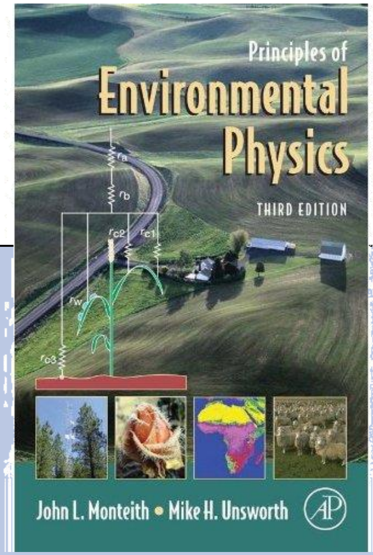
Benoit Beckers

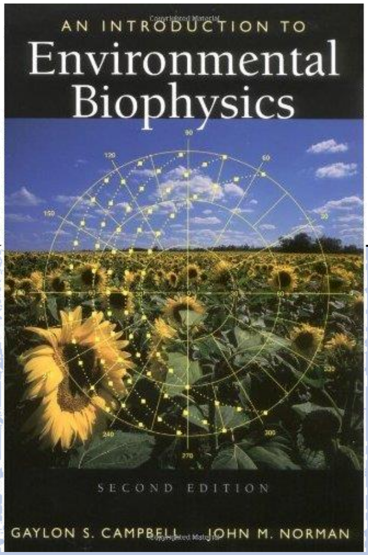
Architecture et Physique Urbaine - ISA BTP
Université de Pau et des Pays de l'Adour
Allée du Parc Montaury, 64600 Anglet (France)
benoit.beckers@univ-pau.fr, www.heliodon.net

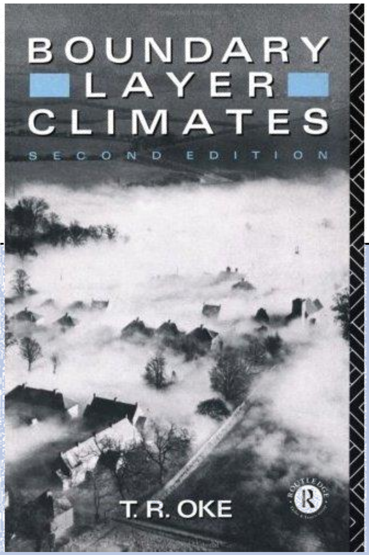
Finite element Simulation of convective heat transfers

Urban Physics - references

Environmental Physics developed in the 1970s (crop yield), 1980 (forest and acid rain), 1990 (hole in the ozone layer), 2000 (climate change). Among its methods: the local balance of energy flows (conductive, radiative, sensible and latent).



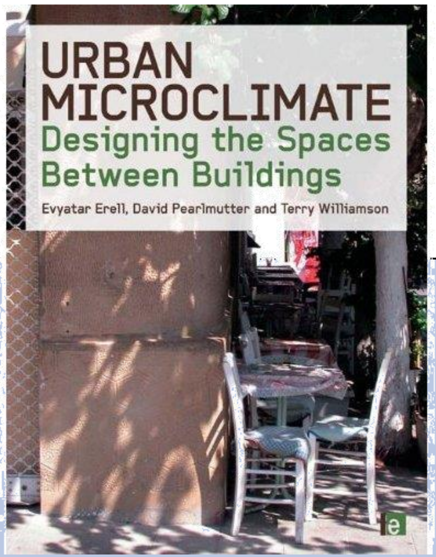


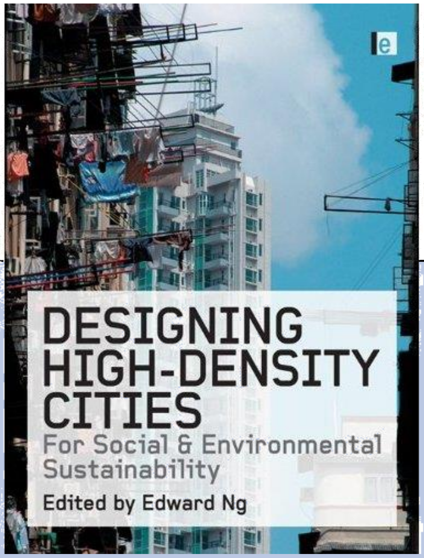


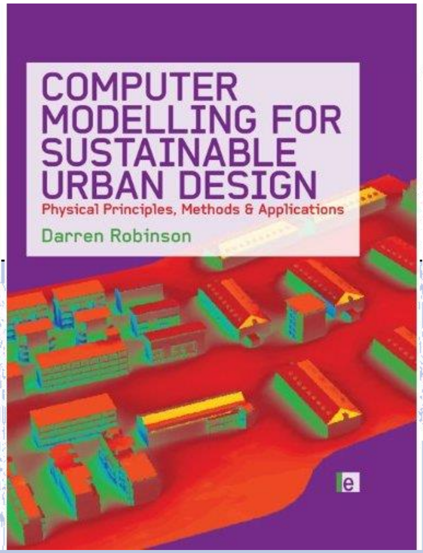
The application of the same methods to urban structures was proposed in the 1970s by T. R. Oke, around the issue of urban heat island. We can therefore speak of Urban Physics.

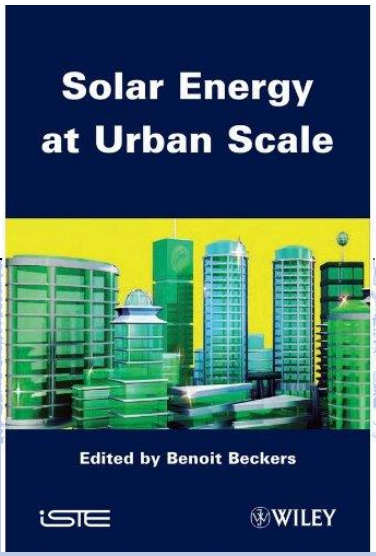
Urban Physics - references

In recent years, considerable progress has been made in several ancillary areas: geographic information systems, standardization of the level of detail of 3D models, terrestrial and satellite measurements, computing power of computers…


URBAN MICROCLIMATE
Designing the Spaces Between Buildings
Evyatar Erell, David Pearlmutter and Terry Williamson


DESIGNING HIGH-DENSITY CITIES
For Social & Environmental Sustainability
Edited by Edward Ng

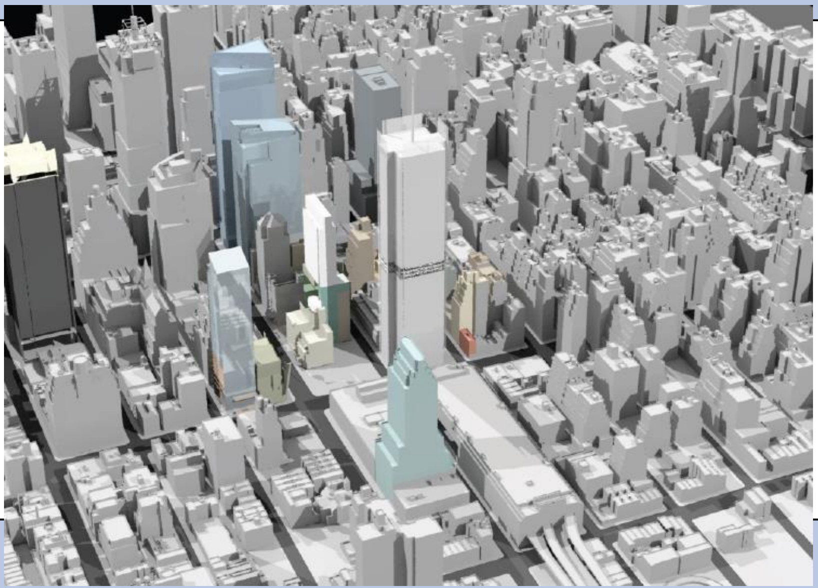

COMPUTER MODELLING FOR SUSTAINABLE URBAN DESIGN
Physical Principles, Methods & Applications
Darren Robinson

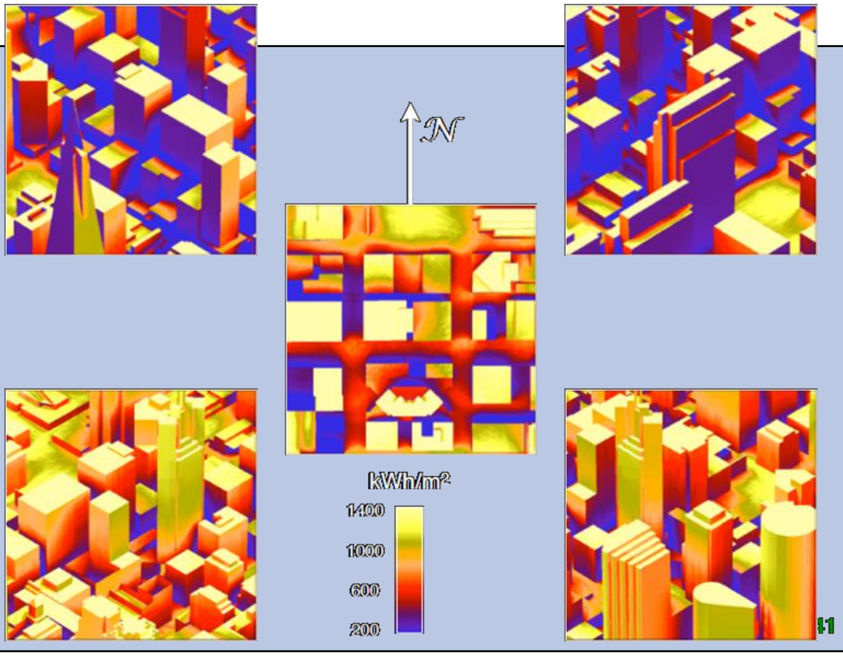

Solar Energy at Urban Scale
Edited by Benoit Beckers
ISTE WILEY

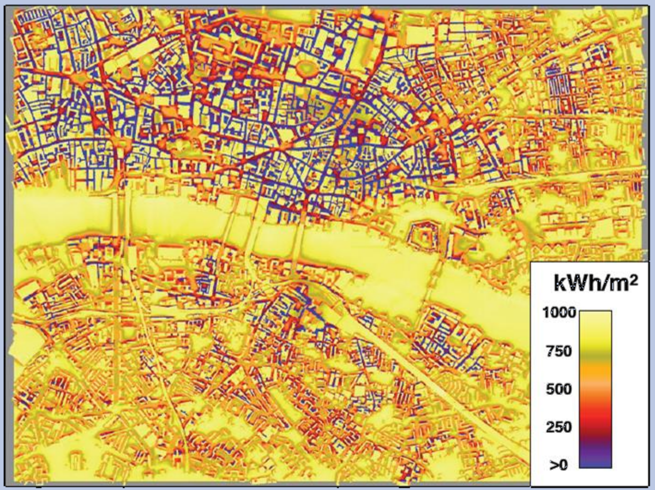
… Allowing for a first apprehension of urban physics by numerical simulation.
Framework: "smart", "safe" and "sustainable" city.
Emergence of urban thermal (urban climate, building-city-territory heat flux, energy efficiency, local production of renewable energy).

SOLAR ENERGY AT URBAN SCALE

- SOLAR ENERGY AT URBAN SCALE
- 24 /25 may 2010, Compiègne
- www.utc.fr/seus

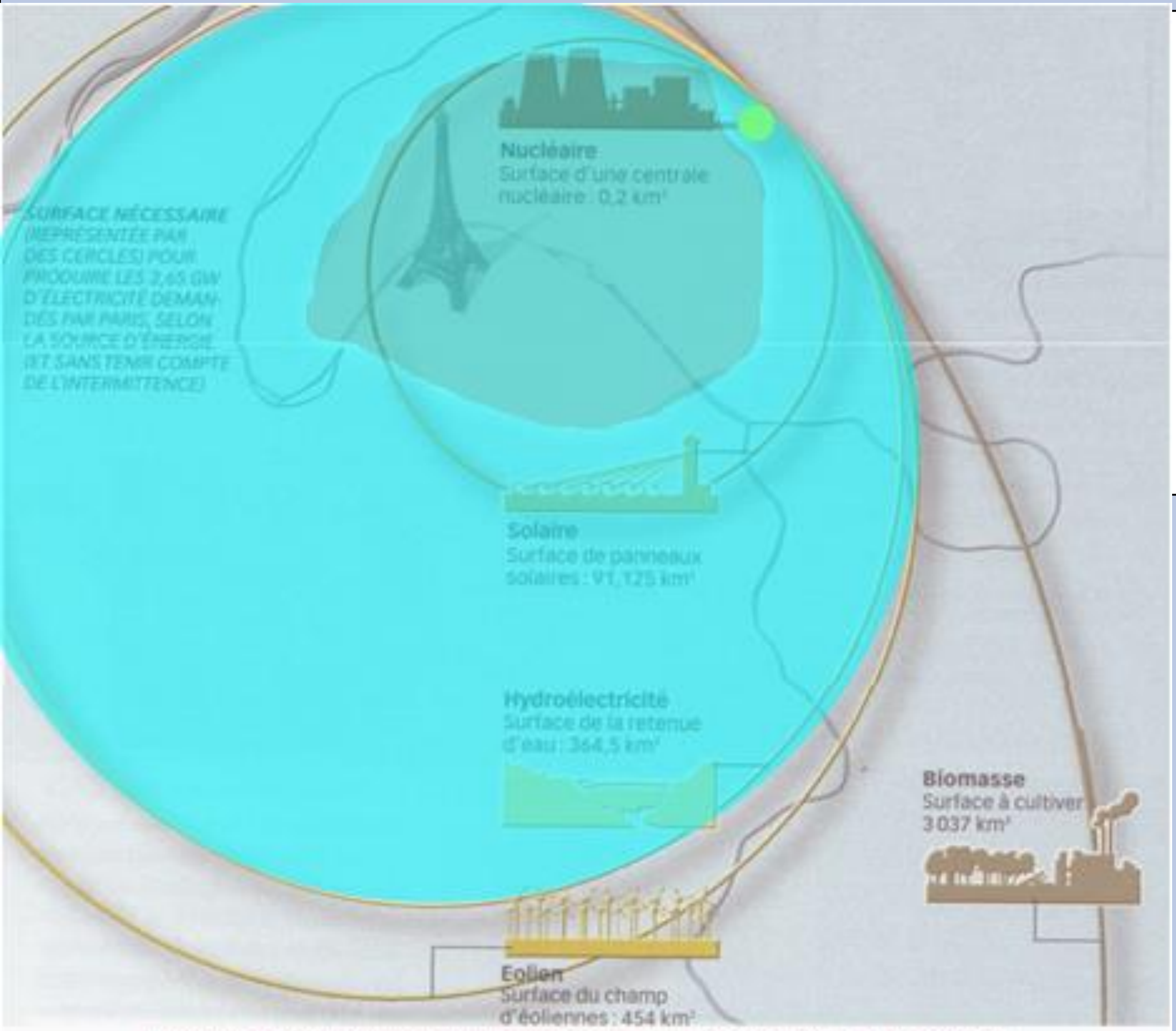






John Mardaljevic De Beaufort University

Daylight and Solar Energy Simulation at Urban Scale


SURFACE NÉCESSAIRE (REPRÉSENTÉE PAR DES CERCLES) POUR PRODUIRE LES 3,65 GW D'ÉLECTRICITÉ DEMANDÉS PAR PARIS, SELON LA SOURCE D'ÉNERGIE (ET SANS TENIR COMPTE DE L'INTERMITTENCE)
Nucléaire
Surface d'une centrale nucléaire : 0,2 km²
Solaire
Surface de panneaux solaires : 91,125 km²
Hydroélectricité
Surface de la retenue d'eau : 364,5 km²
Biomasse
Surface à cultiver 3 037 km²
Eolien
Surface du champ d'éoliennes : 454 km²

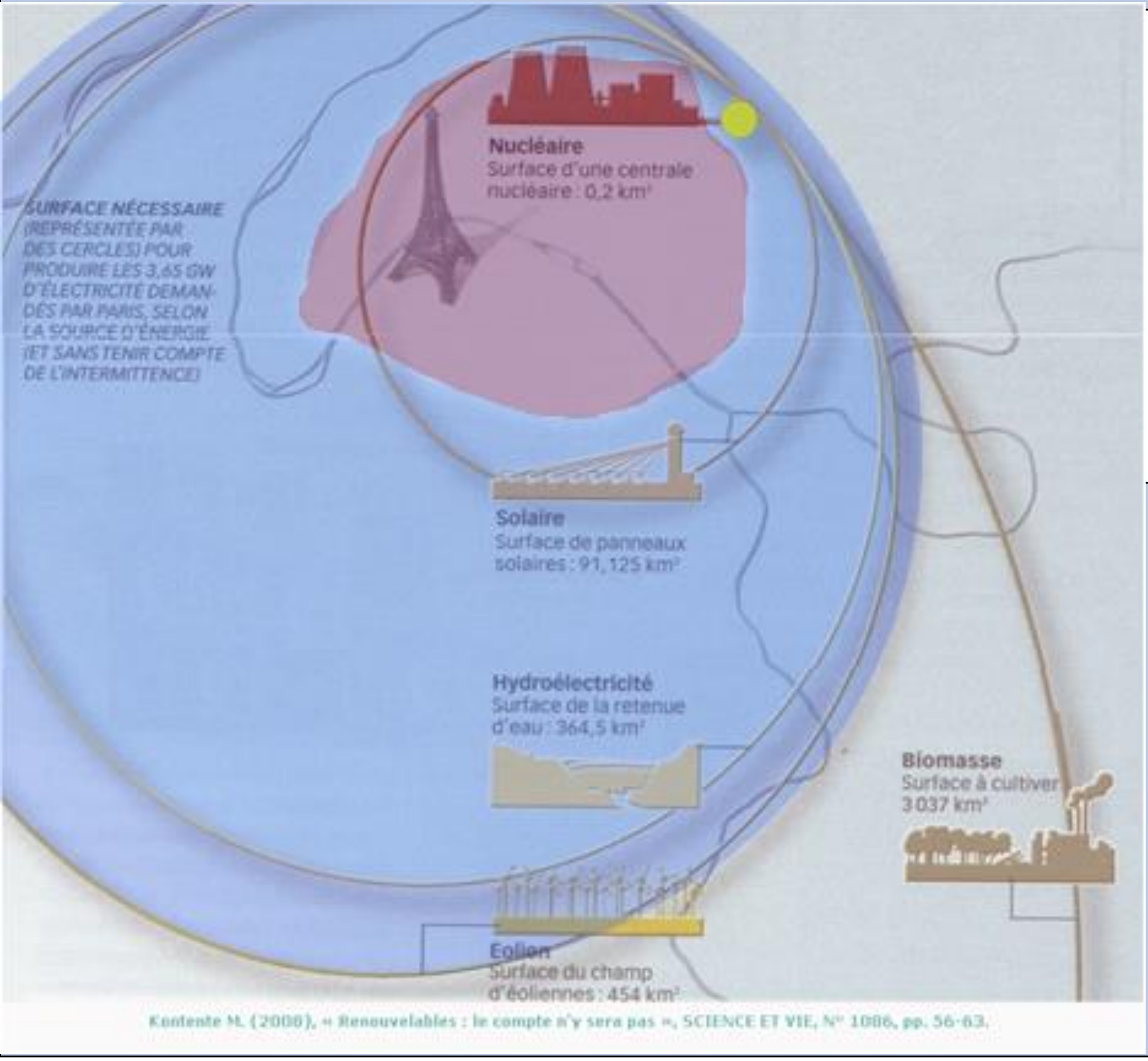
Kontente M. (2008), « Renouvelables : le compte n'y sera pas », SCIENCE ET VIE, N° 1086, pp. 56-63.

Hydropower

260 km2 to produce 3,65 GW

(water retention)

Daylight and Solar Energy Simulation at Urban Scale

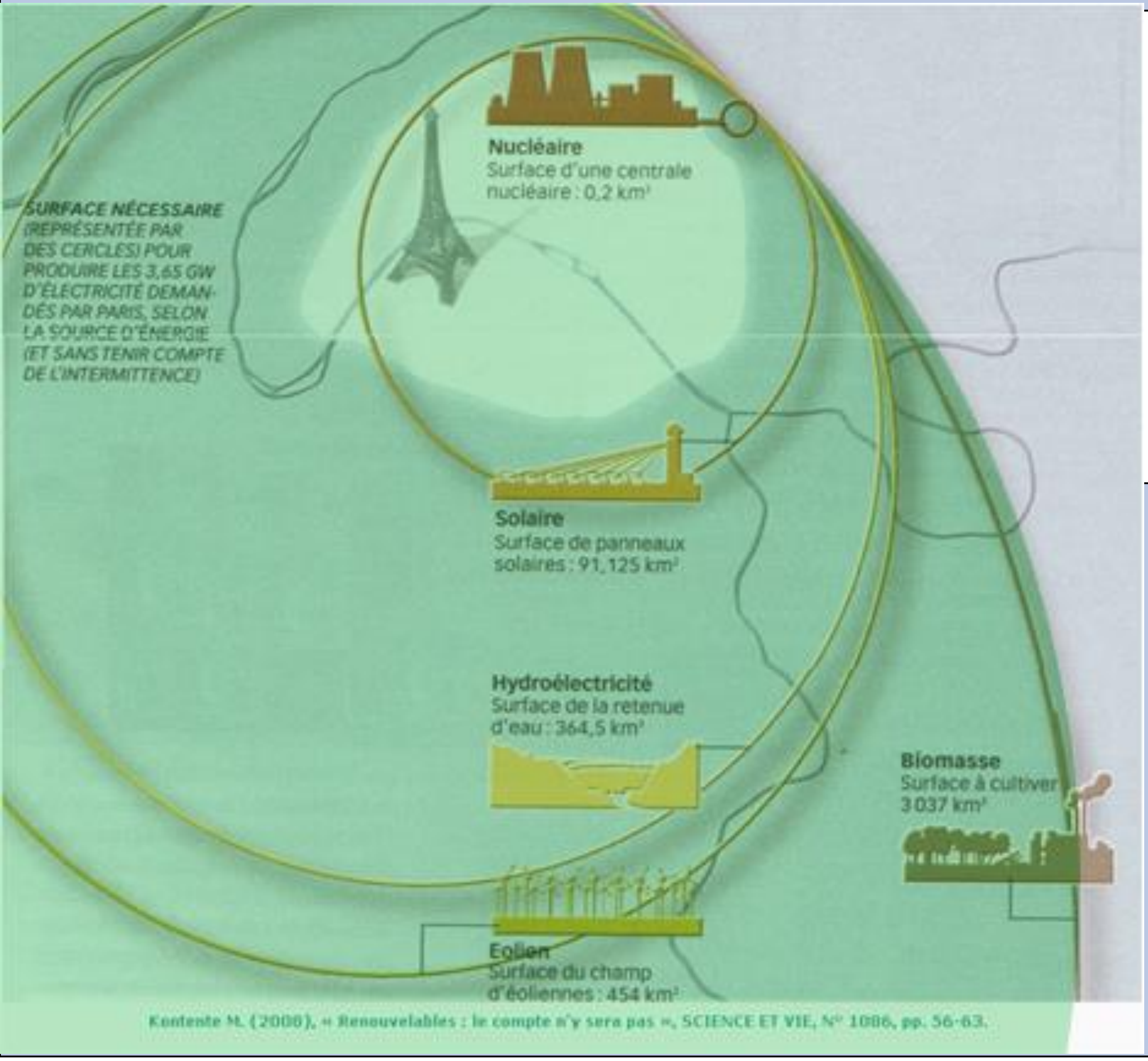

Nucléaire
Surface d'une centrale nucléaire : 0,2 km²
SURFACE NÉCESSAIRE (REPRÉSENTÉE PAR DES CERCLES) POUR PRODUIRE LES 3,65 GW D'ÉLECTRICITÉ DEMANDÉS PAR PARIS, SELON LA SOURCE D'ÉNERGIE (ET SANS TENIR COMPTE DE L'INTERMITTENCE)
Solaire
Surface de panneaux solaires : 91,125 km²
Hydroélectricité
Surface de la retenue d'eau : 364,5 km²
Biomasse
Surface à cultiver 3 037 km²
Eolien
Surface du champ d'éoliennes : 454 km²
Kontente M. (2008), « Renouvelables : le compte n'y sera pas », SCIENCE ET VIE, N° 1086, pp. 56-63.

Wind energy

450 km2 to produce 3,65 GW

(wind farm)

Daylight and Solar Energy Simulation at Urban Scale


Nucléaire
Surface d'une centrale nucléaire : 0,2 km²
SURFACE NÉCESSAIRE (REPRÉSENTÉE PAR DES CERCLES) POUR PRODUIRE LES 3,65 GW D'ÉLECTRICITÉ DEMANDÉS PAR PARIS, SELON LA SOURCE D'ÉNERGIE (ET SANS TENIR COMPTE DE L'INTERMITTENCE)
Solaire
Surface de panneaux solaires : 91,125 km²
Hydroélectricité
Surface de la retenue d'eau : 364,5 km²
Biomasse
Surface à cultiver 3 037 km²
Eolien
Surface du champ d'éoliennes : 454 km²

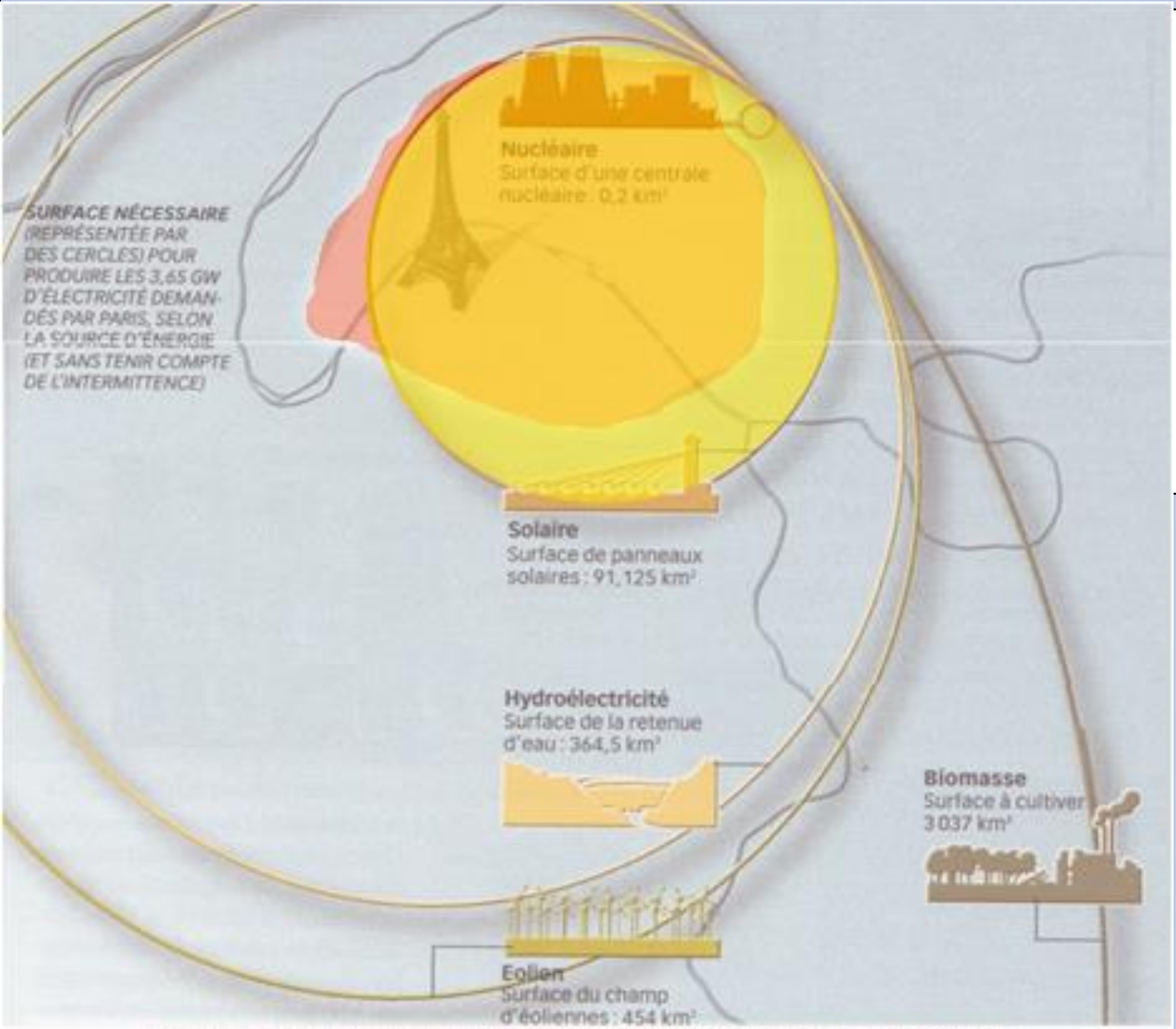
Kontente M. (2008), « Renouvelables : le compte n'y sera pas », SCIENCE ET VIE, N° 1086, pp. 56-63.

Biomass

3000 km^2 to produce 3,65 GW

(surface to cultivate)

Daylight and Solar Energy Simulation at Urban Scale


SURFACE NÉCESSAIRE (REPRÉSENTÉE PAR DES CERCLES) POUR PRODUIRE LES 3,65 GW D'ÉLECTRICITÉ DEMANDÉS PAR PARIS, SELON LA SOURCE D'ÉNERGIE (ET SANS TENIR COMPTE DE L'INTERMITTENCE)
Nucléaire
Surface d'une centrale nucléaire : 0,2 km²
Solaire
Surface de panneaux solaires : 91,125 km²
Hydroélectricité
Surface de la retenue d'eau : 364,5 km²
Biomasse
Surface à cultiver 3 037 km²
Eolien
Surface du champ d'éoliennes : 454 km²

Kontente M. (2008), « Renouvelables : le compte n'y sera pas », SCIENCE ET VIE, N° 1086, pp. 56-63.

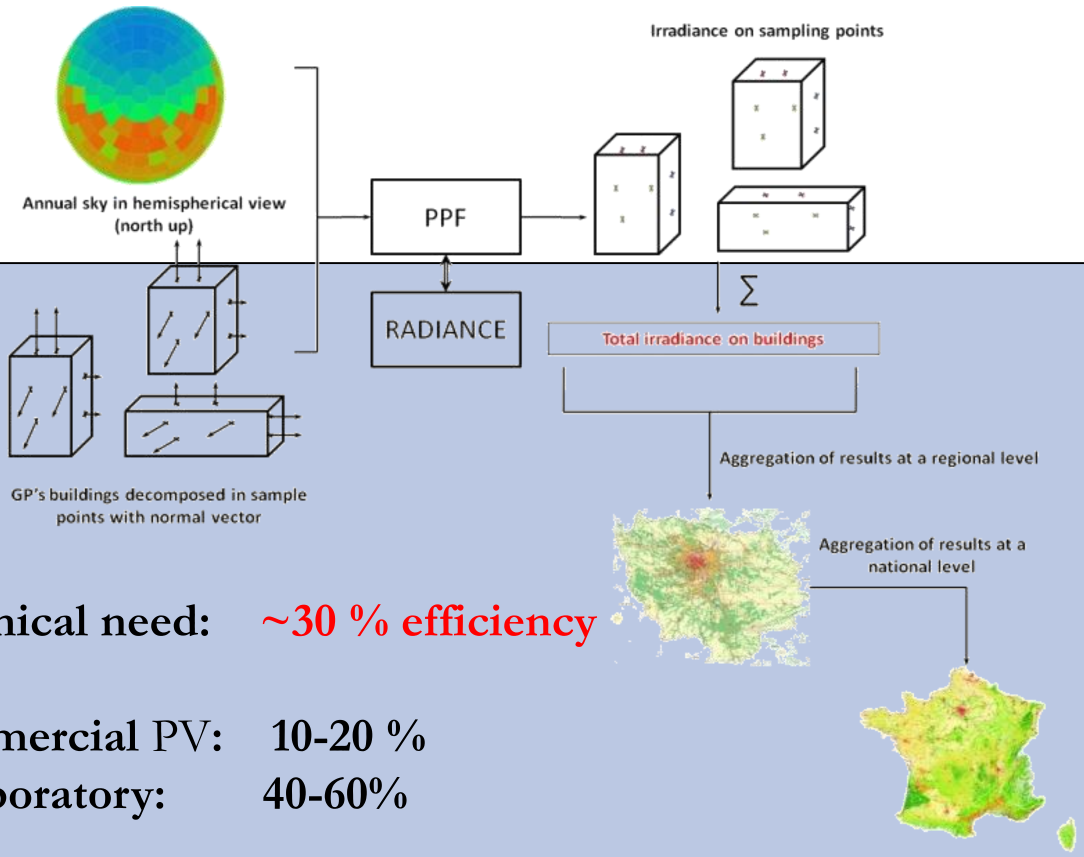
Solar energy

90 km2 to produce 3,65 GW

(solar panels)

General issues

- PhD (2012) : Elie Ghanassia (EDF, dir.: M. Maïzia & B. Beckers)


Annual sky in hemispherical view (north up)
GP's buildings decomposed in sample points with normal vector
PPF
RADIANCE
Irradiance on sampling points
Σ
Total irradiance on buildings
Aggregation of results at a regional level
Aggregation of results at a national level

« The sun will provide 100% of our energy needs in twenty years »

Ray Kurzweil – Le Monde Magazine 26/03/2011

Technical need: ~30 % efficiency

Commercial PV: **10-20 %**

In laboratory: 40-60%

- PhD (2014) : Thibaut Vermeulen (dir.: B. Beckers & P. Villon)

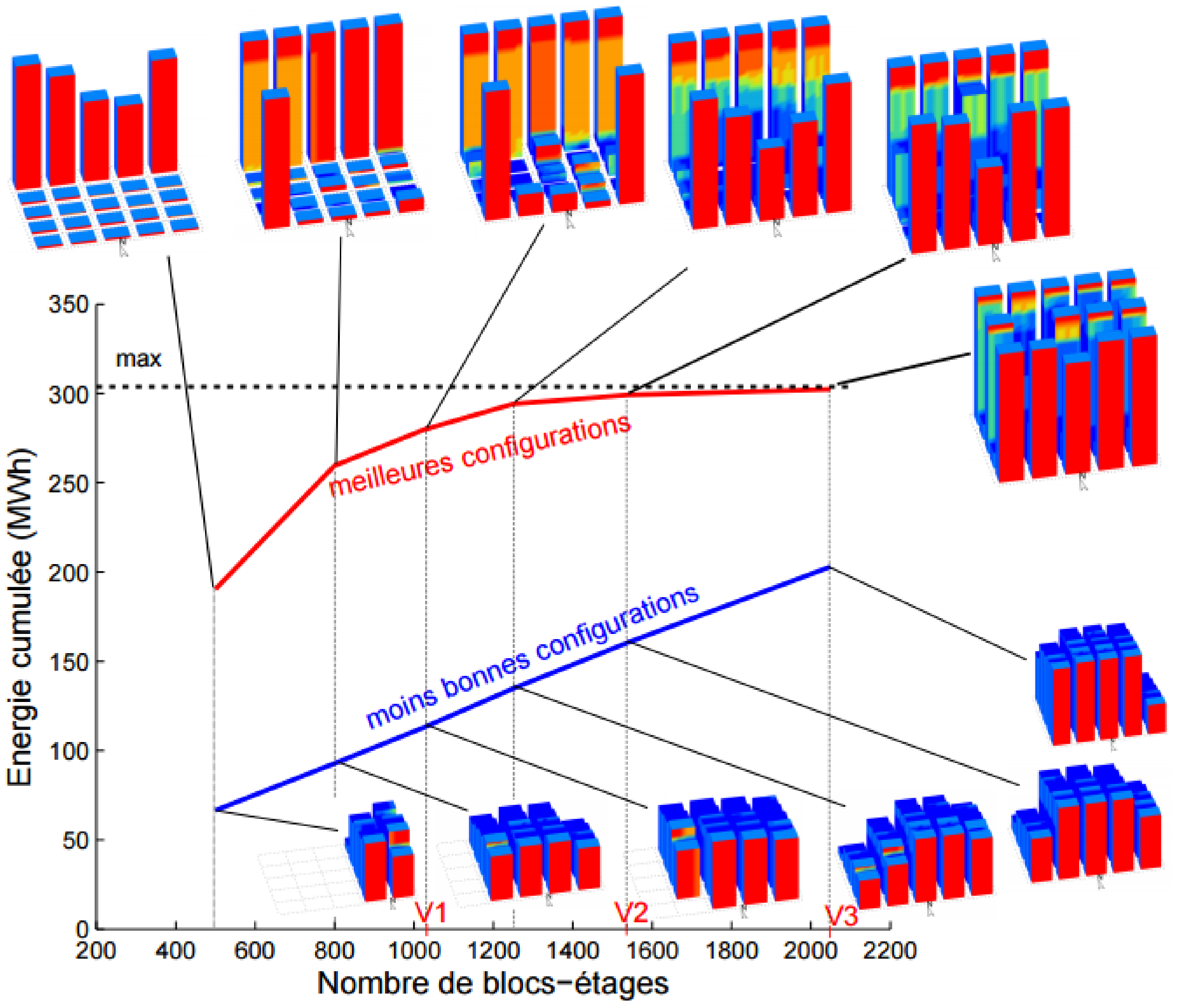


FIGURE 1.10: Evolution de la fonction-objectif avec la densité (21 décembre, 50°N, ciel clair, Liu & Jordan)

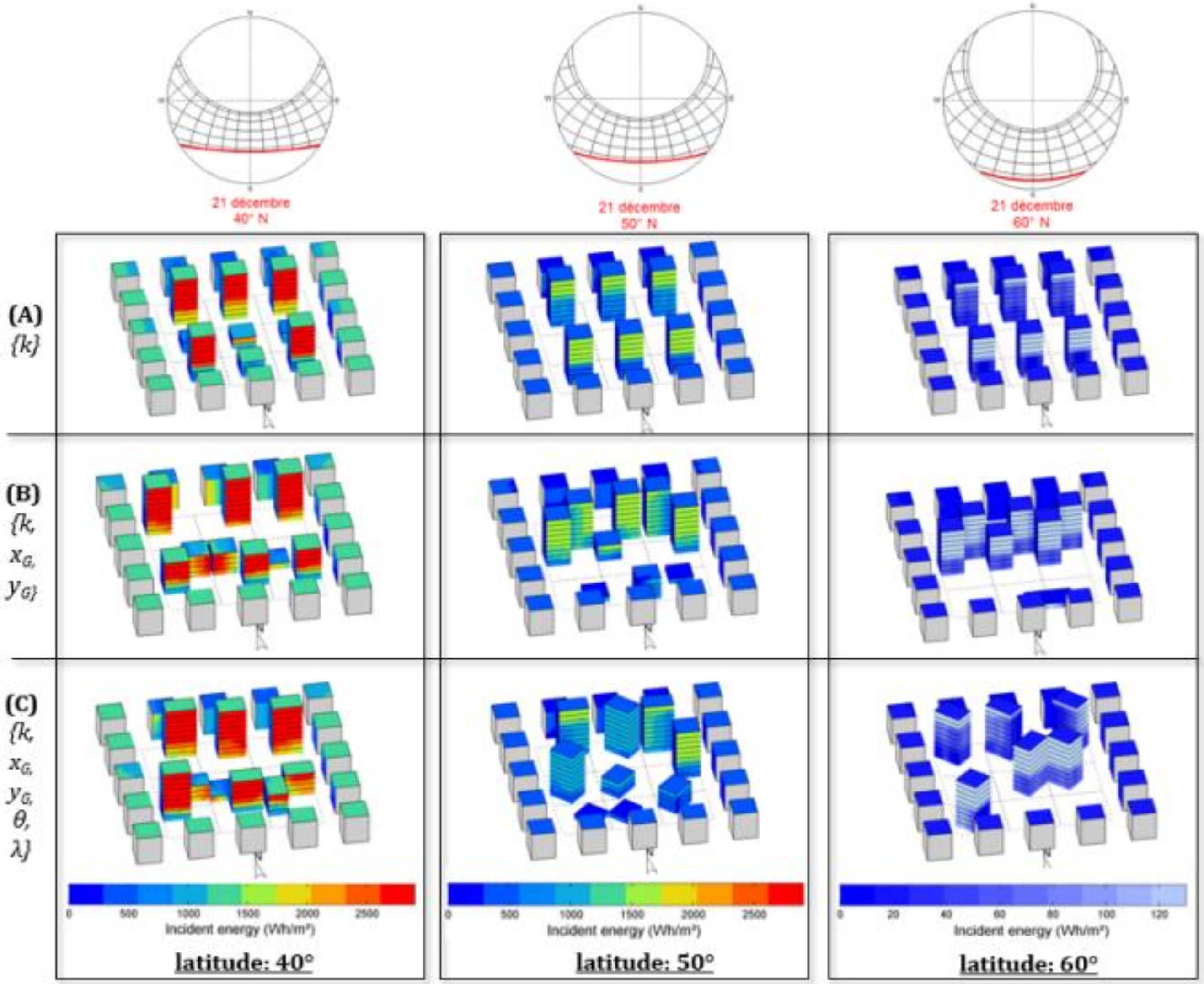


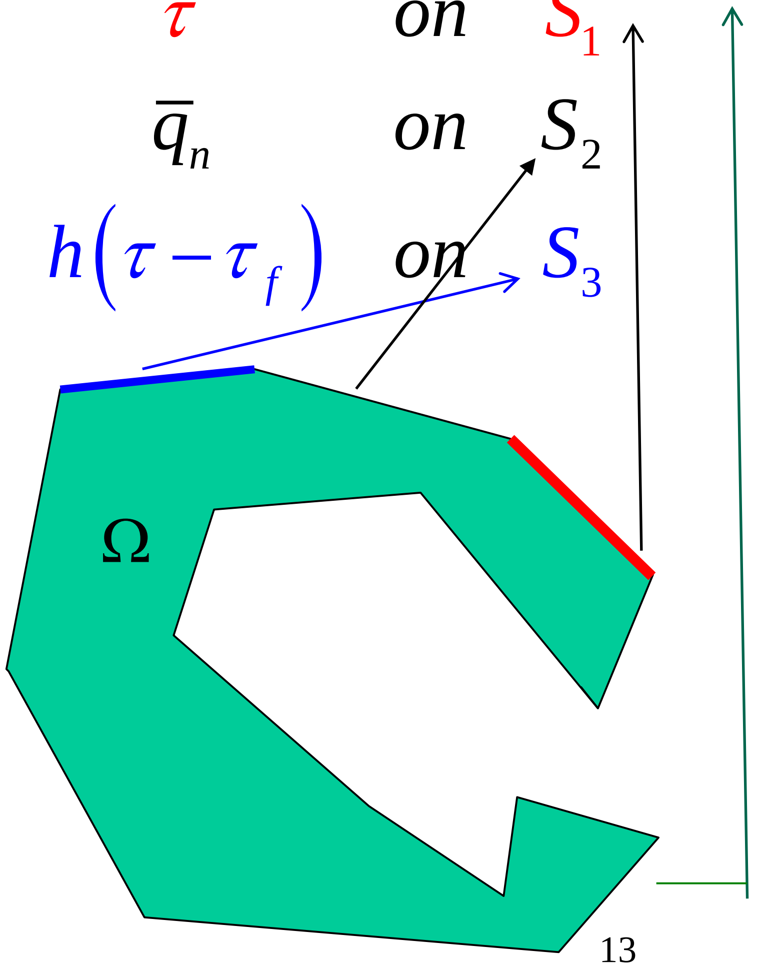
FIGURE 1.17: Formes optimales pour différents ensembles de paramètres et différentes latitudes (21 décembre, ciel clair, Liu & Jordan)

Convective heat transfers

We want to compute the temperature distribution in a domain Ω, subjected to boundary conditions on the temperature and/or the heat flow and convection conditions, namely thermal exchanges with a fluid in which the domain. is fully or partially incorporated The following system of partial differential equations must be solved as well as the boundary conditions associated with it:

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

$$\begin{aligned} \tau &= \bar{\tau} && on \quad S_1 \\ \vec{n}\cdot\vec{q} &= \bar{q}_n && on \quad S_2 \\ \vec{n}\cdot\vec{q} &= h\left(\tau-\tau_f\right) && on \quad S_3 \end{aligned}$$



In these equations, τ is the unknown temperature field expressed in K, k, the thermal conductivity ($Wm^{-1}K^{-1}$) and h, the convection coefficient ($Wm^{-2}K^{-1}$) .
The boundary of the domain is divided into 3 parts: S_1 where the temperature is imposed, S_2 where the normal heat flow is imposed and S_3, where the convective flow is imposed. A bar indicates that the concerned quantity is imposed.

Convective heat transfers

1. Partial differential equations
2. Variational methods
3. Finite element method (local level)
4. Finite element method (global level)
5. Examples
6. Prescribed heat flows
7. Matlab© procedures

In the presence of convective boundary conditions, the solution of the system of partial differential equations is equivalent to the search for the minimum of the past functional in which the convection term, which depends on the coefficient h, has been added:

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

To express the stationnarity condition, it is sufficient to write that under the boundary conditions:

$$\tau = \bar{\tau},\ \delta\tau = 0\ on\ S_1$$

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

Explicitly: $$I(\tau + \delta\tau) - I(\tau) = 0$$

$$\int_{\Omega} \frac{1}{2} k\ (grad(\tau+\delta\tau))^T \cdot grad(\tau+\delta\tau)\ d\Omega - \int_{\Omega} \frac{1}{2} k\ (grad\,\tau)^T \cdot grad\,\tau\ d\Omega$$

$$+\frac{1}{2}\int_{S_3} h\left(\tau+\delta\tau-\tau_f\right)^2 dS - \frac{1}{2}\int_{S_3} h\left(\tau-\tau_f\right)^2 dS$$

$$+\int_{S_2} \bar{q}_n(\tau+\delta\tau)dS - \int_{S_2} \bar{q}_n \tau dS = 0$$

By developing this expression, we obtain:

$$\int_{\Omega} \left(\frac{1}{2}k\ grad(\tau)^T \cdot grad\,\tau + \frac{1}{2}k\ (grad(\delta\tau))^T \cdot grad(\delta\tau)\right) d\Omega +$$

$$\int_{\Omega} \frac{1}{2} 2\ k\ (grad(\tau))^T \cdot grad(\delta\tau)\ d\Omega - \int_{\Omega} \frac{1}{2} k\ (grad\,\tau)^T \cdot grad\,\tau\ d\Omega$$

$$+\frac{1}{2}\int_{S_3} h\left(\tau-\tau_f\right)^2 dS - \frac{1}{2}\int_{S_3} h\left(\tau-\tau_f\right)^2 dS + \frac{1}{2}\int_{S_3}\left(h\left(\delta\tau\right)^2 + 2h\left(\tau-\tau_f\right)\delta\tau\right)dS$$

$$+\int_{S_2} \overline{q}_n(\tau+\delta\tau)dS - \int_{S_2} \overline{q}_n \tau dS = 0$$

The terms in blue are counterbalanced, the second term of the first line (in red) can be ignored because it is second order. So:

$$\int_{\Omega} k\ grad(\tau)^T \cdot grad(\delta\tau)\ d\Omega\ + \int_{S_2} \overline{q}_n\ \delta\tau\ dS + \int_{S_3} h\left(\tau-\tau_f\right)\delta\tau dS = 0$$

To exploit this result, it is necessary to put the term of variation $\delta\tau$ in evidence. We thus carry out an integration by parts of the first term of the previous expression

$$-\int_{\Omega} \operatorname{div}\left(k\ grad(\tau)\right)\ \delta\tau\ d\Omega - \int_{S_1+S_2+S_3} q_n\ \delta\tau\ dS + \int_{S_2} \overline{q}_n\ \delta\tau\ dS + \int_{S_3} h\left(\tau-\tau_f\right)\delta\tau dS = 0$$

Since the variation $\delta\tau$ is zero on the boundary S_1. the expression is reduced to:

$$\left(-\int_{\Omega} \operatorname{div}\,(k\ grad(\tau))\ d\Omega + \int_{S_2} \left(\bar{q}_n - q_n\right)\ dS + \int_{S_3} \left(h\left(\tau - \tau_f\right) - q_n\right) dS \right)\ \delta\tau = 0$$

Because the variation is arbitrary, the factor must be zero, which makes it possible to write the equations that the thermal field must satisfy.

$$\operatorname{div}\,(k\ grad(\tau)) = 0\ in\ \Omega$$
$$\vec{n}\cdot(k\ grad\,\tau) = \bar{q}_n\ on\ S_2$$
$$\vec{n}\cdot(k\ grad\,\tau) = h\left(\tau - \tau_f\right)\ on\ S_3$$

In conclusion, we can replace the computation of the partial differential equations of the heat transfer problem by the expression of the stationnarity conditions of the functional:

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

Convective heat transfers

1. Partial differential equations
2. Variational methods
3. Finite element method (local level)
4. Finite element method (global level)
5. Examples
6. Prescribed heat flows
7. Matlab© procedures

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

Functional at element level:

$$I_{el} = \frac{1}{2}\int_{S_3} h\left(\tau - \tau_f\right)^2 dS$$

In this expression, h represents the convection coefficient. The fluid temperature τ_f is assumed to be uniform. We consider an element side which is a line segment of length L and thickness e so that $dS = e\, dx$ with x varying between 0 and L. We discretize the temperature as follows:

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

In matrix formalism, with T_0 et T_1 the nodal temperatures and $[T]$, the vector of nodal temperatures, we have:

$$T^T = \begin{bmatrix} T_0 & T_1 \end{bmatrix} \qquad \tau = \begin{bmatrix} 1-\frac{x}{L} & \frac{x}{L} \end{bmatrix}[T]$$

Replacing in the functional, we obtain:

$$I_{el} = \frac{1}{2}\int_0^L he\left(\left[1-\frac{x}{L} \quad \frac{x}{L}\right][T]-\tau_f\right)^2 dx$$

$$= \frac{1}{2}he\int_0^L \left\{[T]^T \begin{bmatrix} 1-\frac{x}{L} \\ \frac{x}{L} \end{bmatrix}\left[1-\frac{x}{L} \quad \frac{x}{L}\right][T] - 2\left[1-\frac{x}{L} \quad \frac{x}{L}\right][T]\tau_f + \tau_f^{\,2}\right\} dx$$

With a new definition of the nodal temperatures vector including the fluid temperature, we obtain with the notation $\tau_f = T_f$ where we assimilate the fluid temperature to this of a virtual node:

$$T_e = \left[T_0 \quad T_1 \quad T_f\right]^T$$

$$\frac{1}{2}heT_e^T \left(\int_0^L \begin{bmatrix} 1-\frac{x}{L} \\ \frac{x}{L} \\ -1 \end{bmatrix}\left[1-\frac{x}{L} \quad \frac{x}{L} \quad -1\right] dx\right) T_e$$

$$\frac{1}{2} h e T_e^T \int_0^L \begin{bmatrix} \left(1-\frac{x}{L}\right)^2 & \left(1-\frac{x}{L}\right)\frac{x}{L} & -\left(1-\frac{x}{L}\right) \\ \left(1-\frac{x}{L}\right)\frac{x}{L} & \left(\frac{x}{L}\right)^2 & -\frac{x}{L} \\ -\left(1-\frac{x}{L}\right) & -\frac{x}{L} & 1 \end{bmatrix} dx \; T_e$$

After integrating, we transform the functional into:

$$I_{el} = \frac{1}{12} h \; eL \; T_e^T \begin{bmatrix} 2 & 1 & -3 \\ 1 & 2 & -3 \\ -3 & -3 & 6 \end{bmatrix} T_e$$

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

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

As well as the pure conduction matrix, this one also is singular. It means that with or without convection it is necessary to fix at least one node.

Convective heat transfers

1. Partial differential equations

2. Variational methods

3. Finite element method (local level)

4. Finite element method (global level)

5. Examples

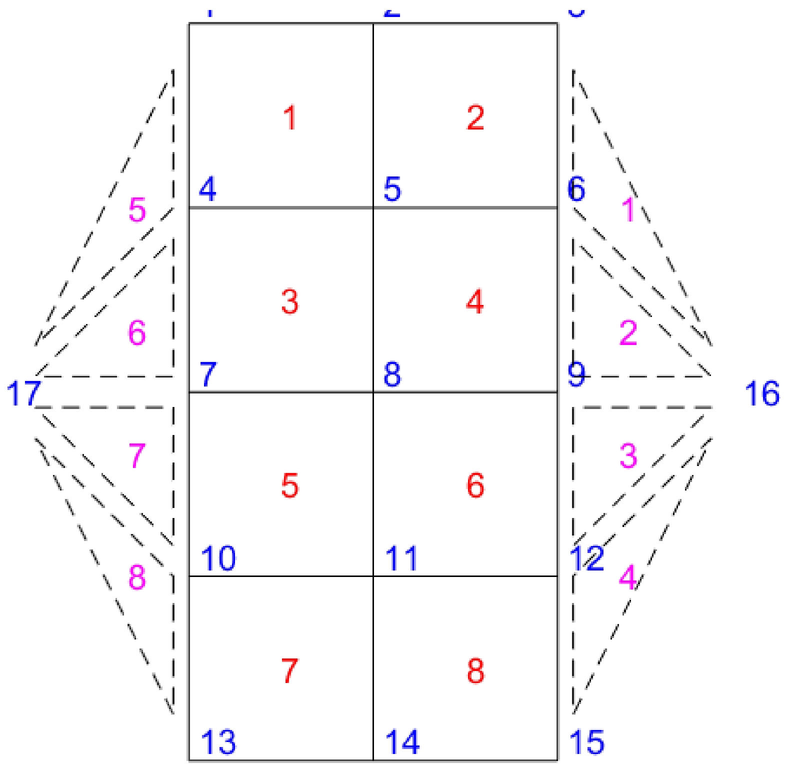
6. Prescribed heat flows

7. Matlab© procedures

Assembling the convection matrices into the global conduction matrix

This function performs the computation of the localization matrix of the convective elements on the sides of the domain (shown in blue). The process starts with the right side, then, the left side and finally the top side. The computation is performed only, it the number of the virtual node is given in the array *ntv* (third argument of the procedure)

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



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 together.

The second one involves extra degrees of freedom corresponding to the virtual convective nodes.

$$K = K_k + K_h$$

The convective nodes may be free or fixed, but the full set of nodal temperatures has to contain at least one fixed node.

The pure conduction matrix introduced in the Matlab© procedure *pp_Conduction.m*

```
K*6/(e*co(1)) = [

     4    -1     0    -1    -2     0     0     0     0     0     0     0     0     0     0
    -1     8    -1    -2    -2    -2     0     0     0     0     0     0     0     0     0
     0    -1     4     0    -2    -1     0     0     0     0     0     0     0     0     0
    -1    -2     0     8    -2     0    -1    -2     0     0     0     0     0     0     0
    -2    -2    -2    -2    16    -2    -2    -2    -2     0     0     0     0     0     0
     0    -2    -1     0    -2     8     0    -2    -1     0     0     0     0     0     0
     0     0     0    -1    -2     0     8    -2     0    -1    -2     0     0     0     0
     0     0     0    -2    -2    -2    -2    16    -2    -2    -2    -2     0     0     0
     0     0     0     0    -2    -1     0    -2     8     0    -2    -1     0     0     0
     0     0     0     0     0     0    -1    -2     0     8    -2     0    -1    -2     0
     0     0     0     0     0     0    -2    -2    -2    -2    16    -2    -2    -2    -2
     0     0     0     0     0     0     0    -2    -1     0    -2     8     0    -2    -1
     0     0     0     0     0     0     0     0     0    -1    -2     0     4    -1     0
     0     0     0     0     0     0     0     0     0    -2    -2    -2    -1     8    -1
     0     0     0     0     0     0     0     0     0     0    -2    -1     0    -1     4
]
```

K_k is a 15 x 15 matrix

The new conduction matrix computed in the Matlab© procedure *pp_Convection.m.* This matrix includes two additional virtual nodes to take into account the convection on the two vertical sides.

```
K*6/(e*co(1)) = [

     6    -1     0     0    -2     0     0     0     0     0     0     0     0     0     0     0    -3
    -1     8    -1    -2    -2    -2     0     0     0     0     0     0     0     0     0     0     0
     0    -1     6     0    -2     0     0     0     0     0     0     0     0     0     0    -3     0
     0    -2     0    12    -2     0     0    -2     0     0     0     0     0     0     0     0    -6
    -2    -2    -2    -2    16    -2    -2    -2    -2     0     0     0     0     0     0     0     0
     0    -2     0     0    -2    12     0    -2     0     0     0     0     0     0     0    -6     0
     0     0     0     0    -2     0    12    -2     0     0    -2     0     0     0     0     0    -6
     0     0     0    -2    -2    -2    -2    16    -2    -2    -2    -2     0     0     0     0     0
     0     0     0     0    -2     0     0    -2    12     0    -2     0     0     0     0    -6     0
     0     0     0     0     0     0     0    -2     0    12    -2     0     0    -2     0     0    -6
     0     0     0     0     0     0    -2    -2    -2    -2    16    -2    -2    -2    -2     0     0
     0     0     0     0     0     0     0    -2     0     0    -2    12     0    -2     0    -6     0
     0     0     0     0     0     0     0     0     0     0    -2     0     6    -1     0     0    -3
     0     0     0     0     0     0     0     0     0    -2    -2    -2    -1     8    -1     0     0
     0     0     0     0     0     0     0     0     0     0    -2     0     0    -1     6    -3     0
     0     0    -3     0     0    -6     0     0    -6     0     0    -6     0     0    -3    24     0
    -3     0     0    -6     0     0    -6     0     0    -6     0     0    -3     0     0     0    24
]
```

K is a 17 x 17 matrix

Band matrices methods are less efficient but they are replaced by the frontal methods

The convection matrix computed in the Matlab© procedure *pp_Convection.m*. This matrix includes two additional virtual nodes to take into account the convection on the two vertical sides. The conduction terms are removed from this global matrix.

```
K*6/(e*co(1)) = [
     2     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0    -3
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     2     0     0     1     0     0     0     0     0     0     0     0     0    -3     0
     1     0     0     4     0     0     1     0     0     0     0     0     0     0     0     0    -6
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     1     0     0     4     0     0     1     0     0     0     0     0     0    -6     0
     0     0     0     1     0     0     4     0     0     1     0     0     0     0     0     0    -6
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     0     0     4     0     0     1     0     0     0    -6     0
     0     0     0     0     0     0     1     0     0     4     0     0     1     0     0     0    -6
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     1     0     0     4     0     0     1    -6     0
     0     0     0     0     0     0     0     0     0     1     0     0     2     0     0     0    -3
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     1     0     0     2    -3     0
     0     0    -3     0     0    -6     0     0    -6     0     0    -6     0     0    -3    24     0
    -3     0     0    -6     0     0    -6     0     0    -6     0     0    -3     0     0     0    24
]
```

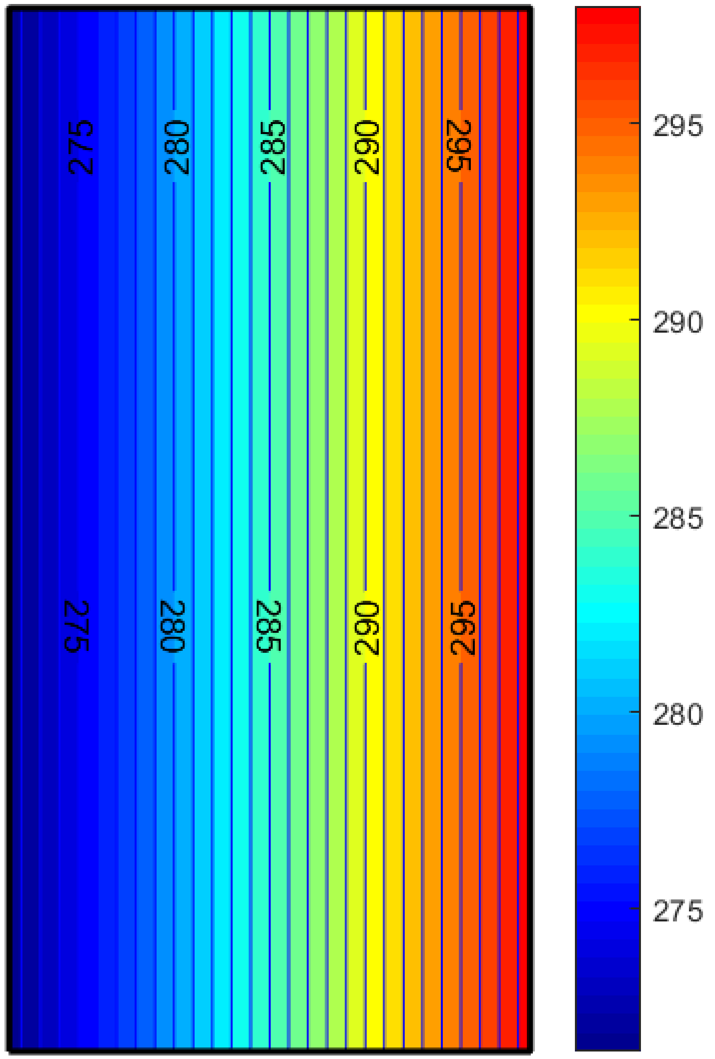
K_h is a 17 x 17 matrix

Convective heat transfers

This example is related to a domain with only convective boundaries. To get a non trivial solution, we have to fix at least 2 nodes with different temperatures in order to generate a temperature gradient. Here we chose to fix the 2 virtual ones related to the vertical sides (300 & 270 *K*).

```
pp_Convection
Thermal conductiv.  : 1  Wm-1K-1
CoKr.m conv. coeff. : 20 Wm-2K-1
Domain dim. w, y, t : 1 2 0.1 m
Biot number h w / k : 20
Delta T in the solid: 27.2727 K
Heat flows balance  : 5.4545 -5.4545  W
Virt. nodes temp.   : 300  270  270 K
Checking heat flows : -2.73 -2.73 -2.73 Wm-2
Checking temper.    : 271.4 271.4 298.6 298.6 K
Temp. virt. nodes   : 300  270 K
```

Multiplying the global conductive matrix by the temperatures gives the second member of the system of equations. Here, we obtain everywhere 0. except for the virtual nodes. Their flow are equal to 5.4545 & -5.4545 *W*. As the exact solution is a linear field, these values do not depend on the mesh, but only on the relative values of the convective and conductive coefficients : Biot number $\beta = h\,w\,/\,k$.

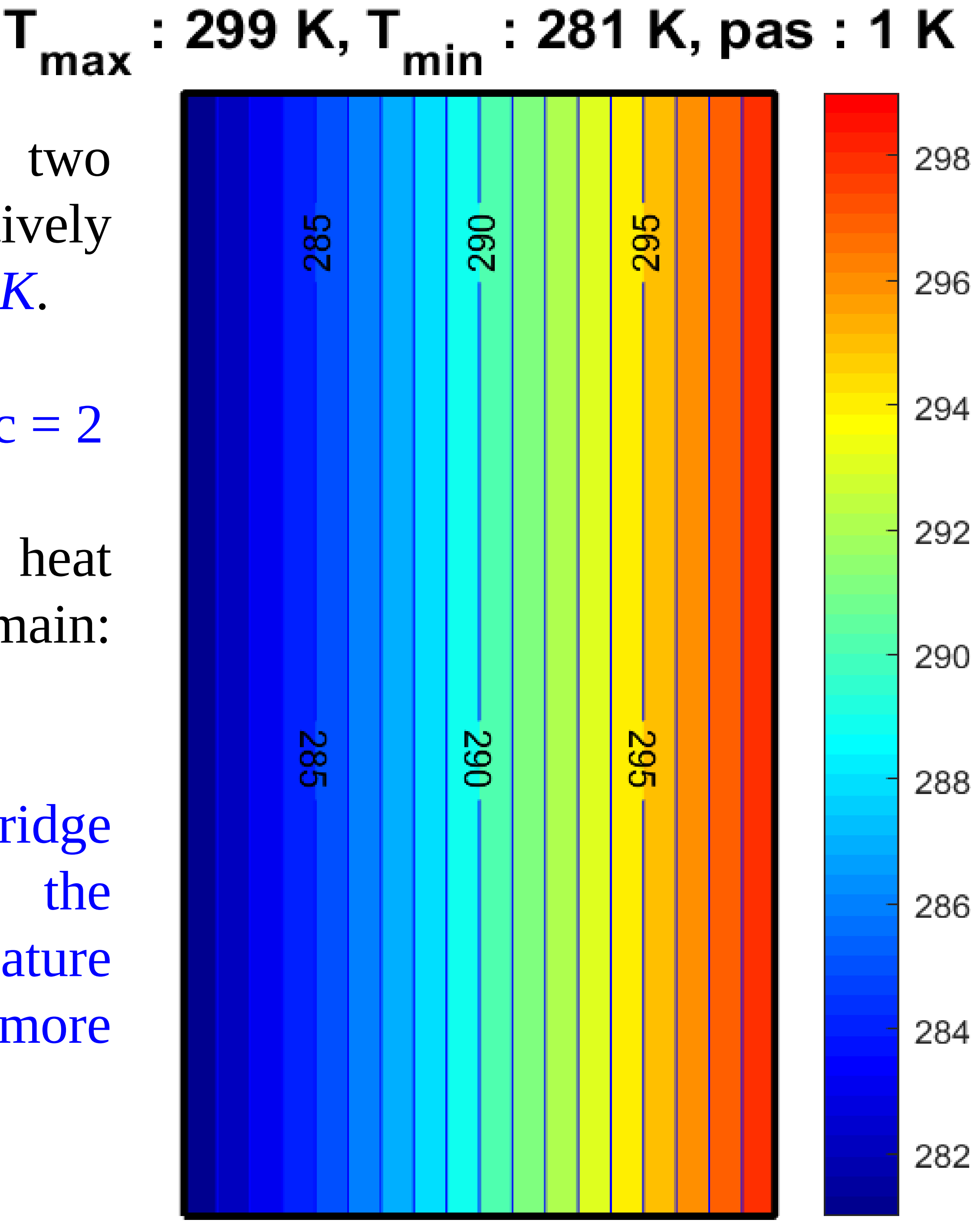


Heat flow between two enclosures with respectively temperatures of 300 and 280 *K*.

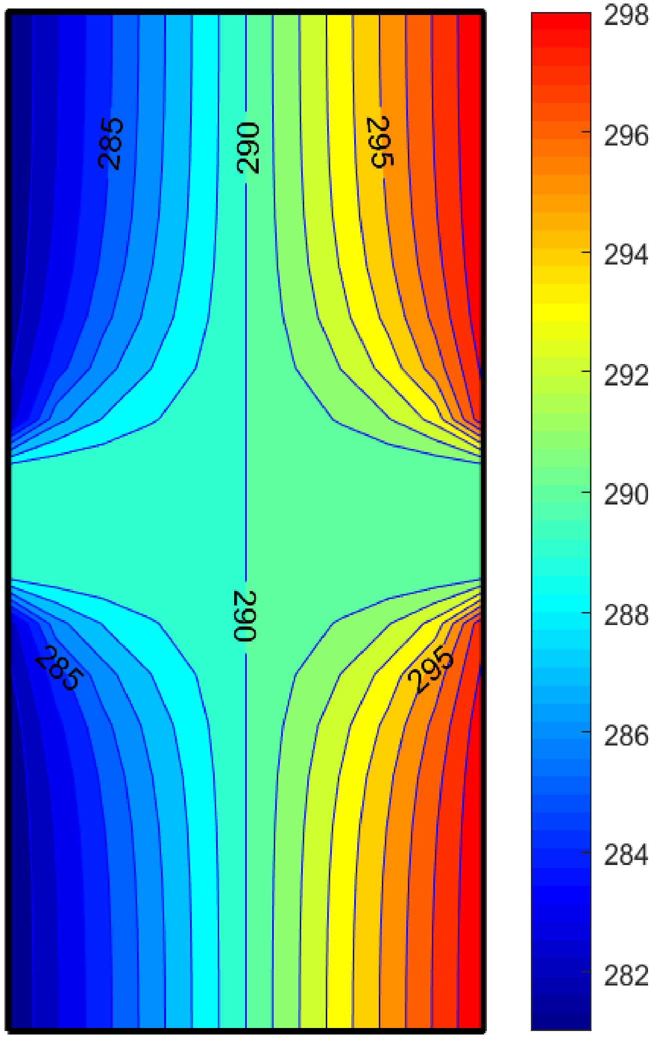
Adiabatic horizontal faces nfc = 2

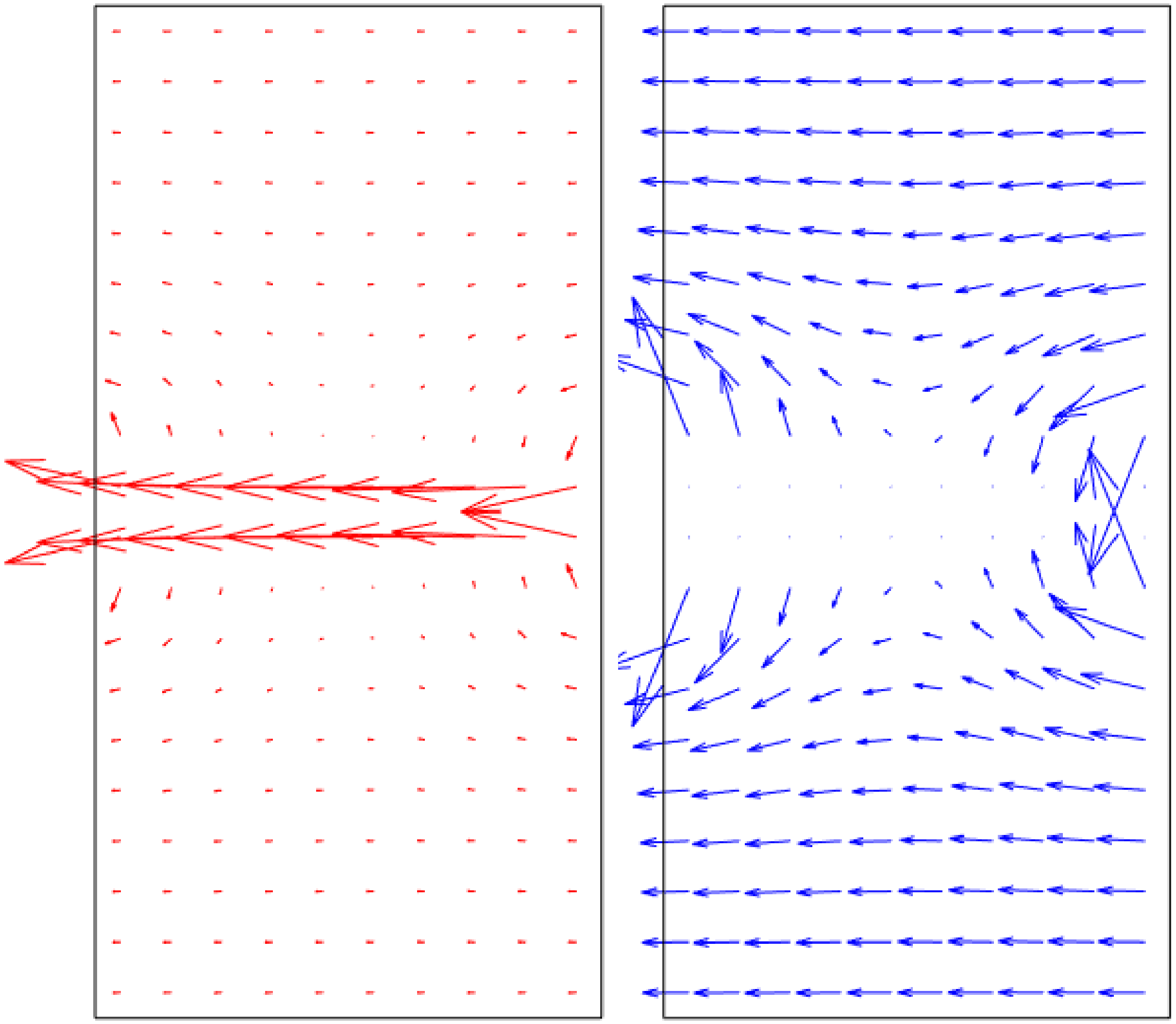
Temperature gradient and heat flow are constant in the domain: $q_x = -1.82\ Wm^{-2}$,

If we introduce a thermal bridge like in the following slide, the visualizations of temperature gradients & heat flows are more effective than isotherms.



Same problem, with horizontal thermal bridge (0.1 height around the horizontal median, with *k ratio* = 1000, *h* = 18 *Wm⁻²K⁻¹*)





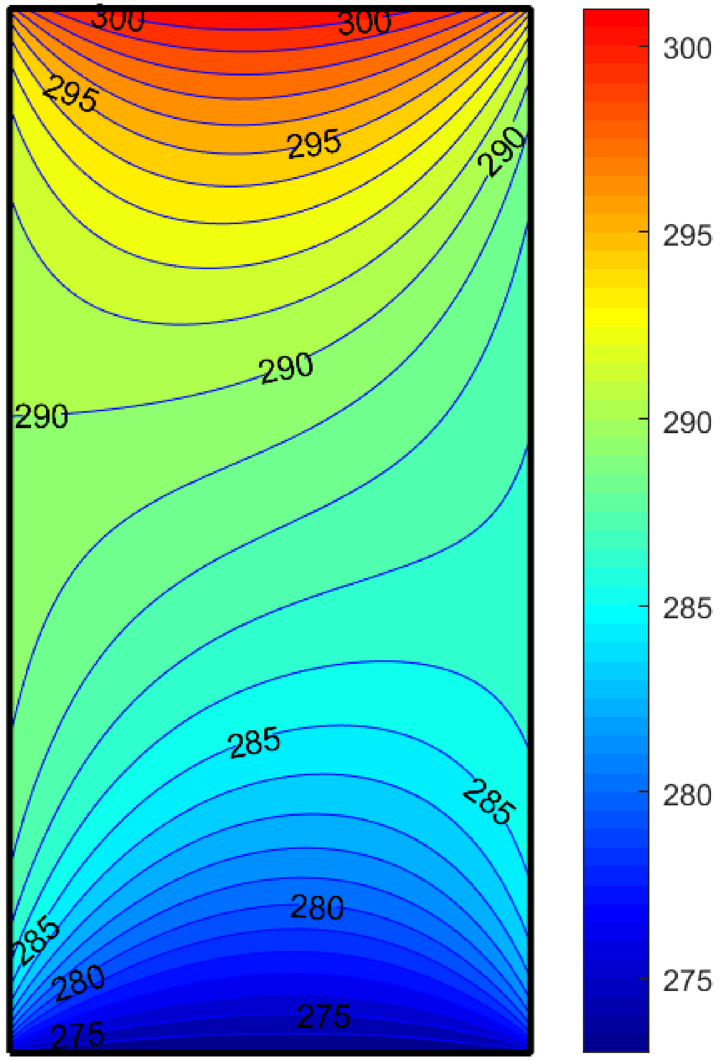
$$\vec{q} = -k \; grad \;\; \tau$$

Heat flow, max: 347 Wm^{-2}, mean: 49 Wm^{-2}

-grad τ: max: 59. Km^{-1}, mean: 17 Km^{-1}

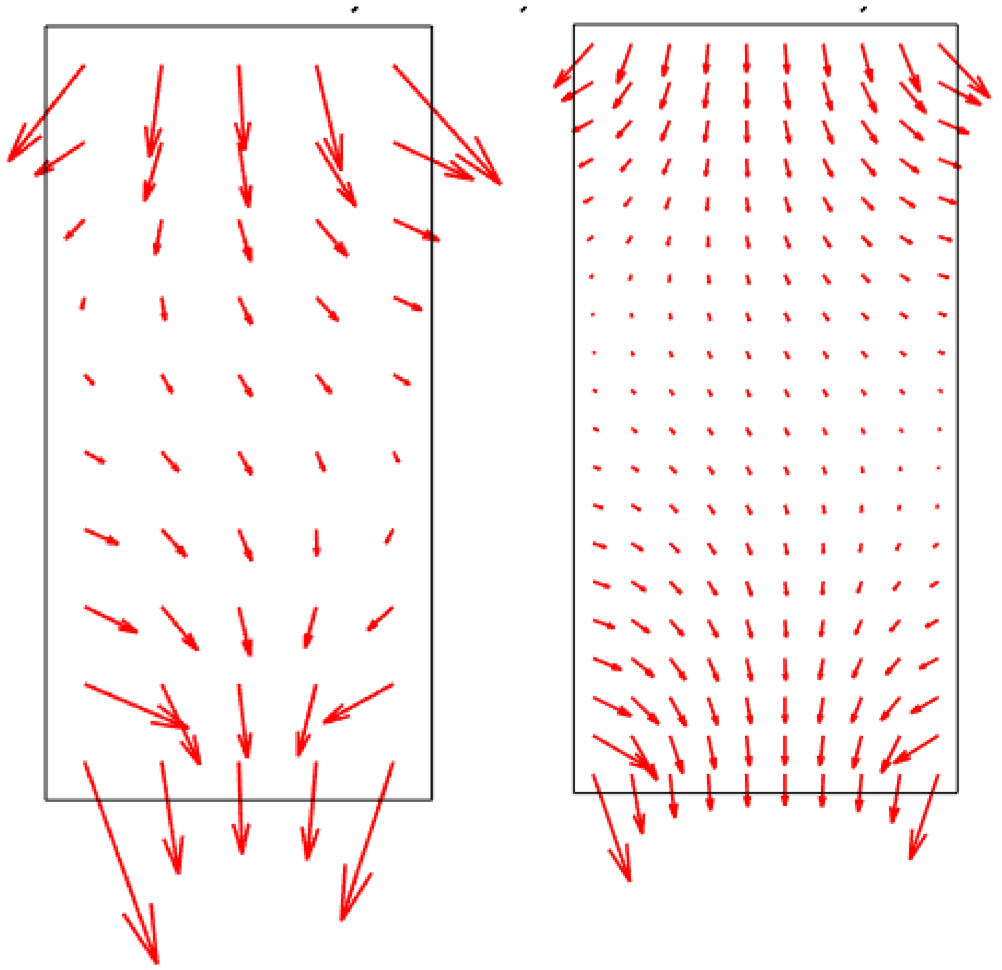
In the next example, we consider convection boundary conditions on 3 faces, the fourth face being fixed (273 K). The temperature of the virtual node related to the right vertical face is free.

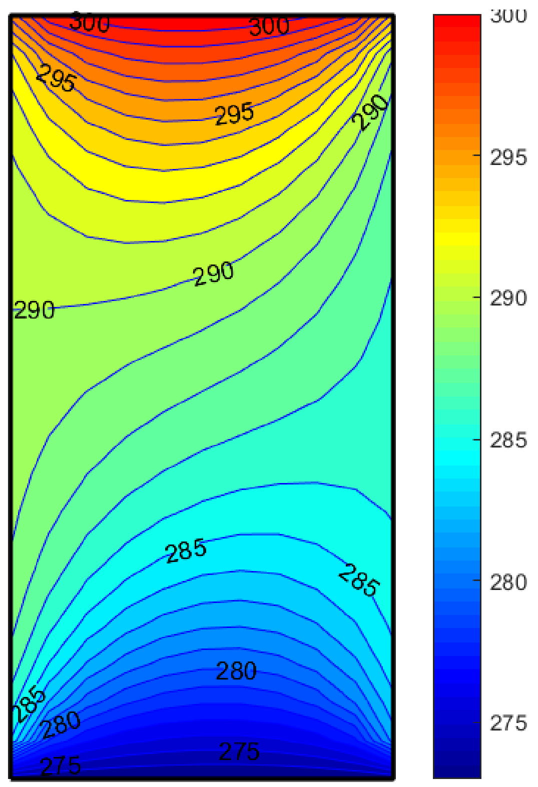
T_{max} : 301 K, T_{min} : 273 K, pas : 1 K



```
Conductivity coeff.: 2  Wm-1K-1
CoKr.m cnv. coeff. : 25 Wm-2K-1
Virtual conv. nodes: 5152  5153  5154
Heat convect. nodes: 2.73 2.73 6.74 W
Heat through base  : -12.1936 W
Size of convect. K : 5154  5154
Domain dim. w, h, t: 1 2 0.1 m
Mesh dimension     : 50 x 100
Virt. nodes temp.  : 0  290  303 K
Nfc                : 3
```

The computed temperature of the free virtual node of the left vertical side is: 286.6 *K.*





Same problem, but: 5 x 10 & 10 x 20 meshes
For both tests, the maximum heat flow = 120 Wm^{-2},
and the mean flow = 33 Wm^{-2}

Properties of a conductivity matrix including convective elements

The conductivity matrix is symmetric and semi-definite positive .

The sum of the terms of each column or of each line is zero.

For any thermal problem, at least 1 nodal temperature has to be specified in order to make the conductivity matrix definite positive.

Convective heat transfers

1. Partial differential equations

2. Variational methods

3. Finite element method (local level)

4. Finite element method (global level)

5. Examples

6. Prescribed heat flows

7. Matlab© procedures

Back to the heat conduction problem involving prescribed heat fluxes according to the functional presented in the first lecture.

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

Limiting the demonstration to 1 element edge, we can write that the second term of the above functional corresponds to the sum of products of heat flows (W) by temperatures (K) and we can write it as follows:

$$\int_{S_{2edge}} \bar{q}_n \tau \; dS_{el} = F_1 T_1 + F_2 T_2$$

If we express the edge temperature in term of blending functions:

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

We can write the discretized functional:

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

In matrix form,

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

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

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

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

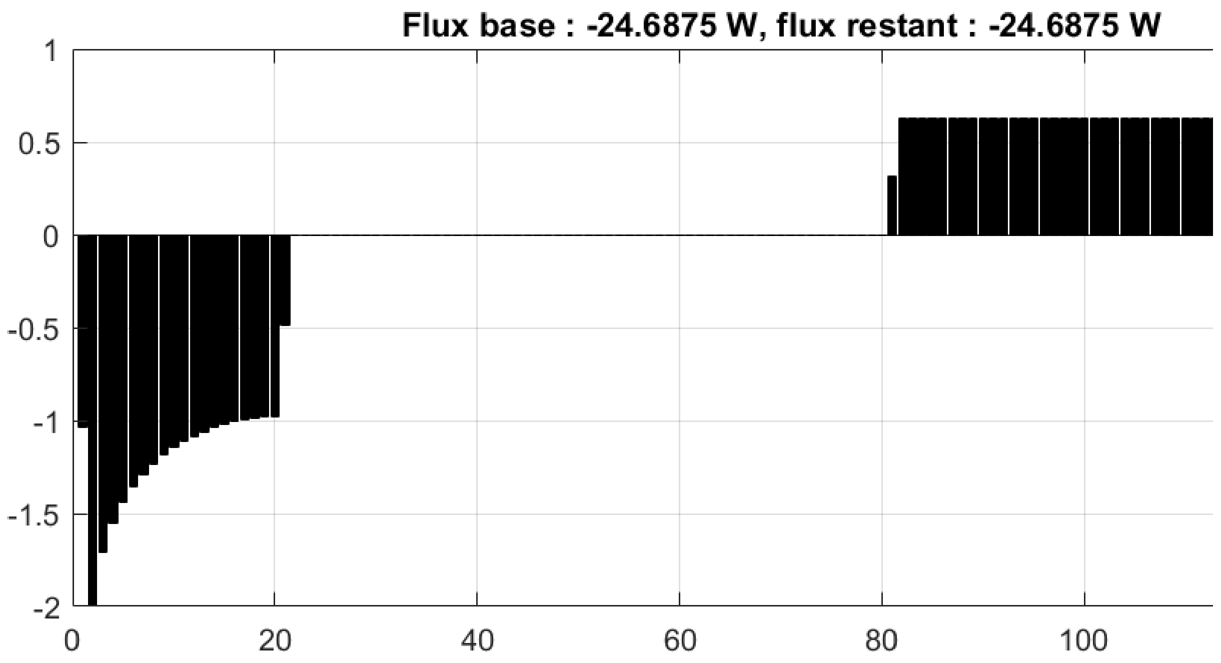
$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix}^T = \int_{S_{2edge}} \bar{q}_n \begin{bmatrix} \left(1 - \frac{x}{l}\right) & \frac{x}{l} \end{bmatrix} dS_{el}$$

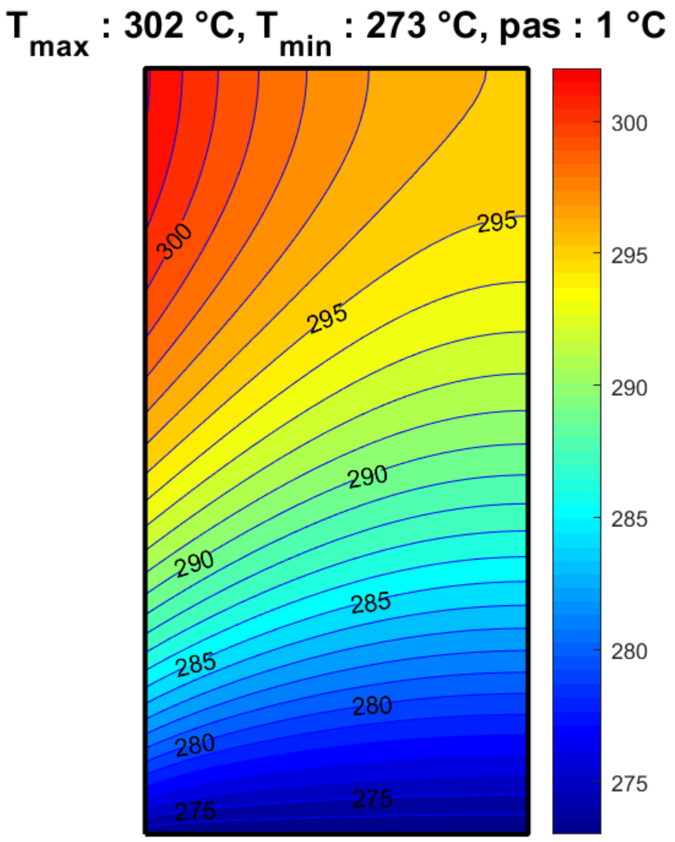
$$F_1 = \int_{S_{2edge}} \bar{q}_n \left(1 - \frac{x}{l}\right) dS_{el} \quad ; \quad F_2 = \int_{S_{2edge}} \bar{q}_n \frac{x}{l} dS_{el}$$

We note that the term dS_{el} is equal to edx, where e is the thickness and dx the differential on the side of the element. We have above, the general method to build the 2^{d} members of the heat transfer equations corresponding to imposed heat flows. If the prescribed heat flow is constant on the edge, for an edge of length l and a thickness e, we obtain:

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

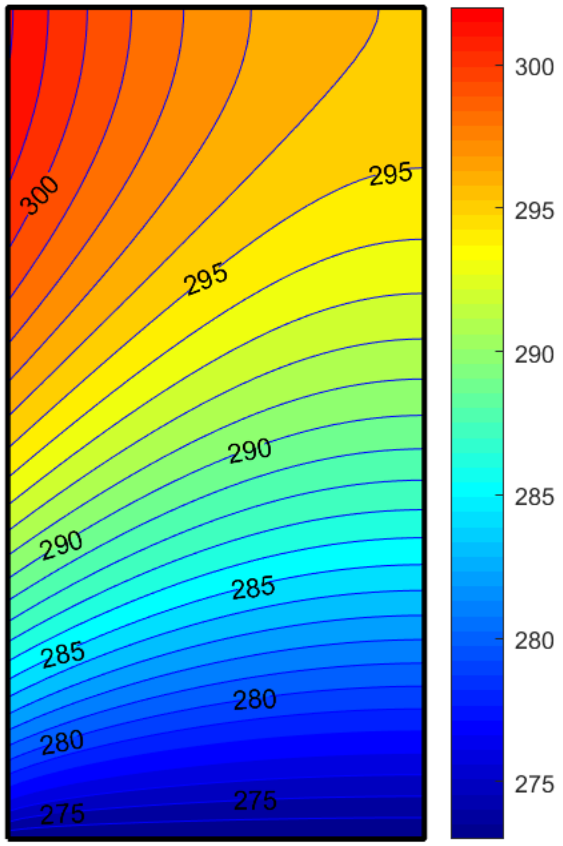
We treat a very simple case with imposed temperatures on the lower face (base) and a constant flow on the left vertical wall. The mesh has 20 x 40 elements. As the upper and right faces are adiabatic, the isotherms are orthogonal to them. The base temperature is fixed to 273 *K*. The total flow on the left side is equal to 25 *Watt*.



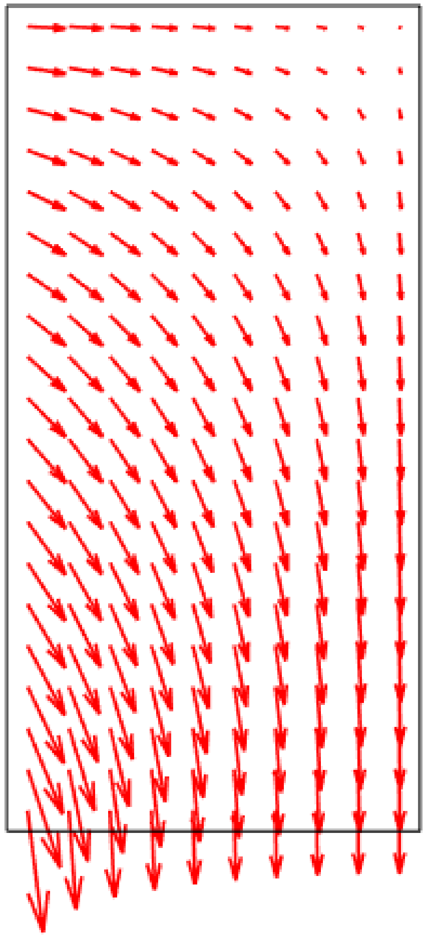


In computing the second member of the system of equations, there is ambiguity for the node of the lower left corner that receives a flux of 0.3125 *W*, but is fixed. This flow is therefore removed from the balance sheet in the above graph.

```
pp_conduction
conde.m uniform k    : 1 W/(mK)
- gradT element max  : 43.3, mean: 14.6 K/m
Max elem. heat flow  : 43.3, mean: 14.6 W/m2
Base temperature     : 273 K
Max  temperature     : 302 K
Mesh size            : 20 x 40
Fix. nod. 2 hor. fa.: 21
Diss tcaT*(K*tca)/2  : 254 WK
Cpu                  : 0.378 sec.
```



Elem. heat flow max : 38, mean: 15 W/m2



Convective heat transfers

1. Partial differential equations
2. Variational methods
3. Finite element method (local level)
4. Finite element method (global level)
5. Examples
6. Prescribed heat flows
7. Matlab© procedures

Matlab© procedure *pp_convection.m* to solve a convection problem

```
 1  hh   = 25;k = 2;nfc = 3;ta = [290; 290; 303];pge = [1;2;.1;50];tb = 273;
 2  th   = pge(3); %tt=tb+50;pf=[0;0;0];
 3  nx   = pge(4);ny = nx*2;nel = nx*ny;no = (nx+1)*(ny+1);%nf = nx+1;  % Mesh
 4  co   = f1_convection(nx,ny,k);                         % k measured in W/(m K)
 5  Kel  = th/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4];   % elem. K
 6  [K ] = f2_convection(nx,ny,pge(2),pge(3),hh,nfc);        % With convection K
 7  lK   = f2_conduction(nx,ny);
 8  for n=1:nel;for i=1:4;for j=1:4      % Assembling nel conduct. matrices Kel
 9            K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+co(n)*Kel(i,j);end;end;end
10  if nfc == 2                                              % 1. Adiabatic base
11      gco  =  K(1:no,no+1:no+size(K,1)-no)*ta(1:size(K,1)-no,1);
12      tca  = -K(1:no,1:no)\gco;
13             f3_conduction(nx,ny,tca(1:no));axis off      % Drawing isotherms
14      sm   = (K*[tca;ta(1:size(K,1)-no)])';% Compute second member of system
15      qxg  = th*hh*(ta(2)-min(tca));
16      qxc  = th*co(1)*(min(tca)-max(tca))/pge(1);
17      qxd  = th*hh*(max(tca)-ta(1));
18      t1   = (co(1)*ta(1)+(co(1)+hh*pge(1))*ta(2))/(2*co(1)+hh*pge(1));
19      t2   = ta(2)+ta(1)-t1;
20  end
21  if nfc == 3              % sequence 1: fixed base and 3 fixed virtual nodes
22  % gco  = K(1:no-nx-1,no-nx:size(K,1))*[ones(nx+1,1)*tb;...
23  %      ta(1:size(K,1)-no,1)];
24  % tca  = -K(1:no-nx-1,1:no-nx-1)\gco;
25  %         grisb(nx,ny,[tca;ones(nx+1,1)*tb]);axis off % Drawing the isot.
26  % sm   = (K*[tca;ones(nx+1,1)*tb; ta(1:size(K,1)-no,1)])';% 2d member sys.
27  %      ni  = size(sm,2)-nfc+1;
28  %      disp(['Heat convect. nodes : ',num2str(sm(ni:size(sm,2))),' W'])
29  %      disp(['Heat through base   : ',num2str(sum(sm(no-nx:no))),' W'])
30      ta(1) = 0.;nfc = 3; % sequence 2: fixed base and 2 fixed virtual nodes
31      K11   = K(1:no-nx-1,1:no-nx-1);     K12 = K(1:no-nx-1,no-nx:no);
32      K13   = K(1:no-nx-1,no+1);          K14 = K(1:no-nx-1,no+2:no+3);
33      K31   = K(no+1,1:no-nx-1);          K32 = K(no+1,no-nx:no);
34      K33   = K(no+1,no+1);               K34 = K(no+1,no+2:no+3);
35      T2    = ones(nx+1,1)*tb;            T4  = [ta(2); ta(3)];
36      tcb   = [K11 K13;K31 K33]\(-[K12 K14;K32 K34]*[T2;T4]);
37      tca   = [tcb(1:no-nx-1); ones(nx+1,1)*tb; tcb(size(tcb,1)); T4];
38      f3_conduction(nx,ny,tca(1:no));axis off          % Drawing the isotherms
39  end
40  disp(['Size of convect. K   : ',num2str(size(K))])
41  disp(['Domain dim. w, h, t  : ',num2str(pge(1:3,1)'),' m'])
42  disp(['Mesh dimension       : ',num2str(nx),' x ',num2str(nx*2)])
43  disp(['Virt. nodes temp.    : ',num2str(ta(1:size(ta,1))'),' K'])
44  if nfc == 2
45    disp(['Checking heat flows : ',num2str([qxg qxc qxd]),' Wm-2'])
46    disp(['Checking temper.     : ',num2str([t1 min(tca) t2 max(tca)]),' K'])
47  end
```

Lines 1 – 3: Data input

Line 4 : Definition of element conductivities (function *f1_convection.m*)

Line 5 : Conductivity matrix of a square element

Line 6 : global convection matrix (function *f2_convection.m*)

Line 7 : localization matrix (function *f2_conduction.m*)

Lines 8 – 9: assembling the global conductivity matrix

Line 11 : computation of the second member of the system

Line 12 : solution of the system

Line 13 : drawing the isotherms in the function *f3_conduction.m*

Lines 1 – 3 : 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.

Matlab© function *kf1_conduction.m* to handle non-uniform k

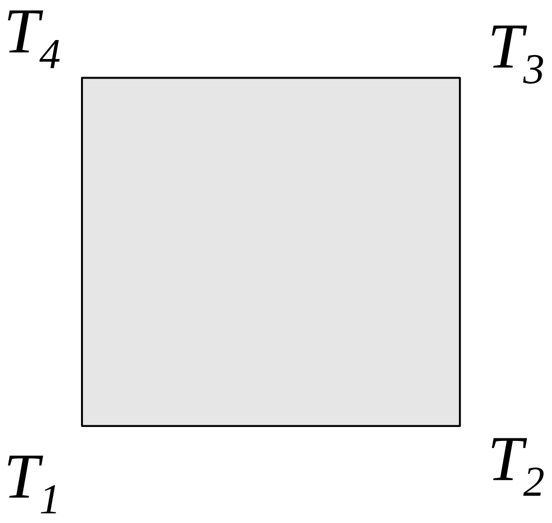
```
function [co] = kf1_conduction(nx,ny,k)% Treatment of non uniform conduct.
nel = nx*ny; % Number of element computed from mesh definition
fa  = 1 ;    % Ratio between the 2 conductivities, if 1, k is a cst
co  = ones(nel,1)*k;
        co(nx*nx+1:nx*nx+nx)  = k*fa;    % Second k on horizontal band 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
```

Line 4 : conductivity coefficients in the elements (function *kf1_conduction.m*)

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

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 6: Compute the conductivity matrix for convection (function *hf1.m*)

The element is a vertical or a horizontal one. Its length is L, its thickness is e and the convection coefficient is h ($Wm^{-2}K^{-1}$). The sequence of nodes start with the two real ones and finish with the virtual one.

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

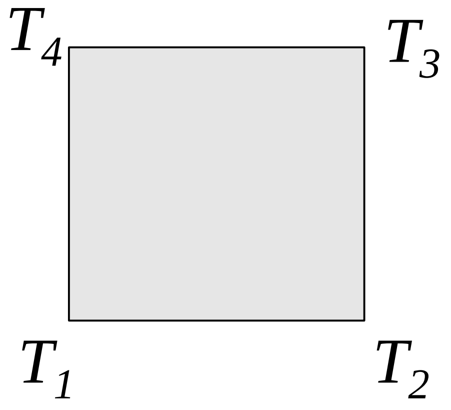
Matlab© function *hf1_convection.m* to assemble the convection matrix

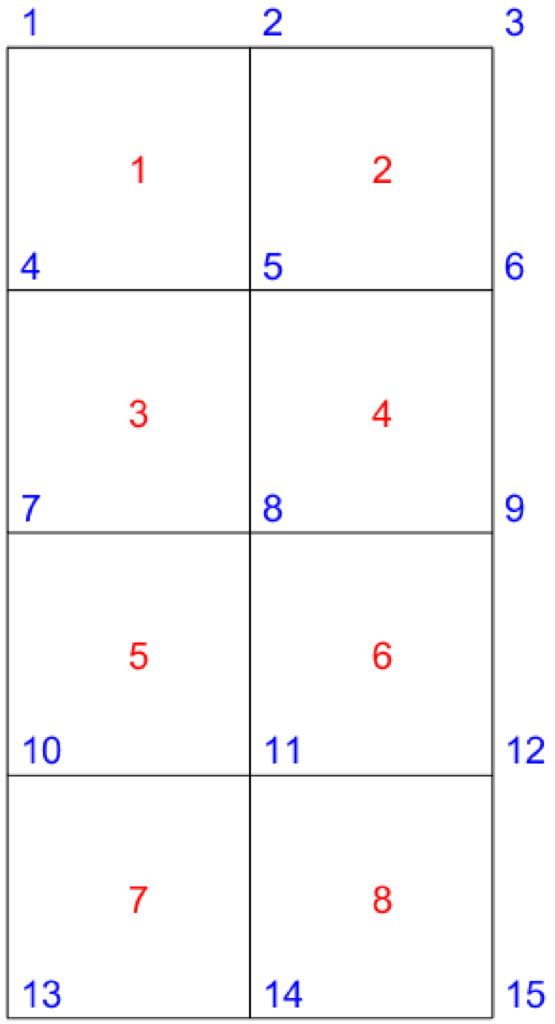
```matlab
function[K] = hf1_convection(nx,ny,ep,he,hh,nfc)
disp(['CoKr.m conv. coeff. : ',num2str(hh),' Wm-2K-1'])
Kelc  = [2 1 -3;1 2 -3;-3 -3 6]*hh*ep*he/ny/6;        % Elem. conv. matrix
if nfc ==2
Ntca  = (nx+1)*(ny+1)+nfc;                          % Mesh nx x ny elements
ntv2  = [ Ntca-1 Ntca];         % Numbering of the convective virtual nodes
lc    = f4_convection(nx,ny,ntv2);     % Local. 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];  % Numbering of the convective virtual nodes
lc    = f3_convection(nx,ny,ntv3); % Local. of the convection mat. 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    = f5_convection(nx,ny,ntv4); % Local. of convection matrices 4 sides
disp(['Virtual conv. nodes : ',num2str(ntv4)])
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
end
```

This function computes the convection matrices of a mesh *nx* x *ny* (arguments 1 and 2 of the function) for a domain of dimensions given by arguments 3 and 4 and a convection coefficient given by argument 5.

Line 7 : compute the localization matrix (function *kf1.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.

Lines 8 – 9 : 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 *kf1.m* (see previous slide).

Moreover, the coefficients of the element matrices *Kel* are multiplied by their conductivity coefficient *co (n)* (see *line 4*).

Lines 10 – 11 : fixing temperatures & solving the system

```
gco=K(1:no-nx-1,no-nx:size(K,1))*[ones(nx+1,1)*tb;ta(1:size(K,1)-no,1)];
tca  = -K(1:no-nx-1,1:no-nx-1)\gco;
```

In the proposed example, we fix the temperature of the base and of the virtual nodes. The solution of a problem including only imposed temperatures is performed as follows. Assuming that $[\bar{T}_2]$ is given, we have only to compute $[T_1]$ and the second member $[?] = [F_2]$. The computation of the incoming or outcoming heat flows through the fixations is simply obtained by multiplying the global conductivity matrix by the vector of temperatures.

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} T_1 \\ \bar{T}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ ? \end{bmatrix}$$

$$[T_1] = -[K_{11}]^{-1}[K_{12}][\bar{T}_2]$$

$$[F_2] = [K_{21}][T_1] + [K_{22}][\bar{T}_2]$$

Ligne 12 : drawing of the isotherms in the function *kf3_conduction.m*

Matlab© function *kf3_conduction.m* to draw isotherm lines

```
 1  function [ ] = kf3_conduction(nx,ny,tca)
 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  gap = 1;     % gap          = max(round((tmax-tmin)/20),.5); %  gap = 0.25;
10  colormap(kf4_conduction);                               % Color map definition
11  [CS,H]    = contourf(x,y,B,(0.:gap:max(tca)),'b');hold on;axis equal
12      clabel(CS,H,[275 280 285 290 295 300 305 310 315 320]);colorbar
13  plot  ([0 nx nx 0 0],[0 0 ny ny 0],'k','LineWidth',2);hold on;axis equal
14      title ([' T_m_a_x : ',num2str(round(max(tca))),' K, T_m_i_n : ',...
15  num2str(round(min(tca))),' K, step : ',num2str(gap),' K'],'fontsize',15);
16  axis off;end                                           % End isotherms drawing
```

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).

In line 9 of the function, the value of the intervals between successive isotherms can be adjusted. Let observe in line 10 the introduction of a particularly effective color bar (function *kf4_conduction.m*)

Matlab© function *hf2_convection.m* to build localizations of convective elements

```
function  [lc]=hf2_convection(nx,ny,ntv)% Loc. vectors for conv. on 3 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;                          % disp(['Number of conv. el. : ',num2str(ii)])
end
```

Matlab© function *hf3_convection.m* to build localizations of convective elements

```
function  [lc]=hf3_convection(nx,ny,ntv)% Loc. 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
```

Matlab© function *hf4_convection.m* to compute localization of convection matrices

```
function [lc]=hf4_convection(nx,ny,ntv)     % Loc. vectors 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
```

Matlab© function *hf5_convection.m* to compact matrices xyz, lK, lc.

```
function [sxyz,slK,slc]=hf5_convection(xyz,lK,lc)             % see cono.m
pn    = zeros(size(xyz,1),1);
for i = 1:size(lK,1);for j = 1:4;if lK(i,j) > 0;pn(lK(i,j))=1;end;end;end
for i = 1:size(lc,1);for j = 1:3;if lc(i,j) > 0;pn(lc(i,j))=1;end;end;end
ii    = 0;
for i = 1 : size(pn,1);if pn(i) == 1;ii=ii+1;pn(i )=ii;end;end
sxyz  = zeros(max(pn),3);
for i = 1:size(pn,1);if pn(i) > 0;sxyz(pn(i),:)=xyz(i,:);end;end
slK   = zeros(size(lK));
for i = 1:size(lK,1);for j=1:4;slK(i,j)=pn(lK(i,j));end;end
ii    = 0;
for i = 1:size(lc,1);if lc(i,1) > 0;ii=ii+1;end;end
slc   = zeros(ii,3);ii=0;
for i = 1:size(lc,1)
    if lc(i,1) > 0;ii = ii+1;for j = 1:3;slc(ii,j) = pn(lc(i,j));end;end;
end
end
```