

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 conductive heat transfers

Dual Analysis for Heat Conduction Problems by Finite Elements,
B. Fraeijs de Veubeke & M. Hogge, IJNME, 1972

A 66 line heat transfer finite element code to highlight the dual approach,
P. Beckers, B. Beckers, Computers & Mathematics with Applications,
Volume 70 Issue 10, November 2015, Pages 2401-2413.

A 33 line heat transfer finite element code, **P. Beckers & B. Beckers**,
Report Helio 010, 2016

Five Lectures on Finite Element Method Applied to Heat Transfer, **B. Beckers**, 2018. www.heliodon.net // daylight // documents

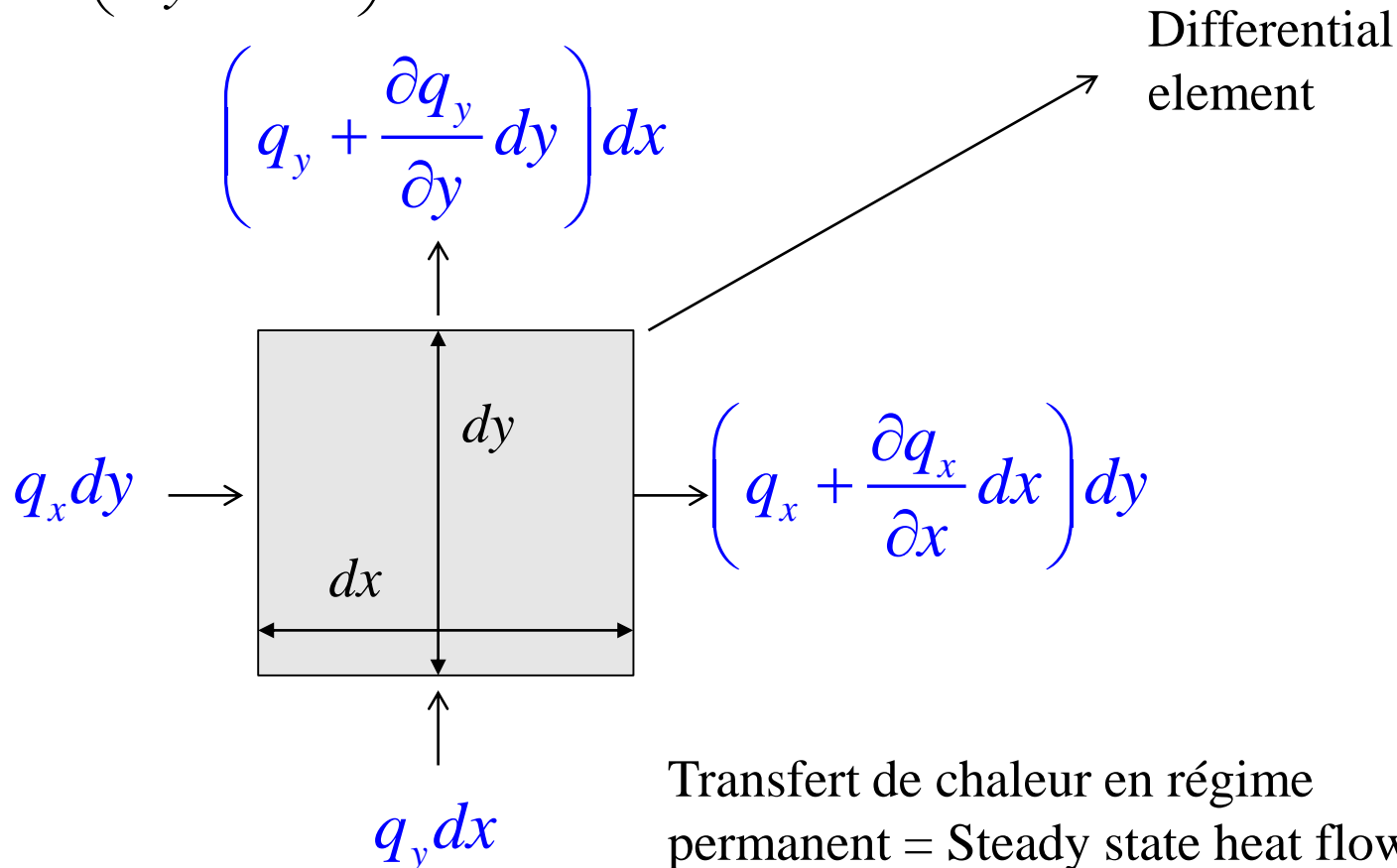
Conductive heat transfers

1. Partial differential equations
2. Variational methods
3. Rayleigh Ritz method
4. Finite element method (local level)
5. Finite element method (global level)
6. Matlab[®] procedures and examples
7. More about the heat flows

Local equilibrium equation for steady state heat flow

$$\left(q_y + \frac{\partial q_y}{\partial y} dy \right) dx + \left(q_x + \frac{\partial q_x}{\partial x} dx \right) dy - q_x dy - q_y dx = 0$$

$$= \left(\frac{\partial q_y}{\partial y} + \frac{\partial q_x}{\partial x} \right) dx dy = \text{div} \vec{q} \, dx dy = 0$$



Partial differential equations to which thermal quantities must satisfy

The Fourier equation links the **temperature** field τ (K) to the **heat flux** vector \vec{q} (Wm^{-2}). This flux is proportional to the temperature gradient and the thermal conductivity k ($Wm^{-1}K^{-1}$).

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

If there is no heat sink or source, the heat flow verifies the continuity equation (or equilibrium equation).

$$\text{div } \vec{q} = 0$$

When the conductivity coefficient is constant, if the continuity equation is expressed in terms of temperature, it becomes the Laplacian.

$$\Delta \tau = 0$$

The basic boundary conditions concern the temperatures fixed on the part S_1 of the boundary and the heat fluxes imposed on its complement S_2 . Their union represents the boundary of the domain.

$$\begin{aligned} \tau &= \bar{\tau} \quad \text{on } S_1 \\ \vec{n} \cdot \vec{q} &= \bar{q}_n \quad \text{on } S_2 \\ S_1 \cup S_2 &= S \end{aligned}$$

To compute the temperature distribution in a domain Ω , subjected to boundary conditions on the temperature and/or the heat flow, the following system of partial differential equations must be solved as well as the boundary conditions associated with it:

$$\text{div}(k \text{ grad } \tau) = 0 \text{ in } \Omega$$

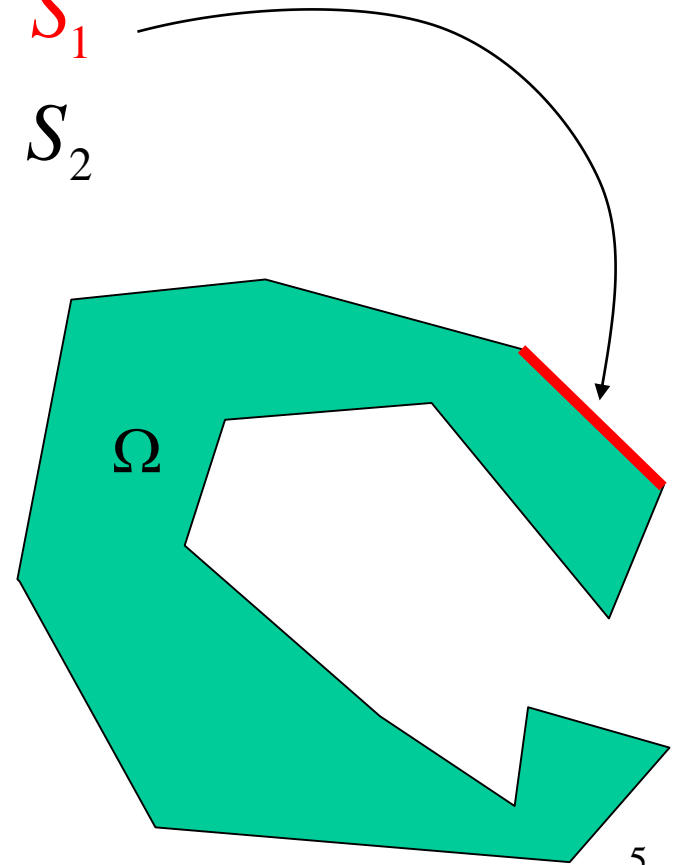
$$\tau = \bar{\tau} \text{ on } S_1$$

$$\vec{n} \cdot \vec{q} = \bar{q}_n \text{ on } S_2$$

In these equations, τ is the unknown temperature field expressed in K , and k , the thermal conductivity ($Wm^{-1}K^{-1}$).

The boundary of the domain is divided into 2 parts: S_1 where the temperature is imposed, S_2 where the normal heat flow is imposed.

A bar indicates that the concerned quantity is imposed.



Conductive heat transfers

1. Partial differential equations
2. Variational methods
3. Rayleigh Ritz method
4. Finite element method (local level)
5. Finite element method (global level)
6. Matlab[®] procedures and examples
7. More about the heat flows

By using the variations method, one shows that solving the system of partial differential equations is equivalent to look for the stationnarity conditions of the functional (expressed in *WK*)

$$\langle I(\tau) = \int_{\Omega} \frac{1}{2} k (\text{grad} \tau)^T \cdot \text{grad} \tau d\Omega + \int_{S_2} \bar{q}_n \tau dS \rangle \text{ minimum}$$

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

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

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

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

$$\int_{\Omega} \frac{1}{2} k (\text{grad}(\tau + \delta\tau))^T \cdot \text{grad}(\tau + \delta\tau) d\Omega - \int_{\Omega} \frac{1}{2} k (\text{grad} \tau)^T \cdot \text{grad} \tau d\Omega + \int_{S_2} \bar{q}_n (\tau + \delta\tau) dS - \int_{S_2} \bar{q}_n \tau dS = 0$$

By developing this expression, we obtain:

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

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

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

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

$$\int_{\Omega} k \text{grad}(\tau)^T \cdot \text{grad}(\delta\tau) d\Omega + \int_{S_2} \bar{q}_n \delta\tau dS = 0$$

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

$$-\int_{\Omega} \text{div} (k \text{grad}(\tau)) \delta\tau d\Omega - \int_{S_1+S_2} q_n \delta\tau dS + \int_{S_2} \bar{q}_n \delta\tau dS = 0$$

Théorème de flux-divergence

En analyse vectorielle, le **théorème de flux-divergence**, appelé aussi **théorème de Green-Ostrogradski**, affirme l'égalité entre l'intégrale de la divergence d'un champ vectoriel sur un volume dans \mathbb{R}^3 et le flux de ce champ à travers la frontière du volume (qui est une intégrale de surface).

L'égalité est la suivante :

$$\iiint_{\mathcal{V}} \vec{\nabla} \cdot \vec{F} \, dV = \oiint_{\partial\mathcal{V}} \vec{F} \cdot d\vec{S}$$

où :

- \mathcal{V} est le volume ;
- $\partial\mathcal{V}$ est la frontière de \mathcal{V} ;
- $d\vec{S}$ est le vecteur normal à la surface, dirigé vers l'extérieur et de norme égale à l'élément de surface qu'il représente ;
- \vec{F} est continument dérivable en tout point de \mathcal{V} ;
- $\vec{\nabla}$ est l'opérateur **nabla** ; $\vec{\nabla} \cdot \vec{F} = \operatorname{div} \vec{F}$.

Ce théorème découle du **théorème de Stokes** qui, lui-même, généralise le **second théorème fondamental de l'analyse**.

Integration by parts

We start by calculating the divergence of the product of a scalar by a vector.

$$\operatorname{div}(f \vec{g}) = \operatorname{grad} f \cdot \vec{g} + f \operatorname{div} \vec{g} \quad (1)$$

If this vector is the gradient of a scalar function τ , equation (1) becomes:

$$\operatorname{div}(f \operatorname{grad} \tau) = \operatorname{grad} f \cdot \operatorname{grad} \tau + f \Delta \tau \quad (2)$$

The Green-Ostrogradsky theorem applied to equation (1) gives:

$$\iiint_{\Omega} \operatorname{div}(f \vec{g}) d\Omega = \iint_S f \vec{g} \cdot \vec{n} dS \quad (3)$$

Applied to equation (2), it gives:

$$\iiint_{\Omega} \operatorname{div}(f \operatorname{grad} \tau) d\Omega = \iint_S f (\operatorname{grad} \tau) \cdot \vec{n} dS \quad (4)$$

The integrand of the left-hand side of equation (4) is replaced by the second member of (2):

$$\iiint_{\Omega} (\mathit{grad} f \cdot \mathit{grad} \tau + f \Delta \tau) d\Omega = \iint_S f (\mathit{grad} \tau) \cdot \vec{n} dS \quad (5)$$

Transfer the second left term of (5) in the right-hand side:

$$\iiint_{\Omega} \mathit{grad} f \cdot \mathit{grad} \tau d\Omega = \iint_S f (\mathit{grad} \tau) \cdot \vec{n} dS - \iiint_{\Omega} f \Delta \tau d\Omega \quad (6)$$

In (6), we replace the scalar function f by $\delta \tau$

$$\iiint_{\Omega} (\mathit{grad} \delta \tau) \cdot \mathit{grad} \tau d\Omega = \iint_S \delta \tau (\mathit{grad} \tau) \cdot \vec{n} dS - \iiint_{\Omega} \delta \tau \Delta \tau d\Omega \quad (7)$$

Using (7), we justify the integration by parts carried out 2 slides higher on the following expression:

$$\int_{\Omega} k \mathit{grad}(\tau)^T \cdot \mathit{grad}(\delta \tau) d\Omega + \int_{S_2} \bar{q}_n \delta \tau dS = 0$$

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

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

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

$$\operatorname{div} (k \operatorname{grad}(\tau)) = 0 \text{ in } \Omega$$

$$\vec{n} \cdot (k \operatorname{grad} \tau) = \bar{q}_n \text{ on } S_2$$

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

$$\langle I(\tau) = \int_{\Omega} \frac{1}{2} k (\operatorname{grad} \tau)^T \cdot \operatorname{grad} \tau \, d\Omega + \int_{S_2} \bar{q}_n \tau \, dS \rangle \text{ minimum}$$

Conductive heat transfers

1. Partial differential equations
2. Variational methods
3. Rayleigh Ritz method
4. Finite element method (local level)
5. Finite element method (global level)
6. Matlab[®] procedures and examples
7. More about the heat flows

$$\langle I(\tau) = \int_{\Omega} \frac{1}{2} k (\text{grad} \tau)^T \cdot \text{grad} \tau \, d\Omega + \int_{S_2} \bar{q}_n \tau \, dS \rangle \text{ minimum}$$

To find the minimum of this functional, one can use an approximation of the temperature field (method of Rayleigh - Ritz). The field τ is represented by a combination of test functions, each one is assigned a coefficient β_i which will be determined in the minimization process.

$$\tau = \sum_{i=1}^n \beta_i f_i(x, y, z)$$

We obtain a new functional:

$$\langle I(\beta) = \int_{\Omega} \frac{1}{2} k (\text{grad} \sum_{i=1}^n \beta_i f_i)^T \cdot \text{grad} \sum_{i=1}^n \beta_i f_i \, d\Omega + \int_{S_2} \bar{q}_n \sum_{i=1}^n \beta_i f_i \, dS \rangle$$

To compute the gradients, we use a matrix formalism.

$$\begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \\ \frac{\partial \tau}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_2}{\partial x} & \frac{\partial f_3}{\partial x} & \dots \\ \frac{\partial f_1}{\partial y} & \frac{\partial f_2}{\partial y} & \frac{\partial f_3}{\partial y} & \dots \\ \frac{\partial f_1}{\partial z} & \frac{\partial f_2}{\partial z} & \frac{\partial f_3}{\partial z} & \dots \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \end{bmatrix}$$

The approximate functional $I(\beta)$ depends on the parameters β_i .

$$\langle I(\beta) = \frac{1}{2} \int_{\Omega} k \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \end{bmatrix}^T \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_2}{\partial x} & \frac{\partial f_3}{\partial x} & \dots \\ \frac{\partial f_1}{\partial y} & \frac{\partial f_2}{\partial y} & \frac{\partial f_3}{\partial y} & \dots \\ \frac{\partial f_1}{\partial z} & \frac{\partial f_2}{\partial z} & \frac{\partial f_3}{\partial z} & \dots \end{bmatrix}^T \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_2}{\partial x} & \frac{\partial f_3}{\partial x} & \dots \\ \frac{\partial f_1}{\partial y} & \frac{\partial f_2}{\partial y} & \frac{\partial f_3}{\partial y} & \dots \\ \frac{\partial f_1}{\partial z} & \frac{\partial f_2}{\partial z} & \frac{\partial f_3}{\partial z} & \dots \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \end{bmatrix} d\Omega + \int_{S_2} \bar{q}_n \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \end{bmatrix}^T \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \dots \end{bmatrix} dS \rangle \text{ minimum}$$

Parameters β_i are get out of integrals that can be directly calculated.

$$\langle I(\beta) = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \end{bmatrix}^T \frac{1}{2} \int_{\Omega} k \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_2}{\partial x} & \frac{\partial f_3}{\partial x} & \dots \\ \frac{\partial f_1}{\partial y} & \frac{\partial f_2}{\partial y} & \frac{\partial f_3}{\partial y} & \dots \\ \frac{\partial f_1}{\partial z} & \frac{\partial f_2}{\partial z} & \frac{\partial f_3}{\partial z} & \dots \end{bmatrix}^T \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_2}{\partial x} & \frac{\partial f_3}{\partial x} & \dots \\ \frac{\partial f_1}{\partial y} & \frac{\partial f_2}{\partial y} & \frac{\partial f_3}{\partial y} & \dots \\ \frac{\partial f_1}{\partial z} & \frac{\partial f_2}{\partial z} & \frac{\partial f_3}{\partial z} & \dots \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \end{bmatrix}^T \int_{S_2} \bar{q}_n \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \dots \end{bmatrix} dS \rangle \text{ minimum}$$

$$\text{with } K = \int_{\Omega} k \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_2}{\partial x} & \frac{\partial f_3}{\partial x} & \dots \\ \frac{\partial f_1}{\partial y} & \frac{\partial f_2}{\partial y} & \frac{\partial f_3}{\partial y} & \dots \\ \frac{\partial f_1}{\partial z} & \frac{\partial f_2}{\partial z} & \frac{\partial f_3}{\partial z} & \dots \end{bmatrix}^T \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_2}{\partial x} & \frac{\partial f_3}{\partial x} & \dots \\ \frac{\partial f_1}{\partial y} & \frac{\partial f_2}{\partial y} & \frac{\partial f_3}{\partial y} & \dots \\ \frac{\partial f_1}{\partial z} & \frac{\partial f_2}{\partial z} & \frac{\partial f_3}{\partial z} & \dots \end{bmatrix} d\Omega \text{ and } G = \int_{S_2} \bar{q}_n \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \dots \end{bmatrix} dS$$

$$\langle I(\beta) = \frac{1}{2} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \end{bmatrix}^T [K] \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \end{bmatrix}^T [G] \rangle \textit{ minimum}$$

The functional has been transformed into a quadratic algebraic function of the parameters β_i . The computation of the derivatives with respect to the β_i is immediate. These derivatives must be null, which leads to the resolution of the linear system of equations:

$$[K] \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \dots \end{bmatrix} = -[G]$$

This formalism is extremely attractive and well suited to numerical computation because it only uses linear equation solving techniques. The only difficulty is that, on the boundary, S_I the approximate solution must satisfy exactly the conditions imposed on the temperatures. Moreover, since the test functions must cover the whole area, it is not easy, except in special cases, to find some that are representative of the exact solution.

Conductive heat transfers

1. Partial differential equations
2. Variational methods
3. Rayleigh Ritz method
4. Finite element method (local level)
5. Finite element method (global level)
6. Matlab[©] procedures and examples
7. More about the heat flows

This leads to imagining test functions that cover only a small part of the field. By doing so for all the partitions whose union covers the domain, it is easier to apply the Rayleigh-Ritz method to all these elements and then to add their contributions. These partitions constitute what is known as a finite element mesh.

The Rayleigh-Ritz method is therefore applied to one of these sub-domains (or finite element):

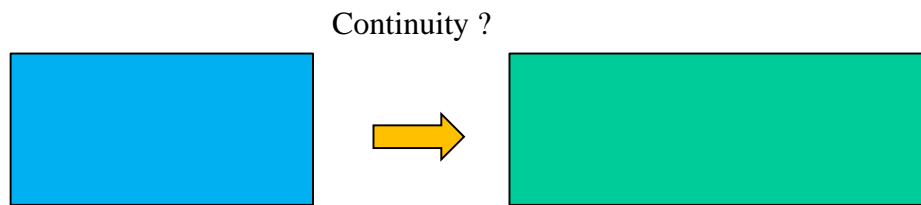
Functional at element level:

$$I_{el} = \int_{\Omega_{el}} \frac{1}{2} k (\mathit{grad} \tau)^T \cdot \mathit{grad} \tau \, d\Omega_{el} + \int_{S_{2el}} \bar{q}_n \tau \, dS_{el}$$

Finite element formulation of a temperature model by connection matrix approach to develop a rectangular element: we start exactly as before with a Rayleigh-Ritz like approximation:

$$\tau = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 xy$$

This trial function has the required properties (it is of first degree on the horizontal and vertical directions) however, we observe that the parameters β_i do not have the same physical meaning and are not easy to handle in order to ensure the continuity of the temperature field in a separation between 2 adjacent elements



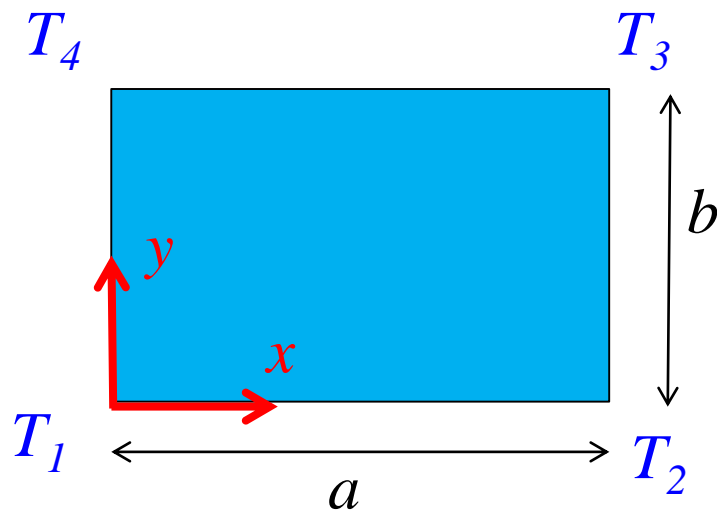
$$\tau = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 xy$$

$$\tau = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 xy$$

So, it is better to use more physical parameters like nodal temperatures. In the interface between 2 elements, it is sufficient to ensure that 2 local temperatures are identical on both sides in order to guarantee the continuity of two 1st degree fields.

So, we first write the definitions of the 4 corner temperatures:

$$\tau = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 xy$$



$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & a & 0 & 0 \\ 1 & a & b & ab \\ 1 & 0 & b & 0 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix}$$

The connection matrix is defined in such a way that it is non singular and thus, can be inverted.

$$\begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & a & 0 & 0 \\ 1 & a & b & ab \\ 1 & 0 & b & 0 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix} = [C] \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix}$$

$$\begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix} = [C]^{-1} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1/a & 1/a & 0 & 0 \\ -1/b & 0 & 0 & 1/b \\ 1/ab & -1/ab & 1/ab & -1/ab \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}$$

Replacing the coefficient β_i in the expression $\tau = \beta_1 + \beta_2x + \beta_3y + \beta_4xy$ we obtain the expression of the field τ in terms of **weight functions**, and **nodal temperatures** which can be obtained directly, using the techniques developed in CAD (see the Coons patch).

Finite element formulation of a temperature model by direct **weight** (or **shape** or **blending**) function approach (Coons patch):

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

The weight functions are adimensional. Each one is equal to 1 at one node and 0 at the others. They also have to respect the property of partition of unity which is mandatory to save the possibility to represent a constant temperature field.

The temperature gradient is computed in term of nodal temperatures.

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

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

$$\int_{\Omega_{el}} \frac{1}{2} k (\text{grad} \tau)^T \cdot \text{grad} \tau d\Omega_{el}$$

The first part of the functional defined above can now be written in compact form by using the so called **conductivity matrix** K , which is symmetric and singular (at least one temperature has to be fixed).

Using the vector of nodal temperatures $[T_1 \ T_2 \ T_3 \ T_4]$ the functional becomes:

Note the presence of the thickness e to ensure that we are working with volume and surface integrals.

$$[T]^T \iint_{\Omega_{el}} \frac{ke}{2} \begin{bmatrix} \frac{\partial \tau}{\partial x} & \frac{\partial \tau}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \end{bmatrix} dx dy [T] = \frac{1}{2} [T]^T [K] [T]$$

$$\begin{bmatrix} \frac{\partial \tau}{\partial x} & \frac{\partial \tau}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \end{bmatrix} = \frac{1}{a^2 b^2} \begin{bmatrix} y-b & x-a \\ b-y & -x \\ y & x \\ -y & a-x \end{bmatrix} \begin{bmatrix} y-b & b-y & y & -y \\ x-a & -x & x & a-x \end{bmatrix}$$

Details of the computation: core of the integral.

$$\frac{1}{h^2 l^2} \begin{bmatrix} y-b & x-a \\ b-y & -x \\ y & x \\ -y & a-x \end{bmatrix} \begin{bmatrix} y-b & b-y & y & -y \\ x-a & -x & x & a-x \end{bmatrix} =$$

$$\frac{1}{b^2 a^2} \begin{bmatrix} (y-b)^2 + (x-a)^2 & -(y-b)^2 - x(x-a) & y(y-b) + x(x-a) & -y(y-b) - (x-a)^2 \\ -(y-b)^2 - x(x-a) & (y-b)^2 + x^2 & yb(1-y) - x^2 & y(y-b) + x(x-a) \\ y(y-b) + x(x-a) & y(b-y) - x^2 & x^2 + y^2 & -y^2 + x(x-a) \\ -y(y-b) - (x-a)^2 & y(y-b) + x(x-a) & -y^2 + x(x-a) & y^2 + (x-a)^2 \end{bmatrix}$$

Integration of the monomial terms:

$$\begin{aligned} 1 &\rightarrow ab & x &\rightarrow \frac{a^2 b}{2} & y &\rightarrow \frac{ab^2}{2} \\ x^2 &\rightarrow \frac{a^3 b}{3} & y^2 &\rightarrow \frac{ab^3}{3} & xy &\rightarrow \frac{a^2 b^2}{4} \end{aligned}$$

Result of the integration:

$$\frac{1}{a^2 b^2} \begin{bmatrix} \frac{ab}{3}(a^2 + b^2) & \dots & \dots & \dots \\ \frac{ab}{3}\left(\frac{a^2}{2} - b^2\right) & \frac{ab}{3}(a^2 + b^2) & \dots & \dots \\ -\frac{ab}{6}(a^2 + b^2) & \frac{ab}{3}\left(\frac{b^2}{2} - a^2\right) & \frac{ab}{3}(a^2 + b^2) & \dots \\ \frac{ab}{3}\left(\frac{b^2}{2} - a^2\right) & -\frac{ab}{6}(a^2 + b^2) & \frac{ab}{3}\left(\frac{a^2}{2} - b^2\right) & \frac{ab}{3}(a^2 + b^2) \end{bmatrix}$$

This is the conductivity matrix of a rectangle of dimensions $a \times b$

$$K_{el} = \frac{ke}{6ab} \begin{bmatrix} 2(a^2 + b^2) & (a^2 - 2b^2) & -(a^2 + b^2) & (b^2 - 2a^2) \\ (a^2 - 2b^2) & 2(a^2 + b^2) & (b^2 - 2a^2) & -(a^2 + b^2) \\ -(a^2 + b^2) & (b^2 - 2a^2) & 2(a^2 + b^2) & (a^2 - 2b^2) \\ (b^2 - 2a^2) & -(a^2 + b^2) & (a^2 - 2b^2) & 2(a^2 + b^2) \end{bmatrix}$$

The conductivity of a square element is:

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

$$[T]^T \iint_{\Omega_{el}} \frac{ke}{2} \begin{bmatrix} \frac{\partial \tau}{\partial x} & \frac{\partial \tau}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial \tau}{\partial x} \\ \frac{\partial \tau}{\partial y} \end{bmatrix} dx dy [T] = \frac{1}{2} [T]^T [K_{el}] [T]$$

Because the conductivity matrix is based on a product of temperature gradients, the above expression vanishes when the temperature field is constant, for instance:

$$\text{if } [T_c]^T = \alpha [1 \ 1 \ 1 \ 1] \rightarrow [T_c]^T [K_{el}] [T_c] \equiv 0$$

$$[K_{el}] = \frac{ke}{6} \begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix}$$

The sum of each line of the $[K_{el}]$ matrix = 0.

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = K_{el} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \frac{ke}{6} \begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix}$$

This expression shows that the sum of the second members:

$$F_1 + F_2 + F_3 + F_4 = 0$$

for any set of nodal temperatures.

The sum of each column of the $[K_{el}]$ matrix = 0.

Conductive heat transfers

1. Partial differential equations
2. Variational methods
3. Rayleigh Ritz method
4. Finite element method (local level)
5. Finite element method (global level)
6. Matlab[©] procedures and examples
7. More about the heat flows

In each sub-domain or finite element j , a Rayleigh-Ritz was applied,

$$\tau_j = \sum_{i=1}^n T_{ij} f_{ij}(x, y, z)$$

Then, the contributions of all the finite elements of the domain were added and the discretized functional was obtained:

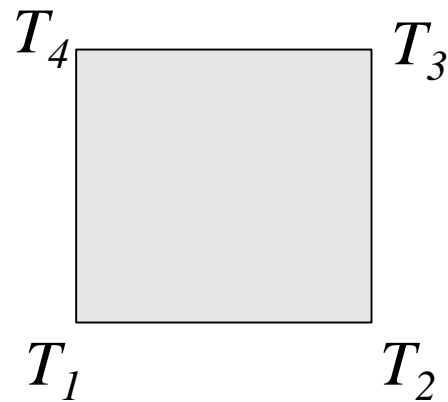
$$\langle I(T) = \sum_{j=1}^{nel} \left(\int_{\Omega_j} \frac{1}{2} k_j (\text{grad} \sum_{i=1}^n T_{ij} f_{ij})^T \cdot \text{grad} \sum_{i=1}^n T_{ij} f_{ij} d\Omega_j + \int_{S_{2j}} \bar{q}_n \sum_{i=1}^n T_{ij} f_{ij} dS_j \right) \rangle \text{ minimum}$$

After introducing the polynomial trial functions, we wrote it in matrix form:

$$\langle I(T) = \sum_{j=1}^{nel} [T_j]^T [K_j][T_j] + [T_j]^T [F_j] \rangle \text{ minimum}$$

It remains to express the continuity of the temperature field across the whole domain. For this, at each interface between two elements, it must be stated that the nodal temperatures are identical, which means to identifying them at the same node.

The nodal temperature vectors of the elements are linked to the global vector of temperatures by location matrices $[L_j]$.

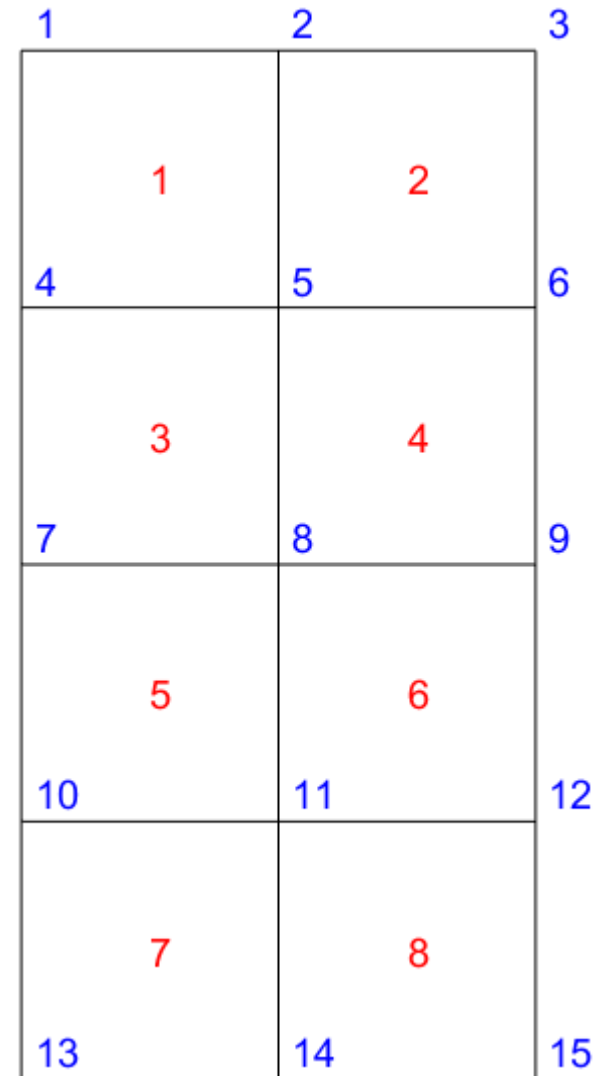


$$[T_j] = [L_j][T]$$

$$L_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$[L_4]^T [K_4][L_4]$$

It is easily verified that by performing this product, the coefficients of the conductivity matrix of element number 4 are placed at positions 8, 9, 6 and 5 of the domain conductivity matrix.



$$\sum_{j=1}^{nel} \{ [L_j]^T [K_j] [L_j] [T] \}$$

Example of element number 4

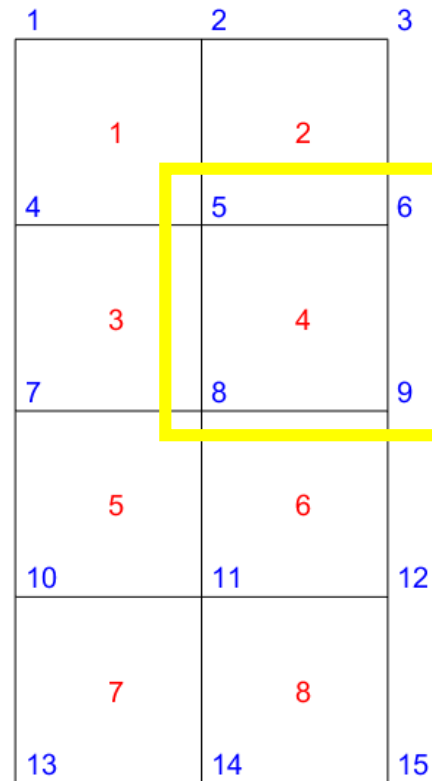
$$L_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$K_4 = \frac{ke}{6} \begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{bmatrix} = \begin{bmatrix} 0.6667 & -0.1667 & -0.3333 & -0.1667 \\ -0.1667 & 0.6667 & -0.1667 & -0.3333 \\ -0.3333 & -0.1667 & 0.6667 & -0.1667 \\ -0.1667 & -0.3333 & -0.1667 & 0.6667 \end{bmatrix}$$

$$k = 1, e = 1$$

$$L_4^T K_4 L_4 =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.6667 & -0.1667 & 0 & -0.1667 & -0.3333 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.1667 & 0.6667 & 0 & -0.3333 & -0.1667 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.1667 & -0.3333 & 0 & 0.6667 & -0.1667 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -0.3333 & -0.1667 & 0 & -0.1667 & 0.6667 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



As we have seen for element number 4 of the domain on the previous slide, the nodal temperature vectors of the elements are linked to the global vector of the temperatures of the domain by localization matrices $[L_j]$. We can therefore perform the summation which, simultaneously, ensures the continuity of the field by identification with the nodes of the domain.

$$[T_j] = [L_j][T] \rightarrow I(T) = \sum_{j=1}^{nel} \left([T]^T [L_j]^T [K_j][L_j][T] + [T]^T [L_j]^T [F_j] \right)$$

The next step is to get out of the sum, the vector $[T]$ in the product above

$$I(T) = [T]^T \left(\sum_{j=1}^{nel} \left\{ [L_j]^T [K_j][L_j][T] + [L_j]^T [F_j] \right\} \right)$$

Deriving with respect to the parameters $[T]$, we have:

$$\sum_{j=1}^{nel} \left([L_j]^T [K_j][L_j] \right) [T] = - \sum_{j=1}^{nel} [L_j]^T [F_j]$$

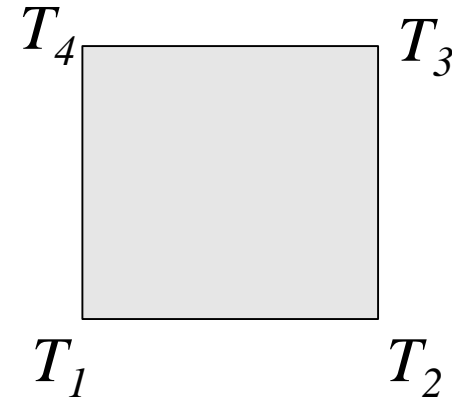
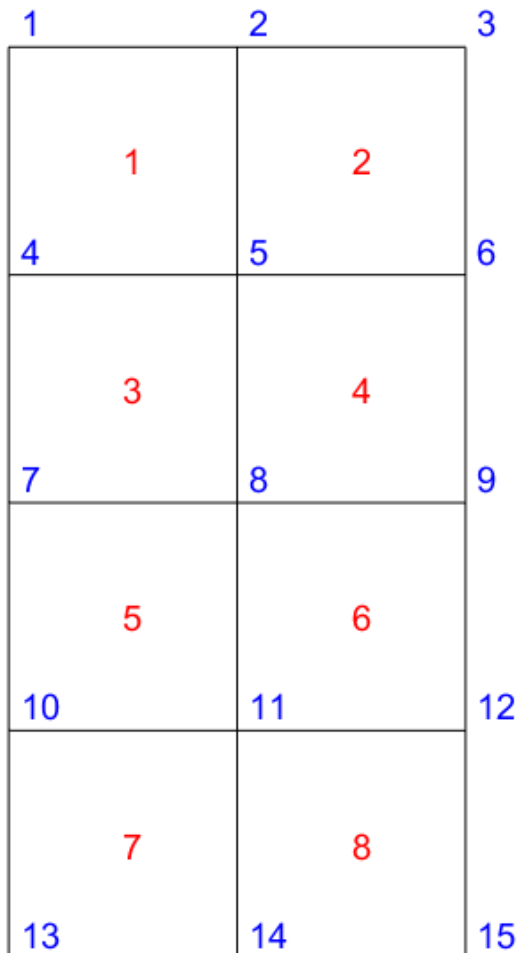
$$\text{with } [K] = \sum_{j=1}^{nel} [L_j]^T [K_j][L_j] \quad \text{and} \quad [F] = - \sum_{j=1}^{nel} [L_j]^T [F_j]$$

We thus reach the solution of a linear system whose matrix of coefficients is the matrix of global conductivity:

$$[K][T] = [F]$$

Using localization vectors

The element node sequence is always the same and must be respected 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.

$$[K] = \frac{ke}{6}$$

$\begin{bmatrix} 4 & -1 & -1 & -2 \\ -1 & 4 & -2 & -1 \\ -1 & -2 & 4 & -1 \\ -2 & -1 & -1 & 4 \end{bmatrix}$	<p style="text-align: center;">Element 1</p>
	<p>Element 1 : 4 5 2 1</p> <p>Element 2 : 5 6 3 2</p> <p>Element 3 : 7 8 5 4</p> <p>Element 4 : 8 9 6 5</p> <p>Element 5 : 10 11 8 7</p> <p>Element 6 : 11 12 9 8</p> <p>Element 7 : 13 14 11 10</p> <p>Element 8 : 14 15 12 11</p>

The conductivity matrix computed in the Matlab[®] procedure [pp_Base_conduction.m](#)

$K \cdot 6 / (t_h \cdot c_o(1)) = [$

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

]

In summary, the principle of the method is to split the domain into elements that, in general, all have the same simple form (for example, triangles or quadrangles in 2D, tetrahedrons or hexahedrons in 3D).

It is necessary to make sure that the discretized field is differentiable by piece (it is necessary to be able to calculate the gradients inside the elements), from where, the choice of polynomial functions.

The coherence of the solution implies the continuity of the fields (same degree on either side of the interfaces between neighboring elements) and the conformity of the mesh

A simple way to satisfy the conditions of continuity is to create a conformal mesh and to define the temperature fields of the elements according to the values of the temperature at the nodes of the mesh.

Properties of the conductivity matrix:

The conductivity matrix is symmetric and semi-definite positive (its determinant is never negative). The totaling of the terms of each column or of each line is zero.

For any heat transfer problem, the temperature has to be specified at least at one point in order to make the conductivity matrix definite positive.

The conductivity matrix of a square does not depend on its size.

Conductive heat transfers

1. Partial differential equations
2. Variational methods
3. Rayleigh Ritz method
4. Finite element method (local level)
5. Finite element method (global level)
6. Matlab[®] procedures and examples
7. More about the heat flows

Matlab[®] procedure *pp_Conduction.m* to solve a conduction problem

```

1  tb      = 270;tt = tb+50 ; pge  = [1;2;1;2];pf = [0;12.5;0]; k = 1;
2  qs      = pf(1)*pge(1)*pge(2);qw=pf(2)*pge(2)*pge(3);qe=pf(3)*pge(2)*pge(3);
3  nx      = pge(4);ny = nx*2;nel = nx*ny;no = (nx+1)*(ny+1);nf = nx+1;  % Mesh
4  co      = kf1_conduction(nx,ny,k);
5  Kel     = pge(3)/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4]; % elem. K
6  lK      = kf2_conduction(nx,ny); K = zeros(no,no);
7  for n = 1:nel;for i=1:4;for j=1:4      % Assembling nel conduct. matrices Kel
8      K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+co(n)*Kel(i,j);end;end;end
9  nl      = max(1,round(nx/5));if nl > nf;nl = nf;end; nu = no-2*nl;
10 K21     = K(nl+1:nu+nl , 1      : nl);K22 = K(nl+1:nu+nl , nl+1      : nu+nl);
11 K23     = K(nl+1:nu+nl , nu+nl+1 : nu+nl*2);ar=ones(nl,1);
12 tca     = [ar*tt;K22\(-K23*ar*tb-K21*ar*tt);ar*tb];
13         kf3_conduction(nx,ny,tca);axis off;          % Drawing the isotherms

```

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

```

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 4 : conductivity coefficients in the elements (function *kf1.m*)

Matlab[®] function *kf1_conduction.m* to handle non uniform k

```
1 function [co] = kf1_conduction(nx,ny) % Treatment of non uniform onductiv.
2 k = 1; % W/(m K)
3 nel = nx*ny; % Number of element computed from mesh definition
4 fa = 1 ; % Ratio between the 2 conductivities, if 1, k is a cst
5 co = ones(nel,1)*k;
6 co(nx*nx+1:nx*nx+nx) = k*fa; % Second k on horizontal band 1
7 if nx>2;co(nx*(nx-1)+1:nx*nx) = k*fa;end % Second k on horizontal band 2
8 disp(['Thermal conductiv. : ',num2str(k,'%0.3g'),' W/(m K)'])
9 disp(['Main & bridge k. : ',num2str([co(1) co(nx*nx+1)]),' W/(m K)'])
10 end
```

For a mesh of $nx \times ny$ 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..

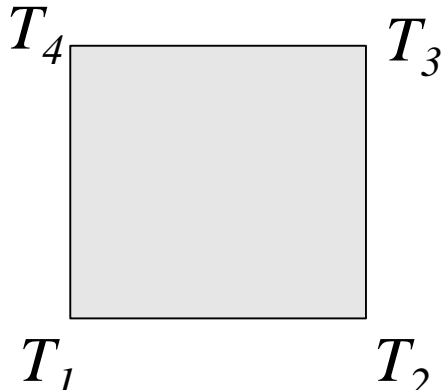
In a non-homogeneous medium, the conductivities may vary from one element to another, it is necessary to add to the previous procedure the two lines defining the elements operating with the second conductivity and modify the assembly of the elements. The conductivity coefficient acts as coefficients in the assembly of the global matrix (*line 8*). 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
```

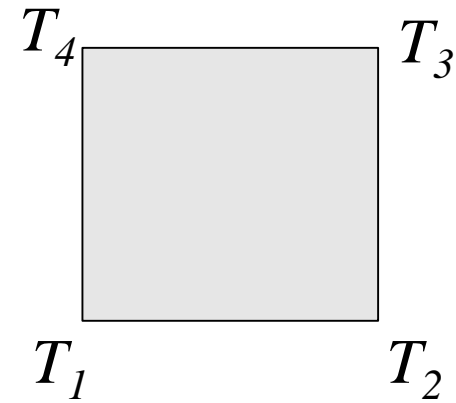
The variable `pge(3)` corresponds to the thickness of the element. In Matlab, the matrix coefficients are written in compact form.

```
Matlab® procedure kf2_conduction.m to build the localization matrix
1 function [lK]=kf2_conduction(nx,ny) % Localization matrix with fixed
2 base
3 nel = nx*ny;
4 nf = nx+1;
5 lK = zeros(nel,4); % Elements are numbered left - right, top - bottom
6 for j = 1:ny % Nodes are numbered left - right, top - bottom
7     for i = 1:nx
8         lK((j-1)*nx+i,1) = j*(nx+1) + i;
9         lK((j-1)*nx+i,2) = lK((j-1)*nx+i,1) + 1;
10        lK((j-1)*nx+i,3) = lK((j-1)*nx+i,1) - nx;
11        lK((j-1)*nx+i,4) = lK((j-1)*nx+i,1) - nf;
12    end
13 end
14 end
```


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

Line 6 : compute the localization matrix (function *kf2.m*)

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



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

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 7 – 8 : 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 *kf2.m* (see previous slide). Moreover, the coefficients of the element matrices Kel are multiplied by their conductivity coefficient $co(n)$ (see *line 4*).

Line 9 : Data for fixed temperatures

```
nl = 10;if nl > nf;nl = nf;end; nu = no-2*nl; % Size of the system
```

In the proposed example, we fix some temperatures on the horizontal sides, starting from opposite corners : left on the top, right in the bottom. The number nl of fixed temperatures is less or equal to the number of nodes on a horizontal line: $nx + 1$ (checked in the procedure). The solution of a problem involving only imposed temperatures is performed as follows. Assuming that $[T_2]$ is given, we have only to compute $[T_1]$. The computation of the incoming or outgoing heat flows through the fixations is simply obtained by multiplying the global conductivity matrix by the vector of temperatures.

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} 0 \\ ? \end{bmatrix}$$
$$[T_1] = -[K_{11}]^{-1} [K_{12}] [T_2]$$
$$[?] = [K_{21}] [T_1] + [K_{22}] [T_2]$$

Lines 10 – 12

: solution of the system

```
K21 = K(nl+1:nu+nl , 1 : nl);K22 = K(nl+1:nu+nl , nl+1 : nu+nl);  
K23 = K(nl+1:nu+nl , nu+nl+1 : nu+nl*2);ar=ones(nl,1);  
tca = [ar*tt;K22\(-K23*ar*tb-K21*ar*tt);ar*tb];
```

The system to be solved is the same as in the previous slide but, now, the fixed temperatures are split into 2 sets of dimensions nl , the first in the begin of the full matrix and the second at the end. The final size of the matrix to be inverted is nu (see the previous slide). As a consequence the global matrix is divided into 9 submatrices. The imposed temperatures are equal to $[T_3]$ in the bottom: $ar*tb$ and $[T_1]$ on the top: $ar*tt$. Note that ar is defined as a vector of dimension nl .

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

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

Line 13 : drawing of the isotherms in the function *kf3_conduction.m*

Matlab[®] function *kf3_conduction.m* to draw isotherm lines

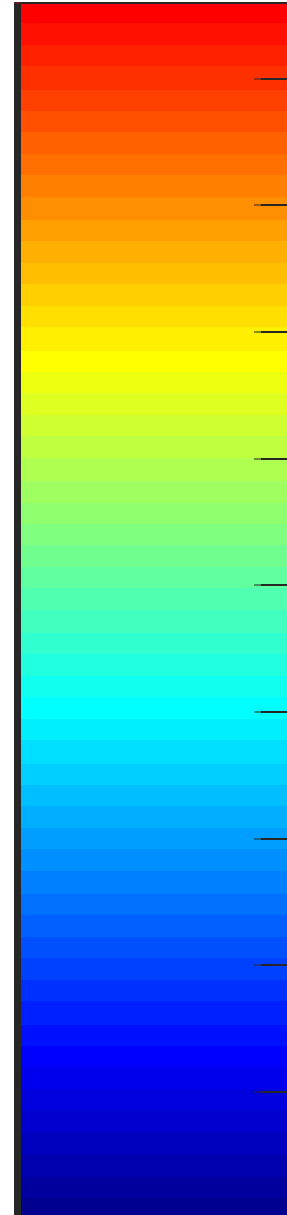
```
1 function [ ] = kf3_conduction(nx,ny,tca)
2 figure('Position',[1 1 600 512]);
3 my = ny+1;no=(nx+1)*(ny+1);
4 B = ones(my,nx+1)*tca(1);x = zeros(my,nx+1);y = zeros(my,nx+1);
5 for j = 1 : nx+1;for i = 1 : ny;x(i,j) = j-1; y(i,j) = my-i;end;end;
6 ii = 0;
7 for i = 1:ny;for j = 1:nx+1;ii = ii+1; 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 $nx \times ny$ (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 at line 10 for the particularly effective color bar (function *kf4_conduction.m*)

Matlab® function *kf4_conduction.m* to
install a convenient color map

```
1 function [bbr] = f4_conduction
2 bbr=[ 0 0 0.5625
3 0 0 0.6250
4 0 0 0.6875
5 0 0 0.7500
6 0 0 0.8125
7 0 0 0.8750
8 0 0 0.9375
9 0 0 1.0000
10 0 0.0625 1.0000
11 0 0.1250 1.0000
12 0 0.1875 1.0000
13 0 0.2500 1.0000
14 0 0.3125 1.0000
15 0 0.3750 1.0000
16 0 0.4375 1.0000
17 0 0.5000 1.0000
18 0 0.5625 1.0000
19 0 0.6250 1.0000
20 0 0.6875 1.0000
21 0 0.7500 1.0000
22 0 0.8125 1.0000
23 0 0.8750 1.0000
24 0 0.9375 1.0000
25 0 1.0000 1.0000
26 0.0625 1.0000 0.9375
27 0.1250 1.0000 0.8750
28 0.1875 1.0000 0.8125
29 0.2500 1.0000 0.7500
30 0.3125 1.0000 0.6875
31 0.3750 1.0000 0.6250
32 0.4375 1.0000 0.5625
33 0.5000 1.0000 0.5000
34 0.5625 1.0000 0.4375
35 0.6250 1.0000 0.3750
36 0.6875 1.0000 0.3125
37 0.7500 1.0000 0.2500
38 0.8125 1.0000 0.1875
39 0.8750 1.0000 0.1250
40 0.9375 1.0000 0.0625
41 1.0000 1.0000 0
42 1.0000 0.9375 0
43 1.0000 0.8750 0
44 1.0000 0.8125 0
45 1.0000 0.7500 0
46 1.0000 0.6875 0
47 1.0000 0.6250 0
48 1.0000 0.5625 0
49 1.0000 0.5000 0
50 1.0000 0.4375 0
51 1.0000 0.3750 0
52 1.0000 0.3125 0
53 1.0000 0.2500 0
54 1.0000 0.1875 0
55 1.0000 0.1250 0
56 1.0000 0.0625 0
57 1.0000 0 0];
58 end
```

Matlab® function *kf4_conduction.m* for
computing a color bar of 56 colors



Matlab[®] procedure *pp_Conduction.m* to solve a conduction problem

```

1  tb      = 270;tt = tb+50; pge  = [1;2;1;50];pf = [0;12.5;0];
2  qs      = pf(1)*pge(1)*pge(2);qw=pf(2)*pge(2)*pge(3);qe=pf(3)*pge(2)*pge(3);
3  nx      = pge(4);ny = nx*2;nel = nx*ny;no = (nx+1)*(ny+1);nf = nx+1;  % Mesh
4  co      = f1_conduction(nx,ny);
5  Kel     = pge(3)/6*[4 -1 -2 -1;-1 4 -1 -2;-2 -1 4 -1;-1 -2 -1 4]; % elem. K
6  lK      = f2_conduction(nx,ny); K = zeros(no,no);
7  for n = 1:nel;for i=1:4;for j=1:4      % Assembling nel conduct. matrices Kel
8          K(lK(n,i),lK(n,j))=K(lK(n,i),lK(n,j))+co(n)*Kel(i,j);end;end;end
9  nl      = 10;if nl > nf;nl = nf;end; nu  = no-2*nl;      % Size of the system
10 K21     = K(nl+1:nu+nl , 1      : nl);K22 = K(nl+1:nu+nl , nl+1      : nu+nl);
11 K23     = K(nl+1:nu+nl , nu+nl+1 : nu+nl*2);ar=ones(nl,1);
12 tca     = [ar*tt;K22\(-K23*ar*tb-K21*ar*tt);ar*tb];
13 f3_conduction(nx,ny,tca);axis off;  % Drawing the isotherms on the domain

```

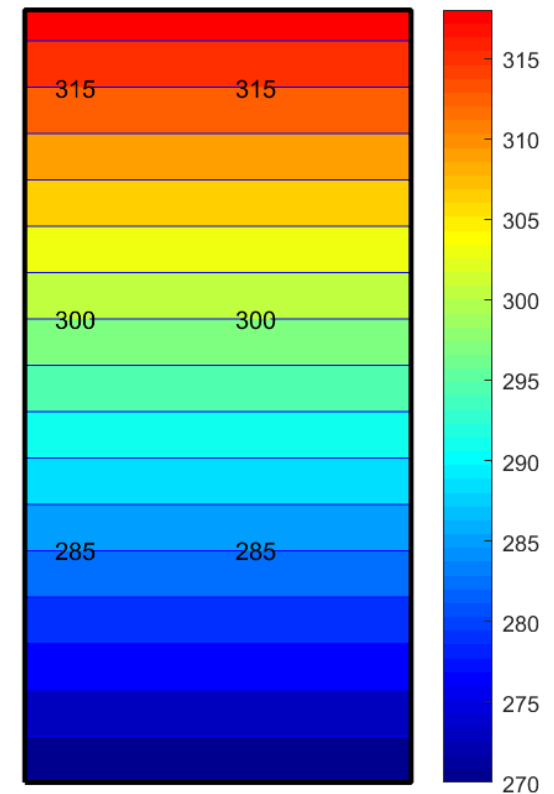
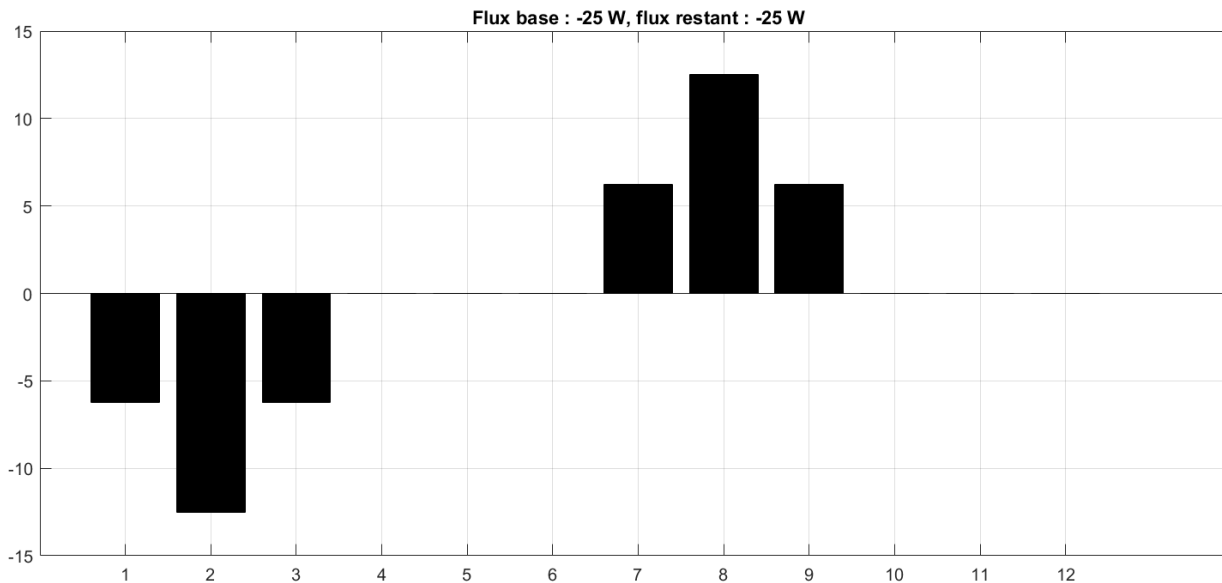
Procedure for thermal conduction problem which involves only imposed temperatures

- Lines 1 – 3 : Data input
- Line 4 : Definition of element conductivities (function *conde.m*)
- Line 5 : conductivity matrix of a square element
- Line 6 : localization matrix (function *loca.m*)
- Lines 7 – 8 : assembling the global conductivity matrix
- Line 9 : prescribed temperatures specification
- Lines 10 – 12 : solution of the system
- Line 13 : drawing the isotherms in the function *grisb.m*

By using a 2 x 4 mesh and imposing the temperatures of 3 nodes on both horizontal faces, we obtain as expected a solution showing a constant vertical temperature gradient.

When calculating the second members of the system of equations, we find the same results on the horizontal faces. The vertical gradient being 25 *K/m*, the flux on these faces are equal to 25 *W*.

$T_{\max} : 320 \text{ }^\circ\text{C}, T_{\min} : 270 \text{ }^\circ\text{C}, \text{pas} : 3 \text{ }^\circ\text{C}$



In this example, the temperatures are imposed on 1 segment of the upper and lower faces: 320 K on the upper face and 270 K on the base. The mesh is 50 x 100. The number of DOF is equal to 5131. The steps between isotherms = 1 K .

As expected, the isotherms are orthogonal to the walls, except in the two zones where the temperatures are imposed.

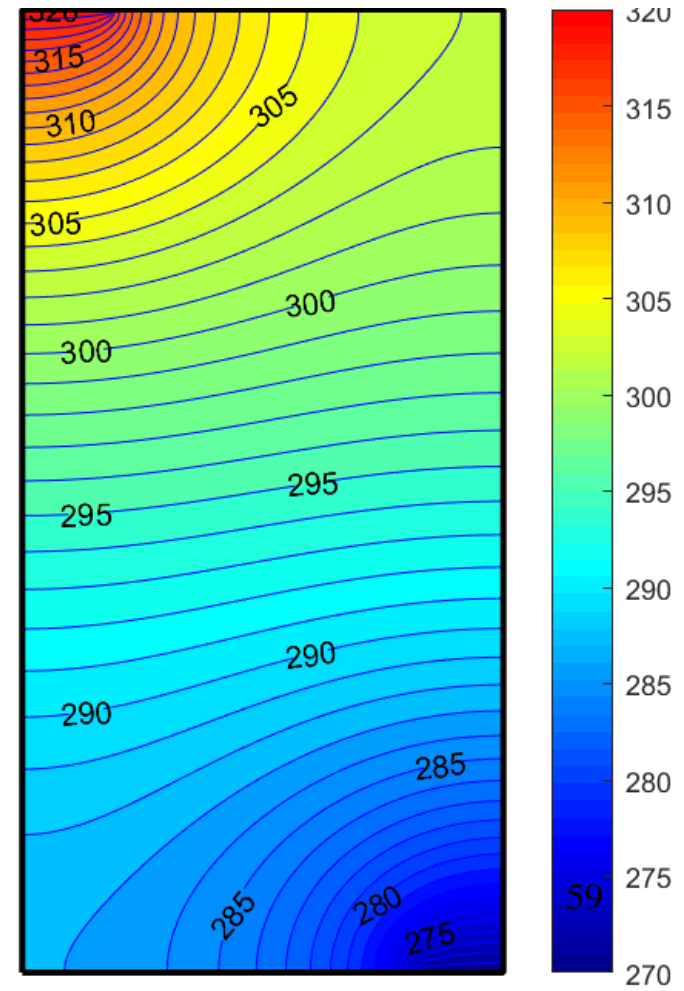
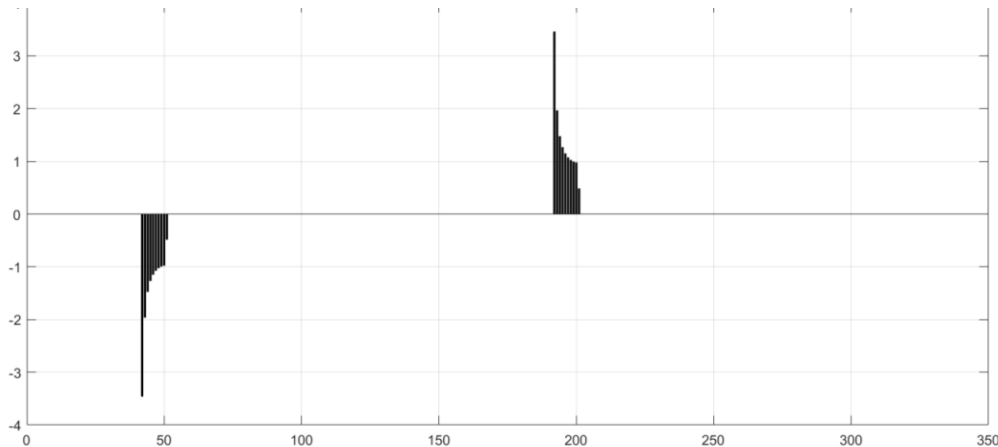
In this example, the temperatures were imposed on 10 nodes of the upper and lower faces.

The nodal fluxes on both horizontal faces = $\pm 13.9 W$.

They are highly concentrated at the junction of imposed and free temperatures.

Matlab sentence to compute them:

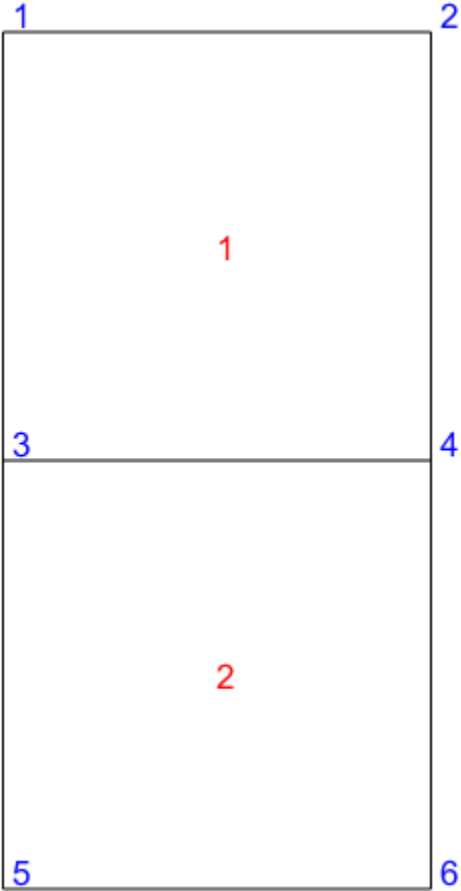
```
a = K*tca;sum (a(1:nl))
```



Conductive heat transfers

1. Partial differential equations
2. Variational methods
3. Rayleigh Ritz method
4. Finite element method (local level)
5. Finite element method (global level)
6. Matlab[©] procedures and examples
7. More about the heat flows

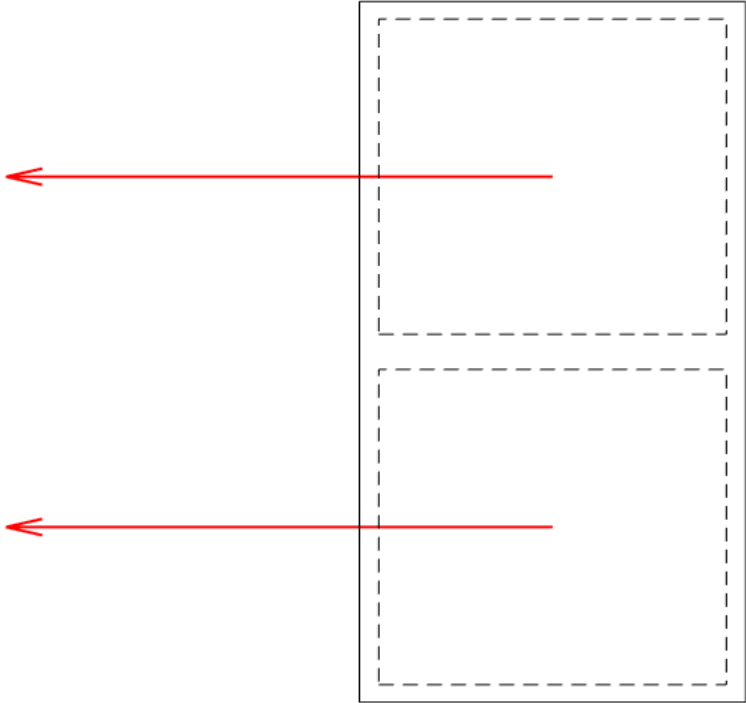
Assumed constant temperature gradient with higher (right) and lower (left) temperatures. The temperature gradient inside an element is computed in term of nodal temperatures.



Mesh and nodes labels

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

Heat flow, max: 18, mean: 18 W/m2



Matlab® function *kf5_conduction.m* to draw element heat flows

```

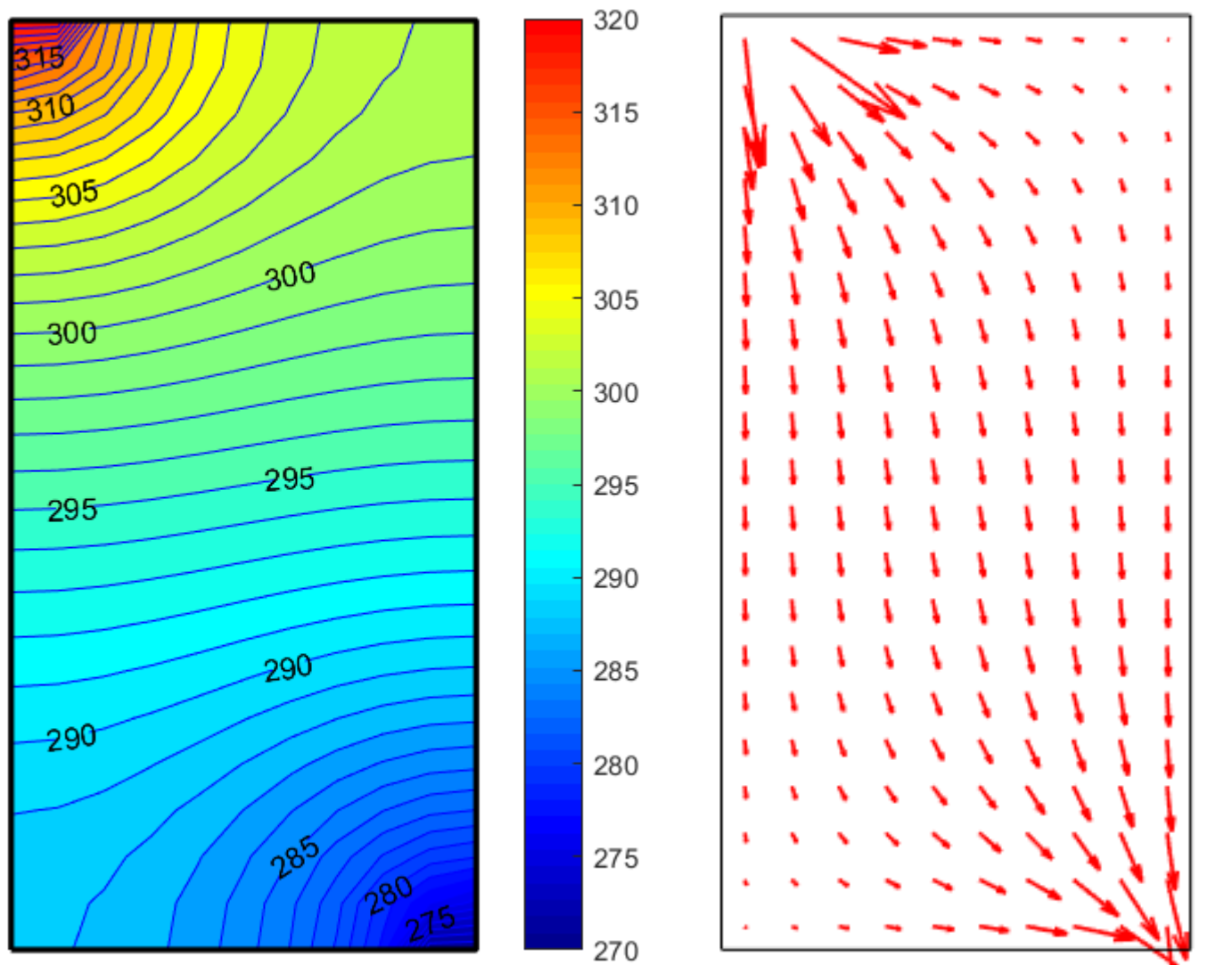
1 function kf5_conduction (nx,ny,lK,tca,co)
2 ii = 0;nn=(nx+1)*(ny+1);xyz = zeros(nn,3);nc=2;% Nodes Coons patch nx x ny
3 P = [0 0 0; 1 0 0; 1 2 0; 0 2 0]; % Flat square domain
4 for i = ny:-1:0
5     for j = 0:nx
6         t = i/ny; s = j/nx; ii = ii +1;
7         for c = 1 : nc
8             xyz(ii,c)= (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 ii=0;
14 X=zeros(nx*ny,1); Y=zeros(nx*ny,1);
15 u=zeros(nx*ny,1); v=zeros(nx*ny,1);
16 xx=zeros(4,1); yy=zeros(4,1);te=zeros(4,1);
17 for i = 1:nx % Loop on the nx columns of elemrnts
18     for j = 1:ny % Loop on the ny liness of elemrnts
19         ii = ii+1;
20         for k=1:4 % Loop on the 4 vertices of the element
21             X(ii) = X(ii)+xyz(lK(ii,k),1)/4; % x coord of the elem. center
22             Y(ii) = Y(ii)+xyz(lK(ii,k),2)/4; % y coord of the elem. center
23             xx(k) = xyz(lK(ii,k),1); % xx contains the 4 vertices x coord.
24             yy(k) = xyz(lK(ii,k),2); % yy contains the 4 vertices y coord.
25             te(k) = tca(lK(ii,k)); % te contains the 4 vertices tempera.
26         end
27         Jacob = 1/4*[xx(2)+xx(3)-xx(1)-xx(4) yy(2)+yy(3)-yy(1)-yy(4);
28                 xx(4)+xx(3)-xx(1)-xx(2) yy(4)+yy(3)-yy(1)-yy(2)];
29         Jm1 = Jacob^(-1)*co(ii);
30         u(ii) = -Jm1(1,:)/4*[-1 1 1 -1;-1 -1 1 1]*te;% x comp. of grad
31         v(ii) = -Jm1(2,:)/4*[-1 1 1 -1;-1 -1 1 1]*te;% y comp. of grad
32     end
33 end
34 gm = [max(sqrt(u.*u+v.*v)); mean(sqrt(u.*u+v.*v))]; % grad : max & av.
35 scale = 2;
36 disp(['Elem. heat flow max : ', num2str(gm(1),3), ', mean: ', ...
37     num2str(gm(2),3), ' W/m2'])
38 quiver(X,Y,u,v,scale,'r','LineWidth',1);hold on;
39 plot([xyz(ny*(nx+1)+1,1) xyz((nx+1)*(ny+1),1) xyz(nx+1,1) xyz(1,1) ...
40     xyz(ny*(nx+1)+1,1)], [xyz(ny*(nx+1)+1,2) xyz((nx+1)*(ny+1),2) ...
41     xyz(nx+1,2) xyz(1,2) xyz(ny*(nx+1)+1,2)], 'k');axis equal;hold on
42 title(['Heat flow, max: ', num2str(gm(1),2), ', mean: ', num2str(gm(2),2), ...
43     ' W/m2'],'fontsize',15);axis off;hold on
44 end

```

Including at the end of the procedure *pp_Conduction.m* the sentence:

`figure;kf5_conduction (nx,ny,lK,tca,co);hold on`

we obtain the heat flows diagram.



The temperatures are imposed on segments of the upper and lower faces: 320 K on the upper face and 270 K on the base. The mesh is 10×20 . In the elements, the maximum heat flow = 68 Wm^{-2} with an average of 14 Wm^{-2}

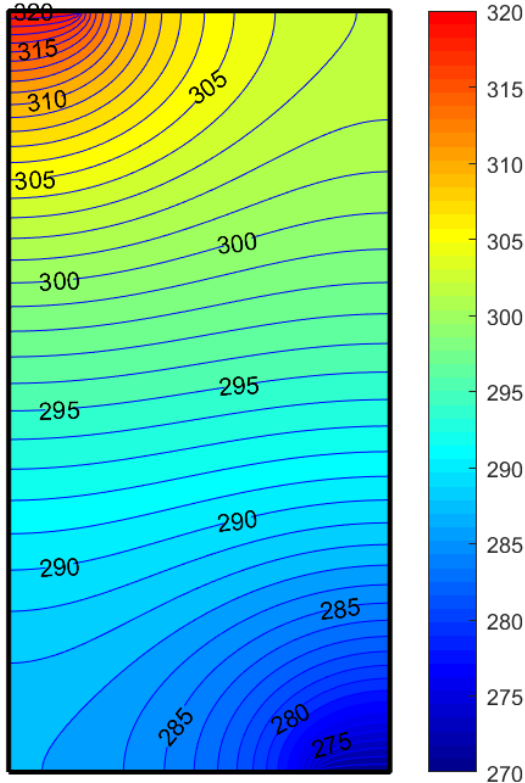
Matlab[®] function *kfl_conduction.m* to handle non uniform k

```

1 function [co] = kfl_conduction(nx,ny)% Treatment of non uniform onductiv.
2 k = 1; % W/(m K)
3 nel = nx*ny; % Number of element computed from mesh definition
4 fa = 1 ; % Ratio between the 2 conductivities, if 1, k is a cst
5 co = ones(nel,1)*k;
6     co(nx*nx+1:nx*nx+nx) = k*fa; % Second k on horizontal band 1
7     if nx>2;co(nx*(nx-1)+1:nx*nx) = k*fa;end % Second k on horizontal band 2
8 disp(['Thermal conductiv. : ',num2str(k,'%0.3g'),' W/(m K)'])
9 disp(['Main & bridge k. : ',num2str([co(1) co(nx*nx+1)]),' W/(m K)'])
10 end

```

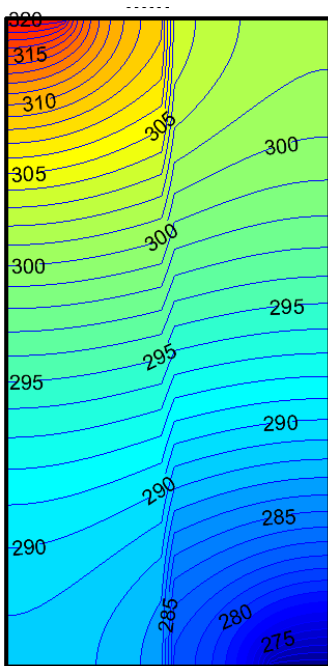
$T_{\max} : 320 \text{ }^{\circ}\text{C}$, $T_{\min} : 270 \text{ }^{\circ}\text{C}$, $\text{pas} : 1 \text{ }^{\circ}\text{C}$



In lines 6 and 7, the conductivity coefficient k is multiplied by the factor fa defined in line 4. The elements are numbered from left to right and from top to bottom (see slide 48).

Introduction of two lines of higher conduction in elements near the horizontal median of the domain.

This slide shows the constant conductivity reference solution with 10 imposed nodal temperatures on each horizontal side (see line 9 of the procedure of slide 45).

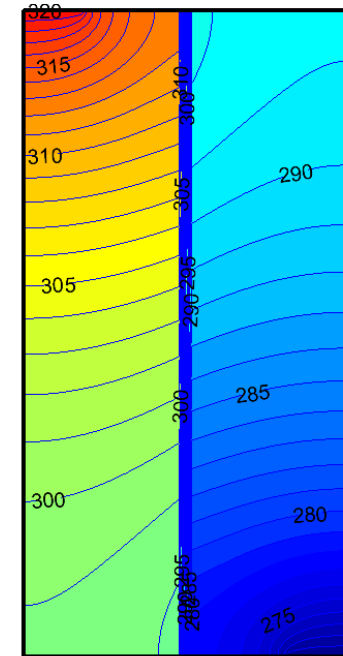


The conductivity change is introduced in the 2 lines of elements at the left and right of the **vertical median**.

$k*0.1$

$k*0.01$

Conductivity : 1 W/(m K)
 Base temperature : 270 K
 Top face temp. : 320 K
 Number of elements : 5000
 N. fix. hor. faces : 20



Matlab[®] function *kf1_conduction.m* to handle non uniform k

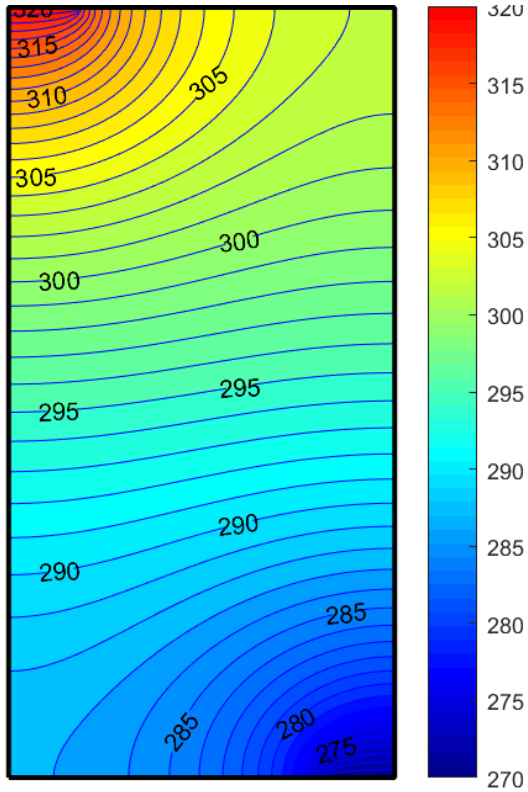
```

1 function [co] = kf1_conduction(nx,ny)% Treatment of non uniform onductiv.
2 k = 1; % W/(m K)
3 nel = nx*ny; % Number of element computed from mesh definition
4 fa = 1 ; % Ratio between the 2 conductivities, if 1, k is a cst
5 co = ones(nel,1)*k;
6 co(nx*nx+1:nx*nx+nx) = k*fa; % Second k on horizontal band 1
7 if nx>2;co(nx*(nx-1)+1:nx*nx) = k*fa;end % Second k on horizontal band 2
8 disp(['Thermal conductiv. : ',num2str(k,'%0.3g'),' W/(m K)'])
9 disp(['Main & bridge k. : ',num2str([co(1) co(nx*nx+1)]),' W/(m K)'])
10 end

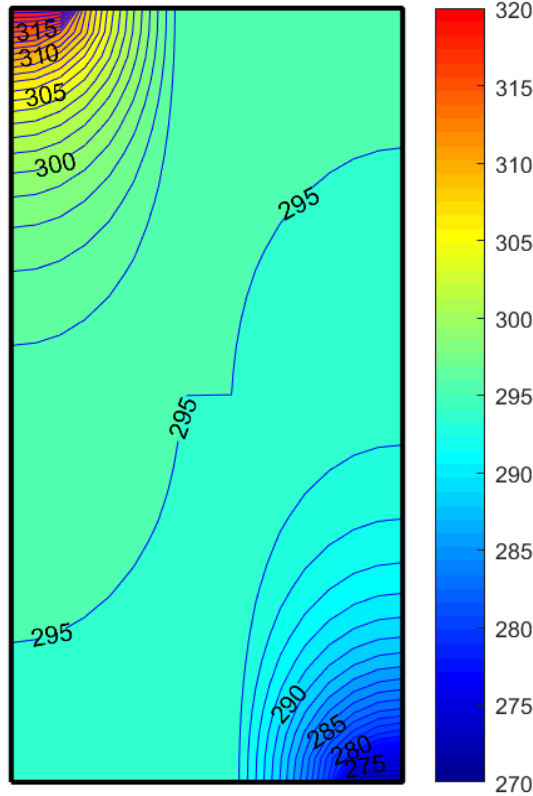
```

Introduction of a central vertical strip with conductivity

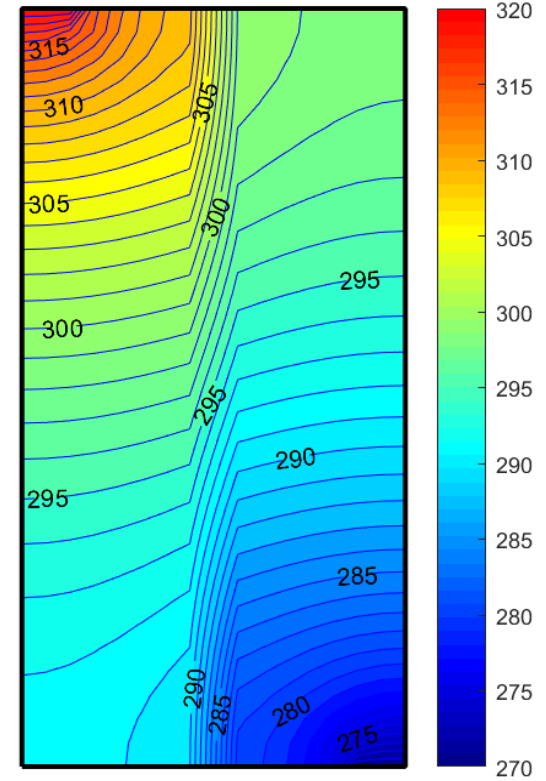
$k \times 1$



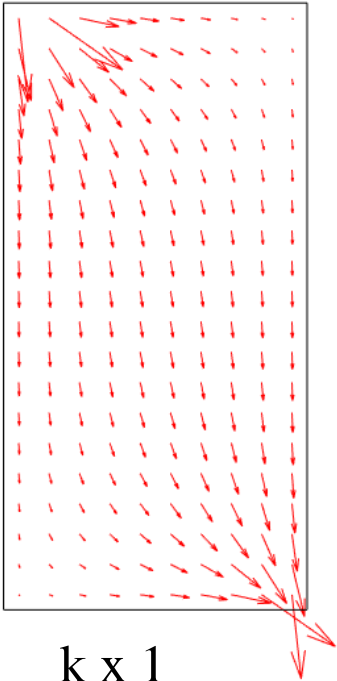
$k \times 1000$



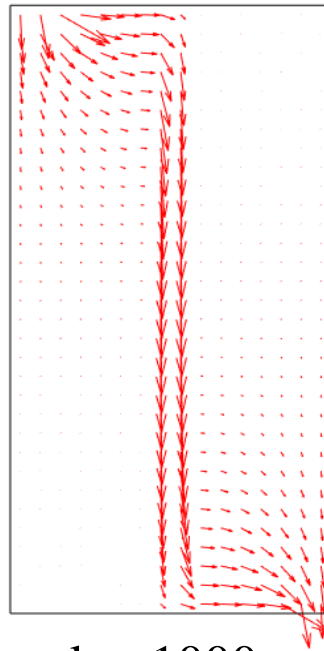
$k \times 0.1$



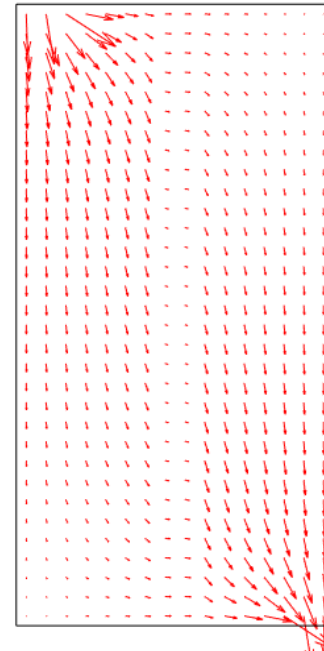
Relative strip thickness: $1/8$ domain width



$k \times 1$

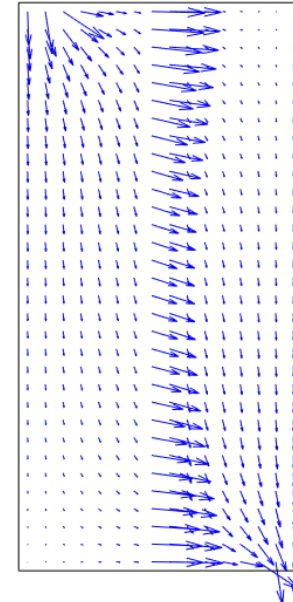
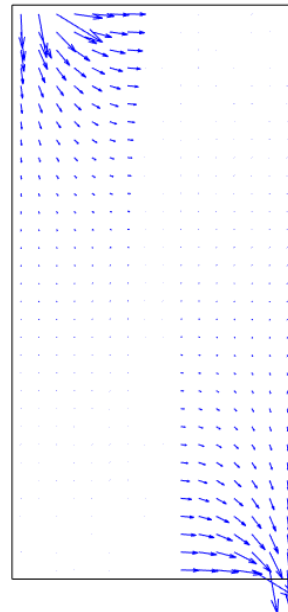
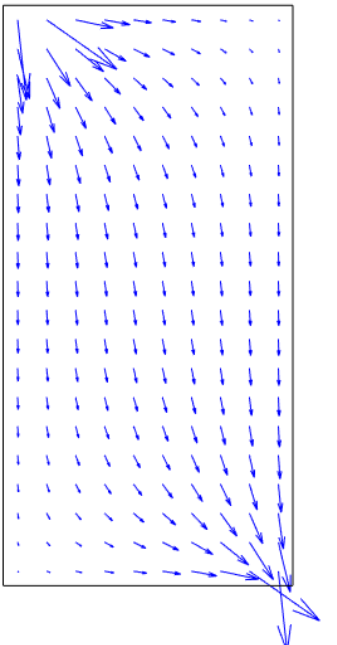


$k \times 1000$



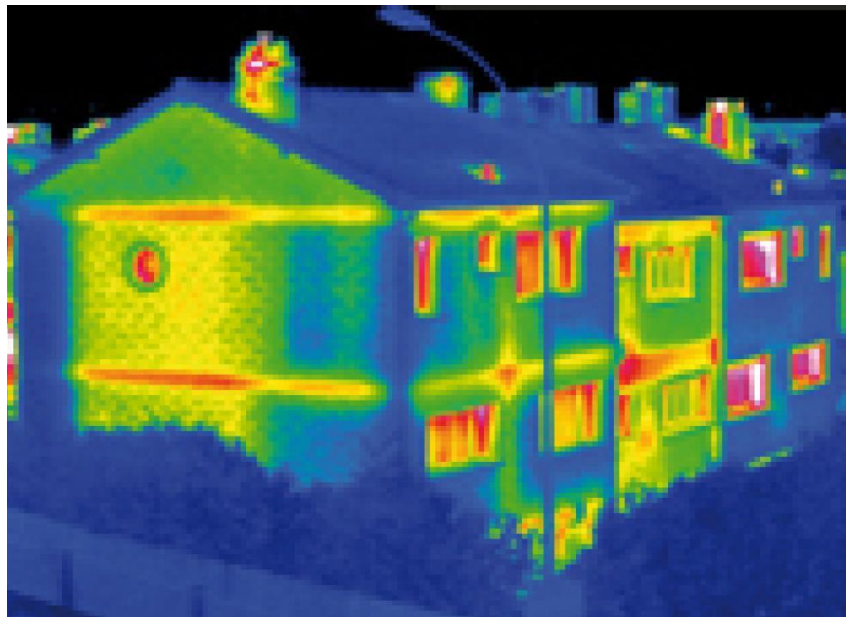
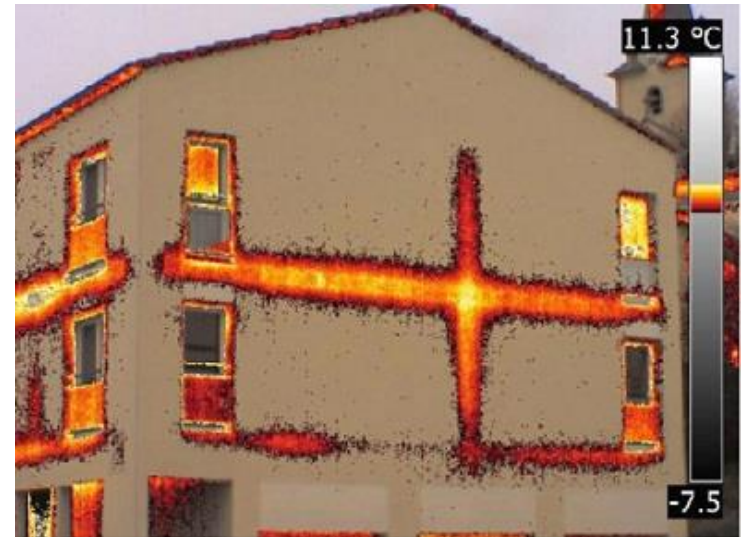
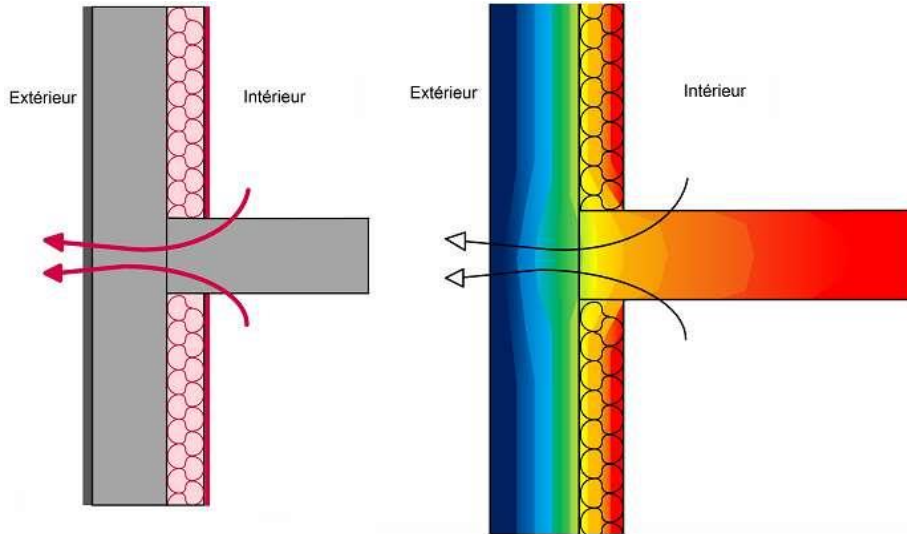
$k \times 0.1$

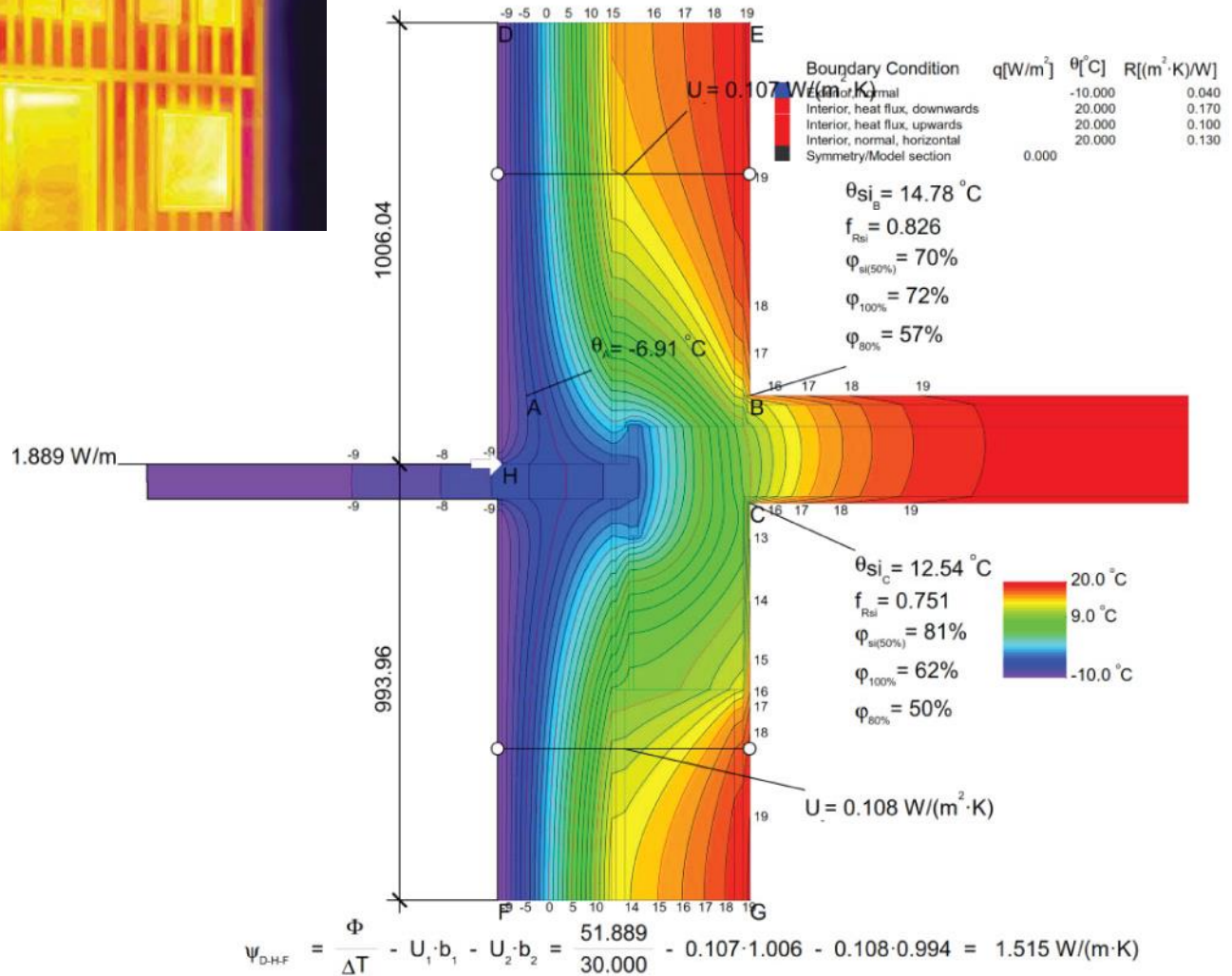
Heat
flow



Temperature
gradient

Relative strip thickness: $1/8$ domain width





Variable	Symbol	Units
Conductivity coefficient & tensor	k & k_{ij}	$W K^{-1} m^{-1}$
Resistivity coefficient	r	$W^{-1} K m$
State variable temperature	T	K
Temperatures array	τ	K
Temperature gradient	g_i	$K m^{-1}$
Temperature gradient array	G	$K m^{-1}$
Thermal loads array	g	W
Heat flux & heat flux array	q_i	$W m^{-2}$
Stream function	ψ	W
Connection matrix	C	m^2
Conductivity matrix	K	$W K^{-1}$
Core resistivity matrix	R	$W^{-1} K m^4$
Dissipation function	I, J, F, P, Q	$W K$
Geometric variable : thickness	e	m
Geometric variable : length	a, b, c	m
Geometric variable : area	$\partial_i D$	m^2
Geometric variable : volume	D	m^3