ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

FINITE ELEMENT METHOD FOR THERMAL PROBLEMS

Bachelor thesis

Autor:
Alessandro Di Giorgio

Supervisors:
Shobhit Jain
Paolo Tiso

May 2018

Contents

Introduction

One of the most important tasks for engineers is to model physical phenomena. They develop conceptual and mathematical models to understand and predict physical events.

Mathematical models are often very complex equations based on the geometrical shape of the system, which are not solvable analytically in most cases. Numerical methods are then used to approximate the solution in such cases. For this reason, before the advent of computing devices, these models were extremely difficult to solve. This was greatly simplified with the support of computers. As a result, numerical simulations are routinely used by industries to predict solutions to realistic and physically relevant problems.

In this study, we use the finite element method (FEM) to solve the partial differential equations (PDE) governing heat transfer using FEM, the goal is to find a numerical solution, which reliably approximates the exact solution. FEM is a popular choice for solving PDE on complex geometries, commonly encountered in the physical world. But how does it work? The domain of the system is represented as a collection of geometrically simple subdomains, called indeed finite elements. Over each finite element, the unknown variables are approximated by a linear combination of algebraic polynomials. This process results in a set of simultaneous algebraic equations, which should be easier to solve and find a general solution.

The goal of this study is to analyze the heat equation in different spatial domains using FEM. To understand the topic thoroughly, we start with examining the simplest case of the steady state in 1 dimensional space. Thereafter, we move towards more complex cases (more spatial dimensions, rotated geometries, dynamic problems,...). All these cases are implemented in a MATLAB code and the results obtained are validated for accuracy against analytically obtained solutions, where ever possible. When analytical solutions are not available, validation is performed against an inbuilt *pdetool* in MATLAB.

Chapter 1

1D problem

1.1 Steady state problem

The steady state temperature distribution, as a result of conduction in a 1D medium, is given by the equation

$$-k\frac{d^2T(x)}{dx^2} = Q(x) \quad x \in \Omega \tag{1.1}$$

where k is the conductivity constant, $Q(x)$ is the known heat source function and $T(x)$ is the unknown temperature. This second order differential equation must be provided with two suitable boundary conditions, at the ends of the domain to make it well-posed. At the boundary, either the value of the unknown (*Dirichlet*) or the value of the first derivative (*Neumann*) is specified in order to solve the problem.

In the Finite element method (FEM), the domain is divided into an established number of elements, in order to obtain the approximate solution T_e in the points (nodes) that connect the elements. The finite element approximation T_e is sought in the form

$$T(x) \approx T_e(x) = \sum_{i=1}^{n} T_i N_i(x) \tag{1.2}$$

where n is the total nodes in the domain, N_i is called the shape function, which can be selected to build the approximate solution. Instead, T_i are temperature coefficients in i^{th} node. In the 1 dimensional case, the shape function can be chosen to be a linear polynomial, in the form

$$N_i(x) = ax + b. \tag{1.3}$$

Since there are n unknown parameters, we need n relations to determine them. If the unknown T is substituted with the approximate solution T_e in the equation (1.1), we obtain an expression that will not be equal to the right-hand side of the equation. The difference between the two sides of the equation is called *residual*

$$-k\frac{d^2T_e(x)}{dx^2} - Q(x) = R(x) \neq 0. \tag{1.4}$$

By definition, the exact solution of the differential equation will make its residual zero at all points of the problem domain. However, the residual will not, in general, vanish when an approximate solution is substituted in it. The basic principle in weighted residual methods is to minimize the residual in a weighted integral sense as follows

$$\int_{\Omega} w(x)R(x)\,\mathrm{d}x = 0, \tag{1.5}$$

which is named the *weak form*.Substituting the equation (1.4) in (1.5), we get

$$\int_{\Omega} \left(-w(x)k\frac{d^2T_e(x)}{dx^2} - w(x)Q(x)\right)\,\mathrm{d}x = 0, \tag{1.6}$$

where $w(x)$ is known as the weight (test) function. The idea of this weight function is to select as many different $w(x)$ as necessary to obtain the required n linear algebraic equations. There are different choices of $w(x)$ that may be used. In this study, the *Galerkin method* will be used: it claims that $w(x)$ is chosen to be a combination of shape functions

$$w(x) = \sum_{j=1}^{n} N_j(x)w_j. \tag{1.7}$$

If we integrate the equation (1.6) by parts, we obtain

$$\int_{\Omega} k\frac{dw(x)}{dx}\frac{dT_e(x)}{dx}\,\mathrm{d}x - k\left[w(x)\frac{dT_e(x)}{dx}\right]_{\partial\Omega} = \int_{\Omega} w(x)Q(x)\,\mathrm{d}x. \tag{1.8}$$

The second term, on the left-hand-side of the equation, depends on the boundary conditions that are supplied. This term is zero if both ends are supplied with Dirichlet boundary conditions. In that case, $w(x_1) = w(x_n) = 0$, where $\partial\Omega = [x_1, x_n]$. From now on, this term will be denoted as "BCT":

$$BCT = k\left[w(x)\frac{dT_e(x)}{dx}\right]_{\partial\Omega}.$$

When we are discussing about boundary conditions, we can introduce a generalize equation for every case:

$$\alpha(x)T(x) + \beta(x)\frac{dT}{dx} = \gamma(x), \tag{1.9}$$

where, if $\beta = 0$, we have the Dirichlet condition, otherwise if α is zero we have the Neumann condition. However, if neither α nor β are zero, we have a third case: the *Robin condition*.

Now, from the expression (1.8), we can substitute the desired approximate solution (1.2) into it. The shape functions $N_{i,j}$ will have a first grade polynomial, as said before and Figure 1.1 shows how they will develop.

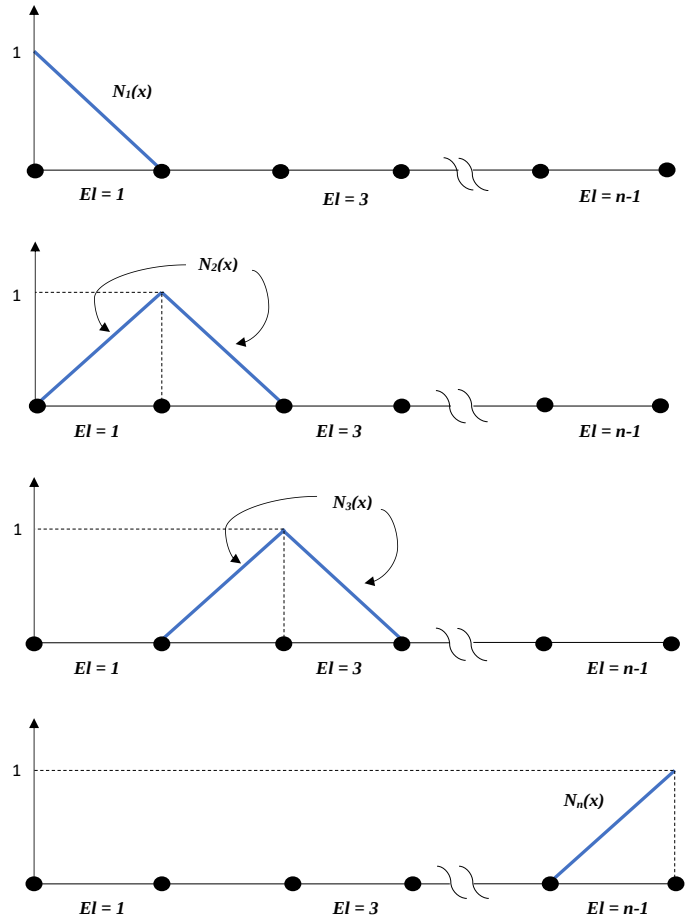


Figure 1.1: Shape functions for a 1D FE mesh of linear elements.

As shown in Figure 1.1, it is possible to associate each shape function with one node of the finite element mesh. Shape functions are nonzero only over the elements connected to the node with which they are associated and they are equal to zero over all other elements. Now, if we substitute $T_e(x)$ and $w(x)$, we get

$$\int_\Omega k \sum_{j=1}^{n} \left(\frac{dN_j(x) w_j}{dx} \right) \sum_{i=1}^{n} T_i \left(\frac{dN_i(x)}{dx} \right) \mathrm{d}x = \int_\Omega \sum_{j=1}^{n} \left(N_j(x) w_j \right) Q(x) \, \mathrm{d}x + \sum_{j=1}^{n} BCT. \tag{1.10}$$

Given that w_j is not a function of spatial position, it can be taken out of the integrals

$$\sum_{j=1}^{n} w_j \int_\Omega k \frac{dN_j(x)}{dx} \sum_{i=1}^{n} T_i \left(\frac{dN_i(x)}{dx} \right) \mathrm{d}x = \sum_{j=1}^{n} w_j \int_\Omega N_j(x) Q(x) \, \mathrm{d}x + \sum_{j=1}^{n} BCT. \tag{1.11}$$

Since, the above eq., must hold for all admissible values of w_j, we can eliminate w_j from both sides

$$\sum_{i=1}^{n} T_i \int_\Omega k \frac{dN_j(x)}{dx} \frac{dN_i(x)}{dx} \, \mathrm{d}x = \int_\Omega N_j(x) Q(x) \, \mathrm{d}x + k \left[N_j \frac{dT(x)}{dx} \right]_{\partial\Omega}. \tag{1.12}$$

To further simplify the equation, we can use the following compact matrix notation

$$\underline{\underline{K}} \, \underline{T} = \underline{f} + \underline{B}, \tag{1.13}$$

which is known as the global equation system. K is the stiffness matrix of size nxn, T is the vector of the unknown with n dimension, f is the body load vector and finally B the vector of the derivative boundary condition are applied. They are defined as

$$\begin{aligned} \left[\underline{\underline{K}} \right]_{ji} &= \int_\Omega k \frac{dN_j(x)}{dx} \frac{dN_i(x)}{dx} \, \mathrm{d}x \\ \left[\underline{T} \right]_i &= T_i \\ \left[\underline{f} \right]_j &= \int_\Omega N_j(x) Q(x) \, \mathrm{d}x \\ \left[\underline{B} \right]_j &= k \left[N_j \frac{dT(x)}{dx} \right]_{\partial\Omega}. \end{aligned} \tag{1.14}$$

Now, to provide a numerical implementation of the 1D case, we need to set firstly the shape function, where the domain $\Omega \in [x_1, x_n]$. The domain is subdivided into $n-1$ elements and delimited by n nodes. The sub-intervals are denoted by

$$h = x_{i+1} - x_i, \tag{1.15}$$

where h is the element size. Looking at the Figure 1.1, we can assert that the shape function is

$$N_i(x) = \begin{cases} \frac{1}{h}(x - x_{i-1}) & x_{i-1} \leq x \leq x_i \\ \frac{1}{h}(x_{i+1} - x) & x_i \leq x \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{1.16}$$

With this choice, the shape functions are zero outside the corresponding element. However, in this way, they have a local support, which means $\underline{\underline{K}}$ is a sparse matrix because we are focusing each time over one node. Therefore, to find the global matrix K, we need to sum all those local matrices:

$$K_{global} = \sum_{el=1}^{n-1} K_{local} \tag{1.17}$$

and it is needed to proceed similarly for the body load vector as well. Finally, we obtain an nxn square Matrix K_{global}.

1.1.1 Isoparametric mapping

Generally, the form of the shape functions might be hard to represent and to calculate for their irregular shapes. So, it is used to build an easier configuration, as for example regular triangles, to solve easily the integration. In order to overcome this problem, we go through the concept of *isoparametric* transformation. In this case, it is common to create a local coordinate system $-1 \leq \xi \leq 1$. We can see this transformation in Figure 1.2.

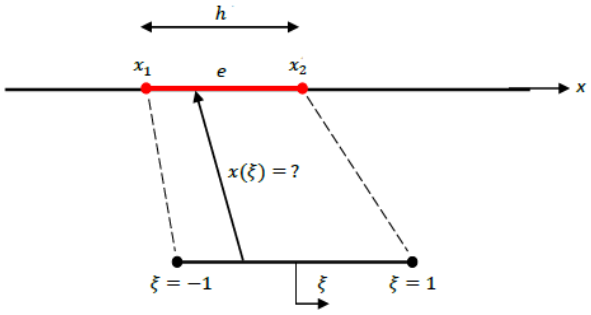


Figure 1.2: Mapping between the global x coordinate and the element coordinate ξ.

Thus, locally our shape functions become

$$N_1(\xi) = \frac{1}{2}(1 - \xi) \text{ and } N_2(\xi) = \frac{1}{2}(1 + \xi), \tag{1.18}$$

where

$$x = \frac{x_2 - x_1}{2}\xi + \frac{x_2 + x_1}{2} = \frac{h}{2}\xi + \frac{x_2 + x_1}{2}. \tag{1.19}$$

Now, we can write the elemental K_{local} integration (1.13) using ξ coordinate

$$K_{ij} = k \int_{-1}^{1} \left(\frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial x} \frac{\partial N_j}{\partial \xi} \frac{\partial \xi}{\partial x} \right) \frac{\partial x}{\partial \xi} \, \mathrm{d}\xi. \tag{1.20}$$

Thus, we can define an important term in our study: the Jacobian parameter J

$$J = \frac{\partial x}{\partial \xi} = \frac{h}{2}. \tag{1.21}$$

Therefore, the stiffness matrix becomes

$$K_{ij} = k \int_{-1}^{1} \left(\frac{\partial N_i}{\partial \xi} \frac{1}{J} \frac{\partial N_j}{\partial \xi} \frac{1}{J} \right) J \, \mathrm{d}\xi. \tag{1.22}$$

The way to get the body load vector is similar, thus the new equation is

$$\underline{f} = \int_{-1}^{1} N_j Q J \, \mathrm{d}\xi. \tag{1.23}$$

However, in this result, we are considering $Q(x)$, of equation (1.1), as a constant. If the heat source is in function of x, we need to proceed just in the same as we did for $T(x)$:

$$Q(x) = \sum_k Q_k \phi_k(x) \tag{1.24}$$

and going through the isoparametric transformation, we get

$$f = \int_{-1}^{1} Q_k N_j \phi_k J \, \mathrm{d}\xi. \tag{1.25}$$

Previously, it was said that we need to assemble a global matrix to solve FEM; let's see the steps how to get it. Firstly, it is needed to define a matrix with n rows, which specifies the local node that we are considering. The table below shows the idea to construct this matrix

Elements	Nodes	
1	1	2
2	2	3
3	3	4
$\vdots$	$\vdots$	$\vdots$
$n-1$	$n-1$	n

and, to make it clearer, we are going to see an example: let's examine the matrix if we were in the first element

$$L_1 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \end{pmatrix} \tag{1.26}$$

so on, for the second element, we obtain

$$L_2 = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \end{pmatrix} \tag{1.27}$$

whereas, if we consult the last element

$$L_n = \begin{pmatrix} 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix}. \tag{1.28}$$

With the support of those L matrix, we are able to build our global stiffness matrix and the body load vector. The way to assemble the global representation is the following

$$K_{global} = \sum_{el=1}^{n-1} L_{el}^T K_{loc} L_{el} \tag{1.29}$$

$$f_{global} = \sum_{el=1}^{n-1} L_{el}^T f_{loc}. \tag{1.30}$$

Thus, we are now able to derive the global equation system that was mentioned in (1.12)

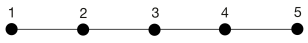
$$\underline{\underline{K}} = \begin{pmatrix} K_{11}^1 & K_{12}^1 & 0 & 0 & \cdots & 0 & 0 \\ K_{21}^1 & K_{22}^1 + K_{11}^2 & K_{12}^2 & 0 & \cdots & 0 & 0 \\ 0 & K_{21}^2 & K_{22}^2 + K_{11}^3 & K_{12}^3 & \cdots & 0 & 0 \\ 0 & 0 & K_{21}^3 & K_{22}^3 + K_{11}^4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & K_{22}^{el-1} + K_{11}^{el} & K_{12}^{el} \\ 0 & 0 & 0 & 0 & \cdots & K_{21}^{el} & K_{22}^{el} \end{pmatrix} \tag{1.31}$$

$$\underline{f} = \begin{pmatrix} f_1^1 \\ f_2^1 + f_1^2 \\ \vdots \\ f_2^{el-1} + f_1^{el} \\ f_2^{el} \end{pmatrix} \tag{1.32}$$

$$\underline{B} = \begin{pmatrix} B_1^1 \\ 0 \\ \vdots \\ 0 \\ B_2^{el} \end{pmatrix}. \tag{1.33}$$

1.1.2 Implementation of boundary conditions

In this section, we analyze how the boundary conditions are included in our calculations. Let us consider a beam divided in 5 nodes as example.



Dirichlet boundary condition

We are dealing with Dirichlet conditions, when, from equation (1.9), $\beta = 0$. These assumptions are applied at the edges of the beam and they are given by the problem. For this case, let us consider $T_1 = T_{0,left}$ and $T_5 = T_{0,right}$ as boundary conditions. Recalling equation (1.13), for the current case, B is excluded. Therefore, the unknown temperature are T_2, T_3 and T_4. So, in order to solve the equation

$$KT = f$$

and to find the middle terms of the beam, we need to rearrange the previous equation so that we can separate the middle terms, which are the unknown, and the boundary terms, which are given, from the vector T.

$$\begin{pmatrix} K_{bb} & K_{bm} \\ K_{mb} & K_{mm} \end{pmatrix} \begin{pmatrix} T_b \\ T_m \end{pmatrix} = \begin{pmatrix} f_b \\ f_m \end{pmatrix}, \tag{1.34}$$

where b and m are indexes, which represents the boundary and middle nodes. They are used to extract the nodal respective values from the original notation. Solving the scalar product on the left in side, one of the two equations is

$$K_{mb}T_b + K_{mm}T_m = f_m \tag{1.35}$$

and, thus, isolating T_m, we get

$$T_m = K_{mm}^{-1}\left[f_m - K_{mb}T_b\right]. \tag{1.36}$$

Mixed boundary conditions

Let us consider, now, both boundary conditions: Neumann and Dirichlet. Thus, in one end exists a constant given temperature and, instead, in the other one the derivative specifies if there is heat flowing or insulation. Considering the same beam, with same number of nodes, we examine the case that $T_1 = T_0$ and $dT_5/dx = g_0$, where g_0 is a constant. Due to Neumann term, we need to include the vector B in the

equation. So, from equation in (1.33), we can claim that $B_5 = kg_0$. Therefore, if we now rearrange equation (1.13), we obtain

$$\begin{pmatrix} K_{bb} & K_{bm} \\ K_{mb} & K_{mm} \end{pmatrix} \begin{pmatrix} T_b \\ T_m \end{pmatrix} = \begin{pmatrix} f_b \\ f_m \end{pmatrix} + \begin{pmatrix} B_b \\ B_m \end{pmatrix}. \tag{1.37}$$

Here, b and m are different: the only nodal solution available is in node 1. Thus, b represents only one edge, in the other hand m includes all the nodes where we need still to find the temperature, even node 5. Summarizing, $T_b = T_1$ and $T_m = (T_2\ T_3\ T_4\ T_5)^T$. Now, if we continue with equation (1.37), the final solution of the unknowns are

$$T_m = K_{mm}^{-1} \left[f_m + B_m - K_{mb} T_b \right]. \tag{1.38}$$

Therefore, specifying the nodal boundary b, where the solution is known, and the nodal middle m, where the solution is unknown, allow us to solve easily the initial equation and to simplify the implementation.

Robin boundary condition

For the Robin conditions, we examine the case that

$$\frac{dT}{dx} + T = \gamma,$$

where γ is constant. For this case, the derivative is included: thus, it is necessary to proceed as for the Neumann case. However, the variable of the unknown has to be included in the equation. So, recalling equation (1.14), we substitute the derivative term in B with the equation above and we can write

$$\underline{\underline{K}}\,\underline{T} = \underline{f} + k \left[N_j(\gamma - N_i T_i) \right]_{\partial\Omega}. \tag{1.39}$$

Introducing the Robin term, we obtain an additional contribution to the body load vector f and the stiffness matrix K. Considering the example that the Robin condition is applied on last node 5, the last term, which represent the node 5, will be modify as follows

$$\begin{pmatrix} K_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & K_{22} + k \end{pmatrix} T = \begin{pmatrix} f_1 \\ \vdots \\ f_2 + k\gamma \end{pmatrix}. \tag{1.40}$$

1.2 Unsteady state problem

After that the steady state case was analyzed, now we start examining the change of the temperature over time. The unsteady heat equation is given as

$$c\frac{dT(x,t)}{dt} - k\frac{d^2T(x,t)}{dx^2} = Q(x,t), \tag{1.41}$$

where c is a known constant. Further, in this initial value problem, an initial condition is given, which provides the solution on the whole domain at initial time.
The weak form of this differential equation is obtained in the same way as we did for the steady problems.

$$\int_\Omega \left(wc\frac{dT}{dt} - wk\frac{d^2T}{dx^2} \right) \mathrm{d}x = \int_\Omega wQ \ \mathrm{d}x. \tag{1.42}$$

In this case the approximation of the solution will be a bit different

$$T_e(x,t) = \sum_{i=1}^{n} T_i(t) N_i(x) \tag{1.43}$$

where the time dependency of the solution is associated with the nodal unknowns and shape functions are taken to be the same as the ones used for the steady problems. If we go on with the integration, we will proceed at the same way it was done before, except for a new term of the time dependency

$$\sum_{i=1}^{n} \left[\underbrace{c\int_\Omega N_i N_j \mathrm{d}x}_{M} \right] \frac{dT_i}{dt} + \sum_{i=1}^{n} \left[\underbrace{k\int_\Omega \frac{dN_j}{dx}\frac{dN_i}{dx}\ \mathrm{d}x}_{K} \right] T_i = \underbrace{\int_\Omega N_j Q \mathrm{d}x}_{f} + \underbrace{BCT}_{B}. \tag{1.44}$$

Thus, the compact matrix form becomes

$$\underline{\underline{M}}\ \underline{\dot{T}} + \underline{\underline{K}}\ \underline{T} = \underline{f} + \underline{B}. \tag{1.45}$$

To summarize, for a time dependent problem an extra matrix (named Mass matrix) needs to be evaluated. Calculation of other matrices and vectors are the same as in steady state problems.

1.3 Time discretization

Since we start dealing with time dependency, time has to be discretized. There are several methods that help us to solve the time derivatives of the nodal unknown. Nevertheless, we are going to look just two of them, which could be useful to implement our code.

1.3.1 The explicit Euler method

The explicit method (forward method) calculates the solution of a system at later time from knowing the current time state. Let us see what this means:

$$\dot{x}(t) = f(x(t), t)$$

the derivative is defined as $\dot{x}(t) = \lim_{h\to 0} \frac{x(t+h)-x(t)}{h}$, but to evaluate numerically a solution we need to consider h as a small time step and neglect the limit. Thus, we can write the approximation equation as

$$\frac{x(t+h) - x(t)}{h} \approx f(x(t), t).$$

Given x at time t, one can compute x at the later time $t + h$, by solving the difference equation

$$x(t + h) = x(t) + hf(x(t), t).$$

Introducing the notation $t_n + h = t_{n+1}$ and $x(t_n) = x_n$, we can rewrite the formula as

$$x_{n+1} = x_n + hf(x_n, t_n)$$

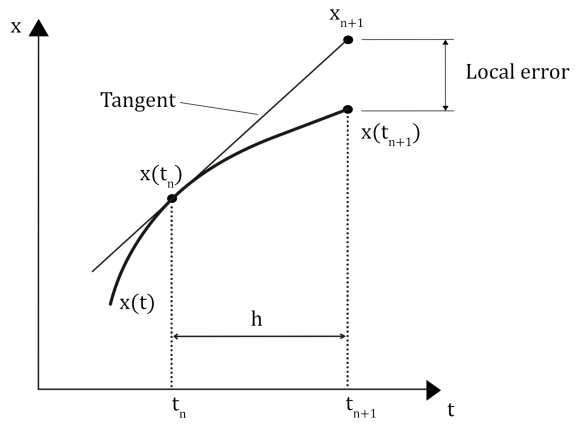


Figure 1.3: Explicit Euler method

Explicit methods are very easy to implement, however, the drawback arises from the limitations on the time step size to ensure numerical stability. Therefore if we approach to our problem by using this method, we can rewrite the equation (1.45) as

$$[M]\left(\frac{\{T\}_{n+1} - \{T\}_n}{\Delta t}\right) + [K]\{T\}_n = \{f\} + \{B\}$$

and so finally we get

$$\{T\}_{n+1} = [M]^{-1}\left[[M]\{T\}_n + \Delta t\left(\{f\} + \{B\} - [K]\{T\}_n\right)\right]. \tag{1.46}$$

1.3.2 The implicit Euler method

The implicit method (backward method) finds a solution by solving an equation involving both, the current state of the system and the later one. For small h, it holds that

$$\frac{x(t) - x(t-h)}{h} \approx f(x(t), t),$$

leading to the scheme

$$x_{n+1} = x_n + hf(x_{n+1}, t_{n+1}).$$

The application of this method needs an iteration to calculate x_{n+1}, because $f(x_{n+1}, t_{n+1})$ is not known, hence it gives us an implicit equation for the computation of x_{n+1}. This is evidently much more time consuming than the explicit one, because we deal with non linear calculations.

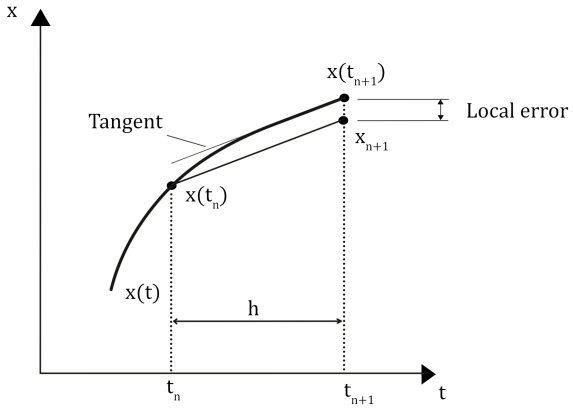


Figure 1.4: Implicit Euler method

Let us see again what happen introducing this method to equation (1.45):

$$[M]\left(\frac{\{T\}_{n+1} - \{T\}_n}{\Delta t}\right) + [K]\{T\}_{n+1} = \{f\} + \{B\}.$$

The unknown vector, $\{T\}_{n+1}$, can be solved as follows

$$\{T\}_{n+1} = ([M] + \Delta t[K])^{-1}\left[[M]\{T\}_n + \Delta t\{f\} + \Delta t\{B\}\right]. \tag{1.47}$$

Explicit Euler method is generally easy to write and to compute. However, it strongly depends on the solution step, which if does not have a proper size, the approximation will blow up quickly. On the other hand, the implicit method requires more computational effort and it would take more time to get the solution, nevertheless this method has a better stability, ensuring a better reliability on the results.
For numerical solving of partial differential equation, Matlab provides already strong algorithms that can help to solve these problems. In the implementation, we are going to use generally the command *ode45*.

1.4 Exact solution

In this section, we examine the analytical solution of the partial differential equation, which would be useful to validate the approximation.
Let us consider a beam of length L. It is subjected to a constant, uniform heat source q and it have a constant temperature at the boundaries $T(0) = T_1$ and $T(L) = T_2$ (Dirichlet condition). Summarizing
Case 1:

$$\begin{cases} -kT_{xx} = q, & x \in [0, L] \\ T(0) = T_1, \\ T(L) = T_2. \end{cases} \tag{1.48}$$

Upon solving (1.48) using successive integration, we obtain the general solution as

$$T(x) = -\frac{q}{2k}x^2 + \left(\frac{T_2 - T_1}{L} + \frac{qL}{2k}\right)x + T_1. \tag{1.49}$$

From the solution we can assert that the temperature profile would be a parabola.

Case 2: Let us change the boundaries to a mixed condition:

$$\begin{cases} -kT_{xx} = q, & x \in [0, L] \\ T(0) = T_1\ (Dirichlet), \\ T_x(L) = g_0\ (Neumann), \end{cases} \tag{1.50}$$

where g_0 is the heat flowing from this edge. Solving the problem, we get

$$T(x) = -\frac{q}{2k}x^2 + \left(g_0 + \frac{q}{k}L\right)x + T_1 \tag{1.51}$$

Case 3: Now considering the unsteady problem, we turn off the heat source q (i.e., $q = 0$) for $t > 0$:

$$\begin{cases} cT_t = kT_{xx}, & x \in [0, L] \\ T(0, t) = T_1, & t > 0 \\ T(L, t) = T_2, & t > 0 \\ T(x, 0) = T_{in}(x) & 0 \le x \le L \end{cases} \tag{1.52}$$

where the initial condition, T_{in}, is chosen to be the solution that we found in (1.49). When we have two or more variables, we use the separation of variables to solve the partial differential equation. Since we consider our general solution as $T(x, t) = X(x)\tilde{T}(t)$, substituting it in the differential equation, we have

$$\frac{c}{k}\frac{\dot{\tilde{T}}}{\tilde{T}} = \frac{X''}{X} = -\lambda.$$

Separating three cases: $\lambda = 0$, $\lambda < 0$ and $\lambda > 0$; we sum then the solution of these cases. However, for simplicity, in the 1D case the command *pdepe*, in MATLAB, will do this job for us, so we do not need to go in further explanations.

1.5 Results for the problem

In this section, we examine the results that we obtained from different chosen cases. Let us examine the case with the following properties:

$$L = 2, \qquad q = 50, \qquad k = 1, \qquad c = 1.$$

Case 1: Steady state case with Dirichlet boundary condition.

$$T(0) = 5, \quad T(2) = 0$$

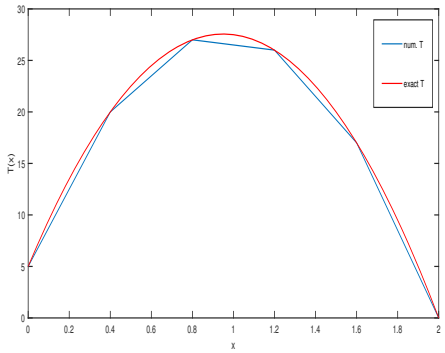


Figure 1.5: Temp. profile with 6 nodes

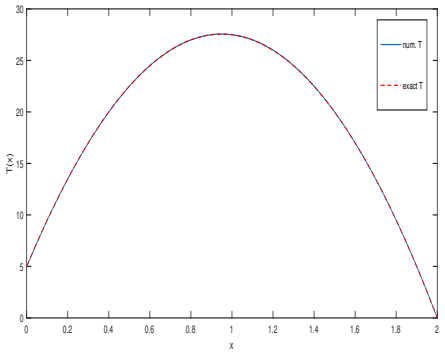


Figure 1.6: Temp. profile with 51 nodes

From Figure 1.5, we can observe that the nodal values match perfectly with the analytical solution. This means that the residual was minimized successfully. However, the numerical solution does not have a smooth distribution, this is duet to the provided shape functions, which their functions express a straight line (1.3). In order to receive a better result, we can increase the number of elements. Indeed, in Figure 1.6, we can see an improvement.

Case 2: Unsteady state case with mixed boundary condition with time $t \in [0, 2]$.

$$T(0, t) = 5, \quad \left.\frac{dT(x,t)}{dx}\right|_{x=2} = 2$$

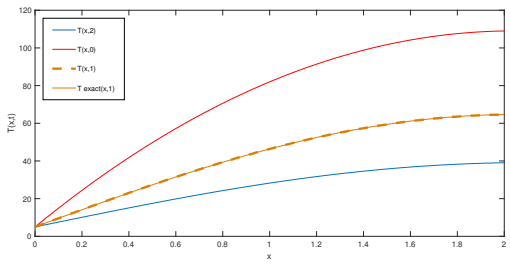


Figure 1.7: Temp. profile evolution

The time step Δt is another important factor to ensure a major accuracy. However, the Matlab command *ode45* is rather reliable although the time step.This is because the command does not use the user-specified time step, it calculates the time step internally. Indeed, in Figure 1.7, we can see the temperature T at different time and the numerical and analytical solution overlapping perfectly. In the next figure, we can see the evolution of the temperature over time.

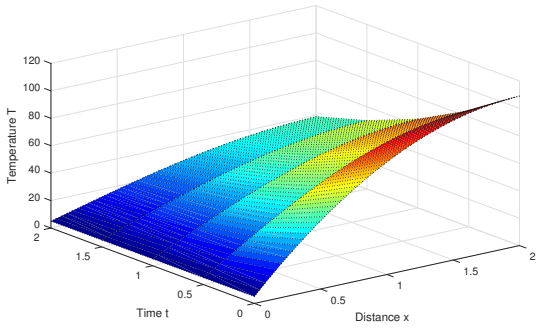


Figure 1.8: Temp. surface change in time

Chapter 2

2D problem

2.1 Steady state problem

We, now, consider the 2 dimensional case. In this problem, we are going through the same procedure as in chapter one, but in a two-dimensional space. Thus, the steady heat equation is

$$-k \bigtriangledown^2 T(x, y) = Q(x, y), \tag{2.1}$$

where $\bigtriangledown^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. We obtain the weak form using Green's Theorem, which is recapitulated below:

$$\iint_\Omega \bigtriangledown \cdot \mathbf{F} \, \mathrm{d}x\mathrm{d}y = \int_{\partial\Omega} \mathbf{F} \cdot \mathbf{n} \, \mathrm{d}s, \tag{2.2}$$

where $\mathbf{F}$ is a vector in 2D, $\mathbf{n}$ is the unit normal vector pointing outward at the boundary $\partial\Omega$ with the line element $\mathrm{d}s$. The second Green's theorem is a corollary of the divergence theorem if we set $\mathbf{F} = v \bigtriangledown u = \left[v\frac{\partial u}{\partial x}, v\frac{\partial u}{\partial y}\right]^T$. Thus, since

$$\begin{aligned} \bigtriangledown \cdot \mathbf{F} &= \frac{\partial}{\partial x}\left(v\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(v\frac{\partial u}{\partial y}\right) \\ &= \bigtriangledown u \cdot \bigtriangledown v + v \bigtriangleup u, \end{aligned}$$

where $\bigtriangleup u = \bigtriangledown \cdot \bigtriangledown u = u_{xx} + u_{yy}$, we obtain

$$\begin{aligned} \iint_\Omega \bigtriangledown \cdot \mathbf{F} \, \mathrm{d}x\mathrm{d}y &= \iint_\Omega \left(\bigtriangledown u \cdot \bigtriangledown v + v \bigtriangleup u\right) \, \mathrm{d}x\mathrm{d}y \\ &= \int_{\partial\Omega} \mathbf{F} \cdot \mathbf{n} \, \mathrm{d}s \\ &= \int_{\partial\Omega} v \bigtriangledown u\mathbf{n} \, \mathrm{d}s = \int_{\partial\Omega} v\frac{\partial u}{\partial n} \, \mathrm{d}s, \end{aligned}$$

where $\mathbf{n} = \left(n_x, n_y\right)^T$ is the unit normal vector , $u_n = \frac{\partial u}{\partial n} = n_x\frac{\partial u}{\partial x} + n_y\frac{\partial u}{\partial y}$ is the normal derivative of u. Therefore, we can rewrite the term on the left side of equation (2.1)

as

$$\int_{\Omega} wk \bigtriangledown^2 T \, \mathrm{d}\Omega = k \int_{\partial\Omega} wT_n \, \mathrm{d}s - k \iint_{\Omega} \bigtriangledown T \bigtriangledown w \, \mathrm{d}x\mathrm{d}y. \tag{2.3}$$

Hence, the equation (2.1) becomes

$$k \iint_{\Omega} \bigtriangledown T \bigtriangledown w \, \mathrm{d}x\mathrm{d}y = \int_{\Omega} Qw \, \mathrm{d}\Omega + k \int_{\partial\Omega} wT_n \, \mathrm{d}s. \tag{2.4}$$

Now, we can approximate the solution using Finite Elements as we did in chapter 1:

$$\begin{aligned} T_e &= \sum_{i=1}^{n+1} T_i N_i(x, y), \\ w &= \sum_{j=1}^{n+1} w_j N_j(x, y), \end{aligned} \tag{2.5}$$

where, again, N_i are the shape functions in the i^{th} node; n is the total number of elements. Upon substituting (2.5) in equation (2.4), we get

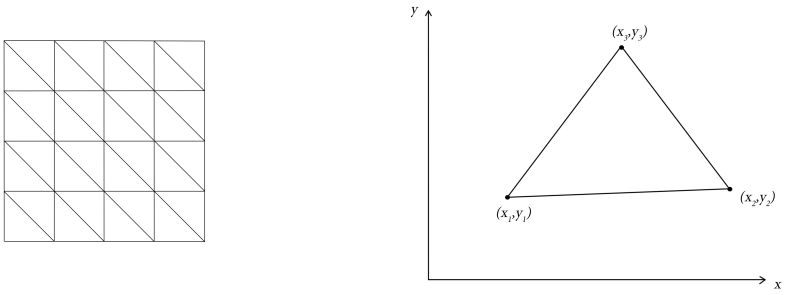
$$\sum_{i=1}^{n+1} \left(\iint_{\Omega} k \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) \mathrm{d}x\mathrm{d}y \right) T_i = \int_{\Omega} QN_j \, \mathrm{d}\Omega + k \int_{\partial\Omega} N_j T_n \, \mathrm{d}s \tag{2.6}$$

The above expression can be rewritten in a matrix notation, as in (1.13)

$$\begin{aligned} K_{ij} &= \iint_{\Omega} k \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) \mathrm{d}x\mathrm{d}y \\ f_j &= \int_{\Omega} QN_j \, \mathrm{d}\Omega \\ B_j &= \int_{\partial\Omega} N_j T_n \, \mathrm{d}s \end{aligned} \quad . \tag{2.7}$$

2.1.1 Linear triangular element

In order to examine a general surface, it is useful to create a mesh of linear element, as triangles or squares. In this way our calculation will adapt much simpler to any complex geometry. Triangles are normally used when we want to mesh a 2D model involving complex geometry with acute corners. In this study, we are going to focus on triangles.
Consider a 2D model in the x-y plane, shown schematically in Figure 2.1 (a). The 2D domain is divided in a proper manner into a number of triangular elements. For each element, we have three nodes at the vertices of the triangle, which are numbered around the element in the counterclockwise direction (Figure 2.1 (b)).



(a) Rectangular mesh

(b) Triangular element

Figure 2.1: 2D Mesh

In this case, we have three different shape functions N_i ($i = 1,2,3$) corresponding to the three nodes of the triangular element. For a linear triangular element, we assume that the shape functions are linear functions of x and y. They should, therefore, have the form

$$N_i = a_i x + b_i y + c_i, \quad i = 1, 2, 3$$

where a_i, b_i and c_i are constants to be determined. Instead, the global coordinates x, y can be defined as

$$x = \sum_{i=1}^{3} N_i x_i, \qquad y = \sum_{i=1}^{3} N_i y_i.$$

The next step is to transform a random triangle to a simpler shape in order to simplify the calculations. We are going to use, again, the isoparametric transformation and we are going to move in the $\xi - \eta$ space. Therefore, the shape functions can be represented simply as

$$N_1 = 1 - \xi - \eta, \quad N_2 = \xi \quad and \quad N_3 = \eta \tag{2.8}$$

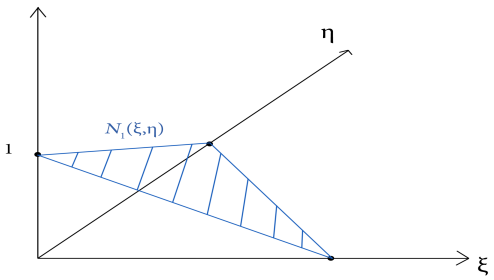


Figure 2.2: Shape function N_1

Notice that

$$N_1 + N_2 + N_3 = 1,$$

which ensures that the domain is not distorted by the chosen shape functions. Also, as in the 1-D case,

$$N_i = \begin{cases} 1 & \text{at node i} \\ 0 & \text{at other nodes} \end{cases}$$

and varies linearly within the element. Thus, our original coordinates, using equation (2.8), can be written as

$$\begin{aligned} x &= (1-\xi-\eta)x_1 + \xi x_2 + \eta x_3 = x_{21}\xi + x_{31}\eta + x_1 \\ y &= (1-\xi-\eta)y_1 + \xi y_2 + \eta y_3 = y_{21}\xi + y_{31}\eta + y_1 \end{aligned} \tag{2.9}$$

where $x_{ij} = x_i - x_j$ and same for y. Now that we have defined the shape functions, we have to adapt them with the new coordinate space in order to compute the element stiffness matrix and the other vectors. Let's rewrite the expression (2.7) for K as follows

$$K_{ij} = k \int_\Omega \begin{pmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial N_j}{\partial x} \\ \frac{\partial N_j}{\partial y} \end{pmatrix} \mathrm{d}\Omega. \tag{2.10}$$

In order to change the shape functions, we have to recall the Jacobian matrix

$$\begin{pmatrix} \frac{\partial N}{\partial \xi} \\ \frac{\partial N}{\partial \eta} \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{pmatrix}}_{J} \begin{pmatrix} \frac{\partial N_j}{\partial x} \\ \frac{\partial N_j}{\partial y} \end{pmatrix}. \tag{2.11}$$

If we calculate the Jacobian matrix with equation (2.9), we obtain

$$J = \begin{pmatrix} x_{21} & y_{21} \\ x_{31} & y_{31} \end{pmatrix}. \tag{2.12}$$

Therefore, the integral in isoparametric coordinates becomes

$$K_{ij} = k \int_0^1 \int_0^{1-\xi} J^{-1} \begin{pmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{pmatrix} J^{-1} \begin{pmatrix} \frac{\partial N_j}{\partial \xi} \\ \frac{\partial N_j}{\partial \eta} \end{pmatrix} |J| \mathrm{d}\eta \mathrm{d}\xi. \tag{2.13}$$

The determinant of the Jacobian matrix is defined as the double triangle's area A

$$|J| = x_{21}y_{31} - x_{31}y_{21} = 2A. \tag{2.14}$$

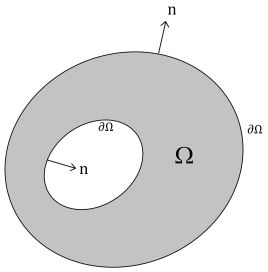
The source vector stays almost the same. Indeed, we must include the determinant of J as a scaling term inside the integral

$$f_j = \int_0^1 \int_0^{1-\xi} Q N_j |J| \mathrm{d}\eta \mathrm{d}\xi. \tag{2.15}$$

Finally, we need to follow the same steps that we already saw in the first chapter (from (1.25)) to get the global matrix and the global vector. Naturally, the localization matrix L is going to have another row because from 2 nodes we passed to 3 nodes for each triangle, for this problem.

2.1.2 Implementation of boundary conditions

Let us focus, now, how boundary conditions influence our computations, as we did in chapter 1. For the Dirichlet boundary conditions, the procedure is exactly the same as we saw in the previous chapter. So, for this case, it is suggested to go back again in section 1.1.2, for the Dirichlet case. On the other hand, when we are dealing with Neumann boundaries, the steps are a little bit different. Assuming a general domain Ω,



the Neumann condition is applied on the boundary $\partial\Omega$, where recalling equation (2.7),

$$B_j = \int_{\partial\Omega} N_j T_n \, \mathrm{d}s.$$

B_j is the boundary term obtained from computing the line integral over the surroundings $\partial\Omega$. Since we are dealing with a line, we can simplify, for the moment, this sub-problem as a 1D case. The line is divided in n elements, which are described by the linear shape function that we used in chapter 1. This approach will lead to a local and global vector for the 1D line. However, in order to have a consistent solution with the 2D geometry, we need to assemble the sub-dimensional vector in the original mesh. Considering an arbitrary boundary line

the shape function that describes the line elements, in the isoparametric form, is

$$N_j = \begin{pmatrix} 1-\xi \\ \xi \end{pmatrix}. \tag{2.16}$$

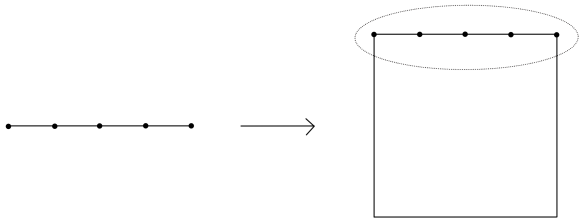
Thus, the line integral can be solved as

$$\int_0^1 N_j T_n |J| \, \mathrm{d}\xi = \frac{dT}{dn} |J| \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \tag{2.17}$$

where, in this case, T_n is considered as a constant and, as we saw in (1.21), $|J|$ is half the length of the local element. The determinant of the Jacobian is

$$|J| = \frac{1}{2}\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \tag{2.18}$$

where i is the i-th node. Here, $|J|$ describes a straight line, so it is important to select a proper number of elements to ensure a better accuracy. After computing the integral, it is essential to generalize the results in the original 2D mesh. To achieve this task, it is important to initialize a vector of zeros $B^{n \cdot mx1}$ and only then assign the value of the line integral where the Neumann condition is applied. In the next Figure, we can see better this process for a rectangular example.



$$B = \begin{pmatrix} b_1 \\ \vdots \\ b_{n+1} \end{pmatrix} \qquad B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n+1} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Where n is the number of elements over the showed line. It is convenient to define in which nodes the Neumann condition is applied, in this way it is easier to move the result of the 1D line into the original dimension.

Robin condition

In the previous chapter, we saw that there are three kind of boundary conditions, until now only two of them were explained. Then, we examine the last condition: Robin condition. Recalling equation (1.9), we have

$$\frac{dT}{dn} = \gamma - T. \tag{2.19}$$

If we put (2.19) in the line integral, we have

$$\int_{\partial\Omega} N_j \left(\gamma - N_i T_i\right) ds = \int_{\partial\Omega} N_j \gamma ds - T_i \int_{\partial\Omega} N_i N_j ds. \tag{2.20}$$

Applying the isoparametric mapping, we get

$$\int_0^1 N_j \gamma |J| d\xi - T_i \int_0^1 N_i N_j |J| d\xi. \tag{2.21}$$

Thus, we finally get

$$\underbrace{\gamma |J| \begin{pmatrix} 1 \\ 1 \end{pmatrix}}_{f_{robin}} - \underbrace{|J| \begin{pmatrix} 2/3 & 1/2 \\ 1/2 & 2/3 \end{pmatrix}}_{K_{robin}} T_i. \tag{2.22}$$

Therefore, we get a local matrix, which is going to change the Stiffness matrix, and a vector, which is going to modify the load body vector.

2.2 Unsteady state problem

Since the time dependent partial differential equation is

$$c\frac{d}{dt}T(x,y,t) - k \bigtriangledown^2 T(x,y,t) = Q(x,y,t), \tag{2.23}$$

we need to include, again, the Mass matrix M in the matrix equation, as in (1.45). Therefore, in 2 dimension, the mass matrix becomes

$$M_{ij} = c \iint_{\Omega} N_i(x,y) N_j(x,y)\, \mathrm{d}x\mathrm{d}y \tag{2.24}$$

and, with isoparametric transformation, we obtain

$$M_{ij} = c \int_0^1 \int_0^{1-\xi} N_i(\xi,\eta) N_j(\xi,\eta)\, |J| \mathrm{d}\eta\mathrm{d}\xi. \tag{2.25}$$

2.3 Building nodes and elements

In order to set our rectangular mesh, we should build two structures: *nodes* and *elements*. *Nodes* describes the nodal coordinates x, y and z for each triangle on the mesh. On the other hand, the second structure, *elements*, specifies the global nodes which are included in each element. The next two figures show the idea behind the two structures.

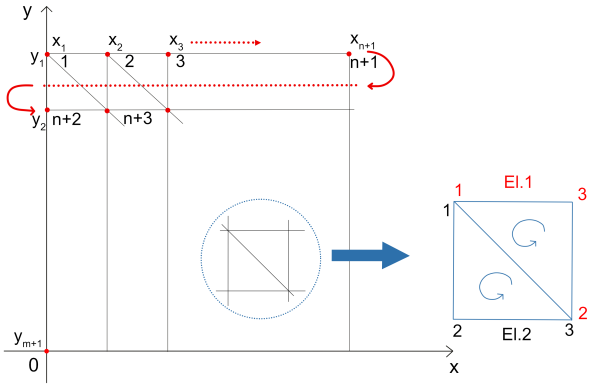
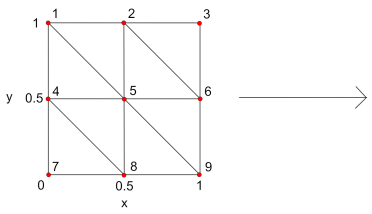


Figure 2.3: Node numbering for 2D space: local and global

In Figure 2.3, m and n are the number of elements for the y and x direction, respectively. In Figure 2.4, it is illustrated how the structures are built from 8 elements mesh with width and height equal to 1.



El.	'1'	'2'	'3'
1	1	5	2
2	1	4	5
3	2	6	3
4	2	5	6
5	4	8	5
6	4	7	8
7	5	9	6
8	5	8	9

nodes	x	y	z
1	0	1	0
2	0.5	1	0
3	1	1	0
4	0	0.5	0
5	0.5	0.5	0
6	1	0.5	0
7	0	0	0
8	0.5	0	0
9	1	0	0

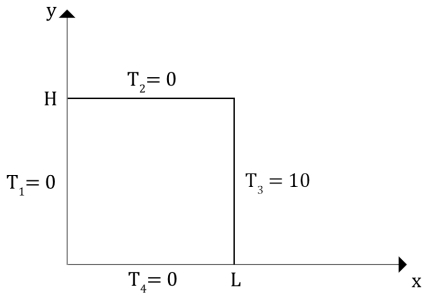
Figure 2.4: *Nodes* and *Elements* structures

2.4 Results for 2D rectangular mesh

For the 2 dimensional case, the analytical solution depend on the boundary conditions, so the procedure will be different for each case. Thus, we are going to solve the PDE for each problem and, afterwards, we are going to analyze it with the numerical solution.

Dirichlet case with steady state condition

As a first case, let us assume a rectangular mesh with the following Dirichlet conditions



where the height $H = 1$ and the width $L = 1$. Now, starting from the PDE

$$-k\left(T_{xx} + T_{yy}\right) = Q, \tag{2.26}$$

where we consider, for this case and the next ones, $k = 1$ and $Q = 50$. In order to solve this equation, we need to compute each single non-homogeneous problem separately and only then, with their respective solutions, we are able to find the general one, by simply summing them. In this case we have only 2 non-hom. cases: the PDE (2.26) and one boundary ($T_3 \neq 0$). Hence, starting from the boundary, let us consider the Laplace eq.:

$$T_{xx} + T_{yy} = 0. \tag{2.27}$$

Proceeding with the separation variable method, we can assume that the solution can be consider as $T = X(x)Y(y)$. We can rewrite equation (2.27) as

$$\frac{X''}{X} = -\frac{Y''}{Y} = -\lambda. \tag{2.28}$$

The only possible solution is when $\lambda < 0$ and, solving the differential equation for $X(x)$ and $Y(y)$, the possible solutions are:

$$\begin{cases} Y = A\cos(\sqrt{\lambda}y) + B\sin(\sqrt{\lambda}y) \\ X = Ce^{\sqrt{\lambda}x} + De^{-\sqrt{\lambda}x} \end{cases} \tag{2.29}$$

where A,B,C and D are unknown constants. Plugging the conditions $Y(0) = Y(H) = X(0) = 0$ in the equation above, we obtain

$$\begin{aligned} A &= 0, \\ \sqrt{\lambda} &= \frac{n\pi}{H} \qquad n = 1, 2, 3, ... \\ C &= -D. \end{aligned} \tag{2.30}$$

Therefore, the general solution

$$T(x, y) = XY = \sum_{n=1}^{\infty} D_n \sin\left(\frac{n\pi}{H}y\right)\sinh\left(\frac{n\pi}{H}x\right) \tag{2.31}$$

is the Fourier series. $D_n = BC$ is called the Fourier coefficient, we can find this ceoff. with the last boundary condition

$$T(L, y) = D_n \sin\left(\frac{n\pi}{H}y\right)\sinh\left(\frac{n\pi}{H}L\right) = T_3 \tag{2.32}$$

and, now, we are able assert that

$$D_n = \frac{2}{H\sinh\left(\frac{n\pi}{H}L\right)}\int_0^H T_3 \sin\left(\frac{n\pi}{H}y\right)\mathrm{d}y = \frac{T_3\left(1-(-1)^n\right)}{n\pi\sinh(n\pi)}. \tag{2.33}$$

The final solution is

$$\boxed{T(x, y) = \sum_{n=1}^{\infty} \frac{T_3\left(1-(-1)^n\right)}{n\pi\sinh(n\pi)} \sin\left(\frac{n\pi}{H}y\right)\sinh\left(\frac{n\pi}{H}L\right)} \tag{2.34}$$

Depending on how many non homogeneous boundary conditions exist, the Laplace equation (2.27) has to be divided into as many sub-problems and then combine them into a general solution. In our case, we have a single non homogeneous boundary, thus we have just one solution for this case. For the heat source (Poisson equation (2.26)), we need to decompose a further sub-problem and to use the superposition principle again to combine them into the final solution of the problem. Contrary to what we did before, now, to solve the Poisson equation, we apply homogeneous boundary conditions: $T_i = 0$, $i = 1, 2, 3, 4$. However, these conditions lead to the trivial solution $T(x, y) = 0$. Therefore, the solution of Poisson's equation with homogeneous boundary conditions is:

$$\left.\begin{aligned} X'' + \mu X &= 0 \\ X(0) &= 0 \\ X(L) &= 0 \end{aligned}\right\} X_m = \sin(\frac{m\pi}{L}x), m = 1, 2, 3, ...$$

$$\left.\begin{array}{l} Y'' + \nu Y = 0 \\ Y(0) = 0 \\ Y(H) = 0 \end{array}\right\} Y_n = \sin(\frac{n\pi}{H}y), n = 1, 2, 3, ...$$

and now the solution is

$$T(x,y) = \sum_{m,n=1}^{\infty} a_{mn} X_m Y_n = \sum_{m,n=1}^{\infty} a_{mn} \sin(\frac{m\pi}{L}x)\sin(\frac{n\pi}{H}y). \tag{2.35}$$

If we substitute the previous equation into the Poisson equation (2.26), we obtain

$$F(x,y) = -\frac{Q}{k} = \sum_{m,n=1}^{\infty} \underbrace{\left[-a_{mn}\left(\left(\frac{m\pi}{L}\right)^2 + \left(\frac{n\pi}{H}\right)^2\right)\right]}_{A_{mn}} \sin(\frac{m\pi}{L}x)\sin(\frac{n\pi}{H}y), \tag{2.36}$$

where A_{mn} is

$$A_{mn} = \frac{\int_0^L \int_0^H F(x,y)\sin(\frac{m\pi}{L}x)\sin(\frac{n\pi}{H}y)\mathrm{d}y\mathrm{d}x}{\int_0^L \int_0^H \sin^2(\frac{m\pi}{L}x)\sin^2(\frac{n\pi}{H}y)\mathrm{d}y\mathrm{d}x}. \tag{2.37}$$

Thus, the value of the constant a_{mn} is

$$a_{mn} = -\frac{4}{LH\left(\left(\frac{m\pi}{L}\right)^2 + \left(\frac{n\pi}{H}\right)^2\right)} \int_0^L \int_0^H F(x,y)\sin(\frac{m\pi}{L}x)\sin(\frac{n\pi}{H}y)\mathrm{d}y\mathrm{d}x. \tag{2.38}$$

Now, if we compute the integral and we plug it into equation (2.35), the solution is

$$\boxed{T(x,y) = \sum_{m,n=1}^{\infty} \frac{4Q((-1)^m - 1)((-1)^n - 1)}{\left(\left(\frac{m}{L}\right)^2 + \left(\frac{n}{H}\right)^2\right)\pi^4 kmn} \sin(\frac{m\pi}{L}x)\sin(\frac{n\pi}{H}y)} \tag{2.39}$$

Therefore, we can assert, finally, that the analytical solution of this problem is the sum of equation (2.34) and (2.39). From now, we are able to use the analytical solution and compare it with the numerical solution of the finite element method.
If we divide in 20 elements in y and x direction, the temperature profile of the numerical solution is as follows

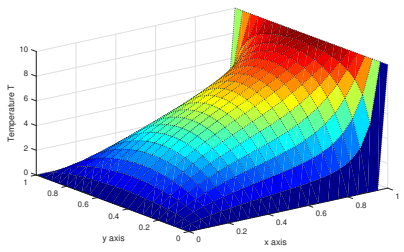


Figure 2.5: Temperature profile

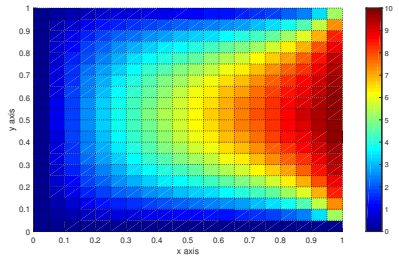


Figure 2.6: Temp. over mesh

Cutting a section of the temperature surface at $y = 0.5$, we can examine the reliability of the approximated solution.

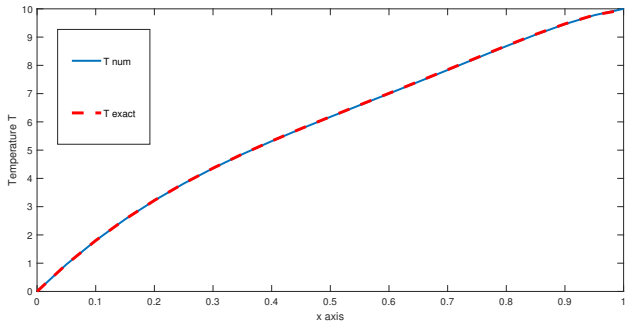


Figure 2.7: Temp. profile at $y = 0.5$

In order to compute the analytical solution, we need to specify the summation's limits m and n, which guarantee the accuracy of the solution. Obviously, a higher value of m and n will give us a more accurate result. In Figure 2.6, these limits are $m = n = 20$.

Unsteady state case

For the unsteady state problem, the analytical solution is more difficult to calculate, especially for non-homogeneous conditions. Thus, for the validation of the unsteady case, we consider the homogeneous Dirichlet conditions as

$$\begin{cases} cT_t(x,y,t) = k(T_{xx}(x,y,t) + T_{yy}(x,y,t)) & t > 0, \\ T(0,y,t) = T(x,0,t) = T(L,y,t) = T(x,H,t) = 0 \\ T(x,y,0) = T_{in}(x,y) \end{cases} \tag{2.40}$$

where $c = 1$ and T_{in} represents the initial solution taken as the solution to the steady state problem. Using the separation of variables, we consider the solution as $T(x,y,t) = X(x)Y(y)\tau(t)$. Thus, if we substitute it in the PDE (2.40), we get

$$\frac{c}{k}\frac{\tau'(t)}{\tau(t)} = \frac{Y''(y)}{Y(y)} + \frac{X''(x)}{X(x)}. \tag{2.41}$$

Since the left side is independent of x and y variables and the right side is independent of t, it follows that the expression must be a constant:

$$\frac{c}{k}\frac{\tau'(t)}{\tau(t)} = \frac{Y''(y)}{Y(y)} + \frac{X''(x)}{X(x)} = \lambda. \tag{2.42}$$

Again, we seek all the possible solutions of λ and the corresponding non zero functions for X,Y and τ. So, we can write

$$\tau'(t) - \frac{k}{c}\lambda\tau(t) = 0 \tag{2.43}$$

and

$$\frac{X''(x)}{X(x)} = \lambda - \frac{Y''(y)}{Y(y)}. \tag{2.44}$$

But, since each side of the equation (2.42) are independent of the other function we can separate them and introduce the constants α and β

$$\begin{aligned} X''(x) - \alpha X(x) = 0 \\ Y''(y) - \beta Y(y) = 0 \end{aligned} \tag{2.45}$$

where $\lambda = \alpha + \beta$. In order to solve the differential equations in (2.44), we substitute temporarily $\alpha = -\mu^2$ and $\beta = -\nu^2$. So, the solution for $X(x)$ is

$$X(x) = C_1 \cos(\mu x) + C_2 \sin(\mu x).$$

Applying the boundary condition, we obtain

$$\mu = \frac{n\pi}{L}$$

and, therefore,

$$\alpha_n = -\mu_n^2 = -\left(\frac{n\pi}{L}\right)^2, \quad X_n(x) = \sqrt{\frac{2}{L}} \sin(\mu_n x), \quad n = 1, 2, 3... \tag{2.46}$$

Going through the same way with $Y(y)$, we get

$$\beta_m = -\nu_n^2 = -\left(\frac{m\pi}{H}\right)^2, \quad Y_n(y) = \sqrt{\frac{2}{H}} \sin(\nu_m y), \quad m = 1, 2, 3... \tag{2.47}$$

Thus, the eigenvalue of the main problem is

$$\lambda_{nm} = -\left(\left(\frac{n\pi}{L}\right)^2 + \left(\frac{m\pi}{H}\right)^2\right) \tag{2.48}$$

and the corresponding eigenfunctions

$$\phi_{nm}(x, y) = \frac{2}{\sqrt{LH}} \sin(\mu_n x) \sin(\nu_m y). \tag{2.49}$$

Now, going back to equation (2.39), its solution is

$$\tau(t) = e^{\frac{k}{c}\lambda_{nm} t}. \tag{2.50}$$

So, rearranging for $T(x, y, t)$

$$\boxed{T(x, y, t) = \frac{2}{\sqrt{LH}} \sum_{m,n=1}^{\infty} C_{nm} e^{\frac{k}{c}\lambda_{nm} t} \sin(\frac{n\pi}{L} x) \sin(\frac{m\pi}{H} y)} \tag{2.51}$$

The final step is to choose the constant C_{nm} so that the initial condition $T(x, y, 0) = T_{in}(x, y)$ is satisfied, so

$$C_{nm} = \frac{2}{\sqrt{LH}} \int_0^L \int_0^H T_{in}(x, y) \sin(\frac{n\pi}{L} x) \sin(\frac{m\pi}{H} y) \mathrm{d}y \mathrm{d}x. \tag{2.52}$$

Mass matrix examination with eigenvalues

It is not always possible to confirm the numerical solution by computing the exact one. However, there is another way to check the reliability of our numerical solution for the time dependency problem: the examination of eigenvalues. In the steady state case, the analytical solution proved that our approximation is trustworthy, consequently the Stiffness Matrix K as well. Now, in the unsteady problem we add just the term of the Mass matrix M, so if we want to check its reliability, let us recall the equation

$$\underline{\underline{M}}\,\underline{\dot{T}} + \underline{\underline{K}}\,\underline{T} = 0. \tag{2.53}$$

This system can be solved by assuming

$$\underline{T}(t) = \underline{x}e^{\delta t}, \tag{2.54}$$

where, comparing with equation (2.50), $\delta = \frac{k}{c}\lambda_{nm}$. Substituting (2.48) into (2.53), we obtain a generalized eigenvalue problem:

$$\left(\delta\underline{\underline{M}} + \underline{\underline{K}}\right)\underline{x} = 0. \tag{2.55}$$

From this equation, we can extract the eigenvalues δ, which will be a diagonal matrix. To verify if the matrix M is reliable, we need to compare the values obtained calculating $\frac{k}{c}\lambda_{nm}$ and the ones coming out from the eigenvalues. If they are approximately equal for each m and n, thus the Mass matrix is reliable. We need to be careful with equation (2.55), because if we want to obtain the proper eigenvalues, we need to take in account just the middle terms of the matrices, i.e. M_{mm} and K_{mm}. Moreover, the number of elements have an important significance on the accuracy of the matrices. Thus, let us see two examples from the case in (2.40): bringing out just the firsts 5 eigenvalues, we have

Eigenvalues		
10 elements	30 elements	$\delta = \frac{k}{c}\lambda_{nm}$
20.228	19.793	$\delta_{11} = 19.739$
51.445	49.580	$\delta_{21} = 49.348$
52.676	49.711	$\delta_{12} = 49.348$
86.546	79.820	$\delta_{22} = 78.956$
108.417	99.762	$\delta_{31} = 98.696$

From the table above, we are considering the cases where we deal with just 10 elements and another one with 30 elements. These results are compared with the ones coming out from δ_{nm}. Taking in account the first try with 10 elements, we can observe that the firsts five values are pretty similar, however from fourth the values start to expand always more. Dividing the sides in 30 elements, the eigenvalues calculated in (2.55) are closer to the original ones. Thus, we can trust on the Mass matrix that we built and from now on it is not required to compute the analytical solution.

However, solving for problem (2.40), we detect unexpectedly a problem: the results blow up if we progress over time.

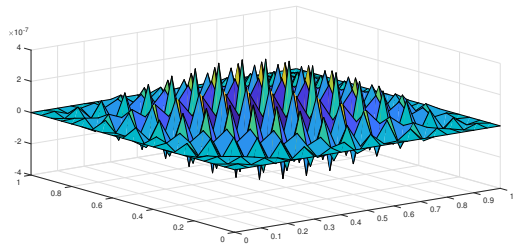


Figure 2.8: Solution blows up in a certain time t

The command *ode45*, inexplicably, doesn't produce the correct solution. Therefore, to overcome this problem, we utilize the time discretization method, which was explained in section 1.3, the Euler method. Setting the explicit Euler method, unfortunately we receive, similarly, the same bizarre results as with *ode45*. On the other hand, with the support of the backward method this problem doesn't appear due to unconditional stability properties. We can see the evolution of the Temperature in Fig. 2.9

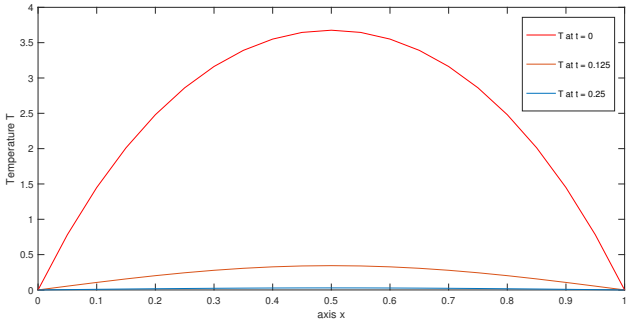
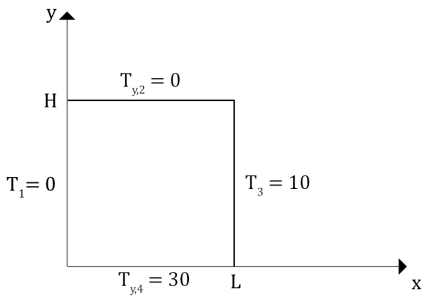


Figure 2.9: Temperature evolution from a section

As we can observe, from the Figure 2.9, the temperature T decays quite quick in a small time. This is because of the constants c and k, which strongly determine heat transfer over the body. The Euler methods are strongly dependent on the step size, in fact the time step is needed to have a small value in order to obtain a reliable discretization. In the case of Figure 2.8, $\Delta t = 0.0001$.

Mixed boundary conditions at steady state

For this example, we analyze the following case



where heat source Q and the conductivity constant k remain the same as in the previous problem. Since the boundary condition changed, we need to focus again in each sub-problem for the analytical solution. There are two Dirichlet cases (homogeneous and non-), two Neumann cases (homogeneous and non-) and a non-homogeneous PDE. Obviously, the homogeneous boundaries will give the banal solution, so we deal with three different cases:

Case 1:

$$\begin{cases} T_{xx} + T_{yy} = 0 \\ T(0, y) = T_y(x, H) = T_y(x, 0) = 0 \\ T(L, y) = T_3 \end{cases}$$

Proceeding with equation (2.28), $\lambda < 0$ and $\lambda > 0$ give none solutions. So, examining just for $\lambda = 0$, the solution is

$$T_1(x, y) = \frac{T_3}{L} x \tag{2.56}$$

Case 2:

$$\begin{cases} T_{xx} + T_{yy} = 0 \\ T(0, y) = T_y(x, H) = T(L, y) = 0 \\ T_y(x, 0) = T_{y,4} \end{cases}$$

The next solution is coming from $\lambda < 0$, where we obtain

$$\begin{aligned} X(x) &= A\cos(\sqrt{\lambda}x) + B\sin(\sqrt{\lambda}x) \\ Y(y) &= C\cosh(\sqrt{\lambda}y) + D\sinh(\sqrt{\lambda}y) \\ Y'(y) &= C\sqrt{\lambda}\sinh(\sqrt{\lambda}y) + D\sqrt{\lambda}\cosh(\sqrt{\lambda}y) \end{aligned}$$

Now, supplying the boundary conditions, $X(0) = X(L) = Y'(H) = 0$,

we get

$$\begin{aligned} A &= 0 \\ \sqrt{\lambda} &= \frac{n\pi}{L}, \text{ where } n = 1, 2, 3, ... \\ D &= -C\tanh(\sqrt{\lambda}H). \end{aligned}$$

Thus, so far, the possible solution can be written as

$$T(x, y) = B_n \sin(\sqrt{\lambda}x)\left(\cosh(\sqrt{\lambda}y) - \tanh(\sqrt{\lambda}H)\sinh(\sqrt{\lambda}y)\right), \tag{2.57}$$

where B_n is the Fourier constant. In order to find this constant, we derive the derivative of y for solution (2.57) and we apply it on the non-homogeneous boundary:

$$T_y(x, 0) = B_n\sqrt{\lambda}sin(\sqrt{\lambda}x) = T_{y,4}. \tag{2.58}$$

Now, if we isolate B_n

$$B_n = \frac{2}{L\sqrt{\lambda}}\int_0^L T_{y,4}\sin(\sqrt{\lambda}x)\mathrm{d}x = \frac{2T_{y,4}}{L\lambda}\left(1 - (-1)^n\right). \tag{2.59}$$

Therefore, the solution of this sub-problem is

$$T_2(x, y) = \sum_{n=1}^{\infty}\frac{2T_{y,4}}{L\lambda}\left(1 - (-1)^n\right)\sin(\sqrt{\lambda}x)\left(\cosh(\sqrt{\lambda}y) - \tanh(\sqrt{\lambda}H)\sinh(\sqrt{\lambda}y)\right). \tag{2.60}$$

Case 3:

$$\begin{cases} T_{xx} + T_{yy} = -\frac{Q}{k} \\ T(0, y) = T_y(x, H) = T_y(x, 0) = T(L, y) = 0 \end{cases}$$

It is always delicate when we try to solve the non-homogeneous PDE. This time, we proceed in a slightly different way, in reference as we did previously. Solving the Laplace equation for $X(x)$, we get

$$X''(x) + \lambda X(x) = 0 \implies X(x) = A\cos(\sqrt{\lambda}x) + B\sin(\sqrt{\lambda}x).$$

Solving for the two boundary conditions, $X(0) = X(L) = 0$, we have

$$\sqrt{\lambda} = \frac{n\pi}{L}.$$

So, now the possible solution can be

$$T(x, y) = Y_n(y)\sin(\sqrt{\lambda}x). \tag{2.61}$$

Normally, at this stage, we obtain a Fourier constant. However, for this problem, we consider the term $Y_n(y)$ a function of y. To find out this function, we need to plug it into the Poisson equation. Therefore, we have

$$-Y_n\left(\frac{n\pi}{L}\right)^2\sin(\frac{n\pi}{L}x) + Y''\sin(\frac{n\pi}{L}x) = a_n\sin(\frac{n\pi}{L}x), \tag{2.62}$$

where the Fourier coefficient a_n is

$$a_n = \frac{2}{L}\int_0^L \left(-\frac{Q}{k}\right)\sin(\frac{n\pi}{L}x)\mathrm{d}x = -\frac{2Q}{kn\pi}\left(1-(-1)^n\right). \tag{2.63}$$

This yields to differential equation

$$Y_n'' - Y_n\left(\frac{n\pi}{L}\right)^2 = \frac{2Q}{kn\pi}\left((-1)^n - 1\right). \tag{2.64}$$

Thus, now, we deal with a classical second order non-homogeneous differential equation. The solution of the differential equation can be written as $Y = Y_{hom} + Y_{part}$. Solving for the homogeneous term, we get

$$Y_{hom} = C_1 e^{\left(\frac{n\pi}{L}\right)^2 y} + C_2 e^{-\left(\frac{n\pi}{L}\right)^2 y}. \tag{2.65}$$

On the other hand, we can guess the particular solution to be

$$Y_{part} = -\frac{2Q\left(1-(-1)^n\right)}{kn\pi}\cdot\frac{1}{\left(\frac{n\pi}{L}\right)^2}. \tag{2.66}$$

Finally, if we apply the boundary conditions, $Y'(0) = Y'(H) = 0$, we obtain

$$Y_n(y) = -\frac{2Q\left(1-(-1)^n\right)}{k(n\pi)^3}. \tag{2.67}$$

Therefore, the solution of this sub-problem is

$$T_3(x,y) = \sum_{n=1}^{\infty} -\frac{2Q\left(1-(-1)^n\right)}{k(n\pi)^3}\sin(\frac{n\pi}{L}x). \tag{2.68}$$

Accordingly, we are able to construct the general solution by summing each sub-problem. So, recalling (2.56), (2.60) and (2.68)

$$\boxed{T(x,y) = T_1(x,y) + T_2(x,y) + T_3(x,y)} \tag{2.69}$$

Now, we are able to verify the numerical solution with the analytical one. Dividing each side with 20 elements, the temperature profile for this problem looks as follow

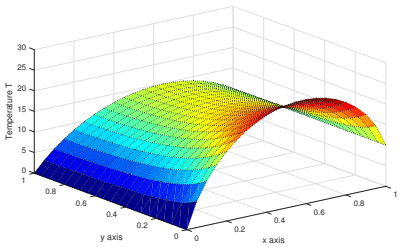


Figure 2.10: Temperature Profile

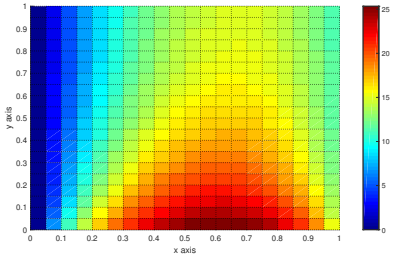


Figure 2.11: Temp. over mesh

and if we focus on a section of our profile, we obtain

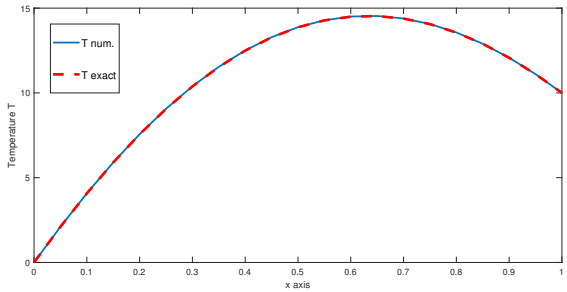


Figure 2.12: Section of the surface at $y = 0.5$

If we want, now, to consider the unsteady state, we need, again, to check the eigenvalues as we saw previously. Even if it is not the same as calculating the analytical solution, we can trust enough by checking the eigenvalues to ensure the reliability of the mass matrix.

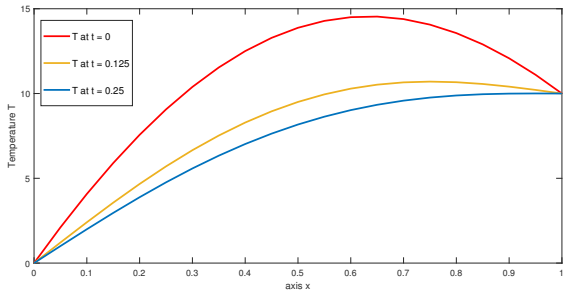


Figure 2.13: Temperature evolution over time

Chapter 3

Surfaces in 3D space

3.1 Rotation matrix

This time, we are not starting with the partial differential equation applied in the 3 dimensional space, which is a straight-forward generalization of the 2D case. In this work, we are not concerned with 3D structures. We are rather going to analyze surfaces, which can be consider a 2D body by using the proper transformation frame. This task can be done by rotating our surface in a new 2 dimensional frame, which is easier to examine.

First, let us see what a rotation frame is by using the figure below

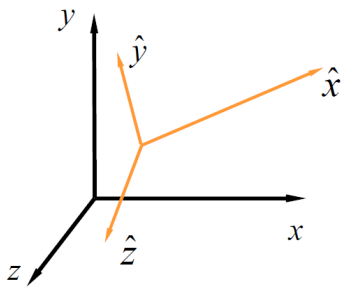


Figure 3.1: Rotation frame

We want to be able to describe the position of an object with the new coordinates system and we can represent it by using the rotation matrix R. The general definition

of R, in 3D, is

$$R = \begin{pmatrix} \cos\theta_x & \cos\varphi_x & \cos\phi_x \\ \cos\theta_y & \cos\varphi_y & \cos\phi_y \\ \cos\theta_z & \cos\varphi_z & \cos\phi_z \end{pmatrix}, \tag{3.1}$$

where θ_x represents the angle between the x' and x axes, θ_y is the angle between the x' and y axes, etc.
Thus, to find the new coordinates, we need to compute

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = R^T \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{3.2}$$

Now that we defined the rotation matrix, we need to study how build to R from the mesh provided. Considering again the triangle element

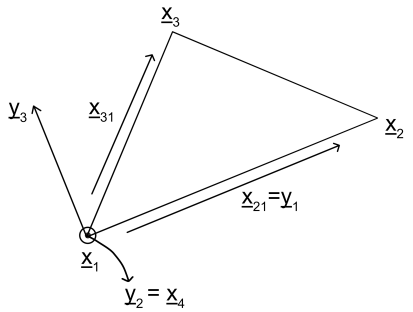


Figure 3.2: New basis frame

we should be able to build a new coordinates system by using the vectors that link the vertices. Obviously, the new basis has to be orthonormal, so

$$\underline{\hat{x}}_{21} = \frac{\underline{x}_2 - \underline{x}_1}{||\underline{x}_2 - \underline{x}_1||} \tag{3.3}$$

$$\underline{x}_4 = \underline{x}_{21} \times \underline{x}_{31} \tag{3.4}$$

$$\underline{\hat{x}}_4 = \frac{\underline{x}_4}{||\underline{x}_4||}. \tag{3.5}$$

Therefore, the new basis can be defined as

$$\underline{\hat{y}}_1 = \underline{\hat{x}}_{21} \qquad \underline{\hat{y}}_2 = \underline{\hat{x}}_4 \qquad \underline{\hat{y}}_3 = \frac{\underline{\hat{x}}_{21} \times \underline{\hat{x}}_4}{||\underline{\hat{x}}_{21} \times \underline{\hat{x}}_4||} \tag{3.6}$$

Finally, the rotation matrix can be build by using the new basis

$$R = \begin{bmatrix} \underline{\hat{y}}_1 & \underline{\hat{y}}_2 & \underline{\hat{y}}_3 \end{bmatrix}^T. \tag{3.7}$$

and applying it, the new coordinates are going to be

$$\begin{pmatrix} x_1' & y_1' & 0 \\ x_2' & y_2' & 0 \\ x_3' & y_3' & 0 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix} \cdot R^T. \tag{3.8}$$

Since we have now the rotation matrix, we are able to implement it in the code. In order to not change much the code, that we obtained in chapter 2, it is useful to apply R on the mesh provided and only then deal with FEM procedure. In this way, we provide the mesh in 2D coordinates, which we already know how to manage it. As an example, let us consider always the square plate, but this time inclined of 45° degree

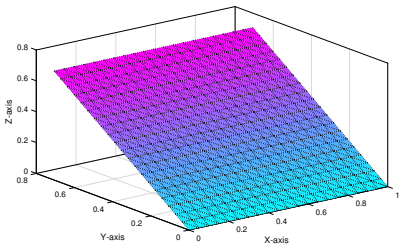


Figure 3.3: Mesh at 45°

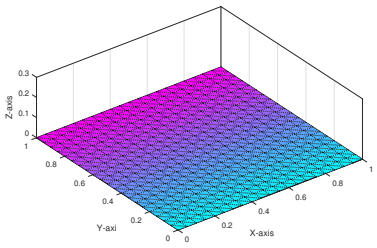


Figure 3.4: Mesh rotated

Applying (3.8), the analysis of the mesh will be the same as we seen in the previous chapter and it should lead to the same result that we saw in the 2D problem. Checking the results of this problem, we can observe that the outcome is exactly the same obtained with the original horizontal plate. This is because the domain has thee same shape, the only difference is that the geometry is rotated.

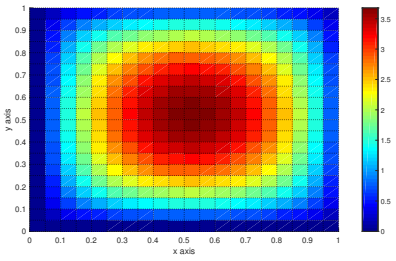


Figure 3.5: Temperature over the mesh

3.2 Curved surfaces

In this section, we observe curved surfaces, which can adapt better for more general complex shapes. A good simplified reference might be a regular cylindrical section, which can be the basis of any more elaborated body.

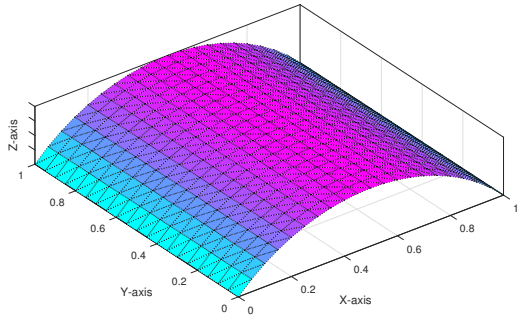


Figure 3.6: Curved Mesh

Unlike the example of the rotating square, here is crucial to rotate each element in the new coordinate frame because the element's area is different from each other. Differently from before, where we could simply rotate the entire body, now we build a rotation matrix for each element. Thus, applying the proper transformation to each triangle, we will obtain the classical 2D problem. In FEM, it is convenient to use this rotation procedure for each element, in order to deal again with a two coordinates space, as shown in (3.8), for the assembly process. So, in comparison with the 2D case, here, just the element initialization is modified by applying the rotation matrix. Applying the same mixed boundary conditions we saw in the second case of section 2.1.5, the temperature obtained is

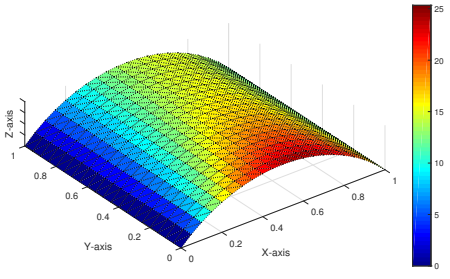


Figure 3.7: Temp. over the surface

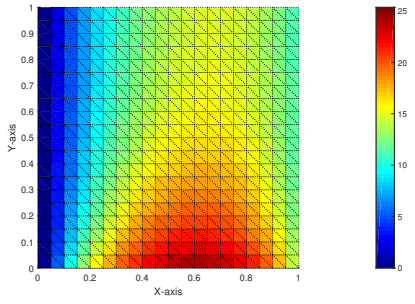


Figure 3.8: Top view of Temp. over the mesh

Chapter 4

Heat transfer

In thermodynamics it distinguish three different mechanisms of heat transfer:

Heat conduction $\dot{Q}''_{cond}$,

Heat convection $\dot{Q}''_{conv}$,

Heat radiation $\dot{Q}''_{rad}$.

In this case, the double apostrophe, over the heat transfer symbol, means that we are considering a heat flow over area. Thus, the unit measure of $\dot{Q}''$ is $\left[\frac{W}{m^2}\right]$.
If we want to introduce the other methods of heat transfer, we need to find a new expression that describes these three phenomenas together. Examining an infinitesimal square surface

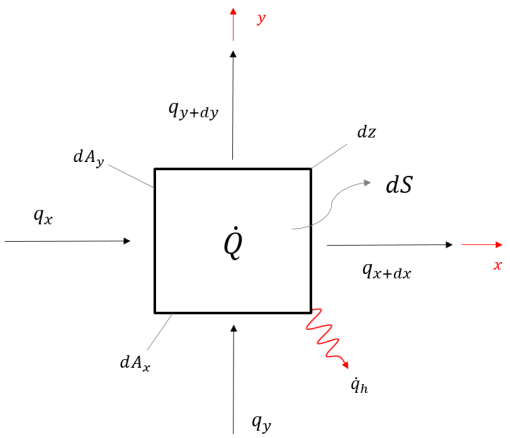


Figure 4.1: Infinitesimal square surface

we are able to derive, from the energy balance, the governing partial differential equation

$$\dot{E}_{in} - \dot{E}_{out} + \dot{E}_{source} = \dot{U} = \frac{\partial}{\partial t}\left(dmc_pT\right). \tag{4.1}$$

Going on, we are able to write

$$\left(q_x dA_y + q_y dA_x\right) - \left(q_{x+dx} dA_y + q_{y+dy} dA_x + \dot{q}_h dS\right) + \dot{Q} dV = \rho dV c_p \frac{\partial T}{\partial t}, \tag{4.2}$$

where $dV = dxdydz$, $dA_x = dxdz$, $dA_y = dydz$ and $dS = dxdy$. Since the heat conduction rate may be expressed as

$$q_{x_i} = -k\frac{\partial T}{\partial x_i}, \qquad q_{x_i+dx_i} = q_{x_i} + \frac{\partial q_{x_i}}{\partial x_i} dx_i. \tag{4.3}$$

Substituting (4.3) in (4.2) and dividing the equation by dV, the final governing PDE is

$$\boxed{k\frac{\partial^2 T}{\partial x^2} + k\frac{\partial^2 T}{\partial y^2} + \dot{Q} - \frac{\dot{q}_h}{dz} = \rho c_p \frac{\partial T}{\partial t}} \tag{4.4}$$

Heat conduction

Until now, in this study we have dealt with heat conduction, which, also known as Fourier's law, claims that the time rate of the heat transfer through a material is

$$\underline{\dot{Q}}''_{cond} = -k\underline{\nabla} T, \tag{4.5}$$

where k is the conductivity term. The minus sign implicates that the heat transfer is in the direction of the decreasing temperature. Besides, the gradient of the temperature tells us that the heat flux is flowing perpendicularly from the tangent plane to the surface at a point.

Heat convection

When a fluid, gas or a liquid, is heated and then travels away from the source, it transports the thermal energy along. This type of heat transfer is convection. The equation for convection is

$$\dot{Q}''_{conv} = \alpha\left(T - T_\infty\right), \tag{4.6}$$

where α is the convection constant $\left[\frac{W}{m^2 T}\right]$ and T_∞ is the ambient temperature. The magnitude of the convection constant specifies how well heat is flowing through a fluid. Observing the value's sign of the heat flow rate term is very important to understand where the heat is flowing. Conventionally, heat flows in direction of the lower temperature. So, if $\dot{Q}''_{conv} > 0$ means that temperature of the body is higher than the fluid's temperature, thus heat is released from the body. On the other hand, if $\dot{Q}''_{conv} < 0$ means that heat is absorbed from the body.

Heat radiation

Thermal radiation is the transfer of internal energy in the form of electromagnetic waves. These waves transport the energy away from the emitting body. The heat radiation is posed by

$$\dot{Q}''_{rad} = \epsilon\sigma \left(T^4 - T_\infty^4\right), \tag{4.7}$$

where ϵ is the emissivity constant,it measures the effective ability of a material to absorb or emit thermal radiation from its surface; σ is the Stefan-Boltzmann constant. Therefore focusing on the boundary of the body, which we are observing, we can combine the heat transfers

$$\rho c_p \frac{dT}{dt} = k \bigtriangledown^2 T + \dot{Q}_{source} - \frac{\alpha}{dz}(T - T_\infty) - \frac{\epsilon\sigma}{dz}\left(T^4 - T_\infty^4\right). \tag{4.8}$$

4.1 Heat convection

Let us see, now, how to include heat convective flow in FEM. Dealing again with the square plate, convection can be applied whether on the boundaries or on the whole domain. For now, focusing on the first case, i.e. we are dealing with Robin condition:

$$\dot{Q} = -k\frac{dT}{dn} = \alpha(T - T_\infty). \tag{4.9}$$

Recalling the line integral, we can substitute equation (4.9) into the last term of (2.7)

$$k\int_{\partial\Omega} N_j T_n \, \mathrm{d}s = k\int_{\partial\Omega} N_j \left(-\frac{\alpha}{k}(T - T_\infty)\right) \mathrm{d}s. \tag{4.10}$$

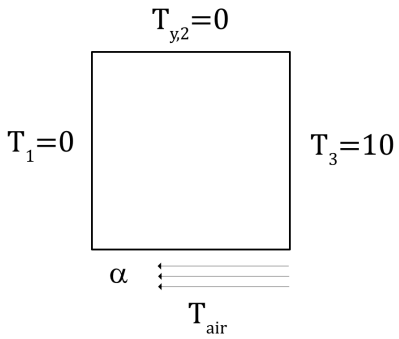
Subsequently, proceeding with the discretization, we obtain

$$\underbrace{\alpha T_\infty \int_{\partial\Omega} N_j \, \mathrm{d}s}_{f_{conv}} - \underbrace{\alpha \int_{\partial\Omega} N_j N_i \, \mathrm{d}s}_{K_{conv}} T_i. \tag{4.11}$$

As we can see from the last equation, we will obtain a matrix which is going to change the Stiffness matrix. Instead, the other term, of the equation above, is the heat source vector. Thus, we summarize the matrix equation as

$$[K + K_{conv}]\, T = f + f_{conv}. \tag{4.12}$$

Let us see, now, how the convection actually influence the results. Having the following geometry and boundaries



and setting $L = 1$, $H = 1$, $f = 50$, $\alpha = 20$, $k = 1$ and $T_\infty = 0$, we obtain

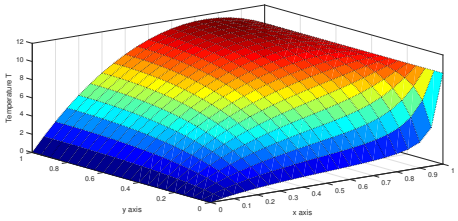


Figure 4.2: Temp. surface

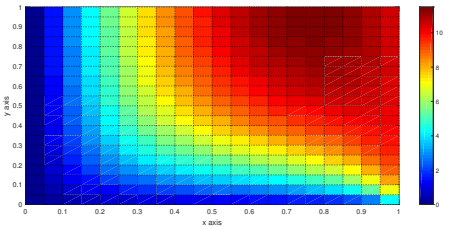


Figure 4.3: Top view of Temp. over the mesh

As we can see from the figures, convection influences clearly dropping the temperature on the bottom side of the geometry.

On the other hand, if we are interested in applying the convective heat flow directly on top of the domain, the calculation is slightly different. This time we will not consider the integral over a line but over the entire domain

$$\dot{q}_h = \int_\Omega N_j \left(\alpha \left(T - T_\infty\right)\right) \, \mathrm{d}\Omega, \tag{4.13}$$

which becomes

$$\underbrace{\frac{\alpha}{dz} \int_\Omega N_j N_i \, \mathrm{d}\Omega \, T_i}_{K_{conv}} - \underbrace{\frac{\alpha T_\infty}{dz} \int_\Omega N_j \, \mathrm{d}\Omega}_{f_{conv}}. \tag{4.14}$$

When we combine conduction and convection on the domain, the matrix notation becomes

$$[K + K_{conv}]\, T = f + f_{conv} \tag{4.15}$$

To test convection, let us take again the previous geometry with the same inputs. But this time we set a constant temperature equal to zero all over the boundaries.

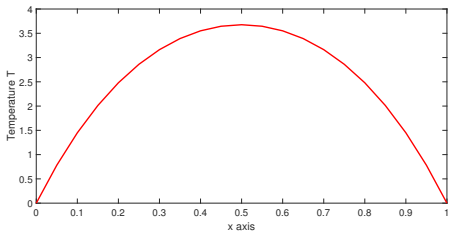


Figure 4.4: Temp. with no convection

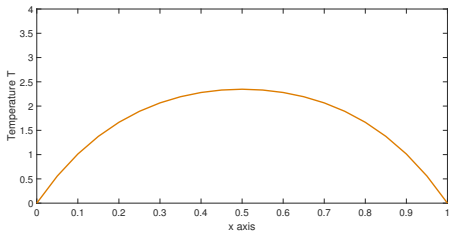


Figure 4.5: Temp. with convection

Examining a cross section sample, it is interesting to see how temperature decrease over the mesh, when convection is applied. The environment temperature T_∞ has great impact in convection, greater is the temperature difference between the surface and the fluid greater effect will have the convective flow.

4.2 Heat radiation

Now, we analyze the radiation heat flow in this study. Radiation makes the problem a bit harder to calculate, because the temperature is raised to the 4^{th} power, which means that now we are dealing with a non-linear problem, as we can see from the calculations:

$$\dot{q}_h = \int_\Omega N_j \left(\epsilon\sigma \left(T^4 - T_\infty^4\right)\right) \, \mathrm{d}\Omega \tag{4.16}$$

Applying the approximation for T, we get

$$\dot{q}_h = \epsilon\sigma \int_\Omega N_j \left(\left(\sum_{i=1}^{3} N_i T_i\right)^4 - T_\infty^4\right) \mathrm{d}\Omega. \tag{4.17}$$

The two terms raised to the 4^{th} power are scalars and N_j is a vector, which means the output is a vector as well. From here, it is convenient to expand the term with the summation symbol and afterwards evaluate the integral for each node of the mesh. Therefore, globalizing the vector, the matrix form equation will be as follow

$$KT + H(T) = f, \tag{4.18}$$

where H(T) is the assembly vector coming out from (4.13). This term is dependent on nodes' temperature, which convert the equation in a non-linear problem. In order to solve this problem, Matlab supports us with different commands. For this case, we borrow the command *fsolve*. Hence, setting the square geometry with zero temperature at the boundary and the given constants are $T_\infty = 100$, $Q = 50$, $\sigma = 5.670373 \cdot 10^{-8}$ and $\epsilon = 0.5$; we get

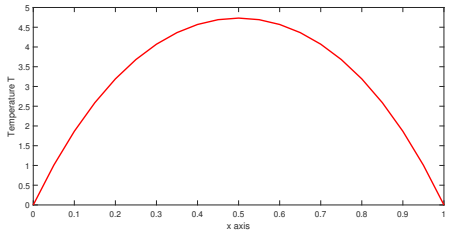


Figure 4.6: Temp. with no radiation

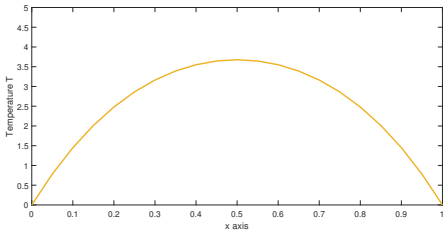


Figure 4.7: Temp. with radiation

Analyzing again a cross section sample, we can observe the emission of the surface when radiation is taken in account. Obviously, as for convection, the temperature difference has a great importance for radiation as well.

4.3 Real combined thermal problem

As a conclusion, we can examine a real material (with its own properties) exposed to the three form of thermal transfer and observe how temperature behave on this surface. Let us explore a copper cylindrical slice surface of Radius 1 m.

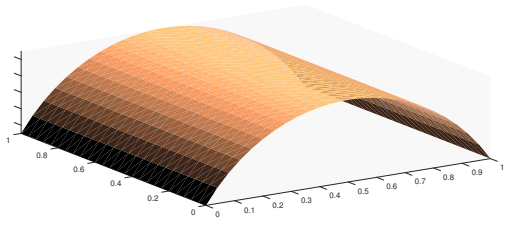


Figure 4.8: Cylindrical slice of copper

The surface, at the beginning, is constantly warmed up with $\dot{Q}''' = 1000\frac{W}{m^2}$. At the two curved sides heat is flowing out, on the other two sides is kept at constant ambient temperature of $25°C$. The properties of Copper and air are:

Property	
Conductivity	$k = 390 \left[\frac{W}{mK}\right]$
Slice's thickness	$b = 0.1\ [m]$
Density	$\rho = 8920 \left[\frac{kg}{m^3}\right]$
Specific Heat	$c_p = 386 \left[\frac{J}{kgK}\right]$
Emissivity(Oxidized)	$\epsilon = 0.65$
Convective const.	$\alpha = 50 \left[\frac{W}{m^2K}\right]$
Stefan's const.	$\sigma = 5.670373 \cdot 10^{-8} \left[\frac{W}{m^2K^4}\right]$

Hence, to recap the mathematical representation, the PDE is described as

$$\rho c_p \frac{dT}{dt} - k \bigtriangledown^2 T = \dot{Q}_{source} - \frac{\dot{q}_{conv}}{b} - \frac{\dot{q}_{rad}}{b}, \tag{4.19}$$

where heat convection and radiation are defined as in (4.6) and (4.7), respectively. On the other hand, if we move forward, the discretization leads to the matrix form:

$$M\dot{T} + [K + K_{conv}]\, T + H(T) = f + f_{conv} + f_{rad}. \tag{4.20}$$

Applying these conditions, from the simulation, at time $t = 0$, we get

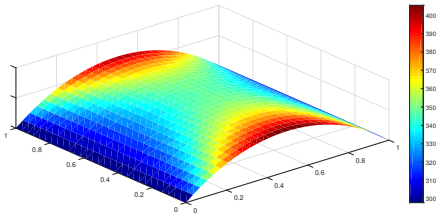


Figure 4.9: Temp. distribution over mesh

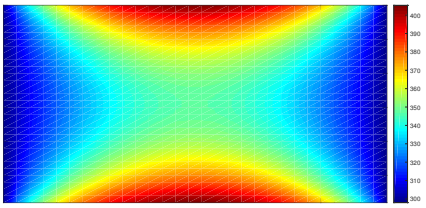


Figure 4.10: Temp. contour

As we can observe, the hotter parts are in the curved sides, where heat is flowing in, which is the same in both. Thus, the temperature distribution is exactly symmetric. The temperature goes from 298 (blue part) to around $400K$ (red side). In the middle of the surface, the temperature is approximately 50 K lower from hotter sides. Since the body is warmer than the air temperature, convection and radiation will cool down the slice. In the next Figure, we can appreciate the difference, in a cross section along y-axis, between the temperature distribution with and without radiation and convection.

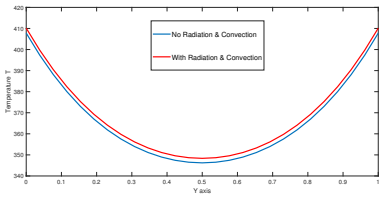


Figure 4.11: Temperature profile in cross section

Bibliography

[1] S. C. Chapra, R. P. Canale, *Numerical Methods for Engineers*, 5^{th} ed., Mc Graw-Hill, 2006.

[2] J.N. Reddy, *An introduction finite element analysis*, Oxford University press, 2004.

[3] Dr. Garth N. Wells, *The finite element method: an introduction*, G.N Wells 2009.

[4] F. P. Incropera, D. P Dewitt, T. L. Bergman, A. S. Lavine, *Fundamentals of Heat and Mass Transfer*, 6^{th} ed., John Wiley and Sons, 2007.

[5] A. F. Mills, *Heat Transfer*, 2^{nd} ed., Prentice Hall, 1999.

[6] Y. Cengel, *Heat and Mass Transfer – A Practical Approach*, Mc Graw-Hill, 2006.

[7] Dr. Noemi Friedman, *Numerical methods for PDEs FEM-implementation*, Lecture notes,Technische Universität Braunschweig.

[8] Zhilin Li, Zhonghua Qiao and Tao Tang, *Numerical Solution of Differential Equations*, 2012.

[9] Bastian Pentenrieder, *Finite Element Solutions of Heat Conduction Problems in Complicated 3D Geometries Using the Multigrid Method*, Diplom Thesis, Technische Universität München, 2005.

[10] G. P. Nikishkov, *INTRODUCTION TO THE FINITE ELEMENT METHOD*, Lecture notes, University of Aizu, 2004.

[11] Quentin Parsons, *Numerical Approximation of the Ohta-Kawasaki Functional MATLAB Implementation Notes*, Thesis, Oxford University, 2012.

[12] ThorstenW. Becker and Boris J. P. Kaus, *Numerical Modeling of Earth Systems*, Lecture notes, University of Southern California, 2017.

[13] Dr. Mohammed HamzaAbdulsada, *Finite Element Analysis of Fins with Convection and Radiation Heat Transfer*, Research report, Al-Qadissyah University, 2014.

[14] Carlos Gonzalez, *What's the difference between conduction, convection, radiation*, Article in "MachineDesign", 2015.

[15] Dave Gilliam, *Heat & Wave Equation in a Rectangle*, Lecture notes, Texas Tech University.

[16] Xu-Yan Chen, *Separation of Variables for Higher Dimensional Heat Equation*, Lecture notes, Georgia Institute of Technology.

[17] Prof. James Vickers, *Solutions using Fourier Series*,Lecture notes, University of Southampton.

[18] Robert Piché, *Partial Differential Equations*, Lecture notes, Tampere University of Technology.

[19] Anthony Peirce, *More Rectangular Domains: Neumann Problems, mixed BC, and semi-infinite strip problems*, Lecture notes, University of British Columbia.

[20] Prof. Dacian N. Daescu, *Nonhomogeneous Problems*, Lecture notes, Portland State University.

[21] Weijiu Liu, *Introduction to Partial Differential Equation*, Lecture notes, University of Central Arkansas.

[22] Victor P. PikulinStanislav I. Pohozaev, *Equations in Mathematical Physics*, Springer Basel AG 2001.

[23] Timothy Sauer, *Numerical Analysis*, Pearson Addison Wesley, 2006.

[24] Dr. Tamás Szabó, *Chapter 7.Numerical Methods for Initial Value Problems*, Lecture notes, University of Miskolc