Benoit Beckers

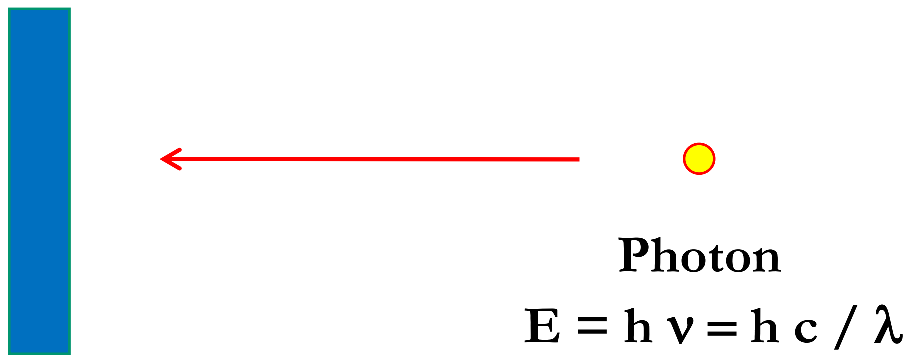
Architecture et Physique Urbaine - ISA BTP
Université de Pau et des Pays de l'Adour
Allée du Parc Montaury, 64600 Anglet (France)
benoit.beckers@univ-pau.fr, www.heliodon.net

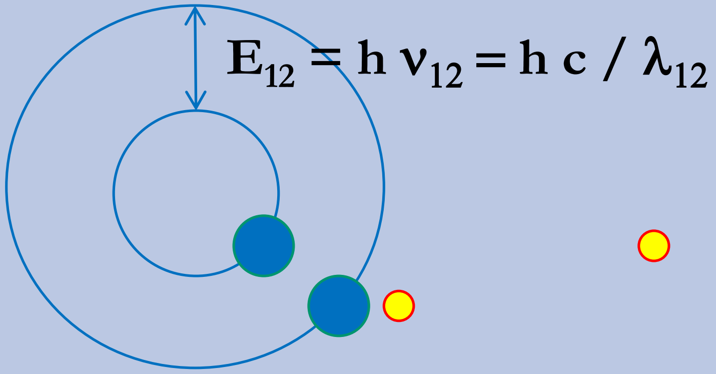
Finite element Simulation of radiative heat transfers

Black body and gray body

- Absorptivity α
- Reflexivity ρ
- Transmissivity τ

$$\alpha + \rho + \tau = 1$$



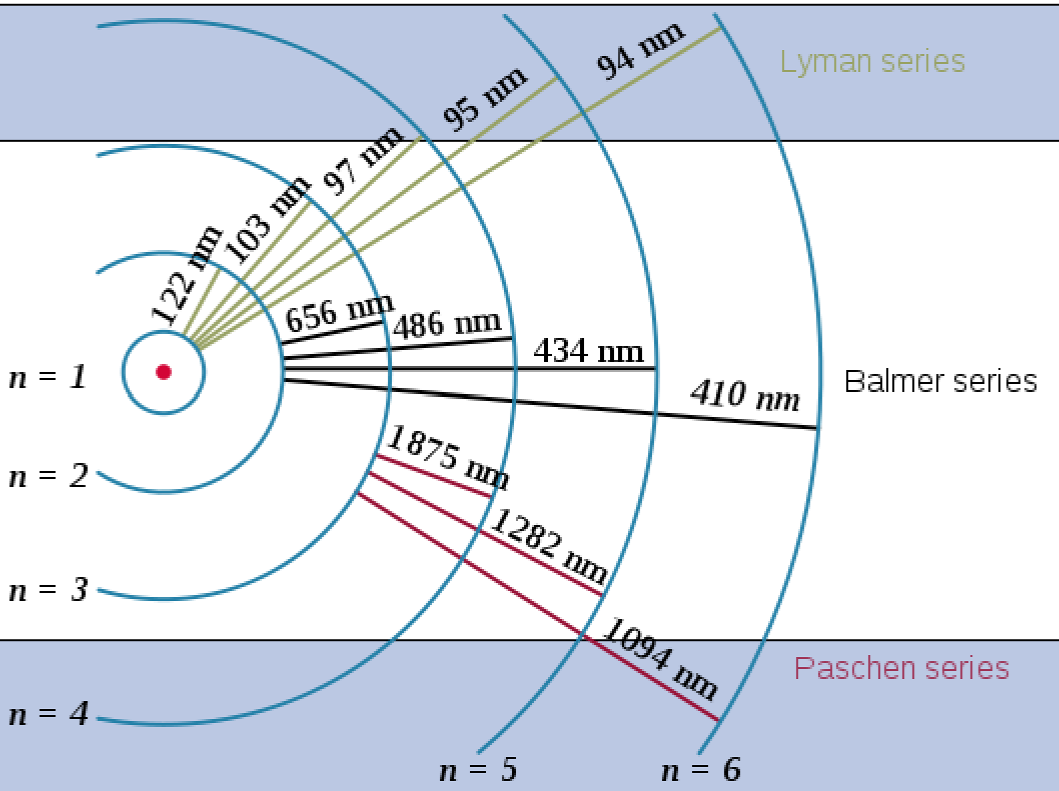


Emissivity ε

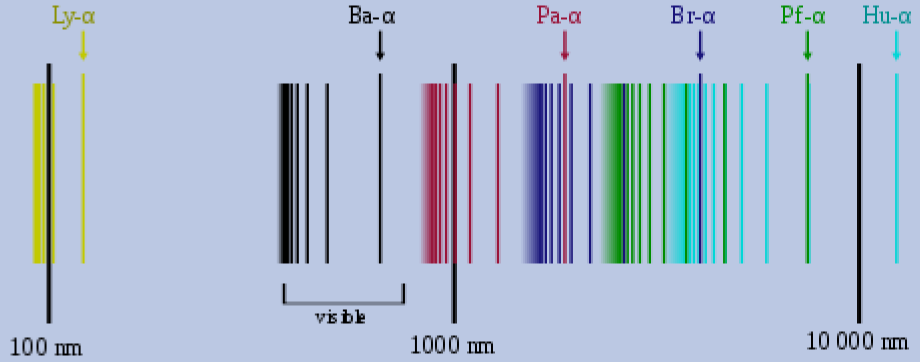
For any λ, we have:

$$\alpha(\lambda) = \varepsilon(\lambda)$$

Kirchhoff's Law



$$\alpha + \rho + \tau = 1$$



For any λ, we have:

$$\alpha(\lambda) = \varepsilon(\lambda)$$

Kirchhoff's Law

http://en.wikipedia.org/wiki/Hydrogen_spectral_series

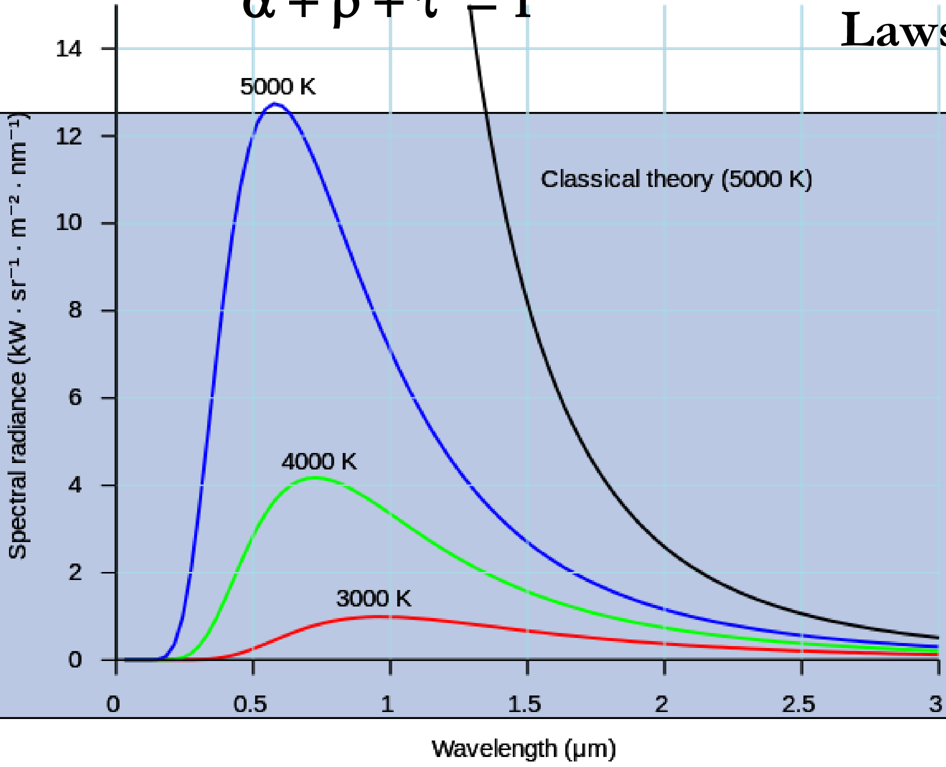
$$L_\lambda = \frac{2hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{k\lambda T}}-1}$$

$$\lambda_{\max} T = 2.898 \times 10^{-3}\, mK$$

$$Q\left(Wm^{-2}\right) = \sigma T_r^4$$

$$\alpha + \rho + \tau = 1$$

Laws of Planck, Wien and Stefan-Boltzmann



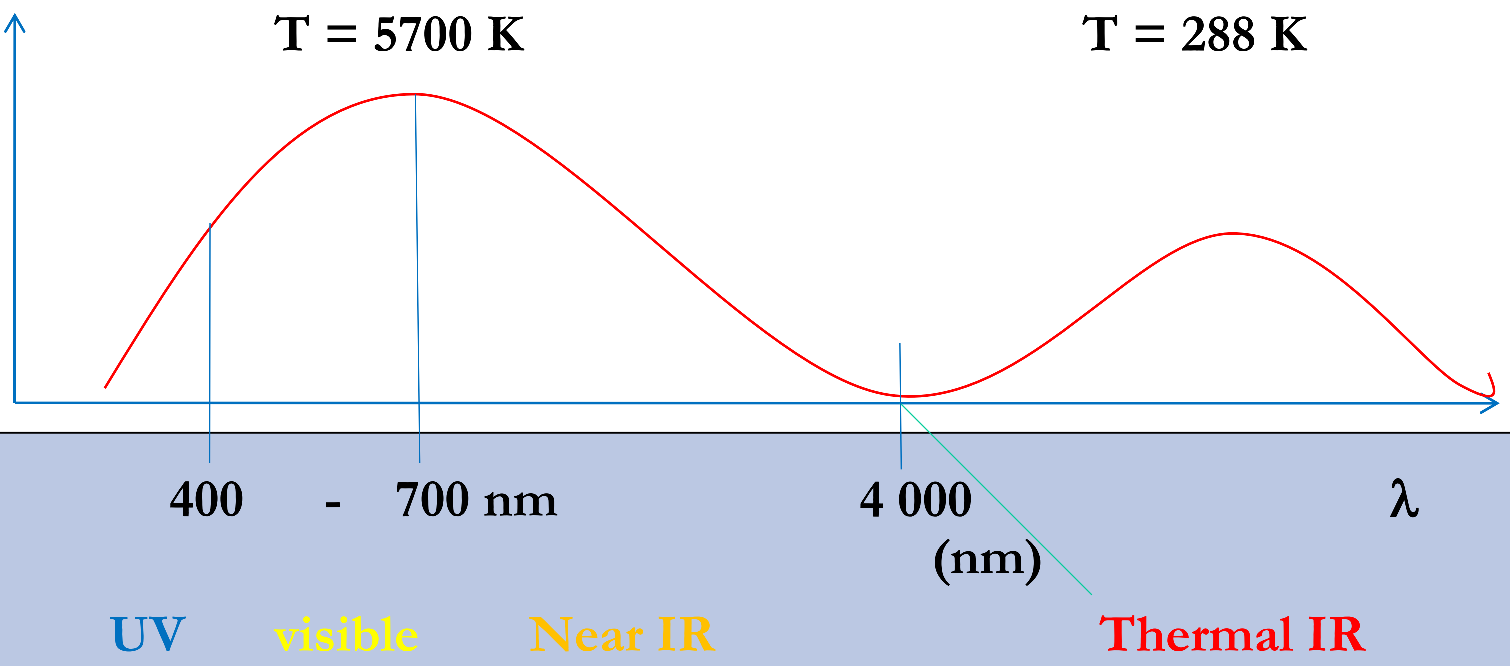
For any λ, we have:

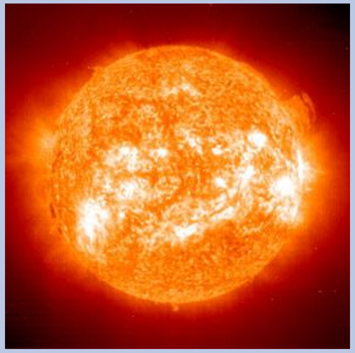
$$\alpha(\lambda) = \varepsilon(\lambda)$$

Kirchhoff's Law

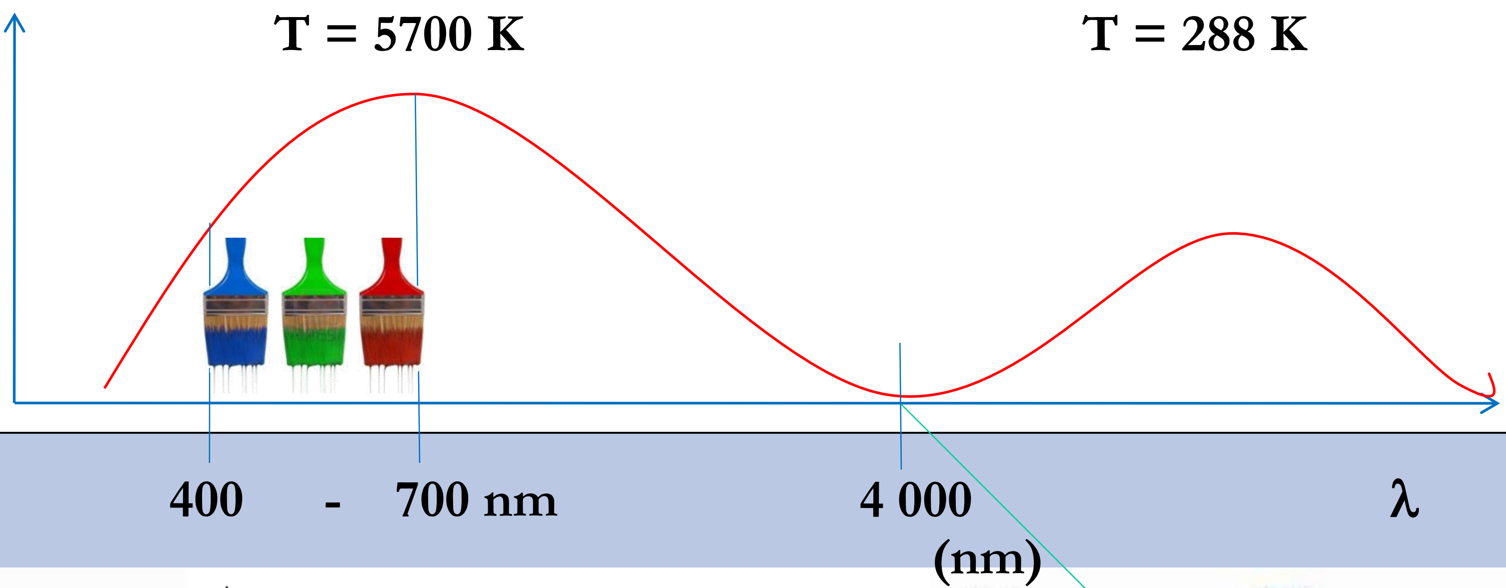
http://en.wikipedia.org/wiki/Black_body

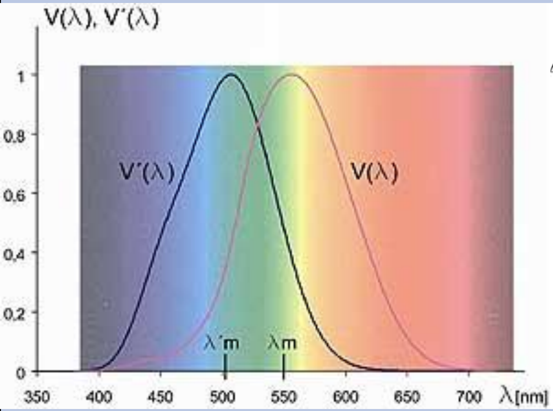
Short waves and long waves

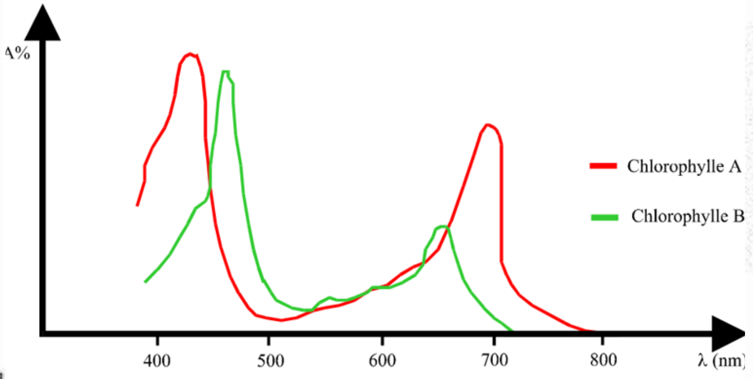


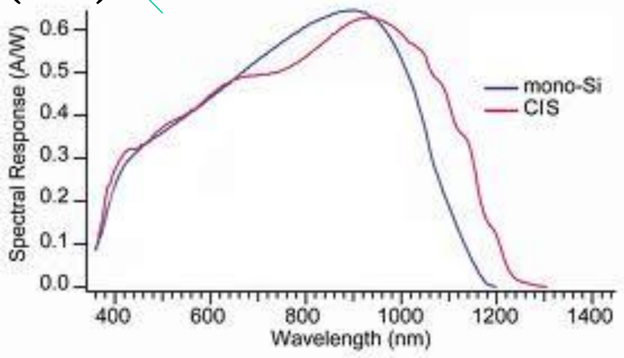


Spectral sensitivity



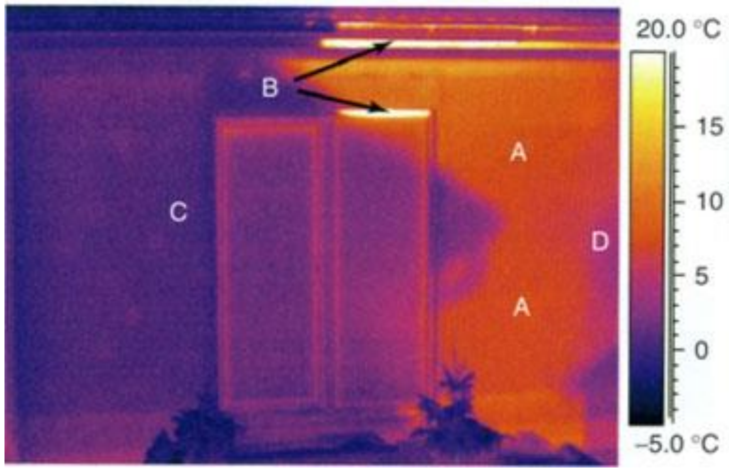


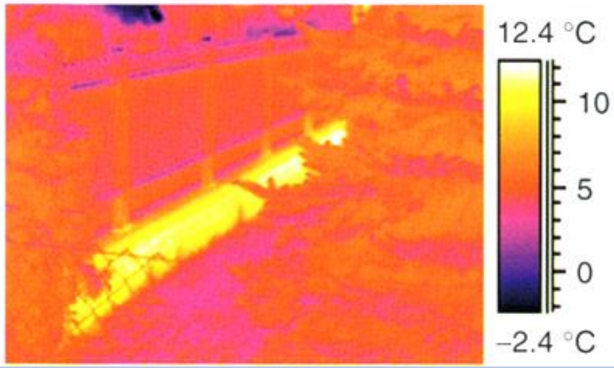




ATRAPA EL RUIDO Y EL CALOR Y LOS DEJA FUERA DE TU CASA.
Los cristales Guardian Sun, con tecnología Lamiglass®, aislan acústica y térmicamente tu casa sin renunciar a la luz.
ruido
calor
luz
GUARDIAN SUN
CRISTAL INTELIGENTE
www.guardiansun.es

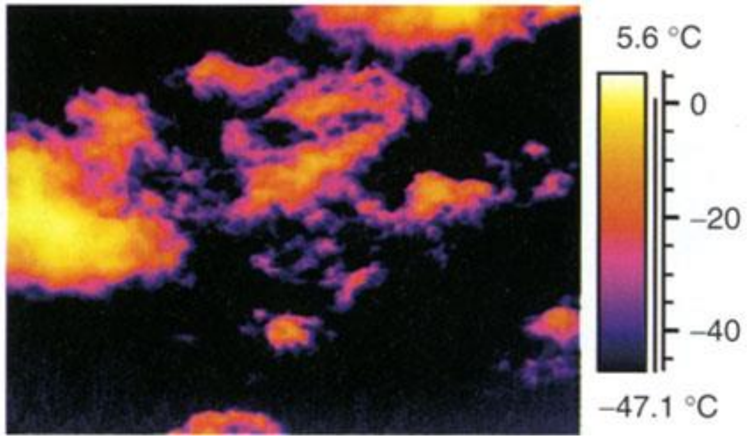
Thermography

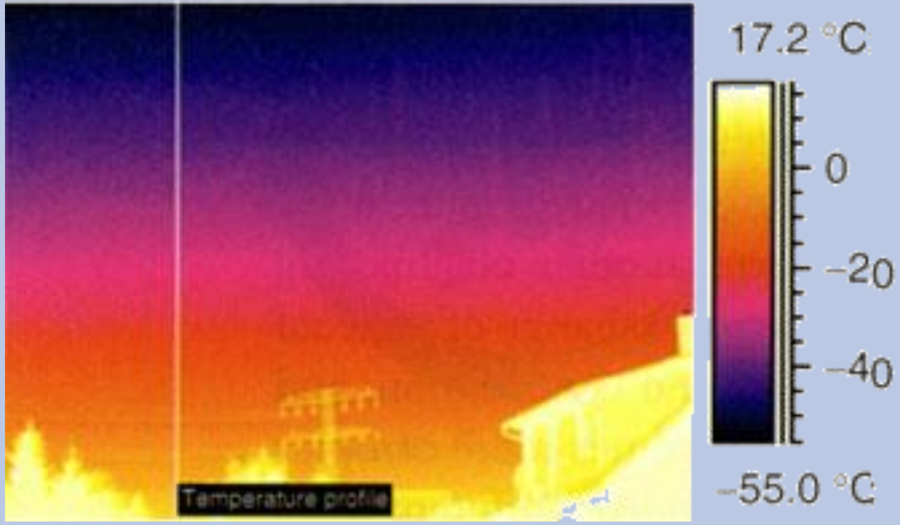


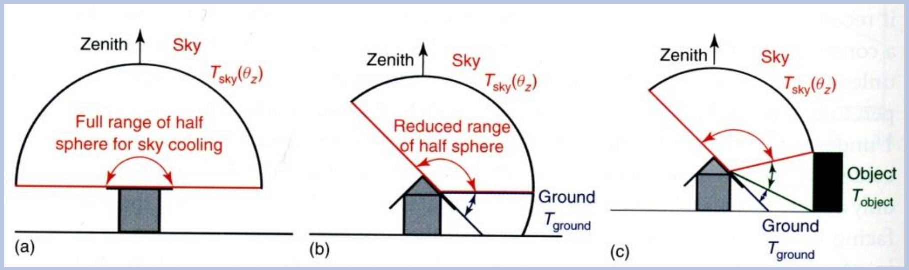


M. Vollmer & K.-P. Möllmann "Infrared Thermal Imaging: Fundamentals, Research and Applications", Wiley-VCH, 612 pages, 2010.

Thermography







M. Vollmer & K.-P. Möllmann "Infrared Thermal Imaging: Fundamentals, Research and Applications", Wiley-VCH, 612 pages, 2010.

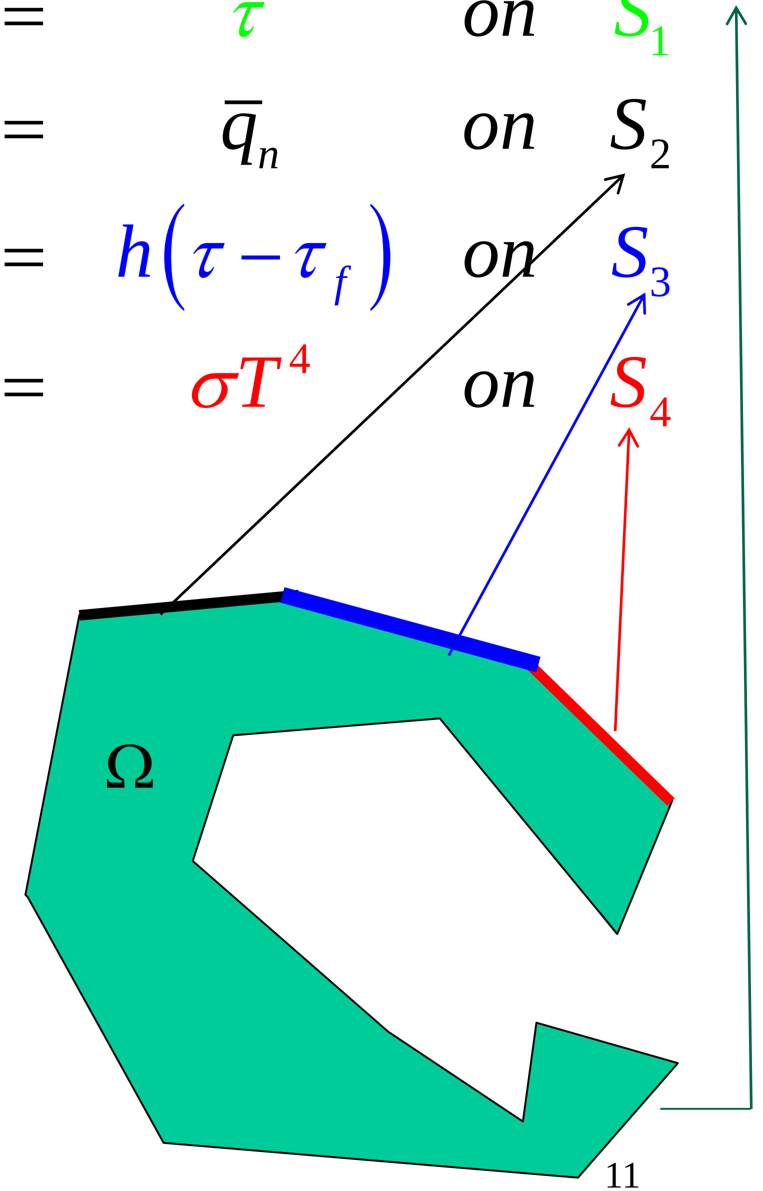
1. Radiation to an identified zone with known temperature

2. Radiation to an other part of the domain

We want to compute the temperature distribution in a domain Ω, subject to various boundary conditions

$$div\left(k\ grad\tau\right)=0\ \ in\ \ \Omega$$

$$\begin{array}{lclcl} \tau & = & \bar{\tau} & on & S_1 \\ \vec{n}\cdot\vec{q} & = & \bar{q}_n & on & S_2 \\ \vec{n}\cdot\vec{q} & = & h\left(\tau-\tau_f\right) & on & S_3 \\ \vec{n}\cdot\vec{q} & = & \sigma T^4 & on & S_4 \end{array}$$



In these equations, τ (K) is the unknown temperature, k ($Wm^{-1}K^{-1}$), the thermal conductivity, h ($Wm^{-2}K^{-1}$), the convection coefficient and σ ($Wm^{-2}K^{-4}$) the Stefan-Boltzmann constant.

The boundary of the domain is divided into 4 parts: $\boldsymbol{S_1}$ where the temperature is imposed, $\boldsymbol{S_2}$ where the normal heat flow is imposed, $\boldsymbol{S_3}$, where the convective flow is imposed and $\boldsymbol{S_4}$, the radiative part of the boundary. A bar indicates that the concerned quantity is imposed.

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

$$Q = \sigma\left(T_k^4 - T_r^4\right)$$

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

$$Q = \sigma\left(T_k^2 + T_r^2\right)\left(T_k + T_r\right)\left(T_k - T_r\right)$$

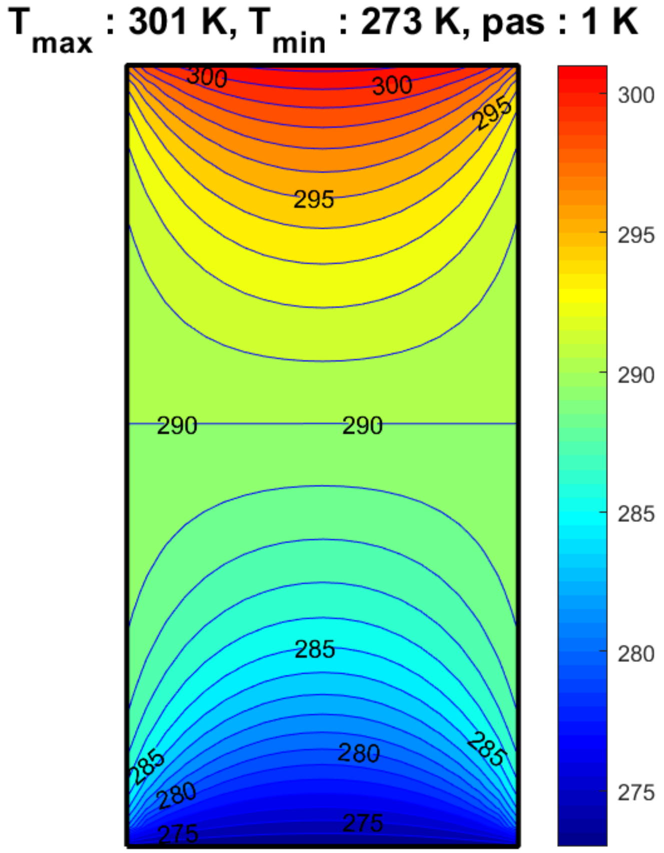
For a radiation element, the "conductivity" matrix is calculated exactly as for convection, but the convection coefficient h is replaced by the coefficient:

$$\sigma\left(T_k^2 + T_r^2\right)\left(T_k + T_r\right)$$

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

In this expression, the temperatures of the radiative boundary of the domain must be evaluated in the same loop as the computed ones. Thus, the above coefficient has to be obtained by successive approximations during the iterations

We start with an example where the boundary conditions consist of radiation on 3 faces and fixed temperatures on the fourth one. The isotherms are shown for the first (left) and the second iteration (right)

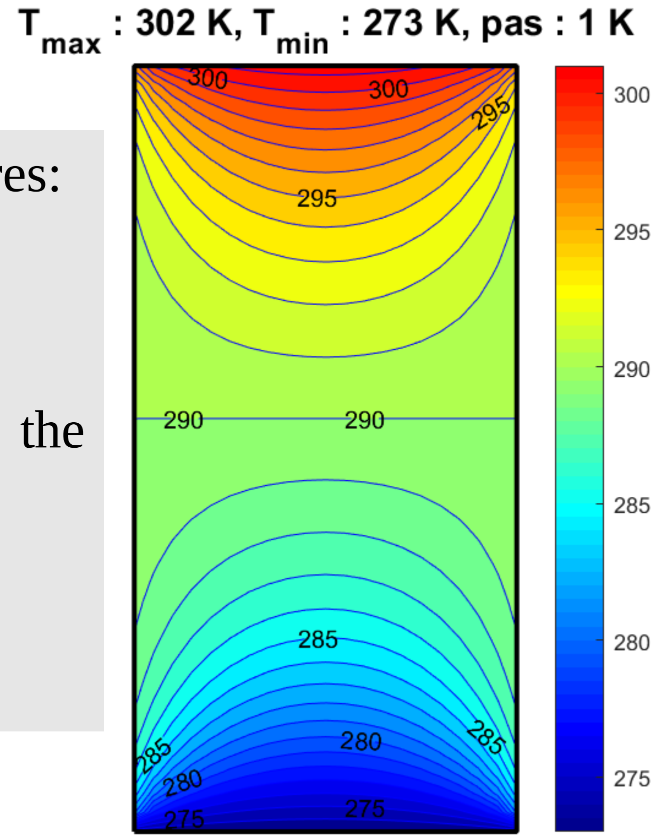


Virtual nodes temperatures:

290, 290, 303 *K*

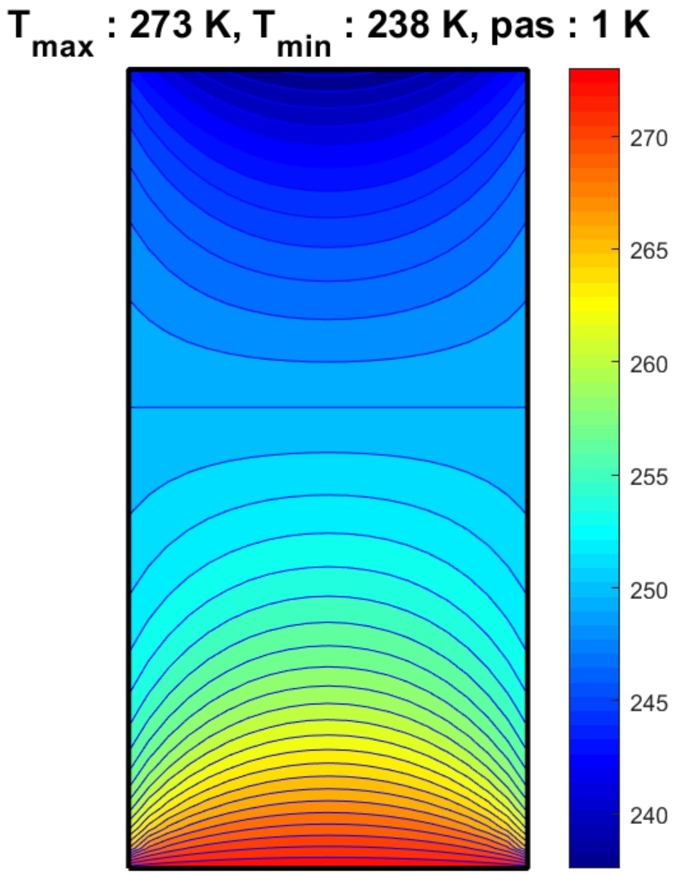
Total heat flow through the 3 virtual nodes

Iteration 1 : 124.5 *W*
Iteration 2 : 126.2 *W*



Iteration 1 : pseudo *h* coefficients: 1.383 1.383 1.4788 $Wm^{-2}K^{-1}$
Iteration 2 : pseudo *h* coefficients: 1.2973 1.2973 1.5373 $Wm^{-2}K^{-1}$
There is not a big difference between the 2 iterations

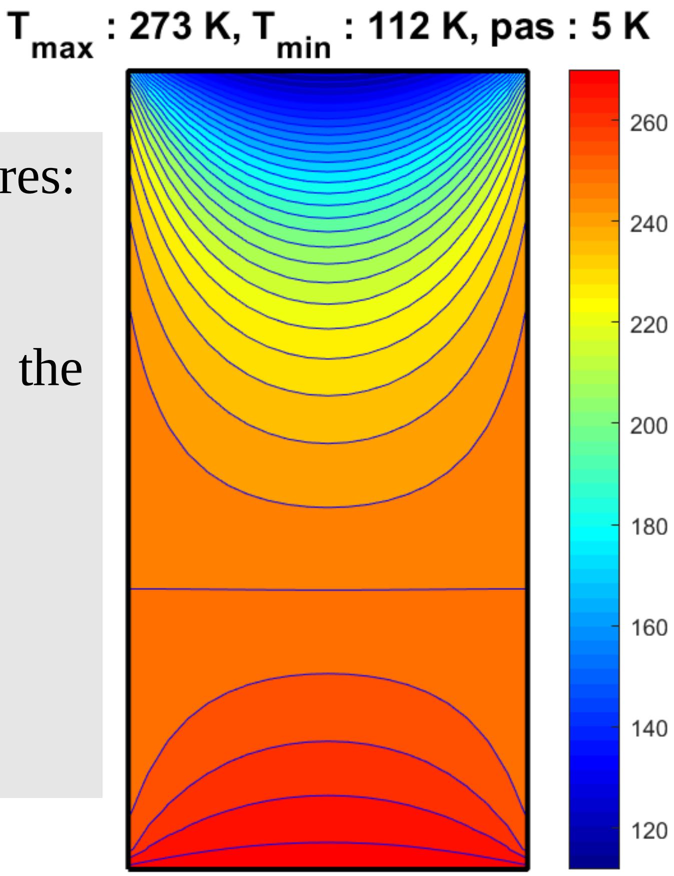
With other boundary conditions, the isotherms are shown for the first (left) and the second iteration (right)



Virtual nodes temperatures:
250, 250, 50 *K*

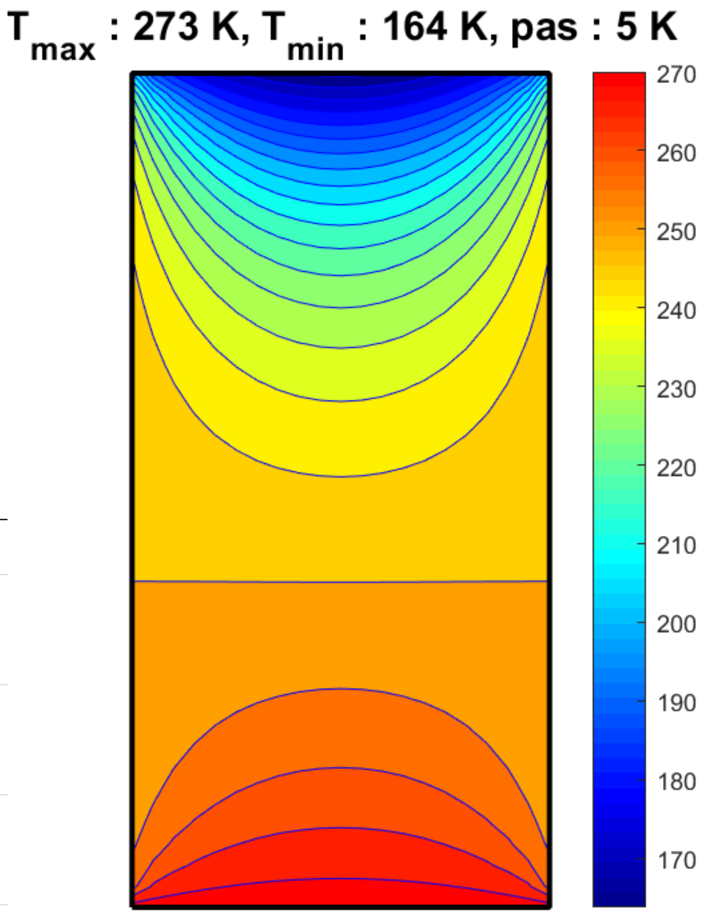
Total heat flow through the 3 virtual nodes

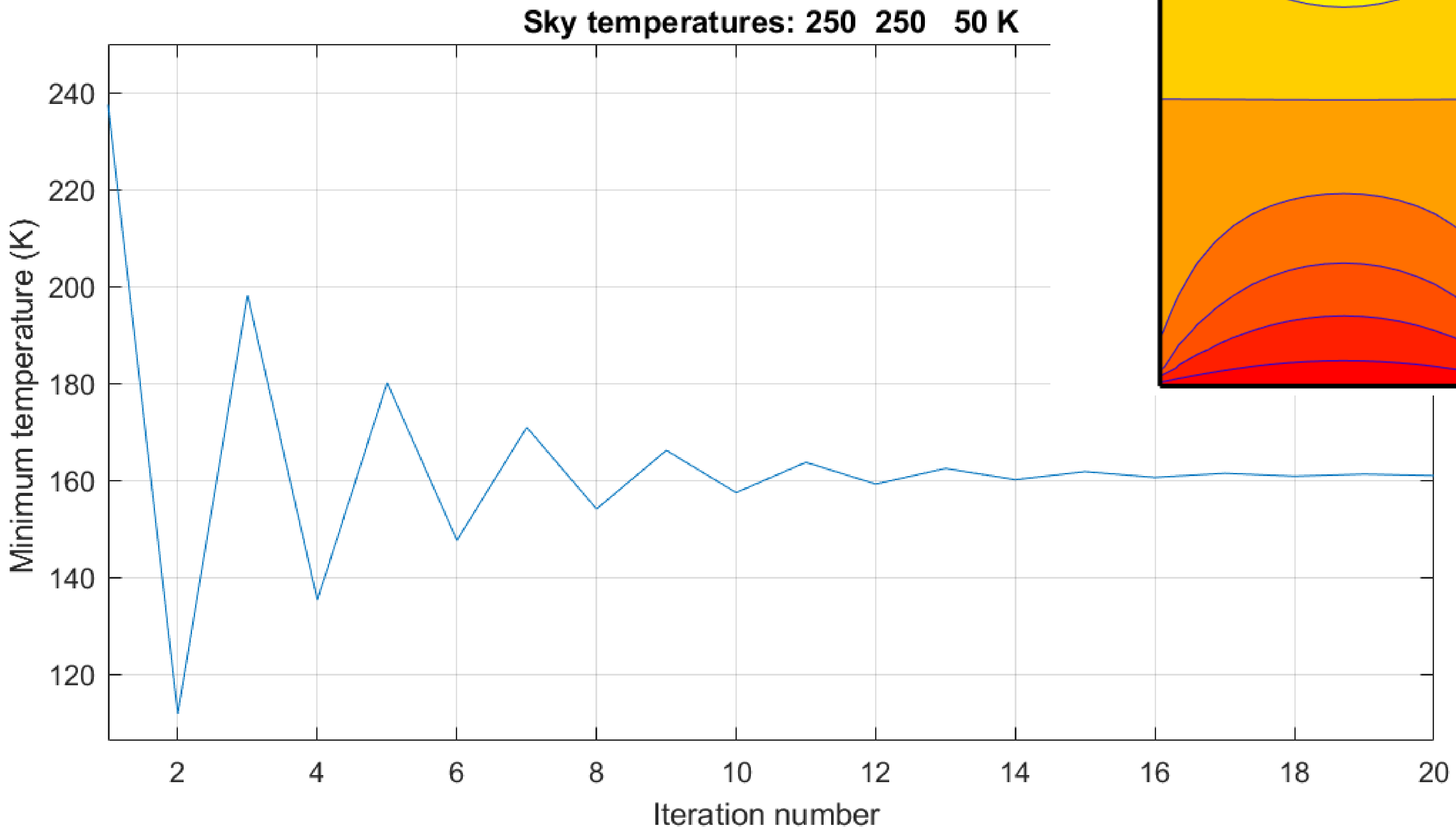
Iteration 1 : -126 *W*
Iteration 2 : -198 *W*



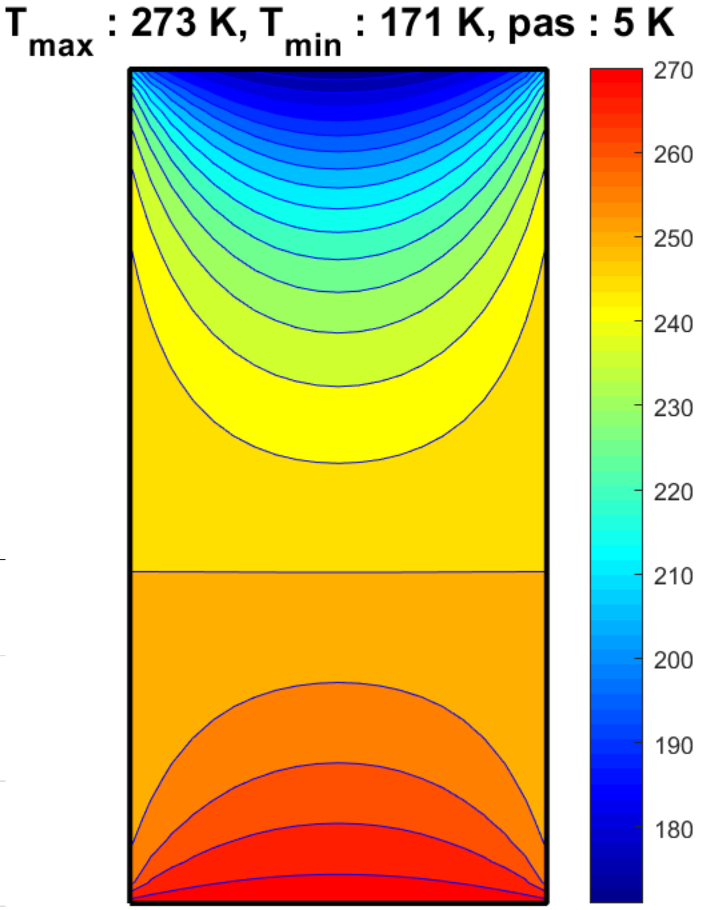
Iteration 1 : pseudo *h* coefficients: 0.552 0.552 0.014 $Wm^{-2}K^{-1}$
Iteration 2 : pseudo *h* coefficients: 1.790 1.790 0.491 $Wm^{-2}K^{-1}$
There is a major difference between the 2 iterations

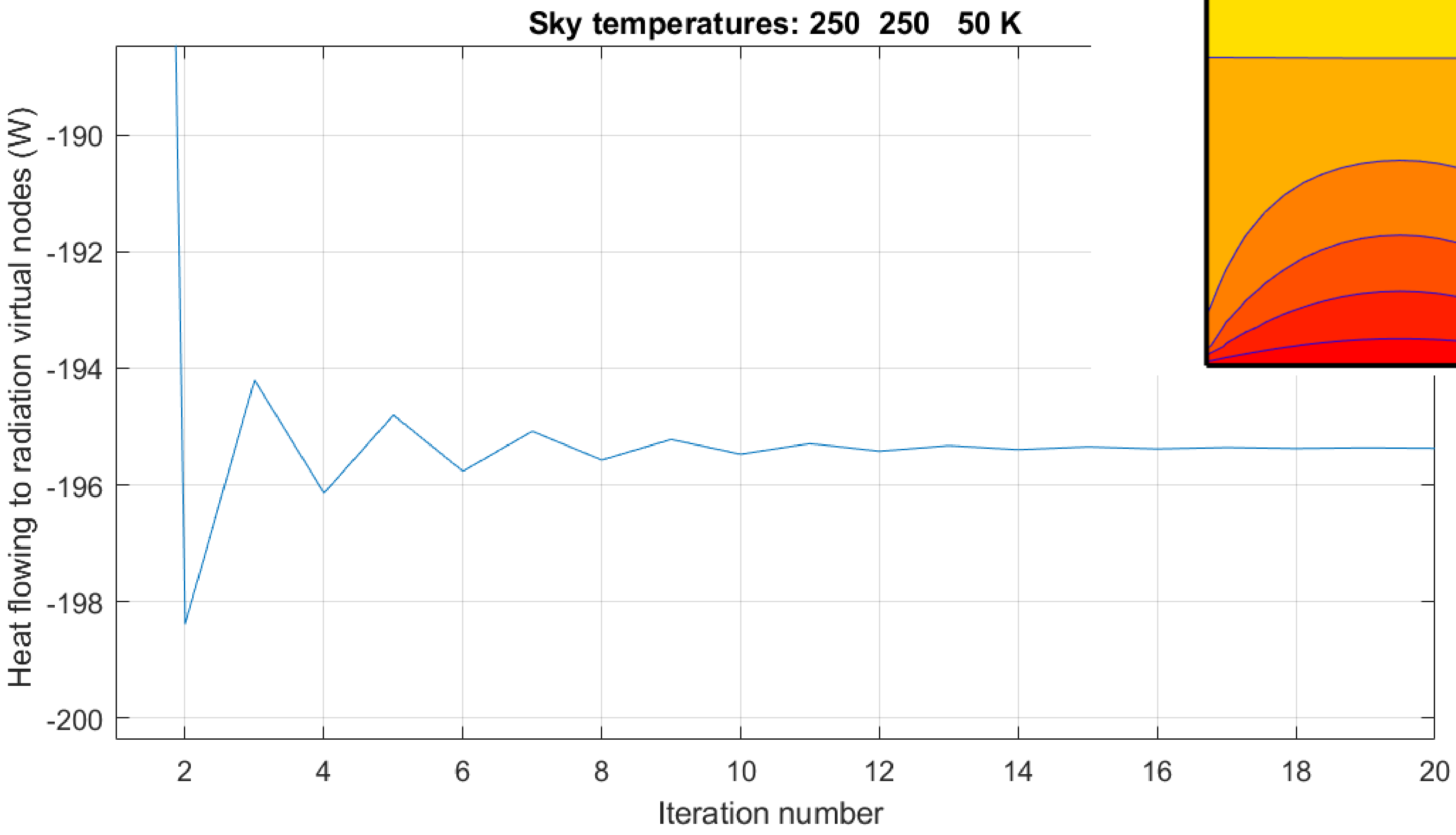
The convergence of the solution is seen trough the value of a global information, i.e., the minimum temperature in the domain. The right drawing shows the iteration 11.





Concerning the value of the total heat flow between the virtual radiation nodes and the base of the domain, the convergence is acceptable. It is achieved after less than ten iterations. The right drawing displays the iteration seven.





The 20 steps of the computation

```
conde.m uniform k     : 2 W/(m K)
Stefan-Boltzmann      : 5.67e-08 W m-2K-4
Mesh dimension        : 20 x 40
Base temperature      : 273 K
Virtual nodes temp.   : 250 K  250 K 50 K
```

Evolution of the pseudo *h* coefficient

```
0.55286     0.55286    0.014176
1.7901      1.7901     0.4908
1.7287      1.7287     0.10582
1.7591      1.7591     0.31042
1.7377      1.7377     0.15106
1.7522      1.7522     0.25327
1.7417      1.7417     0.17607
1.7491      1.7491     0.22881
1.7437      1.7437     0.18981
1.7475      1.7475     0.21717
1.7448      1.7448     0.19719
1.7468      1.7468     0.21138
1.7453      1.7453     0.2011
1.7464      1.7464     0.20845
1.7456      1.7456     0.20314
1.7461      1.7461     0.20694
1.7458      1.7458     0.2042
1.746       1.746      0.20617
1.7458      1.7458     0.20475
1.746       1.746      0.20577
```

T min (*K*) et Heat flow (*W*)

```
    [tmi hvn]

 237.5815 -125.9231
 111.8027 -198.3821
 198.2800 -194.1999
 135.3207 -196.1372
 180.1809 -194.7995
 147.6230 -195.7626
 170.9569 -195.0759
 154.0987 -195.5709
 166.1992 -195.2162
 157.4786 -195.4716
 163.7427 -195.2883
 159.2337 -195.4202
 162.4740 -195.3254
 160.1429 -195.3936
 161.8185 -195.3446
 160.6133 -195.3799
 161.4797 -195.3545
 160.8567 -195.3728
 161.3046 -195.3597
 160.9825 -195.3691
```

Procedure Matlab© *pp_ radiation.m* for radiation

```
tb   = 273;pge=[20;40;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);
th   = pge(3);he=pge(2);tst  = tic;% Beginning analysis, timer init.
nx   = pge(4);ny = nx*2;nel = nx*ny;no = (nx+1)*(ny+1);             % Mesh
disp('********************')% First iteration =============================
co   = conde(nx,ny);
disp(['Stefan-Boltzmann    : ',num2str(SB,'%0.3g'),' Wm-2K-4'])
disp(['Mesh dimension      : ',num2str(nx),' x ',num2str(ny)])
disp(['Base temperature    : ',num2str(tb,3),' K'])
disp(['Virtual nodes temp. : ',num2str(ts'),' K'])
Kel  = th/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];   % elem. K
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,th,he,SB);% he is the height of the domain in m
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
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;
grisb(nx,ny,[tca ;ones(nx+1,1)*tb],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)));
if nit > 1 % Additional iterations ========================================
[hvn,tmi]=radit(tca,nx,ny,tb,ts,th,he,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.'])
```

Procedure used to obtain the previous results.

Procedure for the solution of a thermal conduction problem with radiative boundary conditions

Lines	1 – 5	: Data input
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	: Conductivity-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

Lines 1 – 5 : Data input

```
tb    = 270;tt = tb+50; pge  = [1;2;1;50];
nx    = pge(4);ny = nx*2;nel = nx*ny;no = (nx+1)*(ny+1);nf = nx+1;  % Mesh
```

The variables *tb* & *tt* give the temperatures on the top and the bottom of the domain. The vector *pge* contains the dimensions of the domain: width, height, thickness and the number *nx* of elements in the horizontal direction. *ny* is the number of elements in the vertical direction: twice *nx*. The following items concern the computation of *nel*: number of elements, *no*: number of nodes, *nf*: number of nodes on a horizontal line of the mesh.

Line 6 : conductivity coefficients in the elements (function *conde.m*)

Matlab© function *conde.m* to handle non uniform k

```
function [co] = conde(nx,ny)        % Treatment of non uniform onductiv.
k   = 1;     % W/(m K)
nel = nx*ny; % Number of element computed from mesh definition
fa  = 1 ;    % 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(['Thermal conductiv.  : ',num2str(k,'%0.3g'),' W/(m K)'])
disp(['Main & bridge k.    : ',num2str([co(1) co(nx*nx+1)]),' W/(m K)'])
end
```

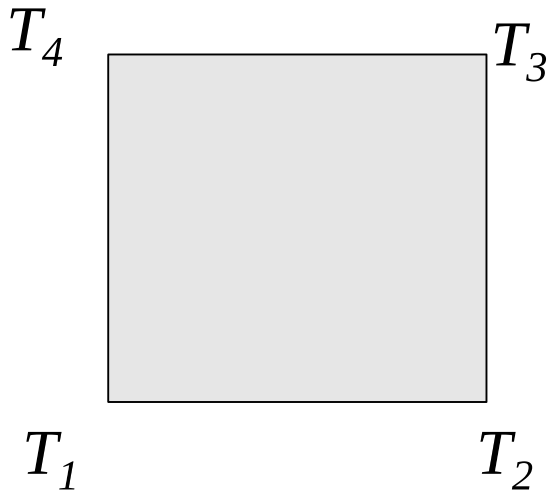
For a mesh of *nx* x *ny* elements (arguments of the function), one defines the vector *co* (output of the function) which contains the values of the coefficients of conductivity of all the elements.

In a non-homogeneous medium, the conductivities may vary from one element to another, it is necessary to add to the previous procedure the two lines defining the elements operating with the second conductivity and modify the assembly of the elements. The conductivity coefficient acts as coefficients in the assembly of the global matrix (*line 9*). The conductivities of the elements are stored in the vector *co* of dimension *nel*.

Line 5 : conductivity matrix of a square element

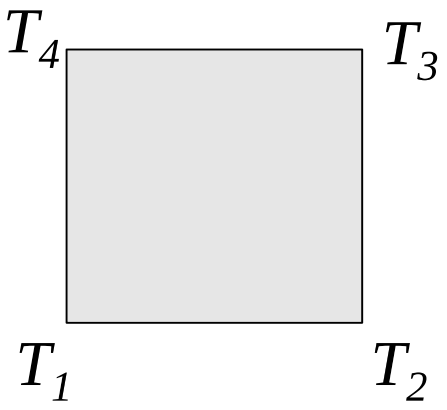
```
Kel    = pge(3)/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4]; % elem. K
```

The variable `pge(3)` corresponds to the thickness of the element. The matrix coefficients are written in compact form.

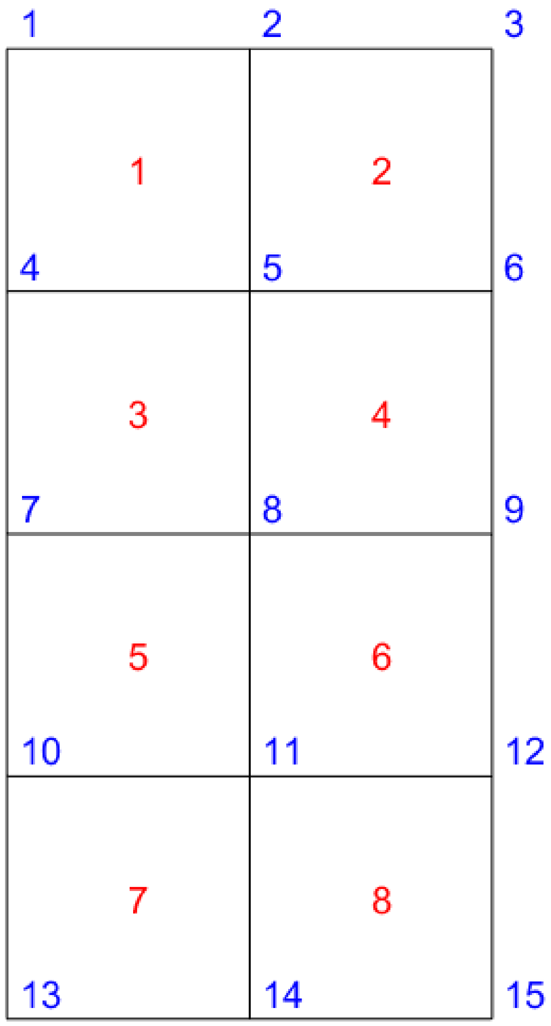


$$K = \frac{ke}{6}\begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix}$$

Line 7 : compute the localization matrix (function *loca.m*)



The element node sequence is always the same and must be satisfied during assembling



For each element, the four nodes must be 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..

Element 1 :	4	5	2	1
Element 2 :	5	6	3	2
Element 3 :	7	8	5	4
Element 4 :	8	9	6	5
Element 5 :	10	11	8	7
Element 6 :	11	12	9	8
Element 7 :	13	14	11	10
Element 8 :	14	15	12	11

Here, to improve the efficiency of the procedure, we use direct localization instead of the matrix formalism seen before in the theoretical presentation.

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
```

The above function creates the localization matrix for *nx* x *ny* meshes defined in a vertical rectangle (1 m x 2 m).

In the previous slide we see a 2 x 4 mesh, and the corresponding localization matrix

To generate this matrix in Matlab,
it is possible to enter directly a command like: *lK = loca(2, 4)*.

Line 10: compute the conductivity matrix for radiation (function *ItKr.m*)

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

The element is a vertical or a horizontal one. Its length is *L*, its thickness is *e* and the Stefan-Boltzmann coefficient is σ ($Wm^{-2}K^{-4}$). The sequence of nodes start with the two real ones and finish with the virtual one.

This function computes the radiation convection matrices of a mesh *nx* x *ny* (arguments 3 and 4 of the function) for a domain of dimensions given by arguments 3 and 4 and the Stefan-Boltzmann constant defined in line 3 of the function. The temperatures of the domain appear in *tca*, the first argument, while the temperatures of the virtual nodes are defined in the vector *ts*

Matlab© function *ItKr.m* to compute the radiation matrix

```
function[Kr]=rf1_radiation(tca,ts,nx,ny,ep,he,SB)
no     = (nx+1)*(ny+1);Ntca = no+3;
Kelc   = [2 1 -3;1 2 -3;-3 -3 6]/6;                 % Element convection matrix
Kr     = zeros(Ntca,Ntca);ntv=no+1:Ntca;% ntv: numbers of the virtual nodes
lc     = rf3_convection(nx,ny,ntv);      % Local. of the convection matrices
crdm   = 0;iel=0;
for i                = nx+1 : nx+1 : no-nx-1            % Right face radiation
    T1               = (tca(i)+tca(i+nx+1))/2;      % Edge mean temperature
    crd              = SB*(T1^2+ts(1)^2)*(T1+ts(1))*ep*he/ny;
    crdm             = crdm+crd;iel=iel+1;
    for ii=1:3;for j = 1:3;Kr(lc(iel,ii),lc(iel,j)) = ...
                Kr(lc(iel,ii),lc(iel,j))+Kelc(ii,j)*crd;end;end
end
crd=crdm/iel;crgm    = 0;iel=0;          % Mean right convection coefficient
for i                = 1 :nx+1 :(ny+1)*nx                 % Left face radiation
    T1               = (tca(i)+tca(i+nx+1))/2;      % Edge mean temperature
    crg              = SB*(T1^2+ts(2)^2)*(T1+ts(2))*ep*he/ny;
    crgm             = crgm+crg;iel=iel+1;
    for ii=1:3;for j = 1:3;Kr(lc(iel+ny,ii),lc(iel+ny,j)) = ...
                Kr(lc(iel+ny,ii),lc(iel+ny,j))+Kelc(ii,j)*crg;end;end
end
crg = crgm/iel;crsm  = 0;iel = 0;         % Mean left convection coefficient
for i                = 1:nx                                % Top face radiation
    T1               = (tca(i)+tca(i+1))/2;         % Edge mean temperature
    crs              = SB*(T1^2+ts(3)^2)*(T1+ts(3))*ep*he/ny;
    crsm             = crsm+crs;iel = iel+1;
    for ii=1:3;for j = 1:3;Kr(lc(iel+ny*2,ii),lc(iel+ny*2,j)) = ...
                Kr(lc(iel+ny*2,ii),lc(iel+ny*2,j))+Kelc(ii,j)*crs;end;end
end
crs = crsm/iel;                             % Mean top convection coefficient
disp(['f1_radiation left up: ',num2str([crd crg crs])]);
end
```

Lines 11 – 12 : global conductivity matrix assembling

```
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
```

The external loop is performed on the elements and the 2 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* (see previous slide). Moreover, the coefficients of the element matrices *Kel* are multiplied by their conductivity coefficient *co (n)* (see *line 4*).

Line 16 : drawing of the isotherms in the function *grisb.m*

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

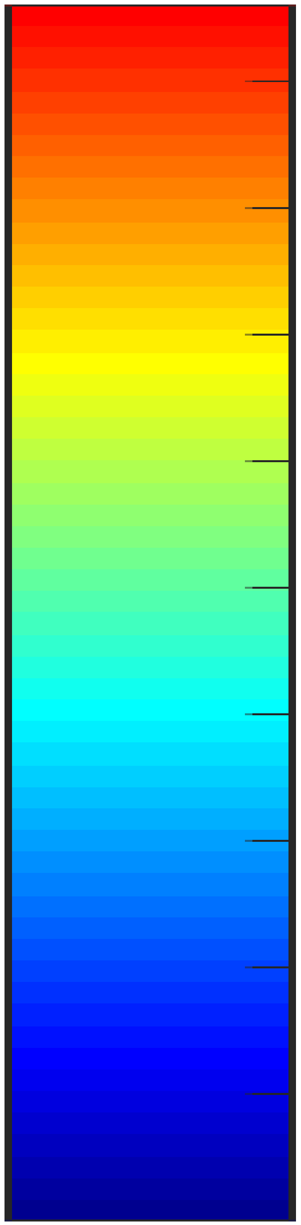
```
1   function [] = grisb(nx,ny,tca,gap)
2   figure('Position',[1 1 600 512]);
3   my = ny+1;no=(nx+1)*(ny+1);
4   B         = ones(my,nx+1)*tca(1);x = zeros(my,nx+1);y = zeros(my,nx+1);
5   for j     = 1 : nx+1;for i = 1 : ny;x(i,j) = j-1; y(i,j) = my-i;end;end;
6   ii        = 0;
7   for i     = 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
```

This function performs the visualization of the isotherms of a mesh *nx* x *ny* (arguments 1 and 2 of the function) based on the nodal temperatures stored in the vector *tca* (argument 3). The value of the intervals between successive isotherms can be adjusted. It is transmitted by the argument gap. Look at line 9 for the particularly effective color bar (function *br56.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
```

Matlab© function *br56.m* for computing a color bar of 56 colors



Line 21 : iterations to update the conduction-radiation matrix *radit.m*

This function performs iterations to take into account the non linearity of the conduction – radiation matrix. It also gives some statistics about the convergence if the number of stipulated iterations is > 10.
The updating of the matrix is performed at line 4 with the function *ItKr.m*. The loading is limited to imposed temperatures and is computed at lines 7 & 8.
At each iteration, an isotherms drawing is produced.

The statistical results deal with:

the minimum temperature of the mesh,
the total of the heat flows related to the virtual nodes.

Function Matlab© *radit.m* for radiation

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

1. Radiation to an identified zone with known temperature

2. Radiation to an other part of the domain

In the previous slides we approached the subject of radiative heat transfer from a body to an environment characterized only by its temperature. We also assumed that we had a black body.

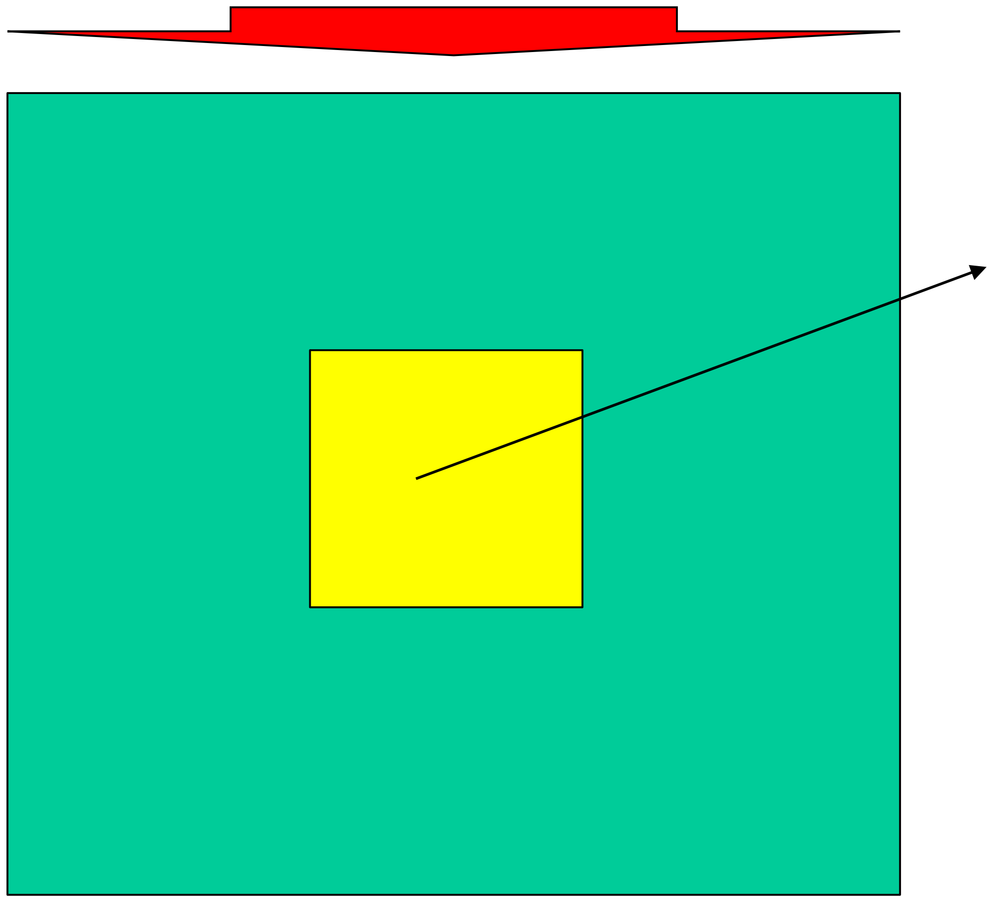
In the frame of urban studies, it should be more appropriate to analyze the radiative interactions between one body and its neighborhood.

It means that we have to define the radiative characteristics of all the bodies: emittance, reflectance, and their relative visibilities called view factor.

It should also be suitable to compute the radiosities and to carry out the balance between all the radiative exchanges in short and in long waves.

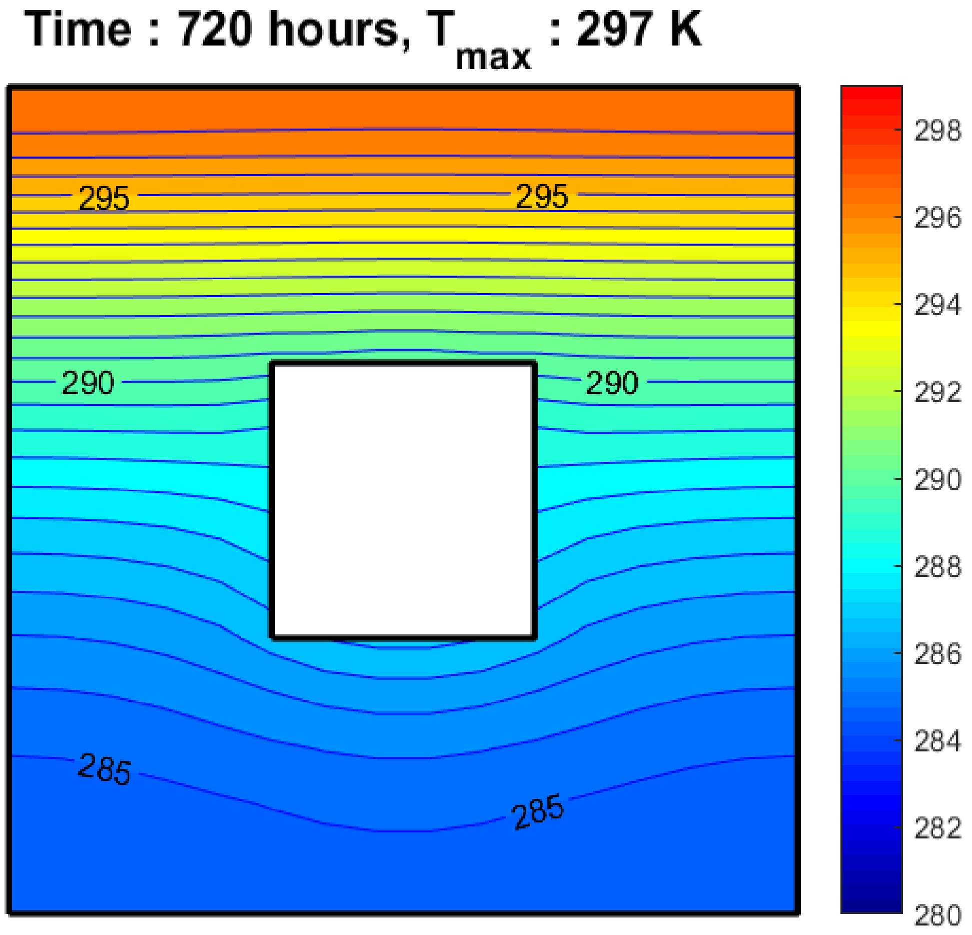
To introduce this important matter, we present some results for a domain receiving some heat load on the upper side and involving a cavity submitted to longwave radiative exchanges.

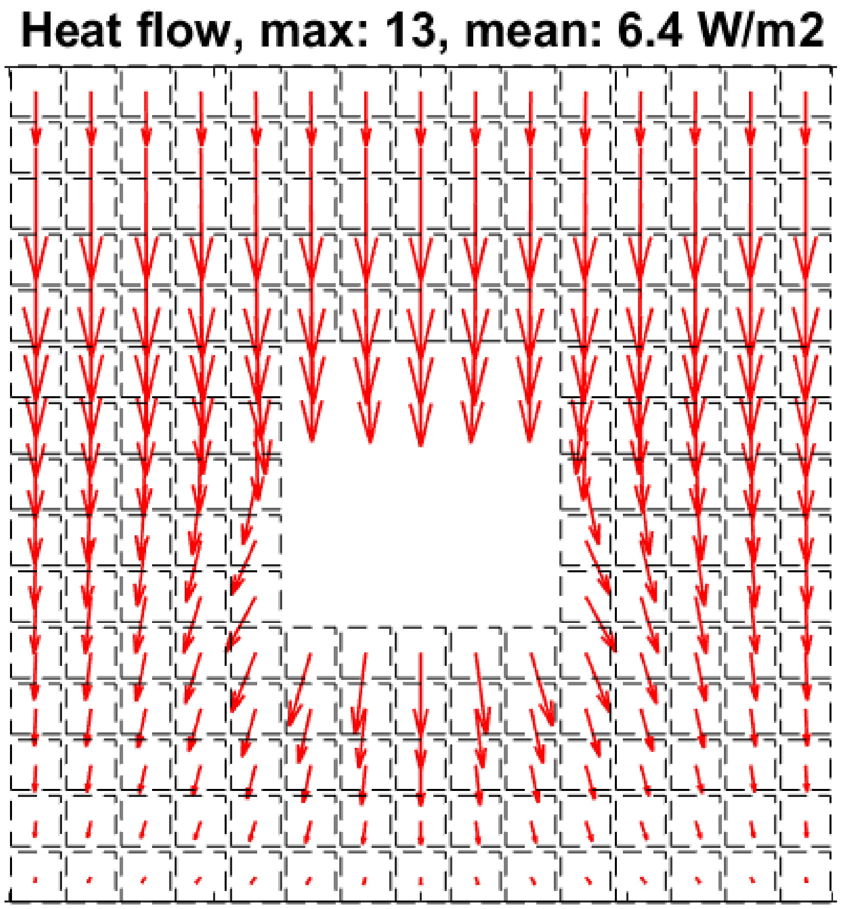
Periodic radiative constant heat load



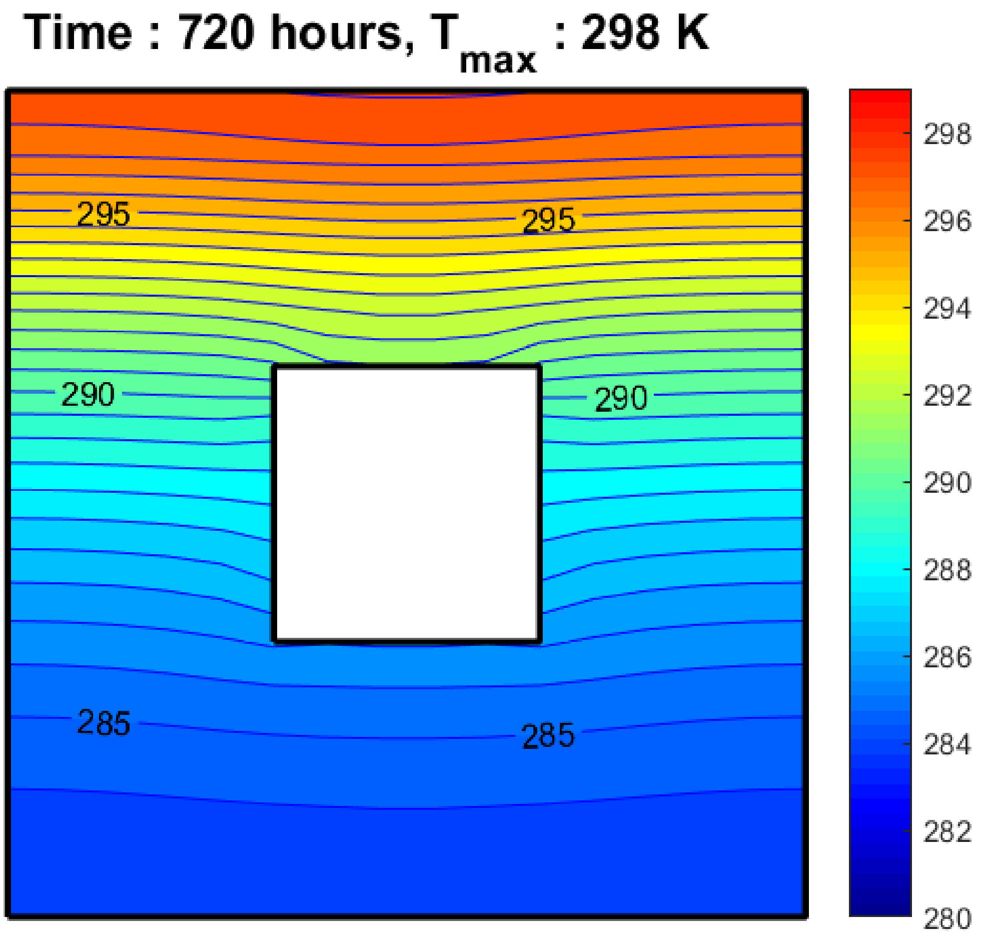
This illustration will provide an introduction to the tomorrow subject of

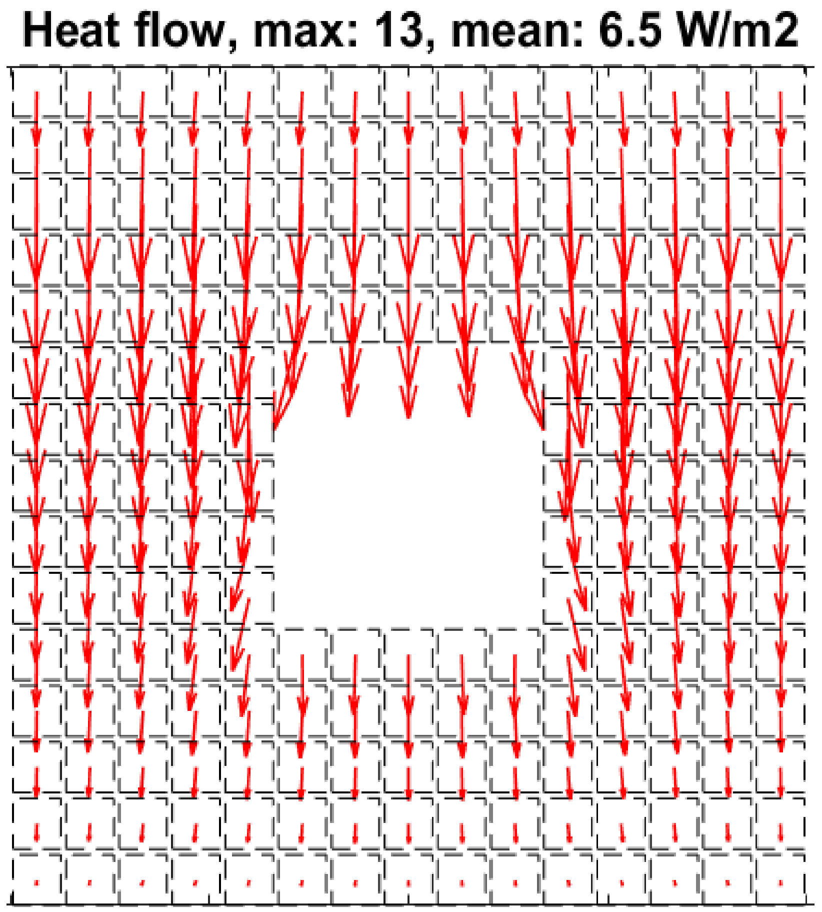
Transient heat transfers



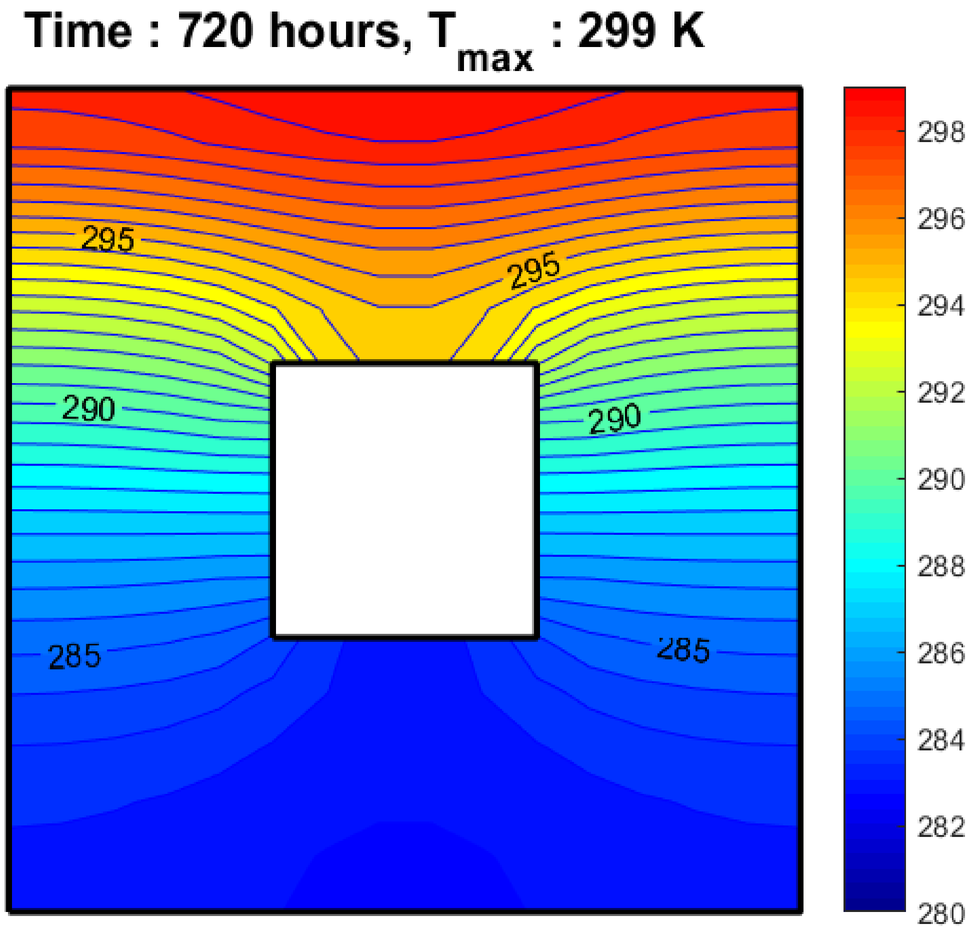


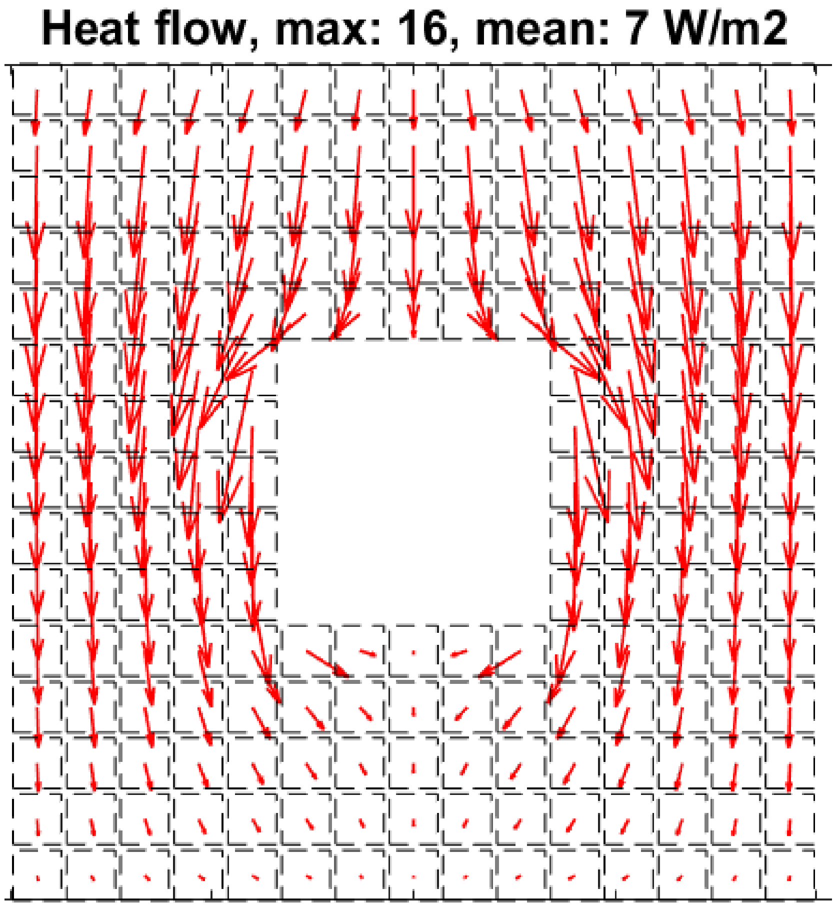
Isotherms, black body, ρ = 0, 30 days





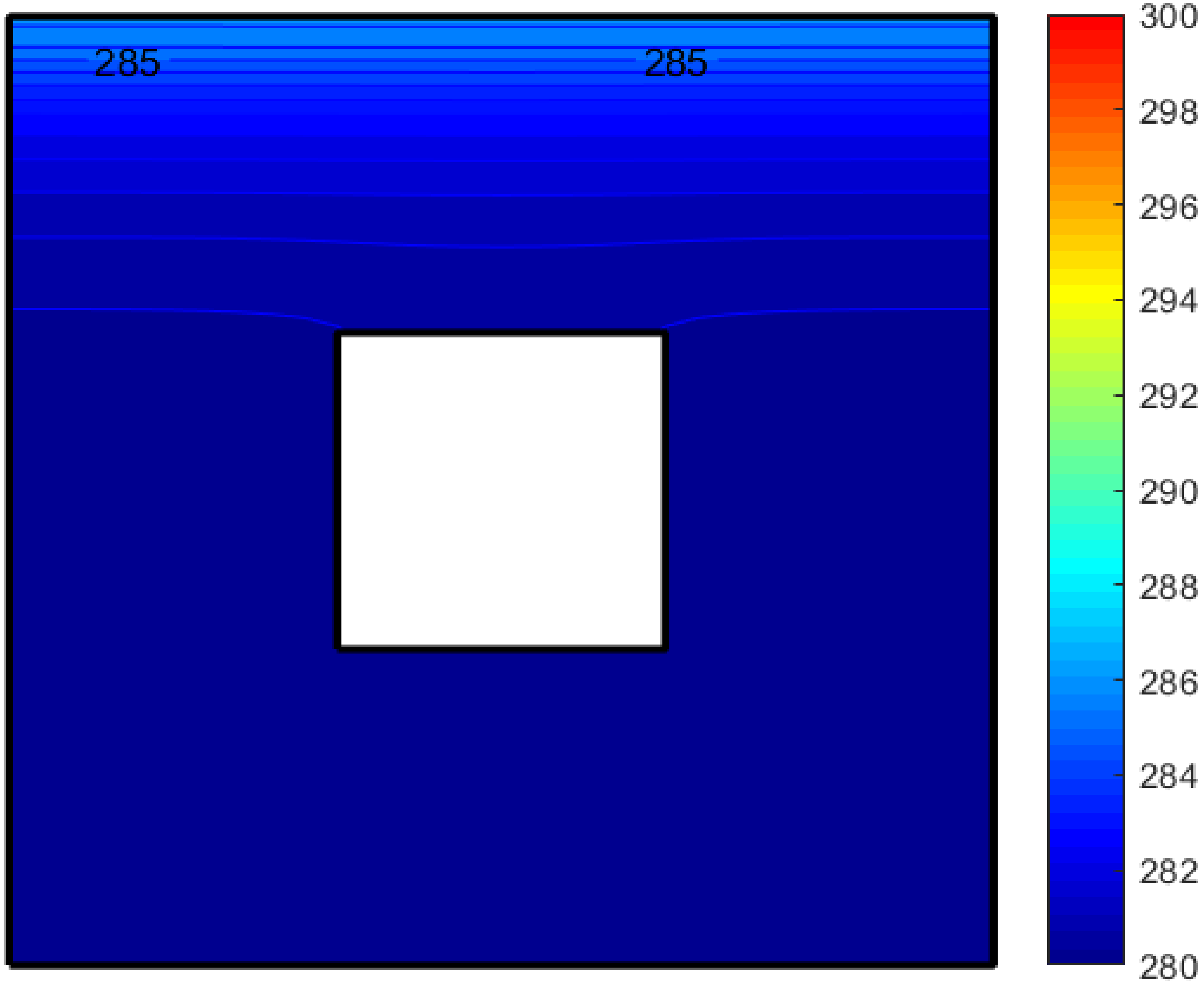
Isotherms, grey body, $\rho = 0.5$, 30 days



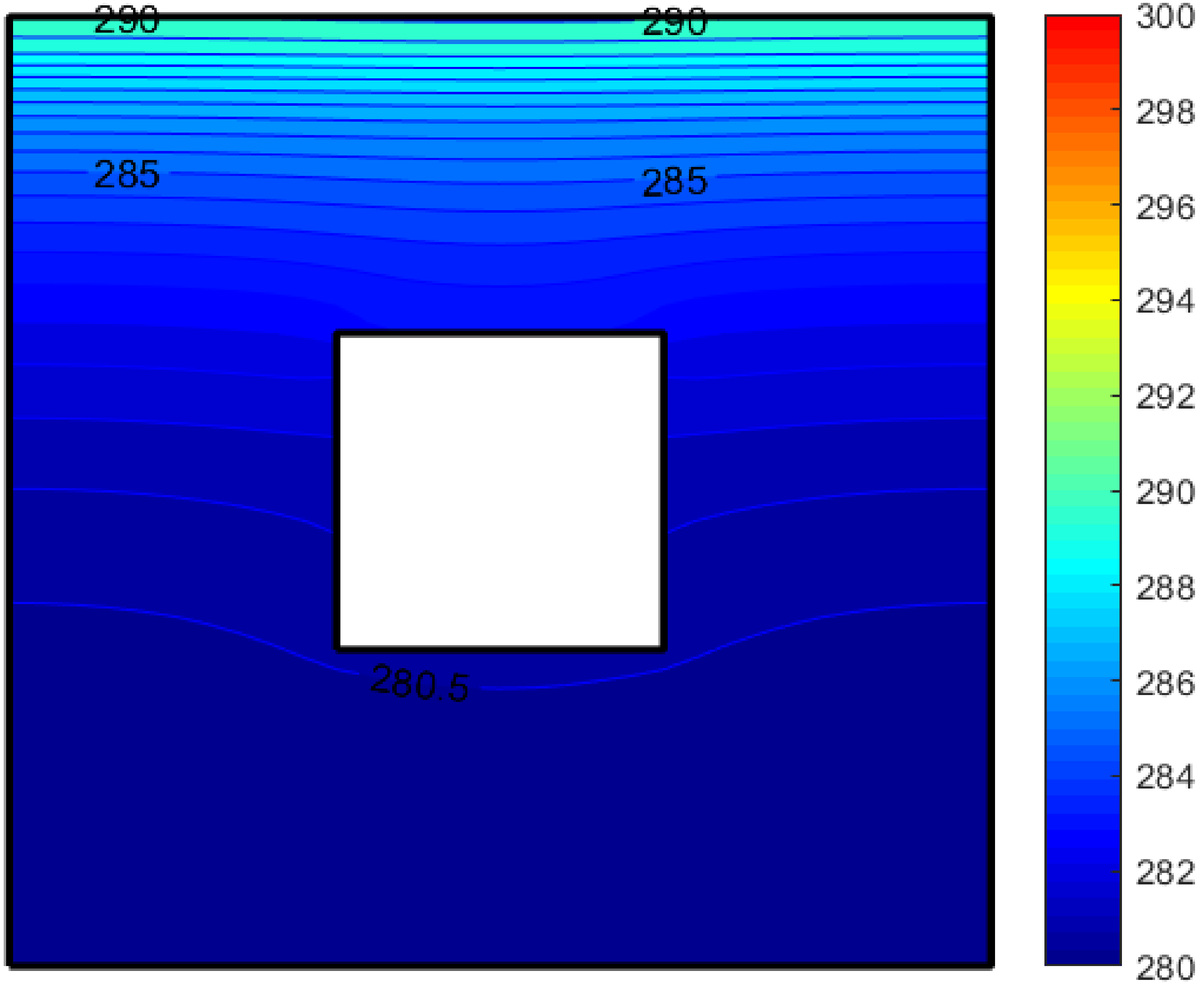


Isotherms, adiabatic cavity, ρ = 1, 30 days

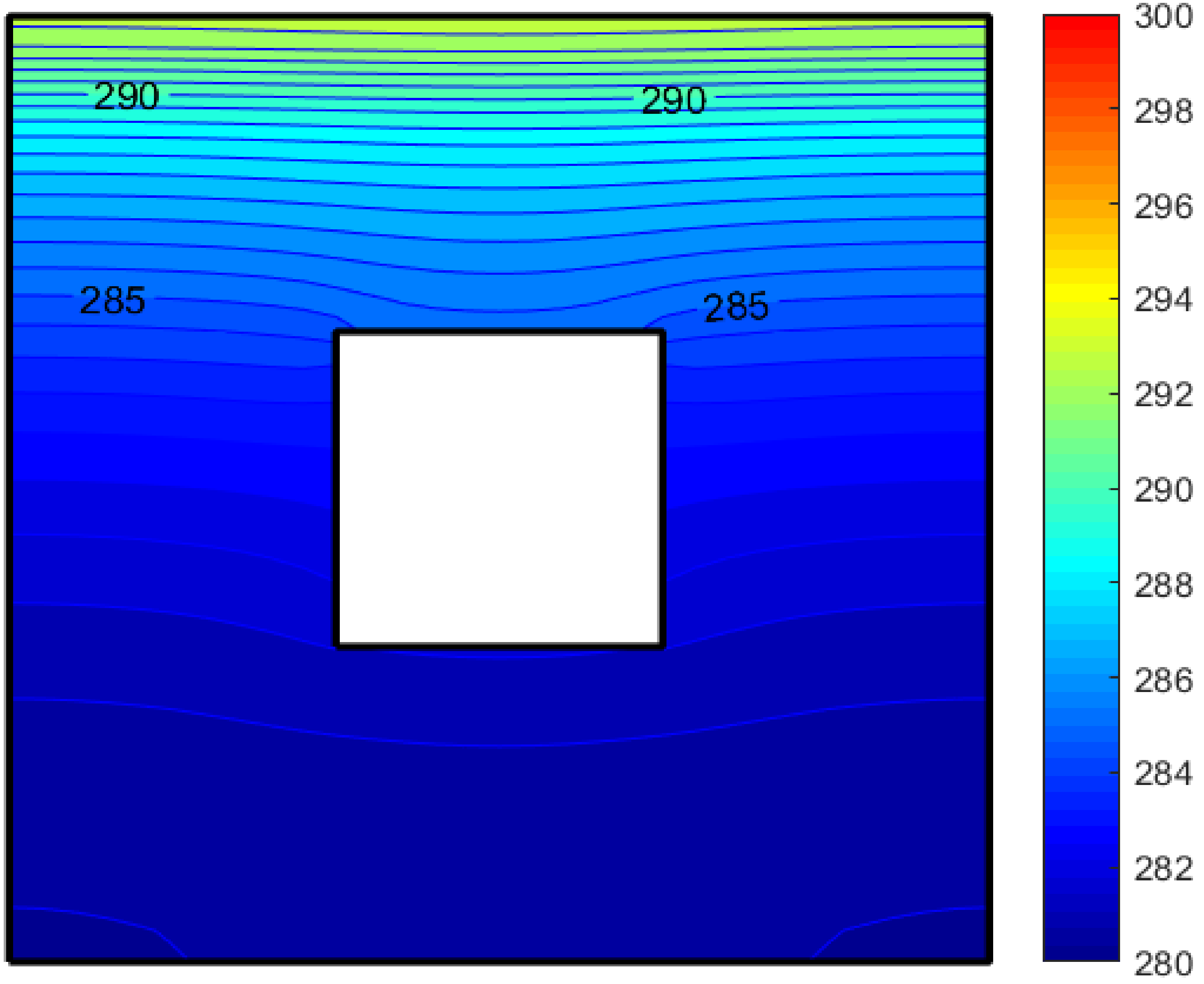
Time : 60 hours, T_{max} : 286 K



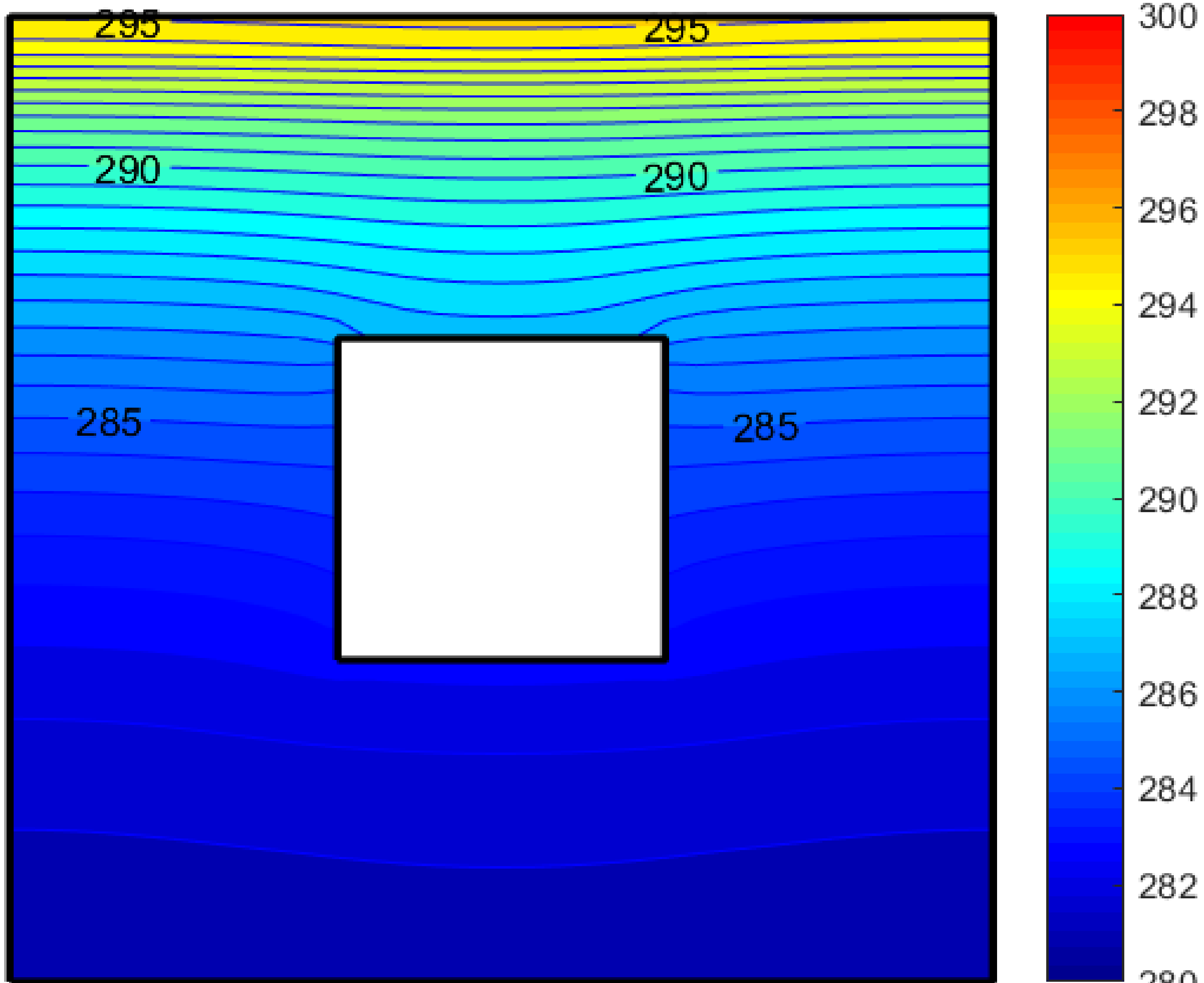
Time : 180 hours, T_{max} : 290 K

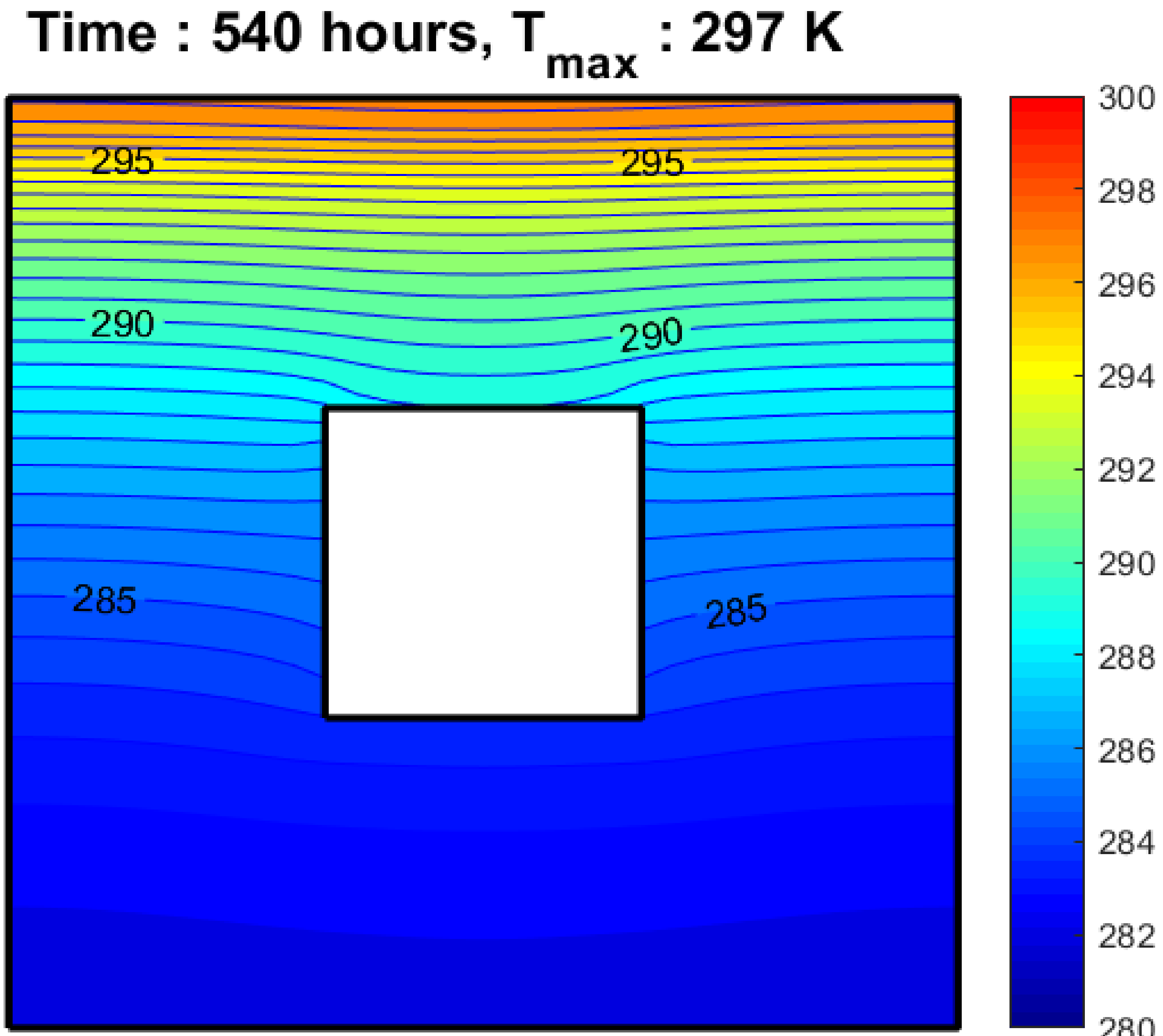


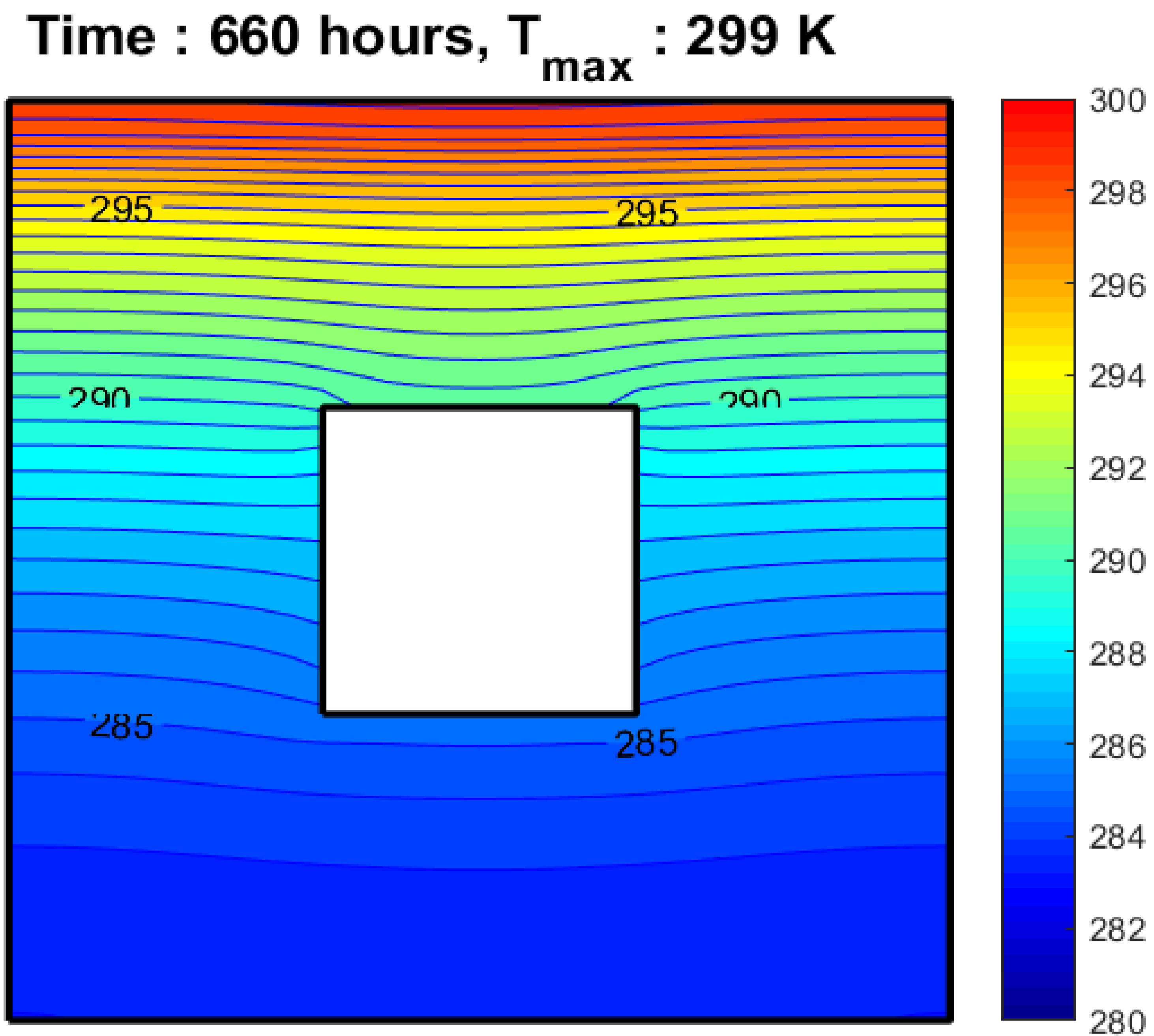
Time : 300 hours, T_{max} : 293 K



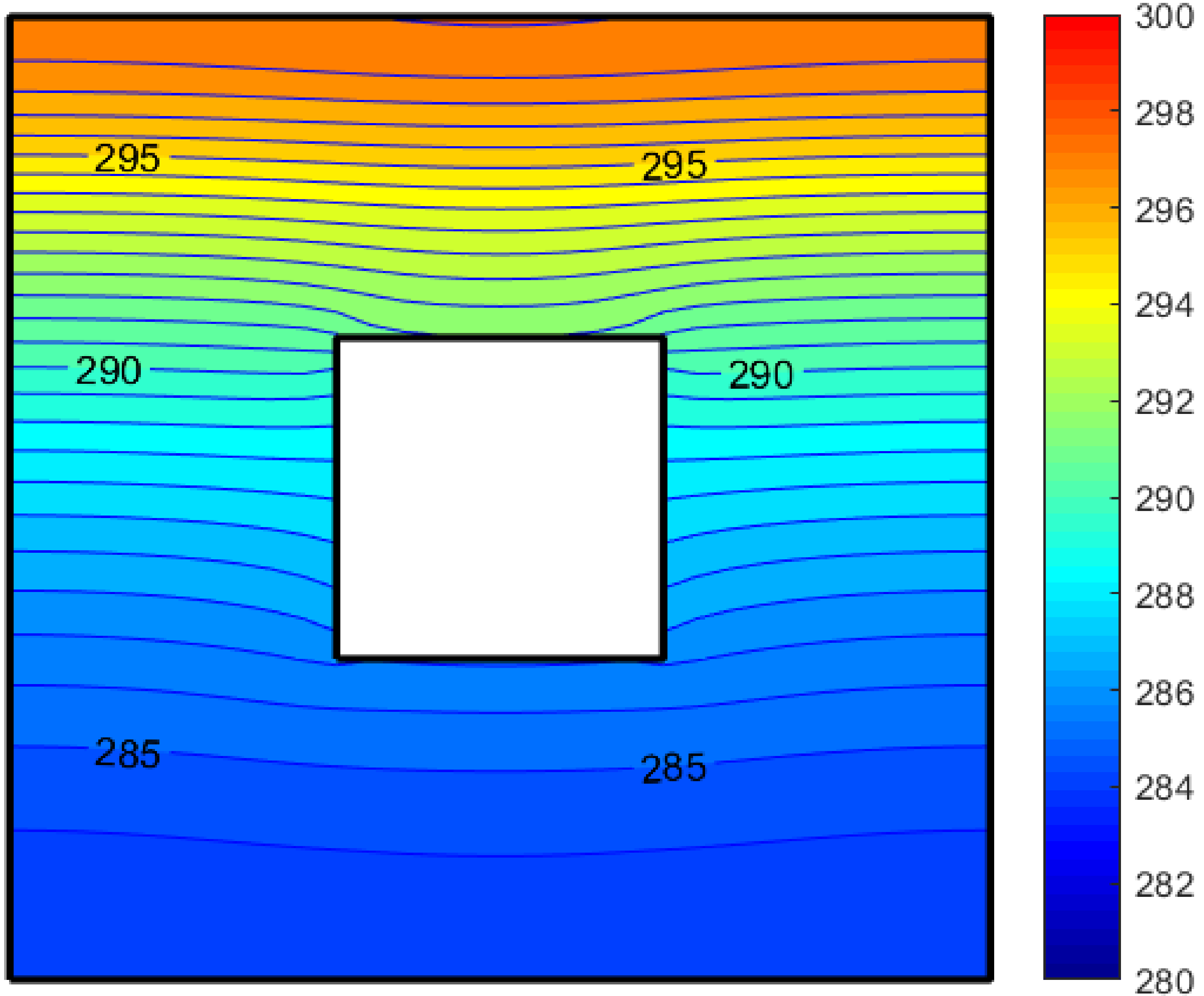
Time : 420 hours, T_{max} : 295 K




Time : 540 hours, T_{max} : 297 K
295
295
290
290
285
285
300
298
296
294
292
290
288
286
284
282
280


Time : 660 hours, T_{max} : 299 K
295
295
290
290
285
285
300
298
296
294
292
290
288
286
284
282
280

Time : 720 hours, T_{max} : 298 K



295
295
290
290
285
285
300
298
296
294
292
290
288
286
284
282
280