = \frac{t_0 + \left(1+\beta\right)t_3}{2+\beta} \tag{1.50}$$

Replacing (1.46) in (1.44), we have:

$$t_1 = \frac{\beta\, t_0}{1+\beta} + \frac{t_0 + (1+\beta)\, t_3}{(2+\beta)\,(1+\beta)} = \frac{t_0}{1+\beta}\left(\beta + \frac{1}{(2+\beta)}\right) + \frac{t_3}{(2+\beta)} = \frac{(1+\beta)\, t_0 + t_3}{(2+\beta)} \qquad (1.51)$$

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

$$\left(t_2 - t_1\right) = \left(t_3 - t_0\right)\frac{\beta}{2+\beta} \qquad (1.52)$$

With t_0 = 270 and t_3 = 300, we obtain both with formula (1.47) to (1.49) and with the *FEM* simulation the results of *Table 15*.

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)
.5	6	282	288	12	6	12
1	10	280	290	10	10	10
2	15	277.5	292.5	7.5	15	7.5
18	27	271.5	298.5	1.5	27	1.5

Table 15: Temperatures and heat flows as functions of Biot number

Basically, we are working with four nodes: two virtual ones numbered 0 (left side) and 3 (right side) and two nodes situated on the surface of the conductive zone: 1 on the left side and 2 on the right one. Their corresponding temperature are: t_0, t_1, t_2, t_3. For *Bi* = 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 finite element model is able to represent the exact solution, this analytical solution is obtained for any mesh.

```
pp_ convection
Boundary cond. type : 2
Domain dim. w, h, t : 1 2 0.1 m
Mesh dimension      : 1 x 2 m
Impos. virt. nod T. : 300  270 K
Impos. base Temp.   : 273 K
Convection coeffic. : 18 W/(m2K)
Numb. convect. faces: 2
conde.m uniform k   : 1 W/(mK)
Co:le./[nx ny nx ny]: 1.8 .9 1.8 .9 W/K
Number of conv. el. : 4
Virtual conv. nodes : 7  8
Biot number hL/k    : 18
Heat flow, max      : 27, mean: 27 W/m2
- gradT, max        : 27, mean: 27 K/m
Heat on virt. nodes : 5.4 -5.4 W
H convl cond. convr : -27 -27 -27 Wm-2
Surf. T. right left : 298.5 271.5 K
Size of full K      : 8  8
Cpu                 : 0.263 sec.
```

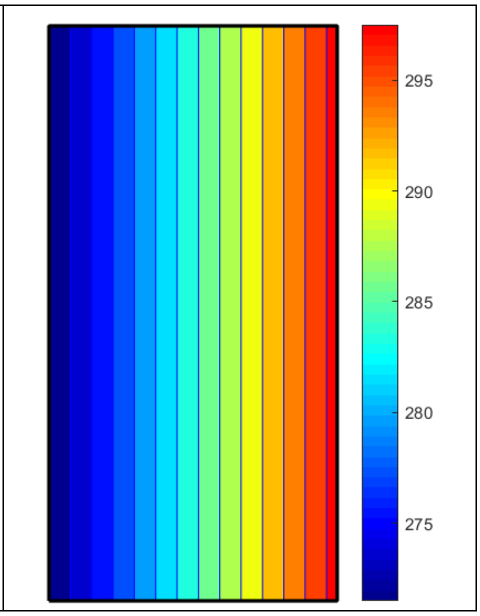


Figure 21: The isocurves are orthogonal to the adiabatic boundaries. β = 18

In the test of *Figure 21*, 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 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 %.

Exercise n°2b: Convection with thermal bridge

In the previous exercise, check the consequence of introducing an horizontal thermal bridge.

With respect to the example of *Figure 19*, we simply modify the function *conde.m* as in *Table 11*. The effect of the thermal bridge is dramatic but, here, the conductivity ratio is 10000. If this ratio falls to 10, the effect is yet visible. This test shows the importance of the thermal bridges in the building design (*Figure 22* & *Figure 23*).
Let observe that the heat rate crossing the domain is equal to 3.6 W when the conductivity is uniform while it reaches the value of 9.4 W if the thermal bridge thickness is 0.1 x the height of the domain and the conductivity in the bridge equal to 10000 times the conductivity in the other part of the domain.

```
pp_convection
Boundary cond. type : 2
Domain dim. w, h, t : 1 2 0.1 m
Mesh dimension      : 10 x 20
Impos. virt. nod T. : 300  280 K
Impos. base Temp.   : 273 K
Numb. convect. faces: 2
G. convection coeff.: 18 W/(m2K)
Conductivity coeff. : 1 W/(m K)
Main & bridge cond. : 1  10000 W/(m K)
Rel. strip thickness: 0.1
Number of conv. el. : 40
Virtual conv. nodes : 232  233
Biot number hL/k    : 18
Elem. heat flow max : 352.2, mean: 49.3 W/m2
- gradT element max : 60.2, mean: 16.8 K/m
Heat on virt. nodes : 9.4258     -9.4258 W
H convl cond. convr : -18.4 -18 -18.4 Wm-2
Surf. T. right left : 299 281 K
Size of full K      : 233  233
Cpu                 : 0.33 sec.
```

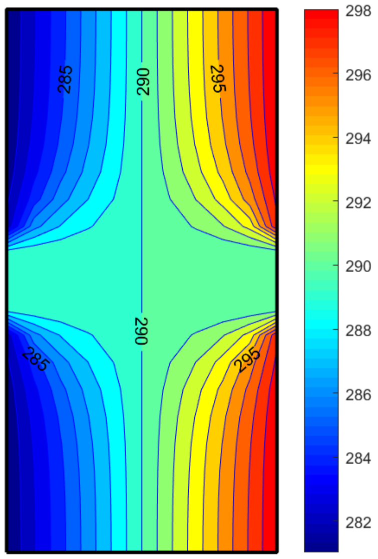
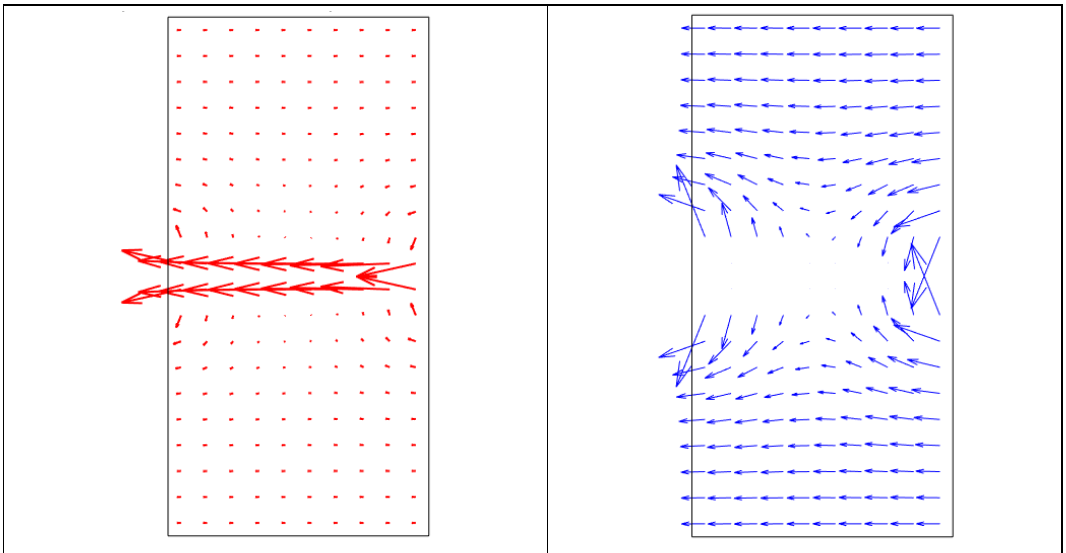


Figure 22: Isocurves with the presence of horizontal thermal bridge



Heat flow, max: 352.2 Wm^{-2}, mean: 49.3 Wm^{-2} -grad τ: 60.2 Km^{-1}, mean: 16.8 Km^{-1}

Figure 23: Heat flows and temperature gradients (horizontal strip with high conductivity)

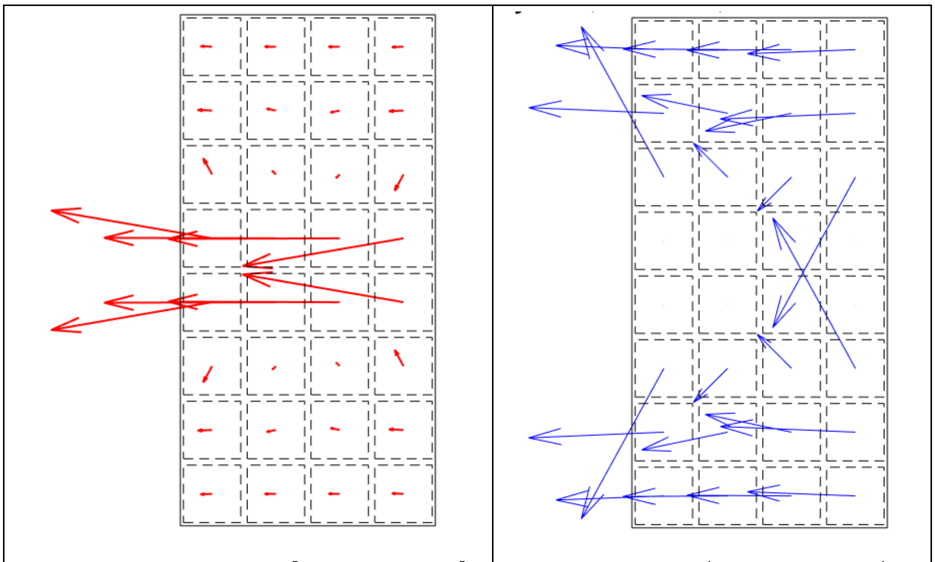


Figure 24: Heat flows and gradients with a coarse mesh (strip / thickness = .25)

Exercise n°3: Modifying the boundary conditions of a presented example

It is proposed to modify the boundary conditions of the application presented in *Figure 25* in order to obtain more or less the same temperatures on both horizontal sides and obtain a temperature gradient mainly oriented from left to right. 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).

We start splitting the system to be solved into four categories. The second members are denoted S_i.

$$\begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} \\ K_{21} & K_{22} & K_{23} & K_{24} \\ K_{31} & K_{32} & K_{33} & K_{34} \\ K_{41} & K_{42} & K_{43} & K_{44} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ S_4 \end{bmatrix} \tag{1.53}$$

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.

$$\begin{aligned} \begin{bmatrix} K_{11} & K_{13} \\ K_{31} & K_{33} \end{bmatrix} \begin{bmatrix} T_1 \\ T_3 \end{bmatrix} &= -\begin{bmatrix} K_{12} & K_{14} \\ K_{32} & K_{34} \end{bmatrix} \begin{bmatrix} T_2 \\ T_4 \end{bmatrix} + \begin{bmatrix} S_1 \\ S_3 \end{bmatrix} \\ \begin{bmatrix} T_2 \\ T_4 \end{bmatrix} &= \begin{bmatrix} K_{11} & K_{13} \\ K_{31} & K_{33} \end{bmatrix}^{-1} \left(\begin{bmatrix} S_1 \\ S_3 \end{bmatrix} - \begin{bmatrix} K_{11} & K_{13} \\ K_{31} & K_{33} \end{bmatrix} \begin{bmatrix} T_2 \\ T_4 \end{bmatrix} \right) \end{aligned} \tag{1.54}$$

```
pp_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
Numb. convect. faces: 3
Convection coeff.   : 25 W/(m2K)
conde.m uniform k   : 2 W/(mK)
Virtual conv. nodes : 5152  5153  5154
Biot number hL/k    : 13
Virtual nodes temp. : 287 290 303 K
Heat outcome        : -10.97 W
Heat virtual nodes  : 0. 3.36 7.61 W
Mesh min max Temp.  : 273 301 K
Cpu                 : 1.1 sec.
```

Left virtual node temp. (tca(no+1)): 286.6 *K*

The sentence:

```
figure;Hf(nx,ny,lK,[tca;ones(nx+1,1)*tb],co);
```

is giving the result:

```
Elem. heat flow max : 451, mean: 32.8 W/m2
```

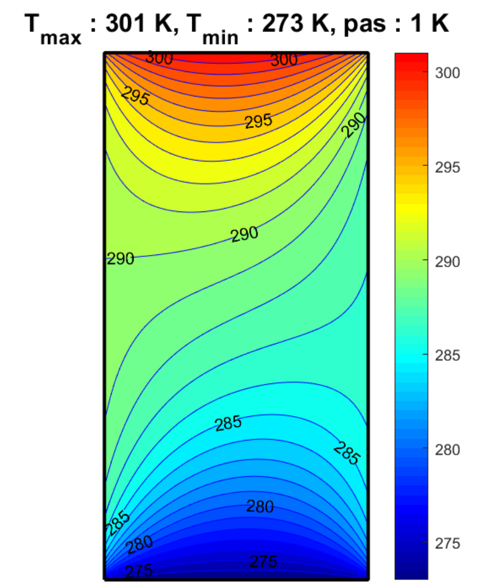


Figure 25: Isocurves for 1 free and 2 imposed temperatures of virtual nodes

```
pp_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  273 K
Impos. base Temp.   : 273 K
Numb. convect. faces: 3
Convection coeff.   : 25 W/(m2K)
conde.m uniform k   : 2 W/(mK)
Virtual conv. nodes : 5152  5153  5154
Biot number hL/k    : 13
Virtual nodes temp. : 277 290 273 K
Heat outcome        : -7.45 W
Heat virtual nodes  : 0.  12.79 -5.34  W
Mesh min max Temp.  : 273 289 K
Mesh min max Temp.  : 273 289 K
Cpu                 : 1.12 sec.

Max elem. heat flow : 449, mean: 31 W/m2
```

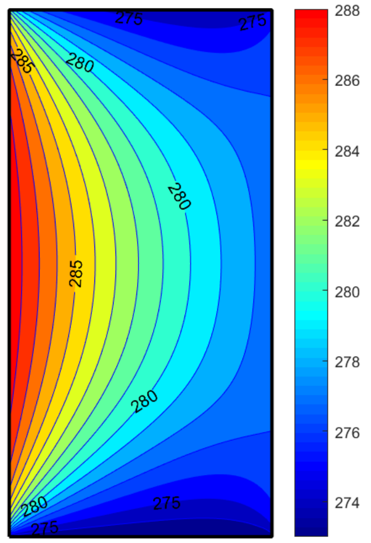
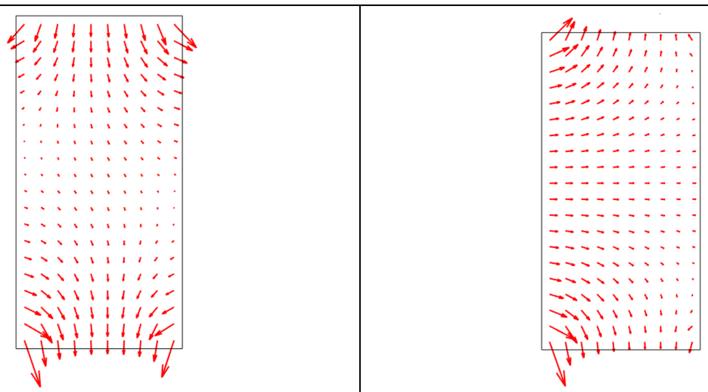


Figure 26: Isocurves for proposed exercise 2



Heat flow, max: 189 Wm^{-2}, mean: 33 Wm^{-2} | Heat flow, max: 186 Wm^{-2}, mean: 31 Wm^{-2}

Figure 27: Heat flows for Figure 25 (left) and Figure 26 (right)

If we compute the second member of the system: g = K*tca (Matlab© notation), we can evaluate the total heat going out through the base: sum (g (5100:5151)) = -7.4503 W, the total heat going in through the two virtual convective nodes (left and top of the domain): sum (g (5153:5154)) = 7.4503 W and the temperature of the free virtual node (right side: tca (5152) = 276.9555 K.

To let free the left virtual node of the problem of *Figure 25*, we use the Matlab sentences of line 1 of *Table 13* 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). With this sequence, the F.E.M. computation is solving the problem with the second line of the development (1.54).

Conclusion: to solve a 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. **All the nodes may be concerned with this rule: mesh or virtual nodes.**

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 black body and per unit of time is proportional to the fourth power of the body's thermodynamic temperature, with the Stefan-Boltzmann coefficient $\sigma = 5.6704\ 10^{-8}$ $Wm^{-2}K^{-4}$

$$Q = \sigma\left(T_k^4 - T_r^4\right) \tag{1.55}$$

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:

$$Q = \sigma\left(T_k^2 + T_r^2\right)\left(T_k + T_r\right)\left(T_k - T_r\right) \tag{1.56}$$

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

$$\sigma\left(T_k^2 + T_r^2\right)\left(T_k + T_r\right) \tag{1.57}$$

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.

$$K_r = \frac{eL\ \sigma\left(T_k^2 + T_r^2\right)\left(T_k + T_r\right)}{6}\begin{bmatrix} 2 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & 6 \end{bmatrix} \tag{1.58}$$

Comparing the definitions of the conductive matrices in convection (1.43) and in radiation (1.58), we deduce, in the radiative exchange, an equivalent convection coefficient measured in $Wm^{-2}K^{-1}$.

$$h_r = \sigma\left(T_k^2 + T_r^2\right)\left(T_k + T_r\right) \tag{1.59}$$

In this expression, the temperatures T_k should be evaluated in the same loop as when it is computed. It means that the radiative problem is no linear and that the above coefficient has to be obtained by successive approximations during the iterations.

Procedures used for radiative heat transfer

Procedure Matlab© *pp_ radiation.m* for radiation

```
1  tb   = 273;pge=[20;40;1;20];pf=[0;12.5;0];ts=[290;290;303];SB = 5.6704e-8;
2  qs   = pf(1)*pge(1)*pge(2);qw=pf(2)*pge(2)*pge(3);qe=pf(3)*pge(2)*pge(3);
3  th   = pge(3);he=pge(2);tst  = tic;% Beginning analysis, timer init.
4  nx   = pge(4);ny = nx*2;nel = nx*ny;no = (nx+1)*(ny+1);             % Mesh
5  disp('********************')% First iteration =============================
6  co   = conde(nx,ny);
7  disp(['Stefan-Boltzmann    : ',num2str(SB,'%0.3g'),' Wm-2K-4'])
8  disp(['Mesh dimension      : ',num2str(nx),' x ',num2str(ny)])
9  disp(['Base temperature    : ',num2str(tb,3),' K'])
10 disp(['Virtual nodes temp. : ',num2str(ts'),' K'])
11 Kel  = th/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];    % elem. K
12 tca  = ones(no,1)*min(ts);
13 lK   = loca(nx,ny);            % Computing the localization matrix (nel x 4)
14 nit  = 2;
15 [K ] = ItKr (tca,ts,nx,ny,th,he,SB);% he is the height of the domain in m
16 for n=1:nel;for i=1:4;for j=1:4      % Assembling nel conduct. matrices Kel
17           K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+co(n)*Kel(i,j);end;end;end
18 gco  =  K(1:no-nx-1,no-nx:no+size(K,1)-no)*[ones(nx+1,1)*tb; ...
19     ts(1:size(K,1)-no,1)];
20 tca  = -K(1:no-nx-1,1:no-nx-1)\gco;gap=1;
21 grisb(nx,ny,[tca ;ones(nx+1,1)*tb],gap);axis off       % Drawing isotherms
22 tmi1    = min(tca(1:no-nx-1));
23 sm      = (K*[tca;ones(nx+1,1)*tb;ts(1:size(K,1)-no)])'; % syst. sec. memb.
24 hvn1    = sum(sm(size(sm,2)-2:size(sm,2)));
25 if nit > 1 % Additional iterations ========================================
26 [hvn,tmi]=radit(tca,nx,ny,tb,ts,th,he,lK,Kel,co,nit,tmi1,hvn1,SB);
27 end
28 disp(['Number of iterations: ',num2str(nit)]) % Numb. iterations
29 disp(['Min. mesh temper.   : ',num2str(tmi(1:min(5,nit))',5),' K'])
30 disp(['Total convect. flow : ',num2str(hvn(1:min(5,nit))',5),' W'])
31 disp(['Cpu                 : ',num2str(toc(tst),'%0.3g'),' sec.'])
```

Table 16: Matlab© procedure pp_radiation.m

The procedure for the solution of a thermal radiation problem with radiative boundary conditions given in *Table 16* has the following characteritics.

Lines 1 – 4 : Data input. In pure conduction problems, the solution was independent of the scale of the geometry. 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.

Line 6 : Definition of element conductivities (function *conde.m*)

Line 11 : Conductivity matrix of a square element
Line 12 : Initialization
Line 13 : Localization matrix (function *loca.m*)
Line 14 : Number of iterations required to test the convergence
Line 15 : Starting the first iteration
Line 15 : Conduction - radiation matrix, function *ItKr.m*
Lines 16 – 17: Assembling the global conductivity matrix
Lines 18 – 20: Solution of the system
Line 21 : Drawing the isotherms in the function *grisb.m*
Lines 12 – 24: Initialization of statistics variables
Line 25 - 27 : Starting the additional iterations in the function *radit.m*
Line 28 - 31 : Displaying additional results

The element can be vertical or horizontal. Its length is L, its thickness, e and the Stefan-Boltzmann coefficient, σ ($Wm^{-2}K^{-4}$). The global nodes sequence starts with the real ones pertaining to the mesh and finishes with the virtual ones. The function *ItKr.m* computes the radiation matrices of a mesh nx x ny (arguments 3 and 4 of the function) as pseudo convection matrices. The Stefan-Boltzmann constant is the last argument of the function. The temperatures of the solid are stored in the vector *tca* (argument 1) while the temperatures of the virtual nodes are in the vector *ts* (argument 2). Note that the quantity *he*/*ny* (*lines 9*, *17* and *25*) is simply the size of the square element. The argument *he* is giving the size of the domain.

Function Matlab© *ItKr.m* for radiation

```
1  function[Kr]=ItKr(tca,ts,nx,ny,th,he,SB)
2  no    = (nx+1)*(ny+1);Ntca = no+3;
3  Kelc  = [2 1 -3;1 2 -3;-3 -3 6]/6;             % Element radiation 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))*th*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 radiation 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))*th*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 radiation 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))*th*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 radiation coefficient
31 disp(['ItKr.m rig left top : ',num2str([crd crg crs])]);
32 end
```

Table 17: Matlab© function ItKr.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));grid 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 18: Matlab© function radit.m

The *radit.m* 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 are providing 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.

```
pp_radiation
conde.m uniform k    : 1 W/(mK)
Stefan-Boltzmann     : 5.67e-08 Wm-2K-4
Mesh dimension       : 20 x 40
Element size         : 0.05 m
Domain volume        : 2 m3
Base temperature     : 273 K
Virtual nodes temp.  : 290  290  303 K
Number of iterations: 2
Equivalent h, r.l.t : 5.53 5.53 5.92 W/(m2K)
Equivalent h, r.l.t : 5.5  5.5  6.19 W/(m2K)
radit.m, heat bottom: -45.7616 W
Min. mesh temper.    : 274.35 274.34 K
Total convect. flow :  46.59  45.76 W
Cpu                  : 0.441 sec.
```

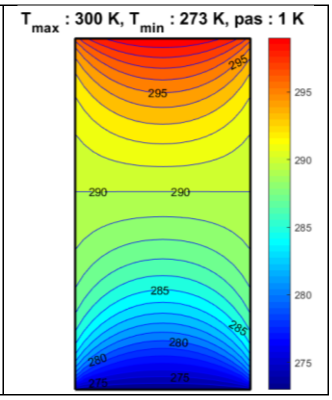


Figure 28: Two iterations of a radiative heat transfer

The isotherms are computed for the first and the second iteration, but, because the difference between iterations is insignificant, only one isotherm diagram is shown. The coefficients equivalent to the convective ones which are computed according to (1.57) are changing with the iteration and can be different on the three faces. Their mean values are given in each iteration (*Figure 29*).

Exercise n°4: Converting convection to radiative boundary conditions

We propose to use the same boundary conditions as in the application presented in *Figure 25*, but the convection conditions are transformed in radiation ones. However, we use 3 prescribed temperatures for the 3 virtual nodes. For the third one, we use the solution of *Figure 25*. What about the convergence of this problem ?

```
pp_radiation

conde.m uniform k      : 2 W/(mK)
Stefan-Boltzmann       : 5.67e-08 Wm-2K-4
Mesh dimension         : 20 x 40
Element size           : 0.05 m
Domain volume          : 2 m3
Base temperature       : 273 K
Virtual nodes temp.    : 286  290  303 K
Mean equivalent h      : 5.4494 W/(m2K)
Mean equivalent h      : 5.5315 W/(m2K)
radit.m, heat bottom:  -62.09 W
Total convect. flow :  62.91 62.09 W
Cpu                    : 0.323 sec.
```

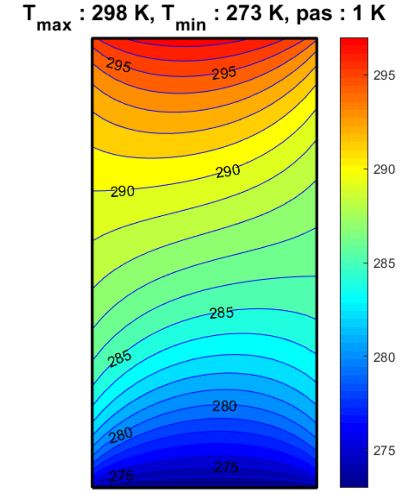


Figure 29: Isocurves for exercise 3

After running the procedure `pp_radiation`

and introducing the Matlab© sequence:

```
figure;Hf(nx,ny,lK,[tca;ones(nx+1,1)*tb], co)
```

we obtain the following numerical and graphical results:

`Elem. heat flow max: 64.3, mean: 25.4` Wm^{-2}

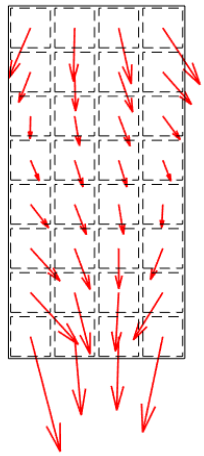


Table 19: Heat flows in the radiative problem of Figure 29 .

```
pp_radiation

conde.m uniform k      : 2 W/(mK)
Stefan-Boltzmann       : 5.67e-08 Wm-2K-4
Mesh dimension         : 10 x 20
Element size           : 0.1 m
Domain volume          : 2 m3
Base temperature       : 273 K
Virtual nodes temp.    : 286  290  303 K
Number of imp. iter.: 2
Mean equivalent h      : 5.4494 W/(m2K)
Mean equivalent h      : 5.5311 W/(m2K)
Max elem. heat flow : 97.3, mean: 25.8 W/m2
radit.m, heat bottom: -62.4592 W
Total convect. flow : 63.31      62.46 W
Cpu                    : 0.562 sec.
```

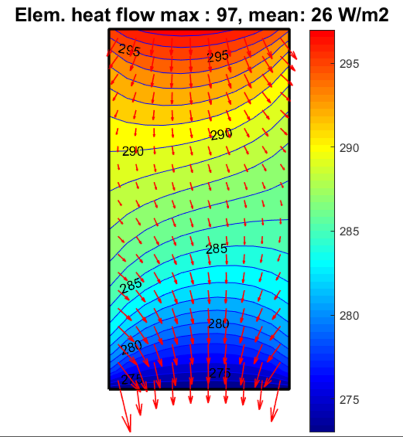


Figure 30: Isocurves for exercise 4 with superposition of heat flow vectors

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).

$$\alpha = \frac{k}{\rho c_p} \tag{1.60}$$

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

$$\left([C]+\theta\ \Delta t\ [K]\right)\left[T^{n+1}\right]=\left([C]-(1-\theta)\ \Delta t\ [K]\right)\left[T^{n}\right]+\Delta t\ \left(\theta\left[f^{n+1}\right]+(1-\theta)\left[f^{n}\right]\right) \tag{1.61}$$

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

$$\left([C]+\Delta t\ [K]\right)\left[T^{n+1}\right]=[C]\left[T^{n}\right]+\Delta t\ \left[f^{n+1}\right] \tag{1.62}$$

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:

$$\left(\begin{bmatrix} C_{11} & C_{1f} \end{bmatrix}+\Delta t\ \begin{bmatrix} K_{11} & K_{1f} \end{bmatrix}\right)\begin{bmatrix} T_{11}^{n+1} \\ T_{1f} \end{bmatrix}=\begin{bmatrix} C_{11} & C_{1f} \end{bmatrix}\begin{bmatrix} T_{11}^{n} \\ T_{1f} \end{bmatrix}+\Delta t\ \left[f^{n+1}\right] \tag{1.63}$$

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

$$\left[C_{11}\right]\left[T_{11}^{n+1}\right]+\Delta t\ \begin{bmatrix} K_{11} & K_{1f} \end{bmatrix}\begin{bmatrix} T_{11}^{n+1} \\ T_{1f} \end{bmatrix}=\left[C_{11}\right]\left[T_{11}^{n}\right]+\Delta t\ \left[f^{n+1}\right] \tag{1.64}$$

$$\left(\left[C_{11}\right]+\Delta t\ \left[K_{11}\right]\right)\left[T_{11}^{n+1}\right]\ =\left[C_{11}\right]\left[T_{11}^{n}\right]-\Delta t\ \left[K_{1f}\right]\left[T_{1f}\right]+\Delta t\ \left[f^{n+1}\right] \tag{1.65}$$

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

$$\left([C]+\Delta t\ [K]\right)\left[T^{n+1}\right]\ =[C]\left[T^{n}\right] \tag{1.66}$$

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

$$\tau = T_1\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right)+T_2\,\frac{x}{a}\left(1-\frac{y}{b}\right)+T_3\,\frac{x}{a}\frac{y}{b}+T_4\left(1-\frac{x}{a}\right)\frac{y}{b} \tag{1.67}$$

$$\begin{aligned}\tau &= [F][T]\\ [F] &= \left[\left(1-\frac{x}{a}\right)\left(1-\frac{y}{b}\right) \quad \frac{x}{a}\left(1-\frac{y}{b}\right) \quad \frac{x}{a}\frac{y}{b} \quad \left(1-\frac{x}{a}\right)\frac{y}{b}\right]\\ [T]^T &= \begin{bmatrix}T_1 & T_2 & T_3 & T_4\end{bmatrix}\end{aligned} \tag{1.68}$$

$$[C]=\int_V \rho c_p [F]^T [F]\,dV \tag{1.69}$$

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.

$$\begin{aligned}C_{33} &= e\int_0^a\left(\int_0^b \rho c_p \frac{x^2y^2}{a^2b^2}\,dy\right)dx\\ &= \rho e c_p\int_0^a \frac{b^3x^2}{3a^2b^2}\,dx = \rho e c_p\,\frac{b}{3}\int_0^a \frac{x^2}{a^2}\,dx = \rho e c_p\,\frac{ab}{9} = \frac{\rho V c_p}{9}\end{aligned} \tag{1.70}$$

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}.

$$[C]=\frac{\rho V c_p}{36}\begin{bmatrix}4 & 2 & 1 & 2\\ 2 & 4 & 2 & 1\\ 1 & 2 & 4 & 2\\ 2 & 1 & 2 & 4\end{bmatrix} \tag{1.71}$$

Procedures used for transient heat transfers

Procedure Matlab© pp_transient.m for transient heat transfer

```
nx  = 21; ni =30 ; pe =33 ; dt=pe*3600;gap=.5;                  % Input data
pte = [290;290;303; % Sky temperature                       East, West, Top
             303;       % Initial temperature in transient analysis
             293;       % Wall base temperature
          300;300;300]; % temperatures of atmosphere :          East, West, Top
pa6= 1       ;  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));
        tcan        = [tca(1:no)'  tcan(no+1:Ntca)']';
    else
        if nf > 0                                   % Fixations are present
            Kif = K(1:no-nf,no-nf+1:Ntca); % dt/ni = time step, see line 1
            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));
            if nf > 0;tca(no-nf+1:no) = tb;end     % Prescribing base temp.
        tcan        = [tca(1:no-nx-1)'  tcan(no-nx:Ntca)']';
        else                                 % No fixations : adiabatic boundary
            tca     = (C+dt/ni*K)\(C*tcan);
            tcan    = tca;
        end
    end
tsmax(it) = max(tca(1:no-nf)); tsmin(it)=min(tca(1:no-nf)); % Inter. nodes
```

```
tmoy(it)  = sum(tca(1:no-nf))/size(tca(1:no-nf),1); % Stat. interior nodes
end
grisb(nx,ny,tca,gap)                       % Drawing the isotherms on the domain
if nx <51;figure;Tg(nx,ny,lK,tca);hold on;end  % Drawing temperature grad.
if nx <51;figure;Hf(nx,ny,lK,tca,co);hold on;end      % Drawing heat flows
% ta=min(tsmax(1),ti);tb=min(tsmin(1),ti);tc=min(tmoy(1),ti);
ta=min(tsmax(1),ti);tb=min(tsmin(1),ti);tc=min(tmoy(1),ti);
grhi(ni,dt,[ta;tsmax],[tb;tsmin],[tc;tmoy])    % 2. Time evolution of temp.
disp(['Cpu                   : ',num2str(toc(tst),'%0.3g'),' sec.'])
```

Table 20: Matlab© procedure pp_ transient.m

The parameter *pa6* allows controlling the procedure (*nx* is the number of elements in the *x* direction *nf* is the number of fixations):

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*

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

```
function [] = grhi(ni,dt,tsmax,tsmin,tmoy)      % Evolution of temperatures
tem         = (0:ni)*dt/3600/ni;         % Time steps for the time graphics
figure('Position',[100 100 700 300]);
plot  (tem,tsmax','r');hold on;
plot  (tem,tsmin','b');hold on;
plot  (tem,tmoy  ,'k');hold on;grid on
legend('T maximum ','T minimum ','T average')
xlabel('Duration (hours) ','fontsize',12)
ylabel('grhi: Temperature (K)','fontsize',12)
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 21: Matlab© function grhi.m

Function Matlab© *CoKr34.m*

```
function[K] = CoKr34(nx,ny,ep,he,hh,pa6)
% disp(['CoKr.m cnv. coeff.  : ',num2str(hh),' W/(m2K)'])
Kelc  = [2 1 -3;1 2 -3;-3 -3 6]*hh*ep*he/ny/6;         % Elem. conv. matrix
if pa6 == 2
    Ntca  = (nx+1)*(ny+1)+4;                        % 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)])
else
    Ntca  = (nx+1)*(ny+1)+4;                        % Mesh nx x ny elements
    ntv3  = [Ntca-3 Ntca-2 Ntca-1 Ntca];  % Numb. convective virtual nodes
    lc    = locc(nx,ny,ntv3);  % Local. of the convection matrices 3 sides
            disp(['Virtual conv. nodes : ',num2str(ntv3)])
end
% if nx*ny < 19; disp (lc); end
K      = zeros(Ntca,Ntca);% Dimension of K including fixed DOF & add. nodes
for n = 1:size(lc,1);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);end;end;end
if pa6==1;Ntca=(nx+1)*(ny+1);K=zeros(Ntca,Ntca);end
end
```

Table 22: Matlab© function CoKr34.m

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.

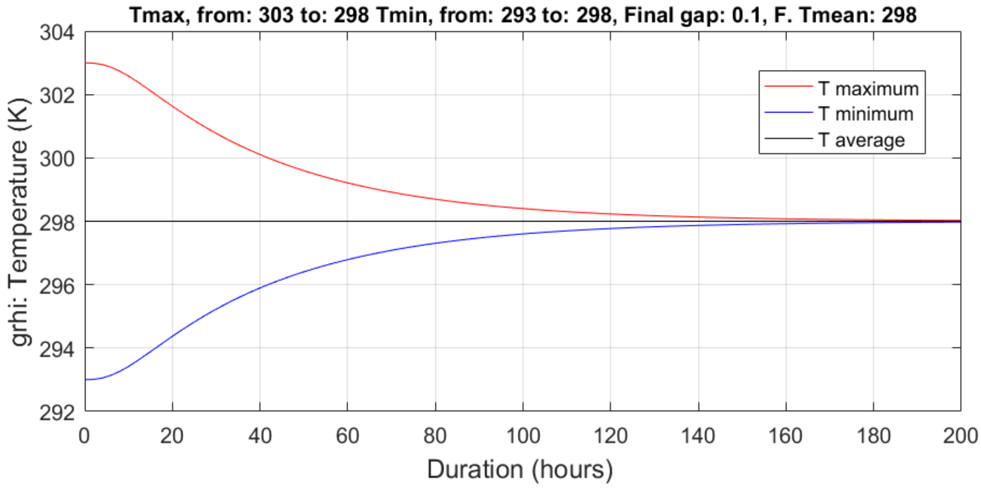


Figure 31: Convergence of the smoothing process

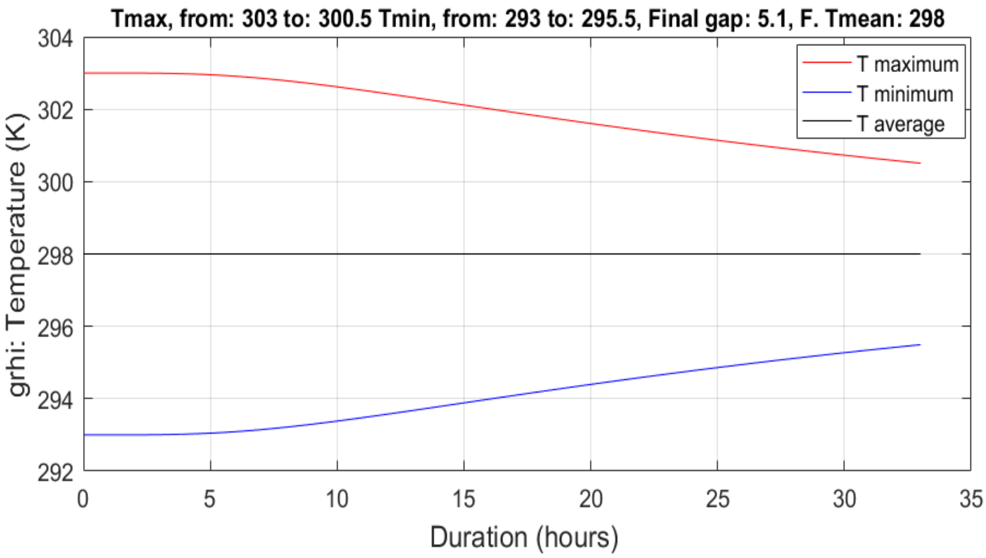


Figure 32: Beginning of the time evolution curve

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

```
pp_transient
Control param.  pa6 : 1
Convection coeff. h : 25 W/(m2K)
Specific heat       : 1000 J.m-3.K-1
Specific mass       : 2500 kg.m-3
Base temperature    : 293 K
Initial temperature : 303 K
conde.m uniform k   : 2 W/(mK)
Domain dimensions   : 1 x 0.1 x 2 m
Air temperatures    : 300  300  300 K
Mesh size           : 21 x 42
N. virtual nodes t3 : 0
Number of fixations : 0
Diffusivity         : 8e-07 m2/s
Total duration      : 33 h
Number of iterations: 30
Virtual conv. nodes : 947  948  949  950
Tot. cap.sum(sum(C)): 500000 J/K
Stat. Ntca no nf t3 : 946  946    0    0
Cpu                 : 0.825 sec.
```

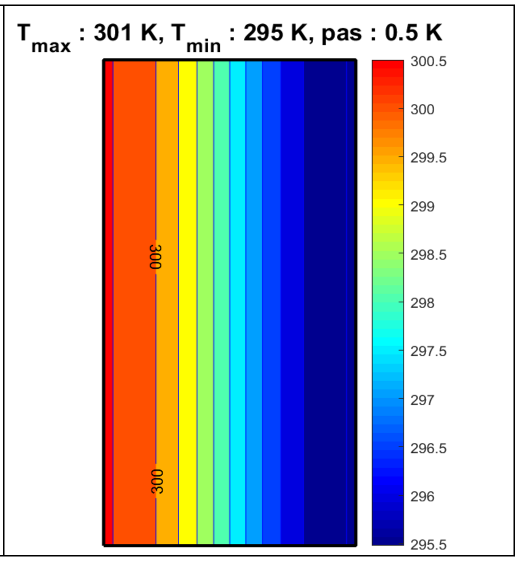


Figure 33: Evolution of the temperature field in a smoothing process

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*.

```
pp_transient
Control param.  pa6 : 2
Convection coeff. h : 25 W/(m2K)
Specific heat       : 1000 J.m-3.K-1
Specific mass       : 2500 kg.m-3
Base temperature    : 280 K
Initial temperature : 280 K
conde.m uniform k   : 1 W/(mK)
Domain dimensions   : 1 x 0.1 x 2 m
Air temperatures    : 300 300 300 300 K
Mesh size           : 20 x 40
N. virtual nodes t3 : 4
Number of fixations : 0
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
Element capacity    : 625 J/K
Tot. cap.sum(sum(C)): 500000 J/K
Stat. Ntca no nf t3 : 865  861    0  4
- gradT element max : 19.5, mean: 10.7 K/m
Max elem. heat flow : 19.5, mean: 10.7 W/m2
Cpu                 : 1.1 sec.
```

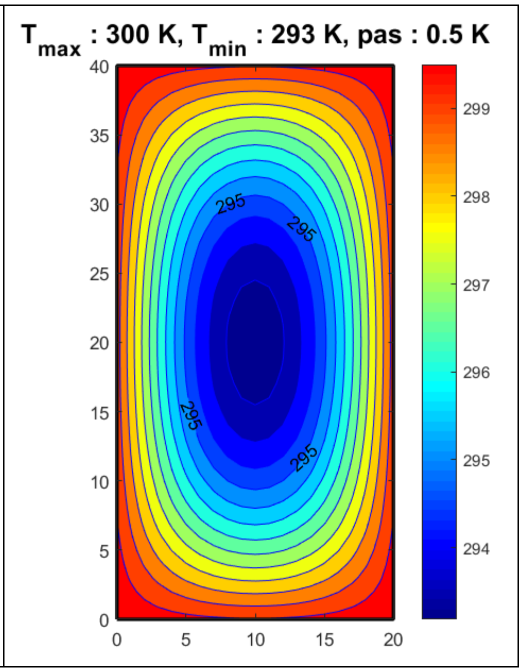


Figure 34: Evolution of the temperature field in a heating operation

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.

To obtain correct result in *Figure 35* and *Figure 36*, the *lines 96* and 97 of *Table 20* have to be swapped.

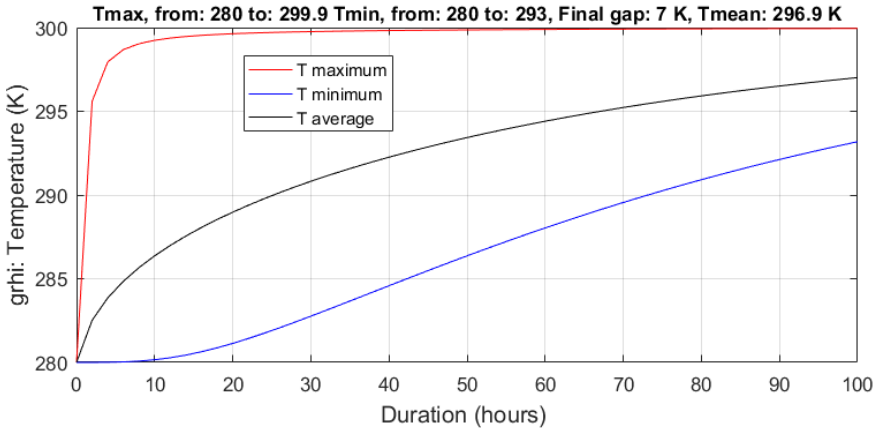


Figure 35: Evolution of max, min and mean temperatures in a heating operation: 100h

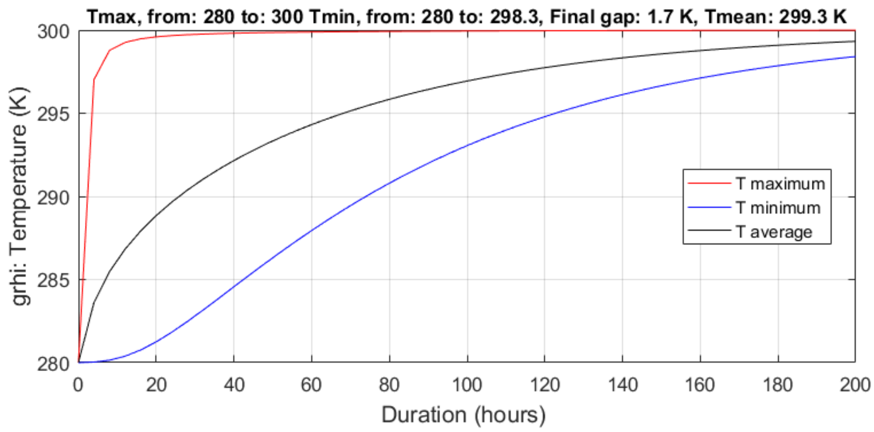


Figure 36: Evolution of max, min and mean temperatures in a heating operation: 200h

Exercise n°5: Cooling of a domain initially at uniform temperature

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.

```
pp_transient
Control param.  pa6 : 2
Convection coeff. h : 25 W/(m2K)
Specific heat       : 1000 J.m-3.K-1
Specific mass       : 2500 kg.m-3
Base temperature    : 280 K
Initial temperature : 300 K
conde.m uniform k   : 1 W/(mK)
Domain dimensions   : 1 x 0.1 x 2 m
Air temperatures    : 280  280  280  280 K
Mesh size           : 20 x 40
N. virtual nodes t3 : 4
Number of fixations : 0
Diffusivity         : 4e-07 m2/s
Total duration      : 100 h
Number of iterations: 40
N. of convective el.: 120
Virtual conv. nodes : 862  863  864  865
Tot. cap.sum(sum(C)): 500000 J/K
Stat. Ntca no nf t3 : 865  861    0    4
- gradT element max : 19.5, mean: 10.7 K/m
Max elem. heat flow : 19.5, mean: 10.7 W/m2
Cpu                 : 0.951 sec.
```

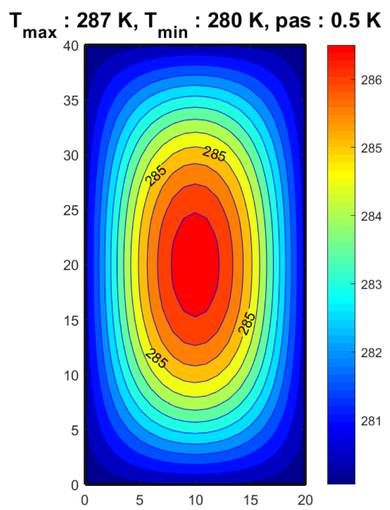


Figure 37: Isocurves for exercise of tutorial IV

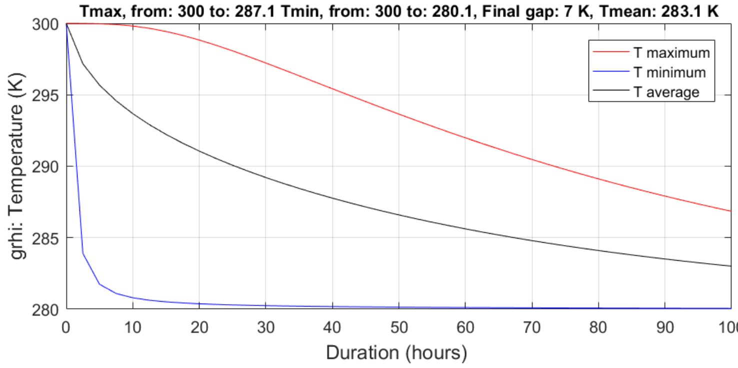
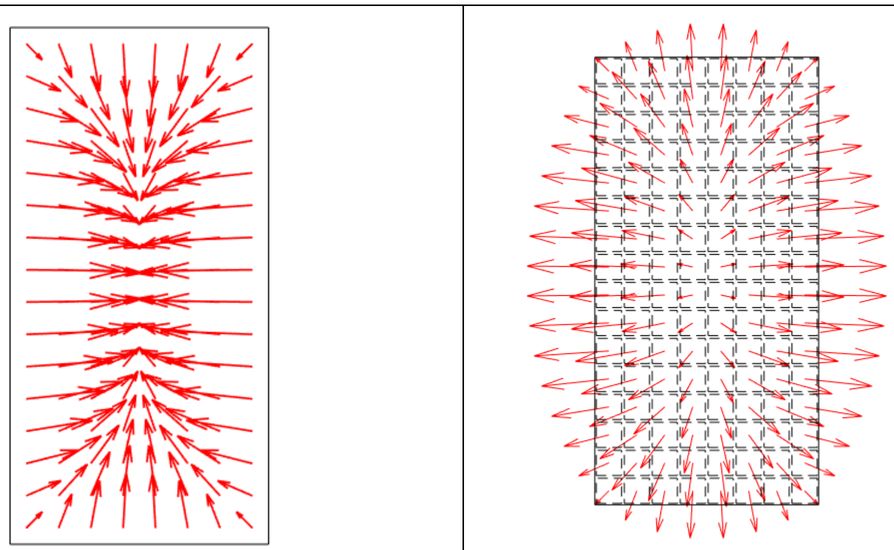


Figure 38: Evolution of temperatures for exercise of tutorial IV



Heat flow, max: 19 Wm^{-2}, mean: 11 Wm^{-2} (*Figure 34*)	Heat flow, max: 19 Wm^{-2}, mean: 11 Wm^{-2}

Figure 39: Heat flows in pure heating (left) or cooling (right) processes

About the time integration

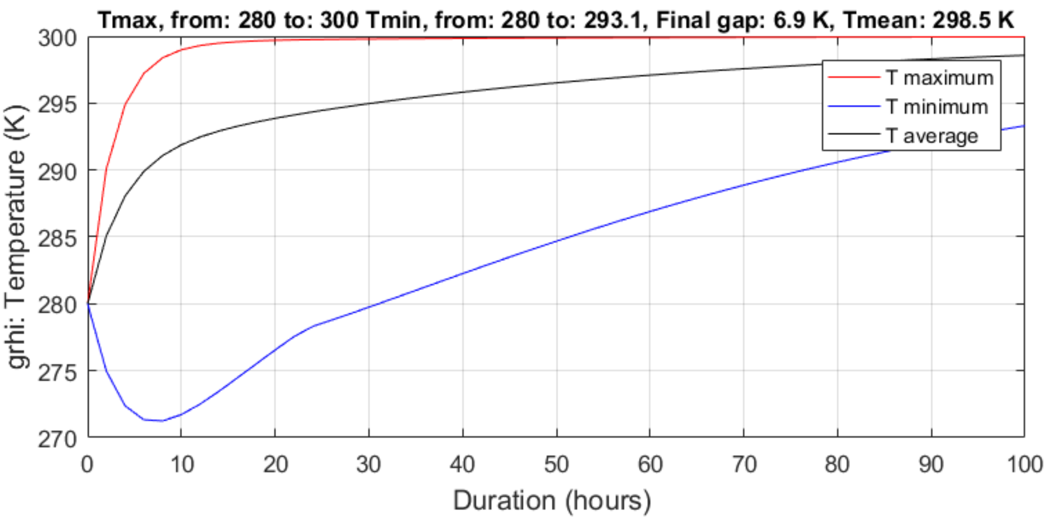
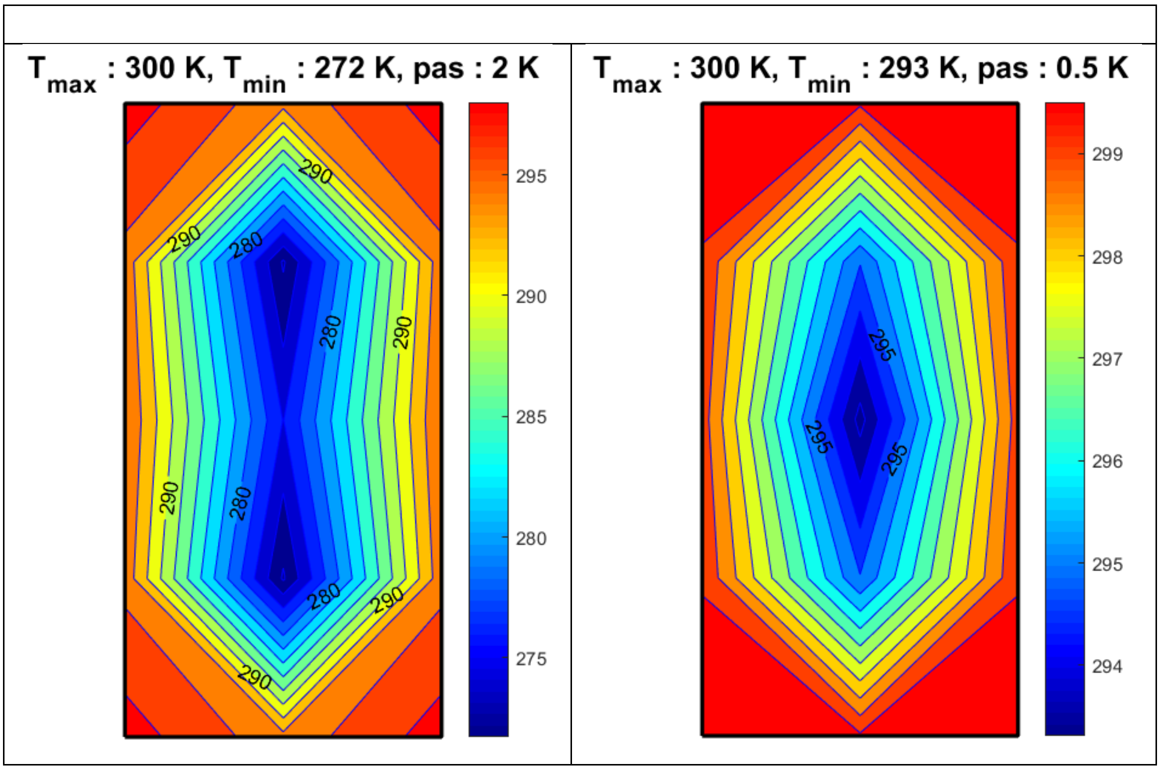


Figure 40: Temperature evolution in a very coarse mesh

If we modify the mesh of the example shown in *Figure 34*, we obtain some strange results when the mesh is very coarse (2 x 4 mesh). We observe in *Figure 40* that the minimum temperature is decreasing inside the mesh and that it needs 30 iterations to become again greater than the initial one.




```
pp_transient
Control param.  pa6 : 2
Convection coeff. h : 25 W/(m2K)
Specific heat       : 1000 J.m-3.K-1
Specific mass       : 2500 kg.m-3
Base temperature    : 280 K
Initial temperature : 280 K
conde.m uniform k   : 1 W/(mK)
Domain dimensions   : 1 x 0.1 x 2 m
Air temperatures    : 300 300 300 300 K
Mesh size           : 20 x 40
N. virtual nodes t3 : 4
Number of fixations : 0
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
Element capacity    : 625 J/K
Tot. cap.sum(sum(C)): 500000 J/K
Stat. Ntca no nf t3 : 865  861    0  4
- gradT element max : 19.5, mean: 10.7 K/m
Max elem. heat flow : 19.5, mean: 10.7 W/m2
Cpu                 : 1.1 sec.
```

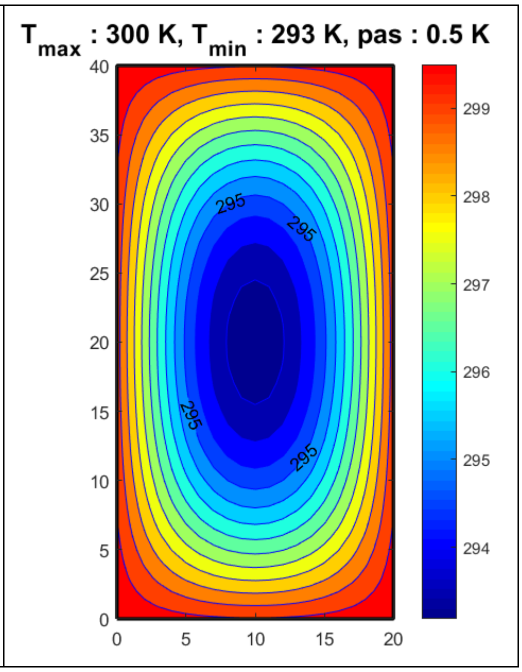


Figure 41: Evolution of the temperature field in a heating operation

Tutorial V: Isoparametric elements for heat transfer

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

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 rewrite the nodes definition of the Coons patch (quadrilateral) in 2D.

$$[Q]=\begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{bmatrix}=\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}=\begin{bmatrix} X & Y \end{bmatrix} \tag{1.72}$$

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 $[F]$:

$$\begin{bmatrix} x(s,t) & y(s,t) \end{bmatrix}=[F][Q]=\begin{bmatrix} (1-s)(1-t) & s(1-t) & st & (1-s)t \end{bmatrix}[Q] \tag{1.73}$$

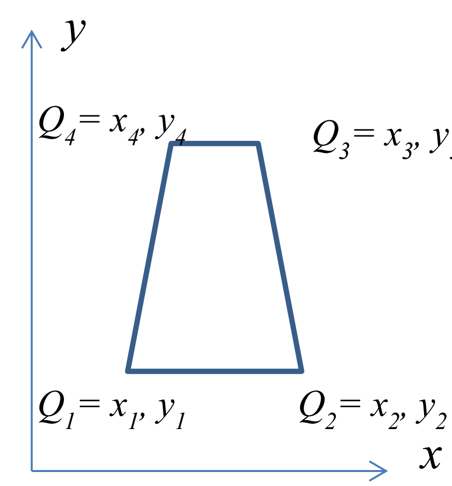
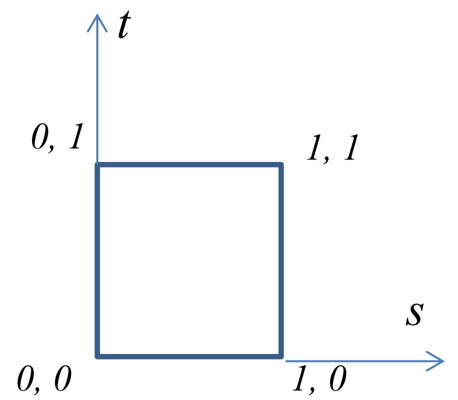
Cartesian space x, y $dS = dx\,dy$	Parametric space s, t $dS = J(s,t)\,ds\,dt$
 $[X]=\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$ $[Y]=\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$	

Table 23: Coons patch definition in Cartesian and parametric spaces

For the bilinear element, the Jacobian matrix $[J]$ is equal to:

$$[J]=\begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{bmatrix}=\begin{bmatrix} \begin{bmatrix} -(1-t) & (1-t) & t & -t \end{bmatrix}[X] & \begin{bmatrix} -(1-t) & (1-t) & t & -t \end{bmatrix}[Y] \\ \begin{bmatrix} -(1-s) & -s & s & (1-s) \end{bmatrix}[X] & \begin{bmatrix} -(1-s) & -s & s & (1-s) \end{bmatrix}[Y] \end{bmatrix} \tag{1.74}$$

Its determinant J is called the **jacobian of the transformation**. In the "center" of the patch computed in parametric coordinates, $s = 0.5$, $t = 0.5$, we have (see *lines 14 &15* of the function of *Table 36*):

$$[J]_{s=.5,\ t=.5} = \frac{1}{4}\begin{bmatrix} [-1 \quad 1 \quad 1 \quad -1][X] & [-1 \quad 1 \quad 1 \quad -1][Y] \\ [-1 \quad -1 \quad 1 \quad 1][X] & [-1 \quad -1 \quad 1 \quad 1][Y] \end{bmatrix} \tag{1.75}$$

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

$$[J]_{s=t=.5} = \frac{1}{4}\begin{bmatrix} x_2 + x_3 - x_1 - x_4 & y_2 + y_3 - y_1 - y_4 \\ x_4 + x_3 - x_1 - x_2 & y_4 + y_3 - y_1 - y_2 \end{bmatrix} \tag{1.76}$$

At point s = 0.5, t = 0.5, the jacobian of the transformation which is the scalar function corresponding the the determinant of the jacobian matrix, is then:

$$\begin{aligned} J_{s=t=.5} &= (x_2 + x_3 - x_1 - x_4)(y_4 + y_3 - y_1 - y_2) \\ &\quad -(x_4 + x_3 - x_1 - x_2)(y_2 + y_3 - y_1 - y_4) \\ J_{s=t=.5} &= (x_2 - x_4)(y_3 - y_1) + (x_3 - x_1)(y_4 - y_2) \end{aligned} \tag{1.77}$$

The gradient of a scalar function, for instance the temperature $\tau(s, t)$, is computed as follows. It was initially computed in parametric coordinates, but we should have it in Cartesian ones (the real world).

$$\begin{bmatrix} \dfrac{\partial \tau}{\partial s} \\ \dfrac{\partial \tau}{\partial t} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial x}{\partial s} & \dfrac{\partial y}{\partial s} \\ \dfrac{\partial x}{\partial t} & \dfrac{\partial y}{\partial t} \end{bmatrix} \begin{bmatrix} \dfrac{\partial \tau}{\partial x} \\ \dfrac{\partial \tau}{\partial y} \end{bmatrix} = [J] \begin{bmatrix} \dfrac{\partial \tau}{\partial x} \\ \dfrac{\partial \tau}{\partial y} \end{bmatrix} \tag{1.78}$$

After inverting (1.78), (see *line 16* of the function of *Table 36*); we obtain:

$$\begin{bmatrix} \dfrac{\partial \tau}{\partial x} \\ \dfrac{\partial \tau}{\partial y} \end{bmatrix} = [J]^{-1} \begin{bmatrix} \dfrac{\partial \tau}{\partial s} \\ \dfrac{\partial \tau}{\partial t} \end{bmatrix} \tag{1.79}$$

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:

$$\tau = \begin{bmatrix} (1-s)(1-t) & s(1-t) & st & (1-s)t \end{bmatrix} [T] \tag{1.80}$$

We can easily compute its derivatives with respect to s and t in the center of the patch:

$$\begin{bmatrix} \dfrac{\partial \tau}{\partial s} \\ \dfrac{\partial \tau}{\partial t} \end{bmatrix} = \begin{bmatrix} -(1-t) & (1-t) & t & -t \\ -(1-s) & -s & s & (1-s) \end{bmatrix} [T] \tag{1.81}$$

$$
\begin{bmatrix} \dfrac{\partial \tau}{\partial s} \\ \dfrac{\partial \tau}{\partial t} \end{bmatrix}_{s=t=.5} = \frac{1}{4}\begin{bmatrix} -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix}[T] \tag{1.82}
$$

The two components of the following equations correspond to the *lines 17 & 18* of the function *Hflo.m* (*Table 36*). They represent the temperature gradient

$$
grad\ \ \tau = \begin{bmatrix} \dfrac{\partial \tau}{\partial x} \\ \dfrac{\partial \tau}{\partial y} \end{bmatrix} = [J]^{-1}\frac{1}{4}\begin{bmatrix} -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{bmatrix}[T] \tag{1.83}
$$

Procedures for conduction, convection, radiation & transient heat transfers

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 24: Matlab© procedure pp_visopa_conduction.m for isoparametric elements

Matlab© Procedure *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 conductivity 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];
    disp(['Heat on virt. nodes : ',num2str(hvn'),' W'])
    disp(['H convl cond. convr : ',num2str([qxg qxc qxd]),' Wm-2'])
    disp(['Surf. T. right left : ',num2str([max(tca) min(tca)]),' K'])
end
if nfc == 3    % 2. One free and two imposed temperatures of virtual nodes
    if ta(2)==0.
        K11 = K(1:no-nx-1,1:no-nx-1);
        K12 = K(1:no-nx-1,no-nx:no);
        K13 = K(1:no-nx-1,no+1);
        K14 = K(1:no-nx-1,no+2:no+3);
        K31 = K(no+1,1:no-nx-1);
        K32 = K(no+1,no-nx:no);
        K33 = K(no+1,no+1);
        K34 = K(no+1,no+2:no+3);
        T2  = ones(nx+1,1)*tb;
        T4  = [ta(3); ta(4)];
        tcb = [K11 K13;K31 K33]\(-[K12 K14;K32 K34]*[T2;T4]);
        tca = [tcb(1:no-nx-1); ones(nx+1,1)*tb; tcb(size(tcb,1)); T4];
        grisc(nx,ny,tca(1:no),xyz,gap);axis off   % Drawing  the isotherms
        Hflo(nx,ny,xyz,lK,[tca;ones(nx+1,1)*tb],co);
        disp(['Virtual nodes temp. : ',...
            num2str(tca(no+1:size(K,1))',3),' K'])
        disp(['Mesh min max Temp.  : ',num2str([min(tca(1:no)) ...
               max(tca(1:no))],3),' K'])
    else
        gco   = K(1:no-nx-1,no-nx:size(K,1))*[ones(nx+1,1)*tb;ta(2:4)];
        disp(['Imposed temperatures: ',num2str([tb;ta(2:4)]',3),' K'])
        tca   = -K(1:no-nx-1,1:no-nx-1)\gco;
        grisc(nx,ny,[tca;ones(nx+1,1)*tb],xyz,gap);axis off % Drawing iso.
%         Hflo(nx,ny,xyz,lK,[tca;ones(nx+1,1)*tb]);
        sm    =(K*[tca;ones(nx+1,1)*tb; ta(2:4)])';          % second member
        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 25: 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*.

$$K_{convection} = h\,\frac{eL}{6}\begin{bmatrix} 2 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & 6 \end{bmatrix} \tag{1.84}$$

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.
The function *Coco.m* allows computing the convection matrices of an *nx* x *ny* mesh (arguments 1 and 2 of the function). The convection coefficient is given by argument 4. Argument 5 is giving the number of faces where convection elements are present (*Table 14*).

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 two convective end two 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 19*), 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.

Lines 38 - 70 : Solution of other problems.

Matlab© procedure *pp_visopa_radiation.m* including isoparametric elements

```
tb    = 273;pge=[10;20;1;20];pf=[0;12.5;0];ts=[290;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 ============================
```

```
tst  = tic;          % Beginning the analysis, initialisation of the timer
co   = conde(nx,ny);
xyz  = Nxyz(nx,ny);                              % Geometry ; nodal coordinates
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'])
th   = pge(3);
tca  = ones(no,1)*min(ts);
lK   = loca(nx,ny);          % Computing the localization matrix (nel x 4)
nit  = 2;
[K ] = ItKr (tca,ts,nx,ny,pge(3),pge(2),SB);
for n=1:nel
    Kel   = th*co(n)*Kelu(xyz,lK(n,:));        % Element conductivity matrix
    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))+Kel(i,j);end;end;
end
gco   =  K(1:no-nx-1,no-nx:no+size(K,1)-no)*[ones(nx+1,1)*tb; ...
    ts(1:size(K,1)-no,1)];
tca  = -K(1:no-nx-1,1:no-nx-1)\gco;gap=1;
grisc(nx,ny,[tca ;ones(nx+1,1)*tb],xyz,gap);axis off   % Drawing isotherms
tmi1    = min(tca(1:no-nx-1));
sm      = (K*[tca;ones(nx+1,1)*tb;ts(1:size(K,1)-no)])'; % syst. sec. memb.
hvn1    = sum(sm(size(sm,2)-2:size(sm,2)));
% Additional iterations =====================================================
if nit > 1
[hvn,tmi]=radit(tca,nx,ny,tb,ts,pge(3),pge(2),lK,Kel,co,nit,tmi1,hvn1,SB);
end
disp(['Number of iterations: ',num2str(nit)]) % Numb. iterations
disp(['Min. mesh temper.   : ',num2str(tmi(1:min(5,nit))',5),' K'])
disp(['Total convect. flow : ',num2str(hvn(1:min(5,nit))',5),' W'])
disp(['Cpu                 : ',num2str(toc(tst),'%0.3g'),' sec.'])
```

Table 26: Matlab© procedure pp_visopa_radiation.m for isoparametric elements

Matlab© Procedure *pp_visopa_transient.m* for transient heat transfer

```
nx  = 8; ni =50; pe =100 ; dt=pe*3600;gap=1;                    % Input data
pte = [300;300;300; % Sky temperature                        East, West, Top
        280;        % Initial temperature in transient analysis
        250;        % Wall base temperature
      300;300;300]; % temperatures of atmosphere :           East, West, Top
                disp('************************')
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'])
th = 0.1    ;  disp(['Thickness           : ',num2str(th),' m'])
wi = 1;he=2;                          % Width, height & thickness of the wall
ny =2*nx;my = ny+1;nel = nx*ny;no = (nx+1)*(ny+1);       % Mesh definition
co = conde(nx,ny);th=co(1);
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)])
nfc   = 4;
C     = zeros(Ntca,Ntca);   % Initialization of the global capacity matrix
xyz   = Nxyz(nx,ny);lK = loca(nx,ny);                    % Geometry & topology
[K ]  = Coco(nx,ny,xyz,hh*th,nfc);               % Computing the convection K
for n     = 1:nel                                  % Loop on the nel elements
    Kel  = th*co(n)*Kelu(xyz,lK(n,:));      % Element conductivity matrices
    Cel  = Cp*ro*th*Celu(xyz,lK(n,:));            % Element capacity 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);
          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));
        tcan        = [tca(1:no)'  tcan(no+1:Ntca)']';
    else
        if nf > 0                                   % Fixations are present
            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));
            if nf > 0;tca(no-nf+1:no) = tb;end    % Prescribing base temp.
        tcan        = [tca(1:no-nx-1)'  tcan(no-nx:Ntca)']';
        else                                 % No fixations : adiabatic boundary
            tca      = (C+dt/ni*K)\(C*tcan);
            tcan     = tca;
        end
    end
tsmax(it)  = max(tca(1:no-nf)); tsmin(it)=min(tca(1:no-nf)); % Inter. nodes
tmoy(it)   = sum(tca(1:no-nf))/size(tca(1:no-nf),1); % Stat. interior nodes
end
grisc (nx,ny,tca,xyz,gap);axis off;        % Graphical output: 1. isotherms
grhi(ni,dt,[ti;tsmax],[ti;tsmin],[ti;tmoy])        % 2. Time evolutions of T
Hflo(nx,ny,xyz,lK,tca,co);% mgra(nx,ny,xyz,lK,tca); % 3. Heat flow, arrows
disp(['Cpu                  : ',num2str(toc(tst),'%0.3g'),' sec.'])
```

Table 27: Matlab© procedure pp_visopa_transient.m

Functions related to the isoparametric formulation

Matlab© function *Coco.m* to compute the conductivity matrix used for convection

```
function[K] = Coco(nx,ny,xyz,hh,nfc)
Kelc  = [2 1 -3;1 2 -3;-3 -3 6] * hh/6;         % Element convection matrix
nuc   = (nx+1)*(ny+1);
Ntca  = nuc + nfc;            % Dimension of matrix K, mesh nx x ny elements
ntv   = zeros(1,4);                 % convective indices of the 4 sides Index
if nfc ==2                % Virtual convection nodes on the 2 vertical sides
ntv   = [Ntca-1 Ntca    0     0];% Numbering of the convective virtual nodes
```

```
end
if nfc ==3                                      % Virtual convection nodes on 3 sides
ntv    = [Ntca-2 Ntca-1 Ntca 0];% Numbering of the convective virtual nodes
end
if nfc ==4                                      % Virtual convection nodes on 4 sides
ntv   = [Ntca-3 Ntca-2 Ntca-1 Ntca];         % Number. convective virtual nodes
end
disp(['Virtual conv. nodes : ',num2str(ntv )])
lc       = locc (nx,ny,ntv );      % Localizations of the convection matrices
nn       = size(xyz,1);                                       % Nodes of the mesh
nuc      = nn;
if nuc < 51;figure;end             % Drawing performed only for coarse meshes
nel=0;for i  = 1:size(lc,1);if lc(i,1)>0;nel=i;end;end        % Compacting lc
K        = zeros(Ntca,Ntca);     % Dim. of K including fixed DOF & virt. nodes
for n  = 1:nel                   % Assembling the convective conductive matrix
    Ls = sqrt((xyz(lc(n,1),1)-xyz(lc(n,2),1))^2+...
        (xyz(lc(n,1),2)-xyz(lc(n,2),2))^2);% K depends of the elem. length
    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
if nuc < 51                 % Drawing the labels of the convective elements
    xyz = povi(nx,ny,xyz,ntv);  % Introducing coord. for the virtual nodes
    mesh_nu(nx,ny,xyz)          % Nodes & elements labels: conductive part
    mesh_co(xyz,lc);axis equal  % Nodes & elements labels: convective part
end
end
```

Table 28: Matlab© function Coco.m for the conductivity matrix used for convection

Matlab© function *povi.m* to compute the positions of the virtual convective nodes

```
function [xyz] = povi(nx,ny,xyz,ntv)% Compute the virtual nodes positions
nn               = (nx+1)*(ny+1);     % and display virtual nodes labels
nuc              = nn;
ra               = 3; % ratio side length over distance side to virtual node
for si = 1 : size(ntv,2)              % Loop on the four sides of the domain
    if ntv(si) > 0
        nuc  = nuc+1;
        if si==1
            P1           = xyz(nn,:);          P2           = xyz(nx+1,:);
            xvi          = (P1(1)+P2(1))/2+(P2(2)-P1(2))/ra;
            yvi          = (P1(2)+P2(2))/2+(P1(1)-P2(1))/ra;
            xyz(nuc,:) = [xvi yvi 0];
        end
        if si==2
            P1           = xyz(1,:);           P2           = xyz(nn-nx,:);
            xvi          = (P1(1)+P2(1))/2+(P2(2)-P1(2))/ra;
            yvi          = (P1(2)+P2(2))/2+(P1(1)-P2(1))/ra;
            xyz(nuc,:) = [xvi yvi 0];
        end
        if si==3
            P1           = xyz(nx+1,:);         P2           = xyz(1,:);
            xvi          = (P1(1)+P2(1))/2+(P2(2)-P1(2))/ra;
            yvi          = (P1(2)+P2(2))/2+(P1(1)-P2(1))/ra;
            xyz(nuc,:) = [xvi yvi 0];
        end
        if si==4
            P1           = xyz(nn-nx,:);         P2           = xyz(nn,:);
            xvi          = (P1(1)+P2(1))/2+(P2(2)-P1(2))/ra;
            yvi          = (P1(2)+P2(2))/2+(P1(1)-P2(1))/ra;
            xyz(nuc,:) = [xvi yvi 0];
        end
        text(xyz(nuc,1),xyz(nuc,2),num2str(nuc),'Color','b');
    end
end;end
```

Table 29: Matlab© function povi.m: positions of the virtual convective nodes

Matlab© function *mesh.m* for drawing shrink mesh

```
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 30: Matlab© function mesh.m

Matlab© function *mesh_nu.m* for vizualization of mesh, nodes & elements labels

```
function []=mesh_nu(nx,ny,xyz)                    % Drawing the shrinked mesh
nel     = nx*ny;X=zeros(5,1);Y=zeros(5,1);lK = loca(nx,ny);
sh      = 1; %0.9;                        % Shrinking coefficient 0 < sh <= 1
for j   = 1:nel           % Elements are numbered left - right, top - bottom
    ce = zeros(2,1);
    for i      = 1:4
        ce(1)  = ce(1)+xyz(lK(j,i),1)/4;         % x coord. of element center
        ce(2)  = ce(2)+xyz(lK(j,i),2)/4;         % y coord. of element center
        X(i)   = xyz(lK(j,i),1);         % x coord. of element nodes sequence
        Y(i)   = xyz(lK(j,i),2);         % y coord. of element nodes sequence
    end
    X(5)=X(1);Y(5)=Y(1);                           % Initial node is repeated
    plot((1-sh)*ce(1)+sh*X,(1-sh)*ce(2)+sh*Y,'k');hold on
    text(ce(1),ce(2),num2str(j),'Color','r');hold on
end
tox =(max(xyz(:,1))-min(xyz(:,1)))/(20*nx); % Tolerance for label position
for i=1:size(xyz,1)
    text(xyz(i,1)+tox/2,xyz(i,2)+1.1*tox,num2str(i),'Color','b');hold on;
    axis off
end;end
```

Table 31: Matlab© function mesh_nu.m: mesh, elements & nodes labels

Matlab© function *mesh_co.m* convective mesh visualization + nodes & elements labels

```
function []=mesh_co(xyz,lc)          % Drawing the shrinked convective mesh
nel                 = size(lc,1);X=zeros(4,1);Y=zeros(4,1);
sh                  = .9;                 % Shrinking coefficient 0 < sh <= 1
for j               = 1:nel  % Elem. are numbered left - right, top - bottom
    if lc(j,1)      > 0
        ce          = zeros(2,1);
        for i       = 1:3
            ce(1) = ce(1)+xyz(lc(j,i),1)/3;     % x coord. of element center
            ce(2) = ce(2)+xyz(lc(j,i),2)/3;     % y coord. of element center
            X(i)  = xyz(lc(j,i),1);     % x coord. of element nodes sequence
            Y(i)  = xyz(lc(j,i),2);     % y coord. of element nodes sequence
        end
        X(4)        = X(1);Y(4)=Y(1);              % Initial node is repeated
        plot((1-sh)*ce(1)+sh*X,(1-sh)*ce(2)+sh*Y,'--k');hold on
        text(ce(1),ce(2),num2str(j),'Color','m');hold on % Elements labels
    end
end
end
```

Table 32: Matlab© function mesh_co.m: convective mesh, virtual nodes labels

The Matlab© function *locc.m* (*Table 33*) allows computing the localizations of the convection matrices on one to four sides of the domain. The function *mesh_co.m* (*Table 32*) draws the convective elements of the domain and display on the same picture the convective nodes and elements numbers (the nodes are shown in blue (*Figure 18*); the conductive elements numbers are shown in red while the convective ones are shown in magenta). The computation is performed only if the edge virtual node pointer is set on in the 1 x 4 array *ntv* (third argument of the function *locc.m* of *Table 33*). The four positions of this array correspond to: right, left, top and bottom sides of the domain.

Matlab© function *locc.m* computes the localization matrix of the convective elements on one to four sides of the domain according to the vector *ntv*

```
function [lc]=locc(nx,ny,ntv)  % Localization vectors for conv. on 4 sides
no              = (nx+1)*(ny+1);lc = zeros(2*(ny+nx),3);ii = 0;
isu      = 0;
if ntv(1) > 0                                                  % Right side
    isu = isu+1;
    for i         = 1:ny
        lc(i,1)   = (nx+1)*i;lc(i,2) = lc(i,1)+nx+1;lc(i,3) = no+isu;
    end;      ii = ii + ny;
end
if ntv(2) > 0                                                   % Left side
    isu = isu+1;
    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+isu;
    end
end
if ntv(3) > 0                                                    % Top side
    isu = isu+1;
    for i         = 1:nx;                 ii = ii + 1;
        lc(ii,1) = i;lc(ii,2) = lc(ii,1)+1;lc(ii,3) = no+isu;
    end
end
if ntv(4) > 0                                                 % Bottom side
    isu = isu+1;
    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+isu;
    end
end
end
```

Table 33: Matlab© function locc.m to compute the localization matrices of convective elements

Matlab© function *locc2.m* computes the localization matrix of the convective elements

```
function [lc]=locc2(nx,ny,ntv) % Localization vectors for conv. on 3 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 34: Matlab© function locc2.m to compute the localization matrices

Matlab© function *locc4.m* computes 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 35: Matlab© function locc4.m to compute the localization matrices

Matlab© function *Hflo.m* for the visualization of the heat flow

```
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                          % Loop on the nx columns of elements
    for j  = 1:ny                      % Loop on the ny liness  of elements
        ii = ii+1;
        for k=1:4                     % Loop on the 4 vertices of the element
            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); % xx contains the 4 vertices x coord.
            yy(k) = xyz(lK(ii,k),2); % yy contains the 4 vertices y coord.
            te(k) = tca(lK(ii,k));   % te contains the 4 vertices temp.
        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;% x comp. of grad
        v(ii)     = -Jm1(2,:)/4*[-1 1 1 -1;-1 -1 1 1]*te;% y comp. of grad
    end
end
gm    = [max(sqrt(u.*u+v.*v)) mean(sqrt(u.*u+v.*v))];   % grad : max & av.
scale = 2;
disp(['Elem. heat flow max : ', num2str(gm(1),3),', average: ',...
    num2str(gm(2),3),' W/m2'])
figure;quiver(X,Y,u,v,scale,'r','LineWidth',1);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 36: Matlab© function Hflo.m: visualization of heat flow with arrows

Matlab© function *mgra.m* for the visualization of the temperature gradients

```
function [xyz] = mgra(nx,ny,xyz,lK,tca)
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);
        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(['- gradT, max         : ', num2str(gm(1)),', mean: ',...
    num2str(gm(2)),' K/m'])
figure;quiver(X,Y,u,v,scale,'b');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
% mesh(nx,ny,xyz,lK);hold on;
if nx < 10;mesh(nx,ny,xyz,lK);end
title(['- gradT, max: ', num2str(gm(1),2),', mean: ',num2str(gm(2),2),...
    ' K/m'],'fontsize',15);axis off
end
```

Table 37: Matlab© function mgra.m: visualization of temperature gradients with arrows

In the function *Hflo.m* of *Table 36*, we compute the gradient of the temperature in the centers of the elements. According to the property of super convergence obtained in the Gauss integration points, we state that the gradient evaluated at this point is suitable for representations using arrows symbols.

Other functions used for isoparametric elements

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

```
function [K] = Kelu(xyz,lo)
Q  = [xyz(lo(1),1:3); xyz(lo(2),1:3); xyz(lo(3),1:3); xyz(lo(4),1:3)];
s  = [.5-sqrt(3)/6 .5+sqrt(3)/6 .5+sqrt(3)/6 .5-sqrt(3)/6];  % 4 Gauss pts
t  = [.5-sqrt(3)/6 .5-sqrt(3)/6 .5+sqrt(3)/6 .5+sqrt(3)/6];  % 4 Gauss pts
K  = zeros(4,4);area=0.;
for i=1:4                                     % Loop on the 4 Gauss points
    fs   = [-(1-t(i)) (1-t(i)) t(i) -t(i)   ];              % Derivative s
    ft   = [-(1-s(i)) -s(i)    s(i) (1-s(i))];              % Derivative t
    gra  = [fs;ft];                % Gradient of the scalar bilinear function
    ds   = fs * Q;
    dt   = ft * Q;
    area = area         + sqrt(dot(cross(ds,dt),cross(ds,dt)))/4;
    J    = [fs*Q(:,1) fs*Q(:,2);ft*Q(:,1) ft*Q(:,2)];
    K=K+((J^(-1)*gra)'*J^(-1)*gra)*sqrt(dot(cross(ds,dt),cross(ds,dt)))/4;
end                                 % disp(['Patch area : ',num2str(area)])
end
```

Table 38: Matlab© function Kelu.m

This Matlab© function *Kelu.m* allows computing the conductivity matrix [*K*] (output of the function) of an isoparametric quadrilateral element with a bilinear temperature field. To obtain the true conductivity matrix, the output of the function 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 localization *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 Matlab evaluation of the conduction matrix is given in *Table 39*, using explicit definitions of both the coordinates and the localization vector. As it was 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 coordinates.

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

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

Table 39: 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 24*. As expected, when we run problems in rectangular domain, we obtain the same results as before. The procedure of *Table 24* exhibits the main characteristics of a finite element program (see the comments of lines 1 – 8).

It needs the definition of nodes coordinates computed in the Matlab© function *Nxyz.m* (*Table 40*). The arguments correspond to the definition of the mesh. The definition of the domain geometry is included inside the function. In the *Table 40*, 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.

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

```
1  function [xyz] = Nxyz(nx,ny)
2  ii = 0;nn=(nx+1)*(ny+1);xyz = zeros(nn,3);nc=2;% Nodes Coons patch nx x ny
3  P = [0 0 0; 2 0 0; 1.5 2 0; 0.5 2 0];                % Trapezoidal domain
4  for i = ny:-1:0
5     for j = 0:nx
6         t = i/ny; s = j/nx; ii = ii +1;
7         for c=1:nc
8             xyz(ii,c)= (1-s)*(1-t)*P(1,c)+ s*(1-t)    *P(2,c)+...
9                            s*t        *P(3,c)+(1-s)*t    *P(4,c);
10        end
11    end
12 end
13 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 40: Matlab© function Nxyz.m

In the *lines 4* and *5* of the procedure of *Table 24*, the functions of *Table 40* and *Table 4* outline the usual bases 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 41*).

Matlab© function *grisc.m* for the isotherms drawing

```
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 41: Matlab© function grisc.m for the drawing of the isotherms

Results for non rectangular domains

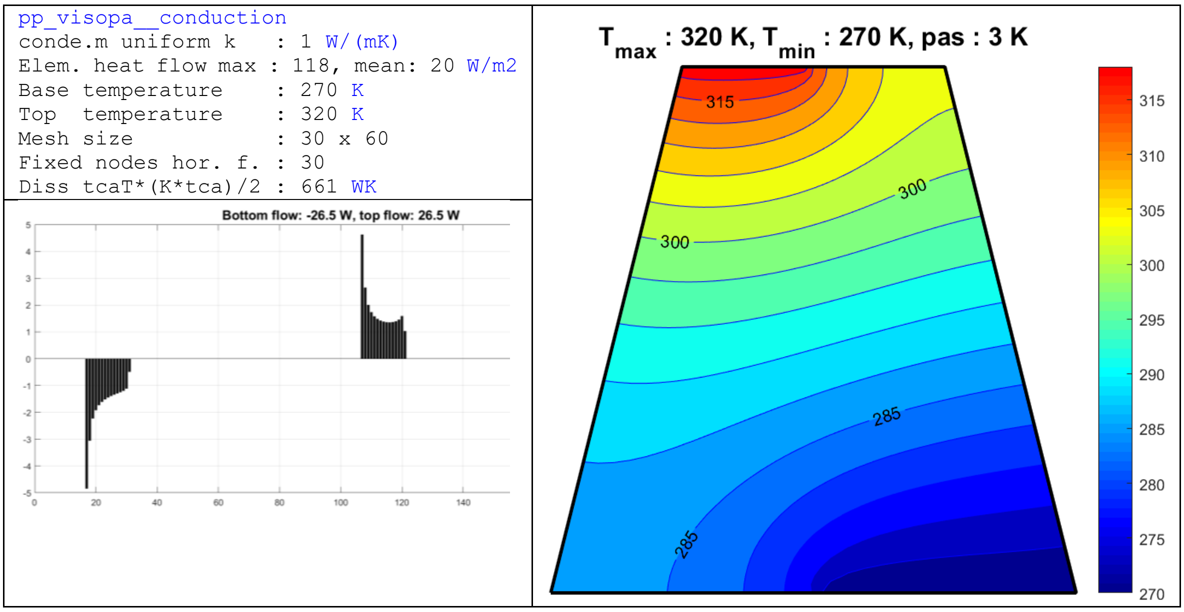


Figure 42: Imposed temperatures on a part of the horizontal faces (see Figure 4)

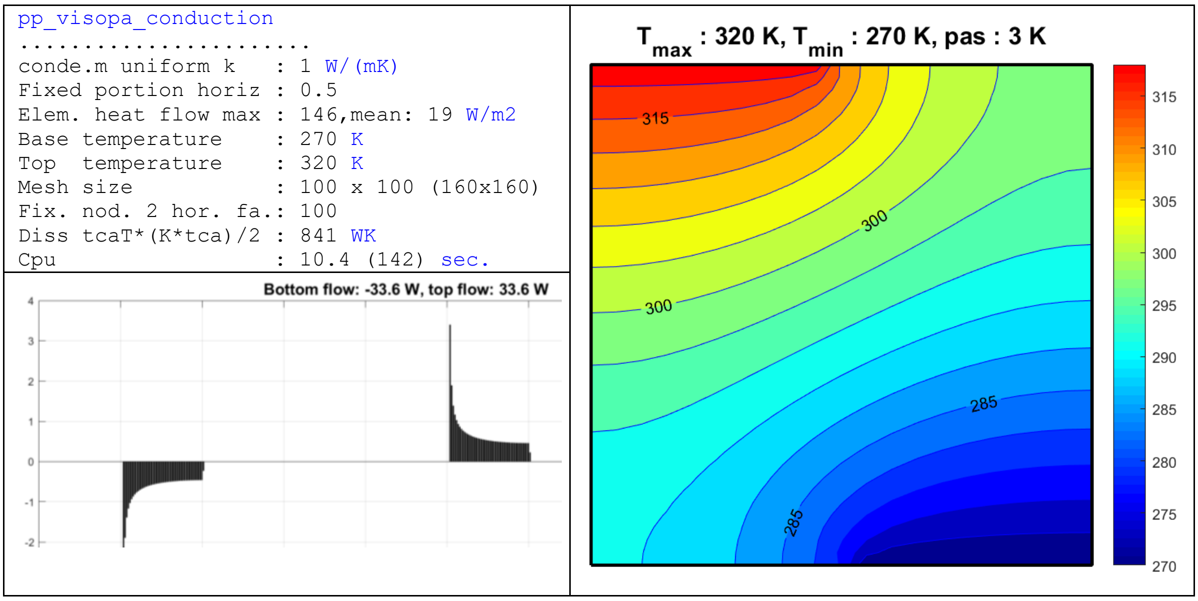


Figure 43: Imposed temperatures on a part of the horizontal faces (see Figure 4)

Exercise n°2c: Thermal bridge in a trapezoidal domain

Modify the shape of the domain analyzed in exercise n°1, *Figure 19*, (chapter on convective heat transfert). Again, check the consequence of introducing an horizontal thermal bridge.

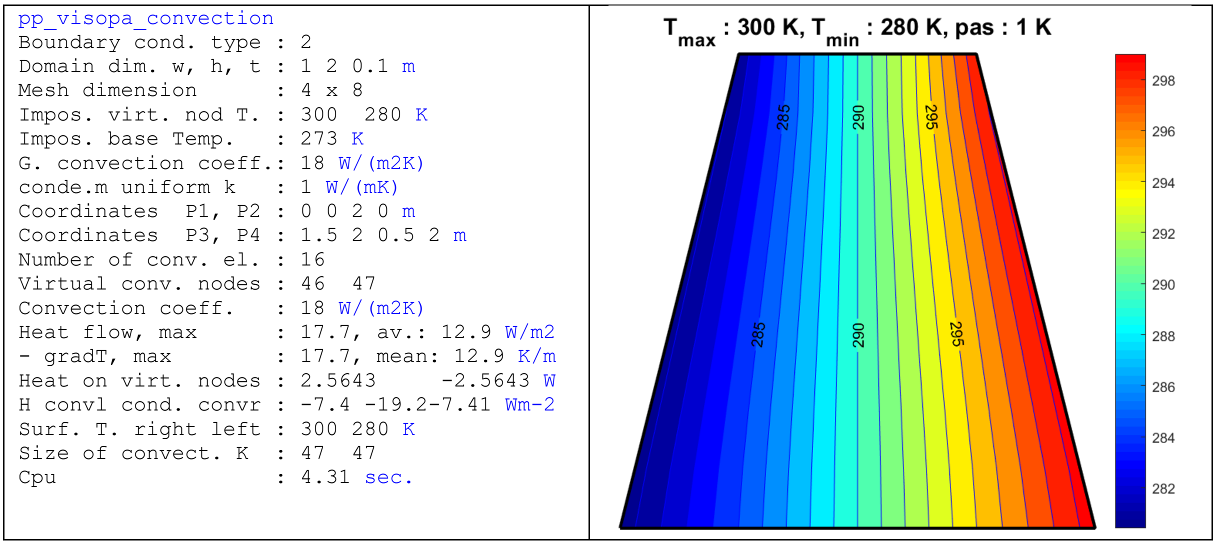


Figure 44: Two imposed temperatures, two adiabatic faces, and trapezoidal domain

```
pp_visopa_convection
Boundary cond. type : 2
Domain dim. w, h, t : 1 2 0.1 m
Mesh dimension      : 16 x 32
Impos. virt. nod T. : 300  280 K
Impos. base Temp.   : 273 K
G. convection coeff.: 18 W/(m2K)
Conductivity coeff. : 1 W/(m K)
Main & bridge cond. : 1  1000 W/(m K)
Rel. strip thickness: 0.0625
Number of conv. el. : 64
Virtual conv. nodes : 562  563
Convection coeff.   : 18 W/(m2K)
Heat flow, max      : 479, mean: 40 W/m2
- gradT, max        : 76, mean: 13  K/m
Heat on virt. nodes : 7.5 -7.5 W
H convl cond. convr : -8.8 -19 -8.8 Wm-2
Surf. T. right left : 300 280 K
K size (incl. conv.): 563  563
Cpu                 : 2.6 sec.
```

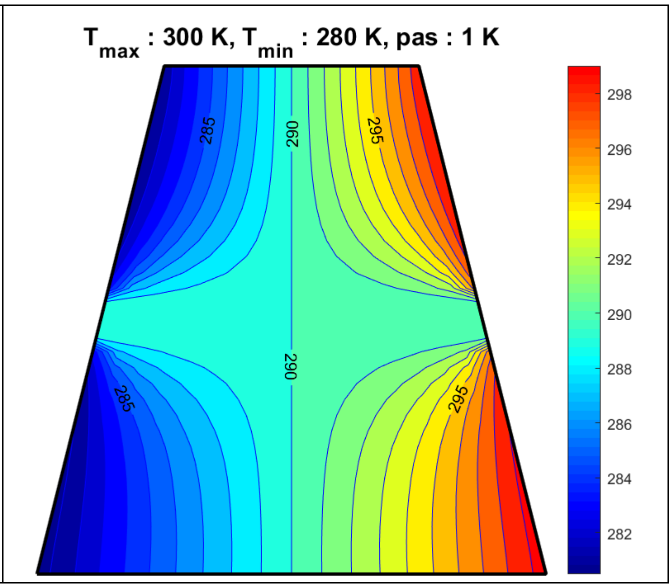


Figure 45: Non homogeneous trapezoidal domain

The introduction of non homogeneous material is performed using the definition, element by element, of the conductivity coefficient (see *Table 11* & *Table 12*).
These functions are writen with the hypotheses that the numbers of elements in the *x* and *y* directions satisfy certain conditions that can be checked in the listings of both functions.
The heat flow drawing is dominated by the arrows in the central zone, while in the gradient one the same zone of hight flows is disappearing due to their small value.

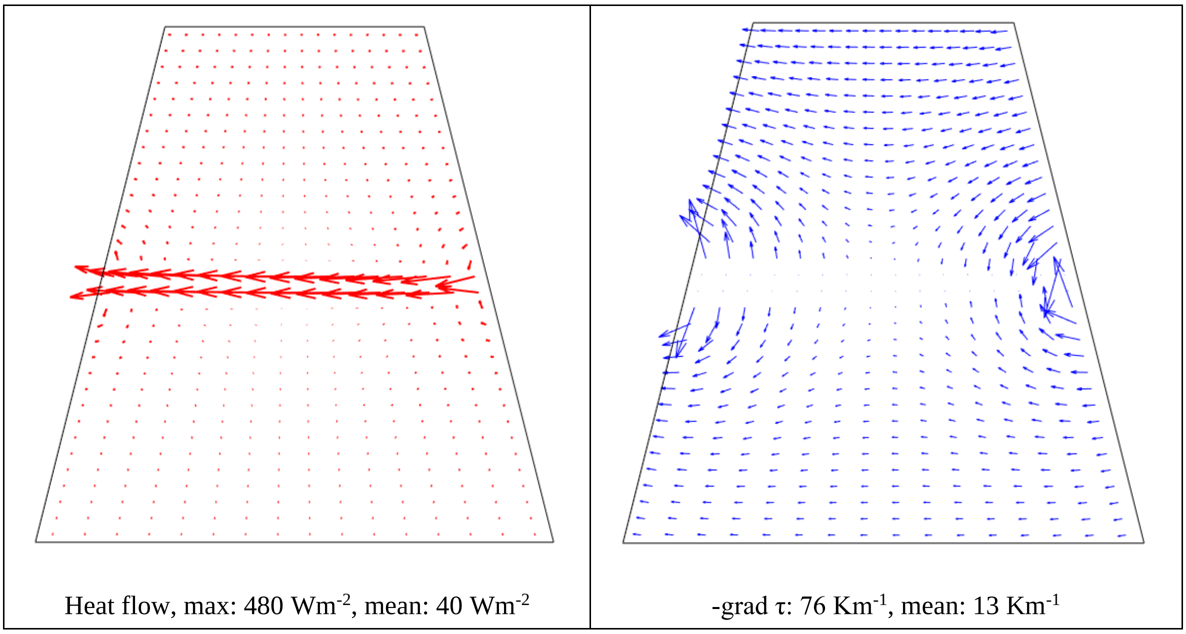


Figure 46: Horizontal strip with high conductivity in a trapezoidal domain

Exercise n°6: Non rectangular domain

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

We solve the same problem as in *Figure 7*, but, wth a constant conductivity coefficient.

```
pp_Base_isopa
conde.m conductivity: 1 W/(mK)
Coordinates  P1, P2 : 0  0  1  0 m
Coordinates  P3, P4 : 1  2  0  2 m
Base temperature    : 270 K
Top  temperature    : 320 K
Mesh size           : 30 x 60
Fix. nod. 2 hor. fa.: 30
Diss tcaT*(K*tca)/2 : 502 WK
```

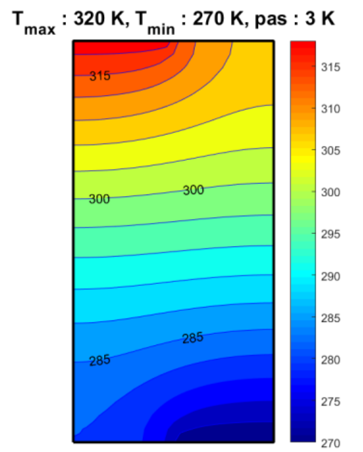


Figure 47: Isocurves for exercise of tutorial I

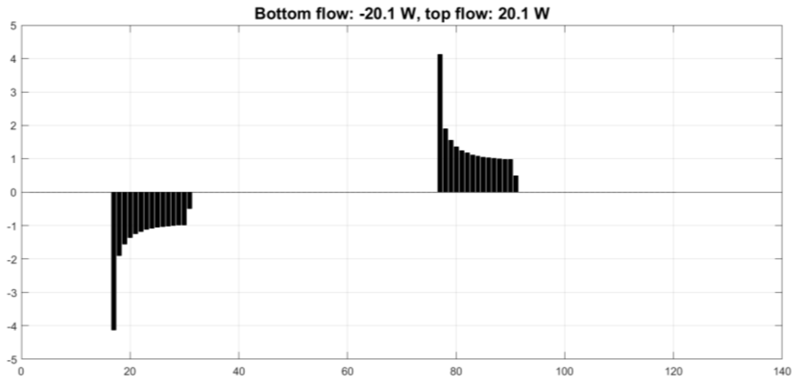
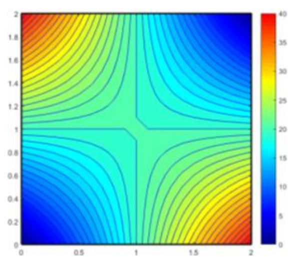


Figure 48: Heat flows

To modify the shape of the domain, we have to introduce the matrix of nodal positions (*Table 40*). According to this function, the shape of the domain is any quadrileteral meshed in *nx* by *ny* elments. When the geometry is 3D, it allows introducing a non planar shell element. However the finite elment developments included in this report are purely 2D. This function allows defining a function $z = f(x, y)$ and representing it with the function *grif.m*, which is able to represent any 3D patch in the plane (x, y) and with contour lines for the z coordinate. For the patch defined by the four points : `P = [0 0 0; 2 0 40; 2 2 0; 0 2 40]`, with the sentence: grif (20, 20, 1), we obtain:

```
function [] = grif(nx,ny,gap)
figure('Position',[1 1 600 512]);
no          = (nx+1)*(ny+1);ii=0;jj=0;my = ny+1;
xx          = zeros(my,nx+1);yy=zeros(my,nx+1);
xyz         = Nxyz(nx,ny);
z           = xyz(1:no,3);
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
```



```
pp_Base_isopa
conde.m conductivity: 1 W/(mK)
Coordinates  P1, P2 : 0   0  2  0 m
Coordinates  P3, P4 : 1.5 2 0.5 2 m
Base temperature    : 270 K
Top  temperature    : 320 K
Mesh size           : 30 x 60
Fix. nod. 2 hor. fa.: 30
Diss tcaT*(K*tca)/2 : 667 WK
```

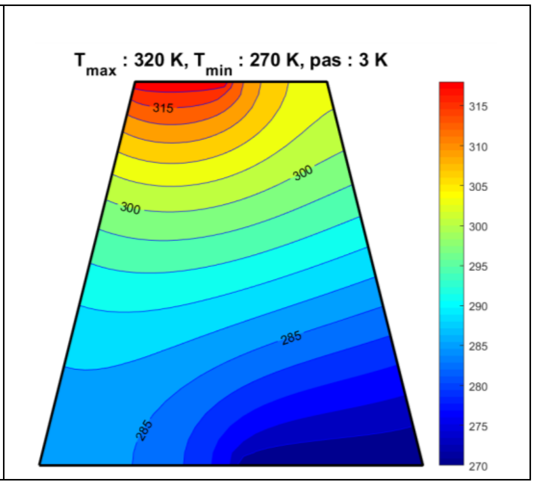


Figure 49: Isocurves for exercise 1

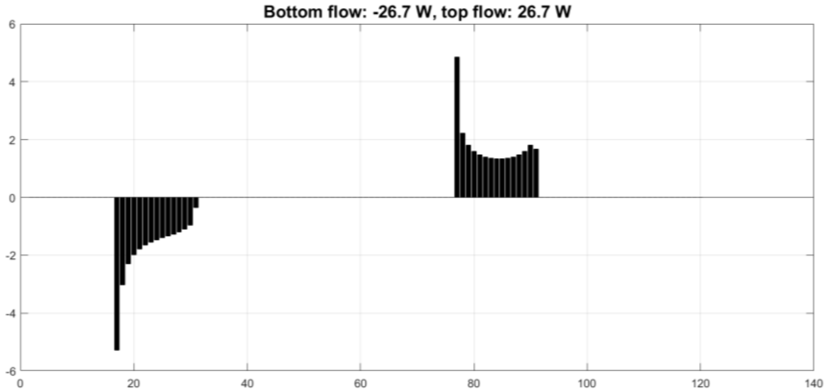


Figure 50: Heat flows

Exercise n°7: Final exercise for computing a capacity matrix

We propose to compute the capacity matix for an isoparametric element and to adapt the procedure of transient heat transfer problems.

Matlab© function *Celu.m* for the integration of the capacity matrix

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

Table 42: Matlab© function Celu.m, for the integration of the capacity matrix

For the example of *Figure 37*, the results are identical:

```
pp_visopa_transient
Control param.  pa6 : 2
Convection coeff. h : 25 W/(m2K)
Specific heat       : 1000 J.m-3.K-1
Specific mass       : 2500 kg.m-3
Base temperature    : 280 K
Initial temperature : 300 K
Thickness           : 0.1 m
conde.m uniform k   : 1 W/(mK)
Air temperatures    : 280  280  280  280 K
Mesh size           : 20 x 40
N. virtual nodes t3 : 4
Number of fixations : 0
Diffusivity         : 4e-07 m2/s
Total duration      : 100 h
Number of iterations: 40
Virtual conv. nodes : 862  863  864  865
Tot. cap.sum(sum(C)): 500000 J/K
Stat. Ntca no nf t3 : 865  861    0    4
Cpu                 : 1.31 sec.
```

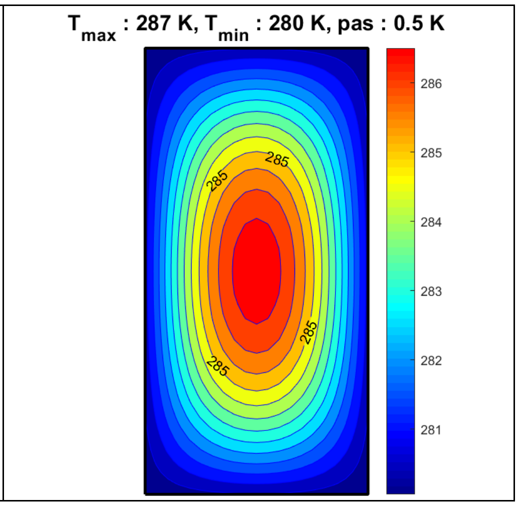


Figure 51: Isocurves for exercise 4

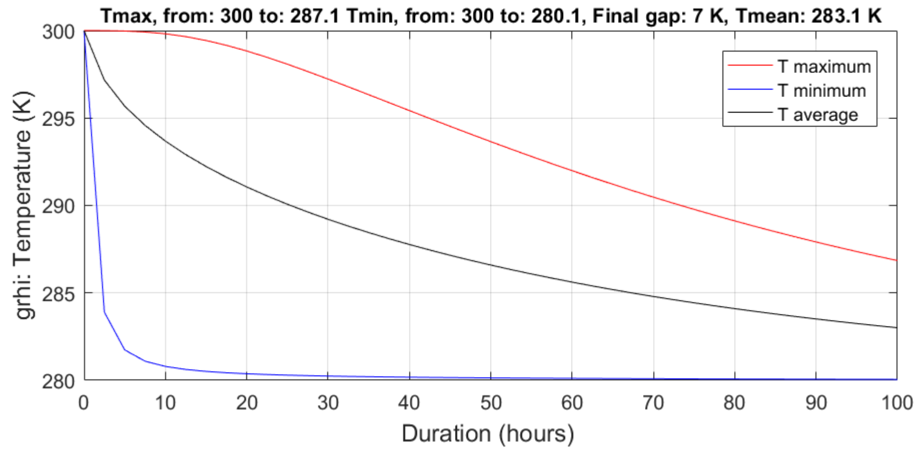


Figure 52: Evolution of temperatures for exercise 4

Matlab© function *CoKrv2.m* for the computation of the conduction-convection matrices

```
function[K] = CoKrv2(nx,ny,le)
co       = le./[nx ny nx ny]';
Kelc     = [2 1 -3;1 2 -3;-3 -3 6]/6;                    % Elem. conv. matrix
Ntca     = (nx+1)*(ny+1)+4;                 % Mesh nx x ny elements + virt nodes
ntv4     = [Ntca-3 Ntca-2 Ntca-1 Ntca];  % Number. convective virtual nodes
[lc,hs]  = loccv2(nx,ny,ntv4,co);      % Local. of the conv. matrices 4 sides
K        = zeros(Ntca,Ntca);
for n    = 1:size(lc,1);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))+hs(n)*Kelc(i,j);end;end;end
           disp(['CoKrv2.m conv. coef.: ',num2str(co',2),' W/K'])
           disp(['Virtual conv. nodes : ',num2str(ntv4)])
end
```

Table 43: Matlab© function CoKrv2.m

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 36*). The elements are shown if there are less than eleven elements in the *x* direction. The arrows are computed and drawn with or without the visualizaton of the shrink elements. The element are drawed with the Matlab© function *mesh.m* (*Table 30*).

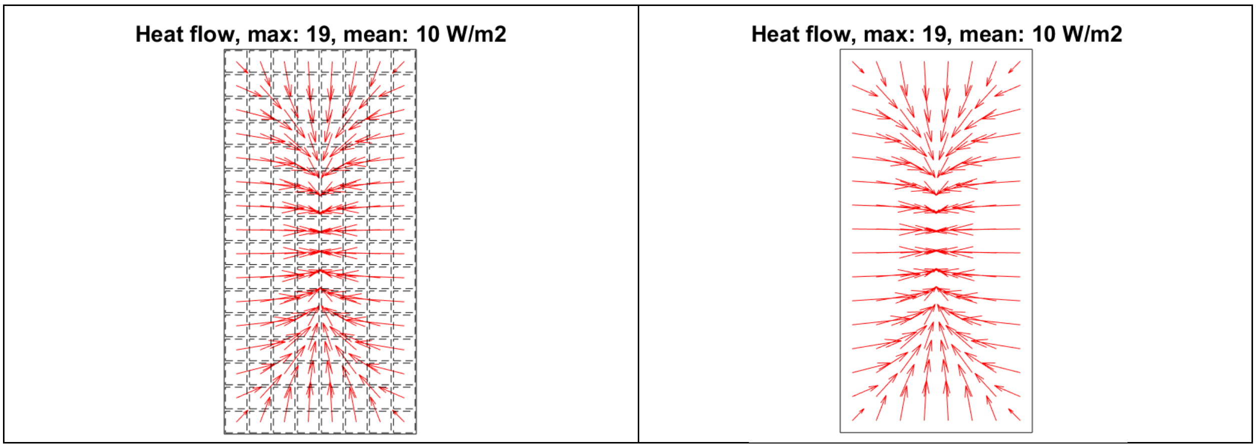


Figure 53: Heat flow represented with (left), or without (right) shrink elements

The same test is performed on a different shape.

```
pp_visopa_transient
Control param.  pa6 : 2
Convection coeff. h : 25 W/(m2K)
Specific heat       : 1000 J.m-3.K-1
Specific mass       : 2500 kg.m-3
Initial temperature : 300 K
Thickness           : 0.1 m
conde.m uniform k   : 1 W/(mK)
Air temperatures    : 280  280  280  280 K
Mesh size           : 20 x 20
N. virtual nodes t3 : 4
Number of fixations : 0
Diffusivity         : 4e-07 m2/s
Total duration      : 100 h
Number of iterations: 50
Coordinates  P1, P2 : 0 0 2 0 m
Coordinates  P3, P4 : 1.5 2 0.5 2 m
Virtual conv. nodes : 442  443  444  445
Tot. cap.sum(sum(C)): 750000 J/K
Stat. Ntca no nf t3 : 445  441   0   4
Elem. heat flow max : 26.2, average: 15.5 W/m2
Cpu                 : 4.84 sec.
```

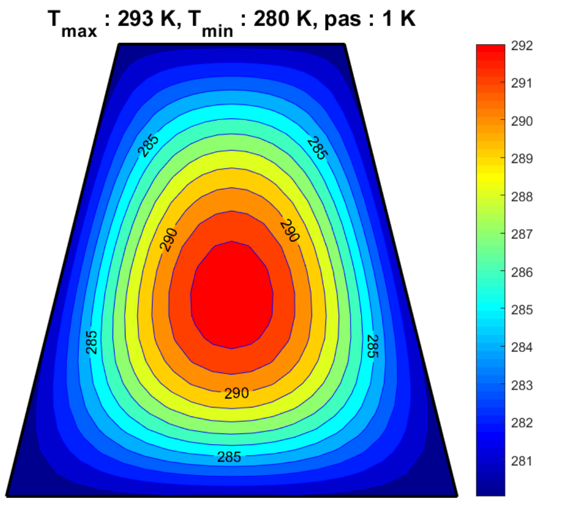


Figure 54: Isocurves for a trapezoidal mesh

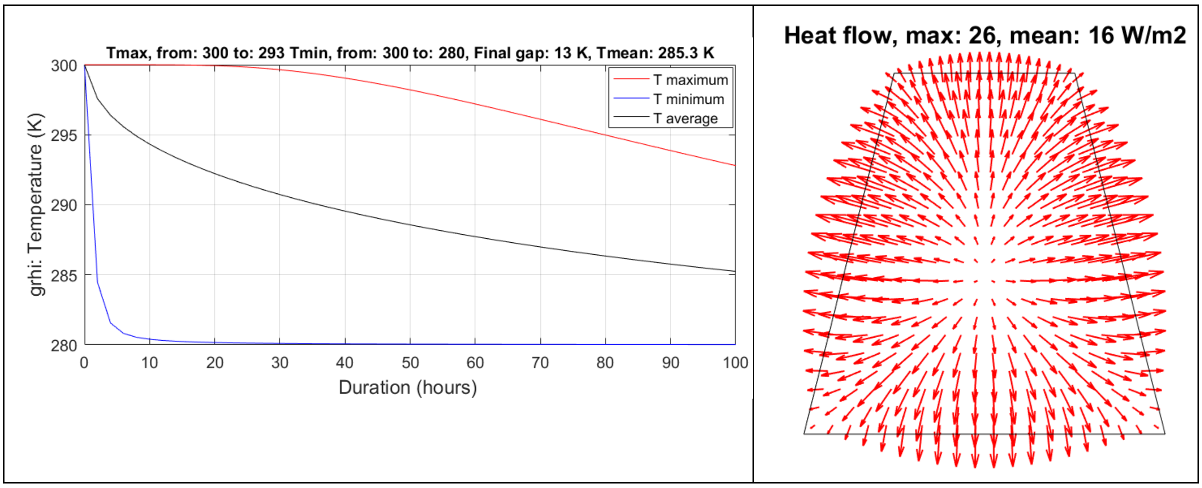


Figure 55: Evolution of temperatures and corresponding heat flows for a trapezoidal mesh

```
pp_visopa_transient
Control param.  pa6 : 2
Convection coeff. h : 25 W/(m2K)
Specific heat       : 1000 J.m-3.K-1
Specific mass       : 2500 kg.m-3
Base temperature    : 250 K
Initial temperature : 300 K
conde.m uniform k   : 1 W/(mK)
Air temperatures    : 280  280  280  280
K
Mesh size           : 3 x 3
N. virtual nodes t3 : 4
Diffusivity         : 4e-07 m2/s
Total duration      : 100 h
Number of iterations: 50
N. of convective el.: 12
Tot. cap.sum(sum(C)): 6250000 J/K
Stat. Ntca no nf t3 : 20  16   0   4
Elem. heat flow max : 19.7, aver.: 13.1
W/m2
Area                : 2.5 m2
Cpu                 : 2.16 sec.
```

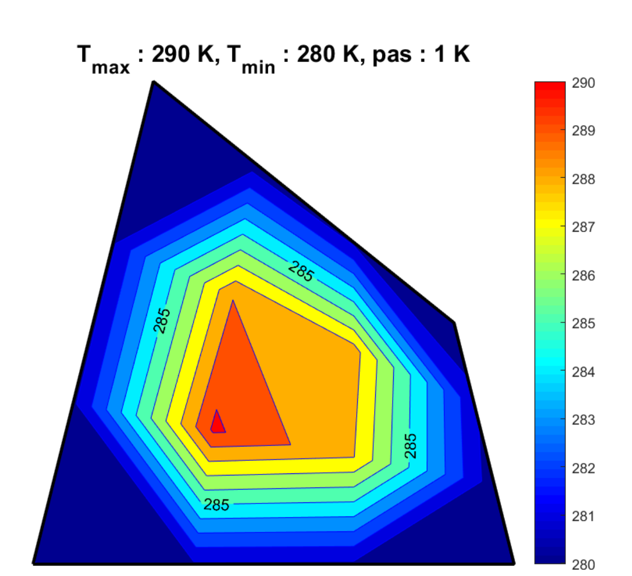


Figure 56: Isocurves for a quadrilateral coarse mesh

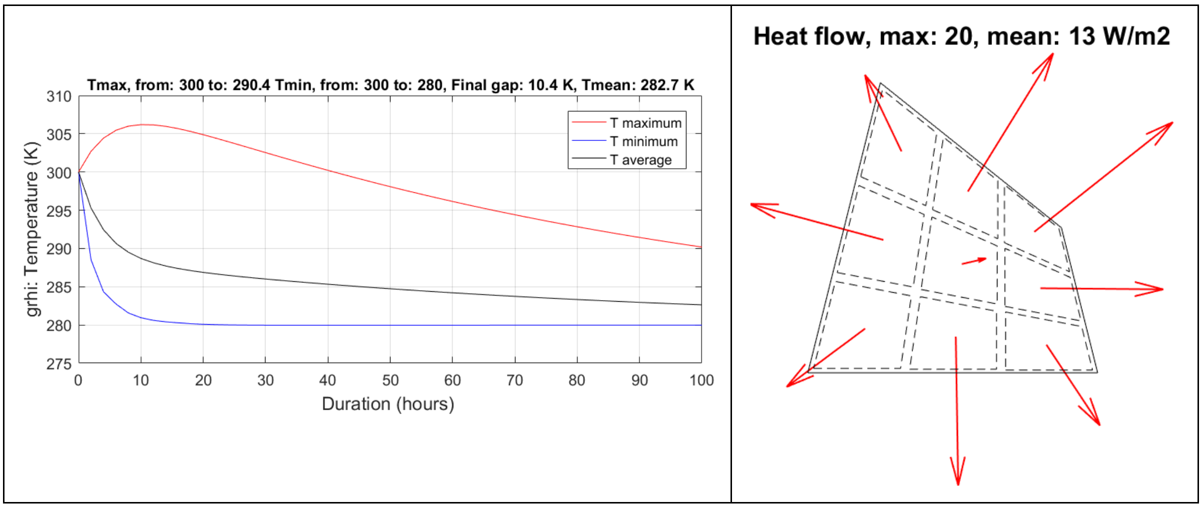


Figure 57: Evolution of temperatures and heat flows for a quadrilateral coarse mesh

```
pp_visopa_transient
Control param.  pa6 : 2
Convection coeff. h : 25 W/(m2K)
Initial temperature : 300 K
Thickness           : 0.1 m
conde.m uniform k   : 1 W/(mK)
Air temperatures    : 280  280  280  280 K
Mesh size           : 20 x 20
N. virtual nodes t3 : 4
Number of fixations : 0
Total duration      : 100 h
Number of iterations: 50
N. of convective el.: 80
Tot. cap.sum(sum(C)): 6250000 J/K
Stat. Ntca no nf t3 : 445  441   0   4
Elem. heat flow max : 27.3, average: 14.5 W/m2
Cpu                 : 6.28 sec.
```

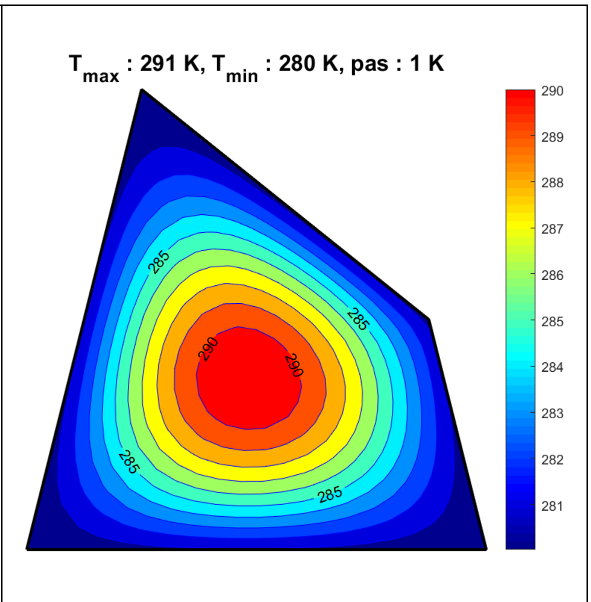


Figure 58: Isocurves for a quadrilateral fine mesh

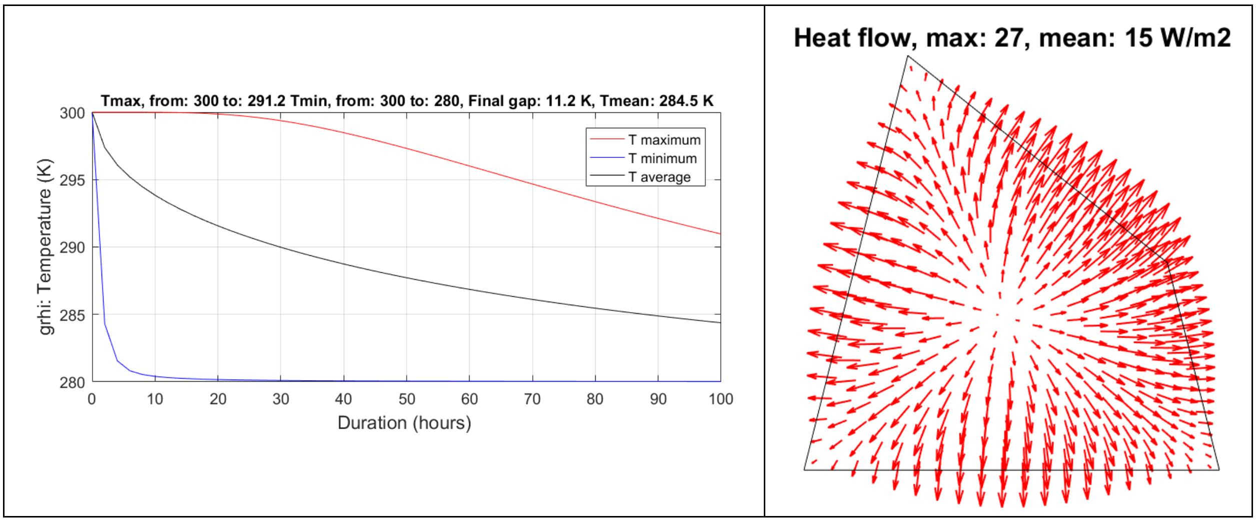
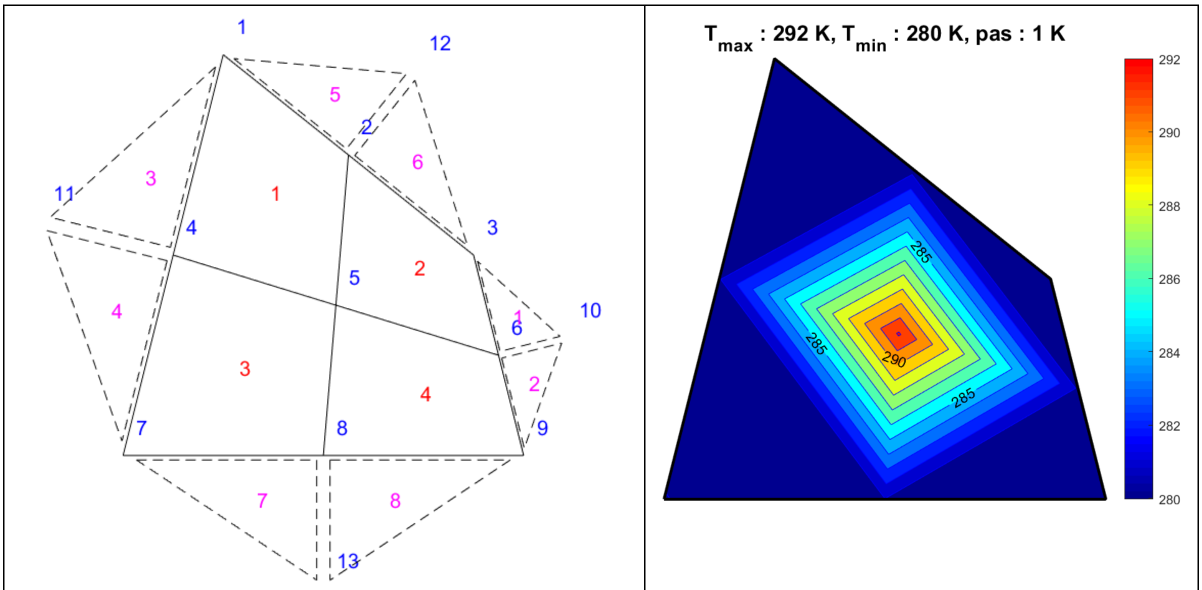


Figure 59: Evolution of temperatures (left) & heat flows (right) for a quadrilateral fine mesh




```
pp_visopa_transient
***********************
Control param.  pa6 : 2
Convection coeff. h : 25 W/(m2K)
Specific heat       : 1000 J.m-3.K-1
Specific mass       : 2500 kg.m-3
Base temperature    : 250 K
Initial temperature : 300 K
Thickness           : 0.1 m
conde.m uniform k   : 1 W/(mK)
Air temperatures    : 280  280  280  280 K
Mesh size           : 2 x 2
N. virtual nodes t3 : 4
Number of fixations : 0
Diffusivity         : 4e-07 m2/s
Total duration      : 100 h
Number of iterations: 50
Virtual conv. nodes : 10  11  12  13
Tot. cap.sum(sum(C)): 6250000 J/K
Stat. Ntca no nf t3 : 13   9   0   4
Elem. heat flow max : 16.5, average: 11.4 W/m2
Cpu                 : 2.46 sec.
```

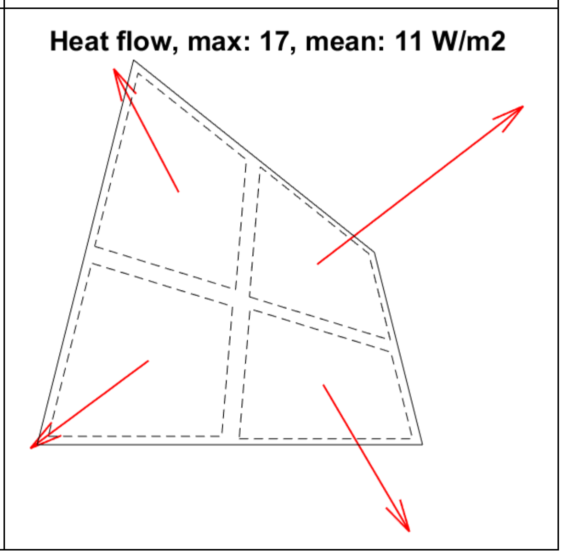


Figure 60: Results of a transient analysis for a very coarse mesh

Procedure name	Aim	Matlab© functions	Ref.
pp_conduction.m Table 1	Heat transfers in conduction with only prescribed temperatures.	*conde.m* *loca.m* *grisb.m* *br56.m* *Tg.m* *Hf.m* *cageco.m*	*Table 3* *Table 4* *Table 5* *Table 6* *Table 7* *Table 8* *Table 9*
pp_ convection.m Table 13	Convective heat transfers	*conde.m* *CoKr.m* *loca.m* *grisb.m* *br56.m* *Tg.m* *Hf.m*	*Table 3* *Table 14* *Table 4* *Table 5* *Table 6* *Table 7* *Table 8*
pp_radiation.m Table 16	Heat transfers including radiation	*conde.m* *loca.m* *ItKr.m* *locc.m* *grisb.m* *br56.m* *radit.m*	*Table 3* *Table 4* *Table 17* *Table 33* *Table 5* *Table 6* *Table 18*
pp_ transient.m Table 20	Transient heat transfers	*conde.m* *loca.m* *CoKr34.m* *grisb.m* *br56.m* *grhi.m*	*Table 3* *Table 4* *Table 22* *Table 5* *Table 6* *Table 21*

Table 44: Matlab© procedures and their linked functions for square elements

Procedure name	Aim	Matlab© functions	Ref.
pp_visopa_conduction.m *Table 24*		*conde.m* *Nxyz.m* *loca.m* *Kelu.m* *grisc.m* *cageco.m* *mesh.m* *mgra.m* *Hflo.m*	*Table 3* *Table 40* *Table 4* *Table 38* *Table 41* *Table 9* *Table 30* *Table 37* *Table 36*
pp_visopa_convection.m *Table 25*	Heat transfers including convection and using isoparametric elements	*conde.m* *Nxyz.m* *Coco.m* *loca.m* *locc.m* *locc2.m* *locc4.m* *Kelu.m*	*Table 3* *Table 40* *Table 14* *Table 4* *Table 33* *Table 34* *Table 35* *Table 38*

		Hflo.m *mgra.m* *grisc.m* *br56.m* *povi.m* *mesh_nu.m* *mesh_co.m*	*Table 36* *Table 37* *Table 41* *Table 6* *Table 29* *Table 31* *Table 32*
pp_visopa_radiation.m	Heat transfers including radiation and using isoparametric elements	*conde.m* *Nxyz.m* *loca.m* *Kelu.m* *grisc.m* *br56.m* *radit.m*	*Table 3* *Table 40* *Table 4* *Table 38* *Table 41* *Table 6* *Table 18*
pp_ visopa_transient.m *Table 27*	Transient heat transfers	*conde.m* *Nxyz.m* *Coco.m* *loca.m* *Kelu.m* *Celu.m* *grisc.m* *br56.m* *grhi.m* *mesh.m* *Hflo.m*	*Table 3* *Table 40* *Table 14* *Table 4* *Table 38* *Table 42* *Table 41* *Table 6* *Table 21* *Table 30* *Table 36*

Table 45: Matlab© procedures and their linked functions for isoparametric elements

Exercices proposed in 2019

Exercise n°1: Conductivity coefficients

Using the Matlab© procedure and the functions presented in the tutorial, it is proposed to examine the effects of a modification of the conductivity coefficients. Let us try, for instance, to introduce an heterogeneity of the conductivity. On the left half side of the domain the conductivity is lower than on the right half side. This modification has to be performed by modifying the function *conde.m*. We assume that the temperature on the upper side = 320 *K* and that the base temperature = 270 *K*. For a coarse mesh, we also want to draw the heat flows. The elements are numbered from left to right and from top to bottom.

Exercise n°2a: One free convective virtual node

In the situation of 2 convective and 2 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 $\beta = w\ h / k$ and to compare with the finite element model result (w is the width of the domain, h and k respectively the convection and conduction coefficient). Let check the consequence of introducing an horizontal thermal bridge (with much higher conductivity in the central band of 4 elements pertaining to a 10 x 20 mesh).

Exercise n°2b: Modifying the boundary conditions of a presented example

It is proposed to modify the boundary conditions of the application presented in *Figure 25* 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).

Exercise n°3 : Converting convection to radiative boundary conditions

We propose to use the same boundary conditions as in the application presented in *Figure 25*, but the convection conditions are transformed in radiation ones. However, we use 3 prescribed temperatures for the 3 virtual nodes. For the third one, we use the solution of *Figure 25*. What about the convergence of this problem ?

Exercise n°4: Cooling of a domain initially at uniform temperature

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.

Exercise n°5a: Thermal bridge in a trapezoidal domain

Modify the shape of the domain analyzed in exercise n°1, *Figure 19*, to transform the rectangular domain into a trapezoidal one (chapter on convective heat transfert). Again, check the consequence of introducing an horizontal thermal bridge.

Exercise n°5b: Non rectangular domain

We propose to compare rectangular and non rectangular shapes for any one of the previous problems.

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., “Surfaces 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