PAT: finite element solver for the heat transfer analysis of infrastructures subjected to environmental actions

User Guide

Noemi Schclar Leitão
Laboratório Nacional de Engenharia Civil
Lisbon, Portugal
nschclar@lnec.pt

Eloísa Castilho
Instituto Superior Técnico
Lisbon, Portugal
eloisa.castilho@tecnico.ulisboa.pt

1 Introduction

`PAT` is an open source code developed by the authors for the thermal analysis of infrastructures subjected to environmental actions. It is written in FORTRAN 95, closely following the structured programming style proposed by Smith and Griffiths [1] and using its companion library of subroutines freely available in [2]. Therefore, to fully understand the philosophy behind the code, the adopted procedures, as well as the general subroutines, the authors highly recommend to see [1].

2 Bridging the gap between FEM and solar radiation formulations

2.1 Mesh orientation

Since solar irradiance depends on the orientation of the sloped surfaces, it is necessary to define the orientation of the global axes. To this aim, the code works with fixed global axes orientation as follows: in 2D problems the y axis must point to the Zenith and the global azimuth is measured with respect to the normal vector pointing outwards from the plane, in 3D problems the z axis must point to the Zenith and the global azimuth is measured with respect to the y axis.

The other necessary geographical data is the Earth's latitude. Both angles, called `azimuth` and `phi` respectively, must be given in decimal degrees.

2.2 Boundary integrals and unit normal vectors

In order to keep the same procedures and subroutines already implemented for domain integration in [1], the concept of an associated line or surface element was used for the numerical integration of the boundary integrals resulting from boundary conditions.

As this program was developed for concrete dam models, only elements associated to 2D quadrilateral and 3D hexahedron were implemented. This results from the fact that these elements naturally suit the representation of horizontal concreting lifts and vertical joints. For other parent elements the implementation is straightforward.

The edge or face of the parent element is identified by the variable `iside` as shown in Figure 1.

The subroutines `surface` and `num_surface` define the necessary data for the associated elements. To facilitate the understanding, the arrays of the associated element keep the same name adopted in [1] plus the termination "`_s`". For example `num_s` is the node numbers vector of the associated element.

Before, to perform their numerical integration, the integrals over the edge or the face of the parent 2D or 3D element, respectively, are transformed to a local coordinate system by

$$\int_{-1}^{1} f(\xi)|\mathrm{G}|d\xi \qquad \text{or} \qquad \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta)|\mathrm{G}|d\xi d\eta \tag{1}$$

where $|\mathrm{G}|$ is the norm of the outward normal vector $\vec{g}$ given in 2D problems by

$$\begin{Bmatrix} g_x \\ g_y \end{Bmatrix} = \begin{Bmatrix} \dfrac{\partial y}{\partial \xi} \\ -\dfrac{\partial x}{\partial \xi} \end{Bmatrix}, \quad |\mathrm{G}| = \sqrt{\left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2} \tag{2}$$

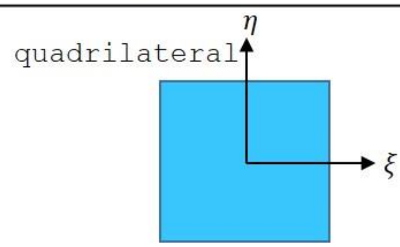
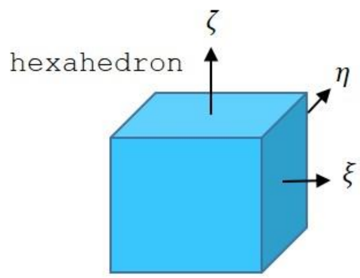
<table>
<tr><th>Parent element</th><th>side</th><th>iside</th></tr>
<tr><td>
</td><td>$\xi = -1$
$\xi = +1$
$\eta = -1$
$\eta = +1$</td><td>1
2
3
4</td></tr>
<tr><td>
</td><td>$\xi = -1$
$\xi = +1$
$\eta = -1$
$\eta = +1$
$\zeta = -1$
$\zeta = +1$</td><td>1
2
3
4
5
6</td></tr>
</table>

Figure 1 - Boundary identification

and in 3D problems by

$$\begin{Bmatrix} g_x \\ g_y \\ g_z \end{Bmatrix} = \begin{Bmatrix} \dfrac{\partial y}{\partial \xi}\dfrac{\partial z}{\partial \eta} - \dfrac{\partial z}{\partial \xi}\dfrac{\partial y}{\partial \eta} \\ \dfrac{\partial z}{\partial \xi}\dfrac{\partial x}{\partial \eta} - \dfrac{\partial x}{\partial \xi}\dfrac{\partial z}{\partial \eta} \\ \dfrac{\partial x}{\partial \xi}\dfrac{\partial y}{\partial \eta} - \dfrac{\partial y}{\partial \xi}\dfrac{\partial x}{\partial \eta} \end{Bmatrix} \tag{3}$$

$$|\mathrm{G}| = \sqrt{\left(\frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial z}{\partial \xi}\frac{\partial y}{\partial \eta}\right)^2 + \left(\frac{\partial z}{\partial \xi}\frac{\partial x}{\partial \eta} - \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta}\right)^2 + \left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta}\right)^2}$$

where the partial derivatives are the components of the Jacobian matrix of the associated line or surface element. The entries of the reduced Jacobian matrix, `jac_s`, are calculated using the derivatives of the associated element shape functions with respect to the local coordinates, `der_s`, and the global coordinates of the associated element nodes, `coord_s`.

Finally the integrals are numerically evaluated using Gauss-Legendre quadrature over the line or quadrilateral region.

In order to compute the surface orientation of each integration point, that is the tilt angle `yy` and the azimuth `beta`, the unit normal vectors components obtained from (2) and (3) by

$$\vec{n} = \frac{\vec{g}}{|G|} \tag{4}$$

are used.

For the 2D case the tilt angle is obtained with the arc-cosine of the n_2 component and the azimuth is computed as the global azimuth plus $\pi/2$ if $n_1 \leq 0$ or $3\,\pi/2$ if $n_1 > 0$.

For the 3D case the integration point azimuth and tilt angles are computed by

```
beta=azimuth+(pi*0.5_iwp-atan2(n2,n1))

yy=acos(n3)
```

where `n1`, `n2` and `n3` are the unit vector components and `pi` is set to π. The second term of the right hand side of `beta` allows to obtain the correct sign of the angle φ, Figure 2.

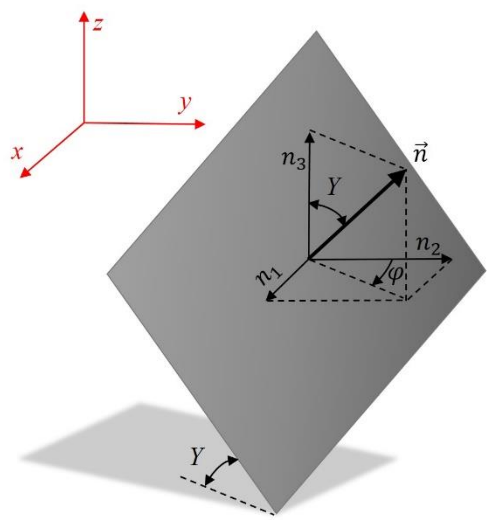


Figure 2 – Integration point orientation

2.3 The counting of time

The environmental actions, mainly solar radiation modelling, depend on the time of the day and the day of the year when they happen. Therefore, the simple consideration that the problem starts at time equal to zero, does not give the necessary information for applying boundary conditions. To solve this problem, the Julian day numbering scheme is used throughout the program `PAT`.

The Julian day numbers, used by astronomers, express the number of days that have elapsed since the Greenwich mean noon of January 1st 4713 BC, which is midday as measured on the Greenwich meridian on January 1st of that year. In this way, the Julian day number is a continuous count of days and fractions thereof from the beginning of the year 4713 BC. It is important to note that each new Julian day begins at 12h 00m UT (Universal Time), half a day out of step with the civil day in time zone 0 [3].

Another advantage of using Julian day numbering is that all dates on input and output data can be given as civil dates which simplify the use of monitoring data.

If the user opts for another unit of time, variable `ucte` will convert the counting of time from days to the adopted unit.

Algorithms for obtaining the Julian day from the ordinary year, month and day exist in the literature and online, in this work the algorithms of [4] were implemented.

Converting civil date to Julian day number (subroutine `julian_day`)

1. Set Y = year, M = month and D = day
2. If M = 1 or 2 subtract 1 from Y and add 12 to M
3. Dropping the fractional part of all results of all multiplications and divisions, let:
   - $A = Y/100$
   - $B = A/4$
   - $C = 2 - A + B$
   - $E = 365.25 \times (Y + 4716)$
   - $F = 30.6001 \times (M+1)$
   - $JD = C + D + E + F - 1524.5$

Converting Julian day number to civil date (subroutine `civil_date`)

1. $Q = JD + 0.5$
2. Set Z = integer part
3. Dropping the fractional part of all results of all multiplications and divisions, let:
   - $W = (Z - 1\,867\,216.2)/36\,524.25$
   - $A = Z + 1 + W - (W/4)$
   - $B = A + 1524$
   - $C = (B - 122.1)/365.25$
   - $D = 365.25 \times C$
   - $E = (B - D)/30.6001$
   - $F = 30.6001 \times E$
4. Day of month = $B - D - F + (Q - Z)$ (including the decimal fraction of the day)
5. Month = $E - 1$ if E is less than 13.5, or = $E - 13$ if E is more than 13.5
6. Year = $C - 4716$ if month is more than 2.5, or = $C - 4715$ if month is less than 2.5

Computing the day of year (subroutine `day_of_year`)

The day of year `d` is defined as the sequential day number starting with day 1 on January 1st. Thus, the Julian day numbers can be obtained by subtracting the Julian day number of December 31st of the previous year from the current Julian day number.

3 Obtaining results in experimental or monitoring points

In cases where experimental or monitoring values are available, the user can choose to compute values in those specific points.

The interpolation technique using FE shape functions was adopted in `PAT`. The main advantage of this approach is that it makes no assumptions other than those already introduced in the finite element model [5].

The strategy implemented in the subroutine `find_points` consists of the following steps:

1. Identify the nearest mesh node to the given data point;
2. For each element containing the nearest mesh node, compute the local coordinates of the data point (ξ_p, η_p, ζ_p);
3. If the point falls inside the domain of the element, that is -1 $\leq \xi_p \leq 1$, -1 $\leq \eta_p \leq$ 1 and -1 $\leq \zeta_p \leq 1$, the element is the owner element and the local coordinates of the data point are (ξ_p, η_p, ζ_p).

The computation of the local coordinates from the global coordinates of the data points, called inverse mapping, is a non-trivial operation and it is obtained using the Newton-Raphson method.

The shape functions $N_i(\xi, \eta, \zeta)$ used to interpolate the coordinate vectors define a one-to-one mapping from the local (ξ, η, ζ) coordinate system to the global (x, y, z) coordinate system, such that

$$\mathbf{x}(\boldsymbol{\xi}) = \sum_{a=1}^{NOD} N_a(\boldsymbol{\xi})\, \mathbf{x}_a^e \tag{5}$$

where $\boldsymbol{\xi} = [\xi, \eta, \zeta]^{\mathrm{T}}$ denotes the local coordinate vector and $\mathbf{x}_a^e = [x_a^e, y_a^e, z_a^e]^{\mathrm{T}}$ are the global coordinates of the element nodes (i.e., vertices of the element within which the point **P** is located).

The inverse of this mapping involves a system of nonlinear equations which can be numerically calculated using an iterative technique in order to minimize the objective functions

$$\mathbf{F}(\boldsymbol{\xi}) = \mathbf{x}_p - \mathbf{x}(\boldsymbol{\xi}) = \mathbf{x}_p - \sum_{a=1}^{NOD} N_a(\boldsymbol{\xi})\, \mathbf{x}_a^e \;=\; \mathbf{0} \tag{6}$$

where $\mathbf{F}(\boldsymbol{\xi})$ denotes the entire vector of functions F_i.

In the neighborhood of point **P**, each of the functions F_i can be expanded in a Taylor series (see [6] §9.6)

$$F_i(\boldsymbol{\xi} + \delta\boldsymbol{\xi}) = F_i(\boldsymbol{\xi}) + \sum_{j=1}^{3} \frac{\partial F_i}{\partial \xi_j} \delta\xi_j \;+\; O(\delta\boldsymbol{\xi}^2) \tag{7}$$

The matrix of the partial derivatives appearing in equation (7) is the Jacobian matrix **J**:

$$J_{ij} \equiv \frac{\partial F_i}{\partial \xi_j} \tag{8}$$

In matrix notation, equation (7) is

$$\mathbf{F}(\boldsymbol{\xi} + \delta\boldsymbol{\xi}) = \mathbf{F}(\boldsymbol{\xi}) + \mathbf{J}.\,\delta\boldsymbol{\xi} + O(\delta\boldsymbol{\xi}^2) \tag{9}$$

By neglecting terms of order $\delta\boldsymbol{\xi}^2$ and higher and by setting $\mathbf{F}(\boldsymbol{\xi} + \delta\boldsymbol{\xi}) = 0$, we obtain a set of linear equations for the corrections $\delta\boldsymbol{\xi}$ that move each function closer to zero simultaneously, namely

$$\mathbf{J}.\,\delta\boldsymbol{\xi} = -\mathbf{F} \tag{10}$$

The corrections are then added to the solution vector

$$\boldsymbol{\xi}_{new} = \boldsymbol{\xi}_{old} + \delta\boldsymbol{\xi} \tag{11}$$

and the process is iterated to convergence.

4 Input data file

To run PAT, an input data file is required. The input data file must be a text-format file. This file describes the finite element model, including the node coordinates, elements and connectivity, boundary and initial conditions, etc. In case of using discrete data of environmental actions, an extra data file must be supplied. All entries are in free form format, which provides some flexibility, but requires that all input variables, even null values, must be given.

The name of the input data file (without extension) is read from the command line and set to `filename`. The program assumes that the data file has a file extension `*.dat`. The program will assign the same file name to the output files.

4.1 Management data

The first group of data defines the main characteristics of the FEM and FDM adopted schemes and allow to allocate memory and initialise several general arrays.

The first line of data identifies the type of element used for the modelling `element`, the number of nodes per element `nod`, the number of elements `nels`, the number of nodes in the mesh `nn`, the number of integration points `nip`, the number of dimensions `ndim`, the calculation time step `dtim` and the time integration parameter `theta`.

The next two lines of data refer to the period of analysis given by the initial and final date and time, `jdate` and `jhour`. These dates must be given in the format `yyyymmdd hhmm`.

Although PAT contains all the element type library defined in [1] it is supposed to use only `quadrilateral` and `hexahedron`, Figure 3. For other element types it is necessary to implement the corresponding associated element to allow the use of prescribed flux or convection boundary conditions. This implementation is straightforward.

It is also important to mention that PAT does not support different element types in the same mesh. However, unlike the extension of element types, this shortcoming is difficult to overcome since it implies a major modification of the program structure.

4.2 Units

The use of the Julian day numbering scheme solves in an efficient way the dependency of the environmental actions on date and time. However, it introduces the shortcoming of fixing a priori the unit of time in days. To address this constraint, the program reads the unit of time chosen for the analysis in order to define the appropriate conversion factor. This is done through the input variable `utemp` by setting “`s`” for seconds, “`h`” for hours or “`d`” for days.

The program does not assume another unit. The only exception can be seen in 4.6.2.

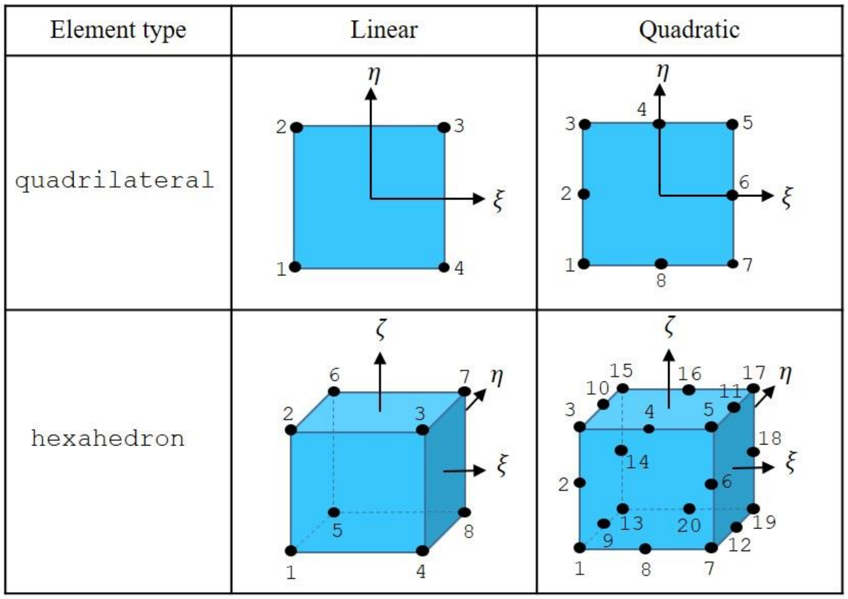


Figure 3 – Type of elements

4.3 Mesh orientation

The general geographic orientation of the global axis is read as the latitude and azimuth angles, `phi` and `azimuth`, expressed in decimal degrees.

The latitude is the angular coordinate of the location in question in reference to the equator with positive values in the north.

The azimuth is defined as the angle between the South and the normal to the surface measured clockwise around the surface's horizon. In order to define the azimuth in an unambiguous way, the global axis orientation should observe the following rules: in 2D problems the y axis must point to the Zenith and the global azimuth is measured with respect to the normal vector pointing outwards from the plane, in 3D problems the z axis must point to the Zenith and the global azimuth is measured with respect to the y axis.

4.4 Material data

The number of properties `np_types` precedes the introduction of the thermal properties values. If there is only one property type (`np_types`=1) then the properties are read and automatically allocated to all elements. If there is more than one property type then the properties are read, followed by the reading of an integer vector `etype` which holds information on which properties are assigned to each element. The properties are stored in the matrix `prop`, and refer to the thermal conductivities k_x and k_y in 2D problems or k_x, k_y and k_z for 3D problems, the material density ρ, the specific heat c, the total thermal transmission coefficient h_t and the absorption coefficient a. These parameters must be provided in consistent units.

4.5 Mesh data

The mesh is defined by the nodal geometry `g_coord` and connectivity `g_num` details. The local numbering of the elements is given in Figure 3.

4.6 Boundary conditions

The boundary conditions must be given in the following order: prescribed temperature, prescribed flux and convection boundary condition.

4.6.1 Prescribed temperature

The prescribed temperature can be given as reservoir water temperature, `fixed_freedom_1`, or as constant or discrete values, `fixed_freedom_2`.

The variable `fixed_freedom_1` indicates the number of nodes in contact with the reservoir water. If there are no nodes with this conditions, `fixed_freedom_1` must be set to 0 and the data input follows with `fixed_freedom_2` boundary condition. Otherwise, the following line must input the parameters of the water temperature equation given by Bofang ([7] §2.5)

$$T(y,d) = T_m(y) - A(y)\cos\left\{\frac{2\pi}{365}[d - \tau_o - d_o(y)]\right\} \tag{12}$$

with

$$T_m(y) = c + (T_s - c)\exp(-e_1 y) \tag{13}$$

$$A(y) = A_o \exp(-e_2 y) \tag{14}$$

$$d_o(y) = [e_3 - e_4 \exp(-e_5 y)]\,\frac{365}{12} \qquad \text{[days]} \tag{15}$$

where y is the depth of the water, d is the fractional day of the year, $T_m(y)$, $A(y)$ and $t_o(y)$ are the annual mean temperature, the amplitude of annual variation and the phase difference of water temperature at depth y, τ_o is the time for maximum air temperature, and T_s, A_o, c and e_1 to e_5 are obtained through the monitored temperatures.

The depth of the water y is calculated as the difference between an adopted reference level of the reservoir `wl` and the vertical coordinate of the node.

The data to input are the adopted reference level of de reservoir followed for T_s, A_o, τ_o, c and e_1 to e_5, that is `wl`, `ts`, `ao`, `tauo`, `ce`, `e1`, `e2`, `e3`, `e4` and `e5`, respectively.

The variable `fixed_freedom_2` indicates the number of nodes with prescribed temperature equal to constant or discrete values. As before, if this boundary condition is not assigned to any node, `fixed_freedom_2` must be set to 0 and follow with the next boundary condition. If `fixed_freedom_2` is not equal to 0, the following line indicates which of the two options for fixed temperatures will be used, variable `tempin`.

If `tempin=1` constant temperatures are assigned to each `fixed_freedom_2` nodes. Therefore, the following piece of data must indicate for each `fixed_freedom_2` the node and the value.

If `tempin=2` the temperatures are given as function of time through a table of discrete values. Each line of the tables must indicate date, time and temperature value. The date and time must be given in `yyyymmdd hhmm` format. This data must be provided in a separate text-format file. The file name, including file extension, corresponds to the next line of the input file followed by the number of nodes with these prescribed temperatures.

4.6.2 Prescribed flux

The prescribed flux corresponds in `PAT` to the solar radiation boundary conditions and is given by the variable `hfbc`. If there is no prescribed flux, `hfbc` must be set to 0, otherwise `hfbc` indicates the number of boundaries with prescribed flux. The solar irradiation data can be introduced as an exponential function, `fluxin`=1, or as discrete values, `fluxin`=3. There is also the possibility of computing the beam and diffuse irradiations internally by the Kumar's model, `fluxin`=2.

If `fluxin`=1, the irradiance is computed through the function

$$\frac{I_b}{\cos Z} = I_o \exp(A + B \cos Z) \tag{16}$$

where I_o is the solar constant (1367 W/m^2), and A and B must be given by the user. It is important to note that the program only converts units of time. If the unit of length is different from meter, or the unit of energy is different from joule, the user must modify the constant I_o in subroutine `radiation_parameters`.

If `fluxin`=3, the beam and diffuse irradiance are given as a function of time through a table of discrete values. Each line of the table must indicate date, time, value of beam irradiation and value of diffuse irradiation. The date and time must be given in the form `yyyymmdd hhmm`. The data table must be provided in a separate text-format file, which name, including file extension, must be given following `fluxin`.

If `fluxin`=2, the beam and diffuse irradiance are computed internally by Kumar's model and no extra data is necessary. The subroutine `Kumar_model` works with a fixed I_o= 1367 W/m^2, and no conversion unit is addressed.

Whatever the value of `fluxin`, the radiation boundary conditions end with the indication for each `hfbc` boundary condition of the number of the parent element and the number of the side with solar radiation following Figure 1.

It is worth noting that the formulas for the Sun-angle relationships use solar time which depends on the Sun's position and is different from clock or local time. Therefore, when introducing discrete values the user is recommended to change clock hour to solar hour. This transformation contemplates three corrections: (1) the difference in longitude between the observer's meridian (longitude) and the meridian on which the local standard time is based, (2) daylight saving time, and (3) the earth's slightly-irregular motion around the Sun (Equation of time). To help in this transformation, several local to solar time calculators are available online.

4.6.3 Convection boundary condition

The convection boundary condition is indicated by the variable `htbc`. If there is no convection boundary condition, `htbc` must be set to 0, otherwise `htbc` indicates the number of boundaries with convection heat transfer. The air temperature can be introduced as a harmonic function, `airin`=1, or as a table of discrete values, `airin`=2.

If `airin`=1, the daily air temperature is represented as the superposition of two harmonic functions, one of annual period and another with a one day period

$$T(d) = T_m + T_o^a \cos\left[\frac{2\pi}{365}\,(d - t_o^a)\right] + \frac{A(d)}{2}\cos[2\pi\,(d - t_o^d)] \tag{17}$$

with

$$A(d) = A_m + A_o^a \cos\left[\frac{2\pi}{365}\,(d - \theta_o^a)\right] \tag{18}$$

where d is the fractional day of the year, T_m is the yearly mean temperature, T_o^a is the amplitude of annual variation, t_o^a is the yearly phase difference, $A(d)$ is the yearly variation of the daily amplitude defined by the mean amplitude A_m, amplitude A_o^a and the phase difference θ_o^a, and t_o^d is the daily phase difference.

The parameters must be given in the following order: T_m, T_o^a, t_o^a, A_m, A_o^a, θ_o^a and t_o^d which are attributed to the variables `t_m1`, `t_p1`, `delay_1`, `t_m2`, `t_p2`, `delay_2` and `delay_3`, respectively.

If `airin`=2, the air temperature and the total thermal transmission coefficient are both introduced as a function of time through a table of discrete values. Each line of the table must contain date, time, air temperature and total thermal transmission coefficient. The date and time must be given in the form `yyyymmdd hhmm`. These data are given in a separate text-format file, which name, including file extension, must be given in the input file. The program provides the possibility of introducing two different groups of convection boundary conditions (air temperature and total thermal transmission coefficient), so after `airin`, it is necessary to indicate 1 or 2 followed by the name(s) of the file(s).

The convection boundary condition ends with the indication for each `htbc` boundary condition of the number of the element and the number of the side of the element with convection. In the case that there are two tables of discrete data, it must be given also the number of the group (1 or 2).

It is worth noting that in case of `airin`=2 the total thermal transmission coefficient given in matrix `prop` is not used.

4.7 Initial conditions

After the boundary conditions, the program reads the initial temperature indicator `indic`. If `indic`=0, the program assigns an initial constant temperature to all the nodes of value `val0`.

Otherwise `indic` must be set to the number of nodes with initial value different from 0, followed by the `indic` number of node and temperature value.

The last option can be used to restart the analysis from a previous run.

4.8 Output data

The final part of the data file refers to the outputs. The `npri` variable indicates that the results will be printed every `npri` time steps. The output files will be called `filename_yyyymmddhhmmss.res`.

If the user wants results in some specific points inside the mesh, the variable `nsensors` must be set to the number of points to be computed, otherwise this variable must be set to 0. For each point, it must be provided a `name` to identify the point and its global coordinates. The output files will be called `filename_name.res`.

5 Verification examples

5.1 One-dimensional heat conduction through a plane wall with one side at prescribed periodic temperature and the other at 0°C

Consider the one-dimensional heat transfer through a plane wall with thickness L, constant thermal diffusivity h^2 and uniform initial temperature T_i with the boundary conditions given by

$$T(0,t) = -T_o \cos\left[\frac{2\pi}{P}\ (t - t_o\right] \tag{19}$$

$$T(L,t) = 0 \tag{20}$$

as shown in Figure 4.

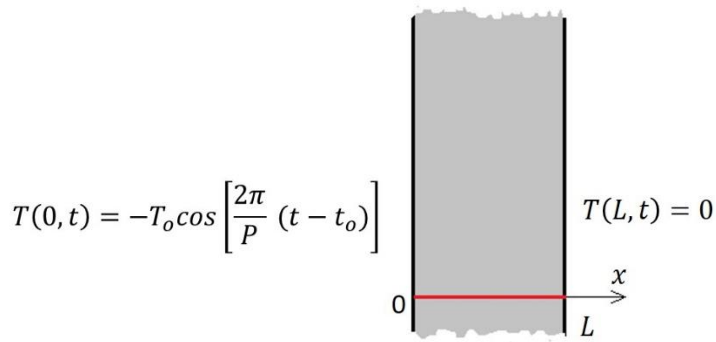


Figure 4 - One dimensional heat conduction through a plane wall with periodic boundary conditions on one side and 0°C in the other

If the surface oscillation of temperature $T(0,t)$ has been going on for so long that the steady periodic conditions are established and the influence of the initial temperature has disappeared, temperature development through the thickness of the wall at time t can be calculated based on Puppini's formula [8] and using trigonometric relationships

$$\begin{aligned} T(x,t) = \ -T_o \Bigg\{ & \cos\left(\frac{2\pi t_o}{P}\right)\left[A_x\ \cos\left(\frac{2\pi t}{P}\right) + B_x \sin\left(\frac{2\pi t}{P}\right)\right] \\ & + \sin\left(\frac{2\pi t_o}{P}\right)\left[A_x\ \sin\left(\frac{2\pi t}{P}\right) - B_x \cos\left(\frac{2\pi t}{P}\right)\right]\Bigg\} \end{aligned} \tag{21}$$

with

$$A_x = \ -2M\ \sinh z\ \cos z - \ 2\,N\ \cosh z\ \sin z\ + e^z \cos z \tag{22}$$

$$B_x = 2M \cosh z \sin z - 2N \sinh z \cos z - e^z \sin z \tag{23}$$

and

$$M = \frac{e^{2z_o} - \cos 2z_o}{2(\cosh 2z_o - \cos 2z_o)} \tag{24}$$

$$N = \frac{\sin 2z_o}{2(\cosh 2z_o - \cos 2z_o)} \tag{25}$$

$$z_o = \sqrt{\frac{\pi}{P\, h^2}}\, L \tag{26}$$

$$z = \sqrt{\frac{\pi}{P\, h^2}}\, x \tag{27}$$

Since the model is infinitely long in one direction, the model is essentially one-dimensional, and horizontal boundaries may be represented as adiabatic boundaries, Figure 5.

In this example the thermal diffusivity $h^2 = k/(\rho\, C)$ was set to 1 m^2/day, the period P to 365 days, and the periodic parameters T_o to 40°C and t_o to 73 days.

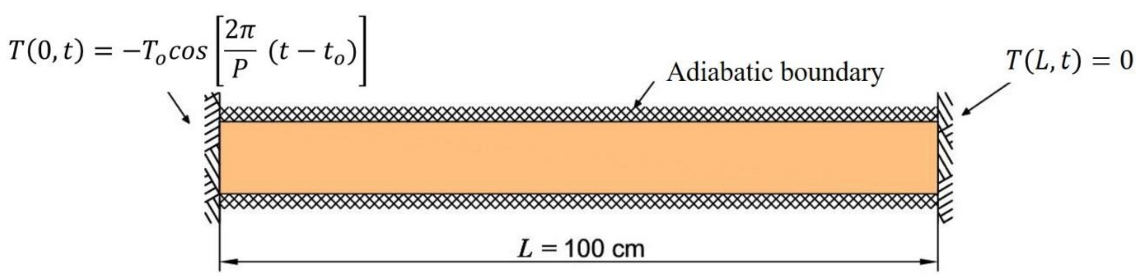


Figure 5 – Idealization of the plane wall

The wall was divided into a row of 8 quadratic quadrilateral elements. Figure 6 shows the mesh and data used for the analysis. It is worth noting that the words in bold of Figure 6 are just explanatory text, they do not form part of the input file.

The input file begins with the general data of the adopted mesh. Since the problem must be run until it reaches a steady temperature variation, a long period between 2001/01/01 and 2019/01/01 was chosen. In this example the orientation of the mesh is not necessary, therefore the angles `phi` and `azimuth` can be set to any value.

As it was mentioned above, prescribed temperature boundary conditions are applied to the two sides of the wall and adiabatic boundary conditions to the horizontal boundaries.

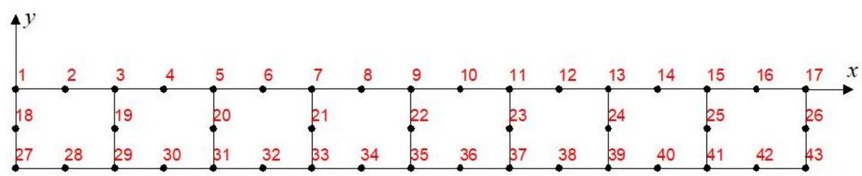
The periodic temperature is introduced as reservoir water temperature in nodes 1, 18 and 27, thus `fixed_freedom_1=3`. In order to adapt expression (12) to expression (19) A_o was set to 40°C, τ_o to 73 days and the rest of the parameters to zero. To indicate that the model is submerged it was set `wl=10`.

On the right side, the prescribed temperature is introduced as a fixed temperature in nodes 17, 26 and 43, that is `fixed_freedom_2=3` and `tempin=1`.

The adiabatic boundary condition is simulated as a prescribed flux boundary condition with $q_r = 0$. However, as $q_r = 0$ does not contribute to the heat load vector, it is not necessary to specify this boundary conditions. In other words, the adiabatic boundary condition is the default boundary condition.

The last part of the input file refers to the output selection. In this case it was chosen to print node temperatures every 365 time steps by setting `npri`=365, and to print the whole temperature history of node 22 in order to control the convergence of the results to steady periodic variation.

```
element           nod  nels    nn    nip  ndim
quadrilateral       8     8    43      9     2
dtim
theta
1.0E+00
0.5E+00
jdate     jhour
20010101   0000
20190101   0000
utemp
d
phi     azimute
9999.     9999.
np_types
1
prop(kx, ky, ρ, c, h, a)
1.0E+00   1.0E+00   1.0E+00   1.0E+00   1.0E+00   0.0E+00
g_coord
  0.0000E+00  0.0000E+00
  0.6250E-01  0.0000E+00
  0.1250E+00  0.0000E+00
  ⋮
  0.8750E+00 -0.1000E+00
  0.9375E+00 -0.1000E+00
  0.1000E+01 -0.1000E+00
g_num
   27    18     1     2     3    19    29    28
   29    19     3     4     5    20    31    30
  ⋮
```



```
   39   24   13   14   15   25   41   40
   41   25   15   16   17   26   43   42
```

fixed_freedoms_1
```
3
```

```
wl             ts         ao      tauo       ce       e1       e2       23       e4
10.0E+00  0.0E+00 40.0E+00 73.0E+00 0.0E+00 0.0E+00 0.0E+00 0.0E+00 0.0E+00
e5
0.0E+00
```

(node_1(i),i=1,fixed_freedoms_1)
```
1  18  27
```

fixed_freedoms_2
```
3
```

tempin
```
1
```

(node_2(i),value_2(i),i=1,fixed_freedoms_2)
```
17 0.0E+00  26 0.0E+00  43 0.0E+00
```

hfbc
```
 0
```

htbc
```
 0
```

indic
```
 0
```

val0
```
 0.0E+00
```

npri
```
365
```

nsensors
```
1
```

```
ifile     sensors(j,:)
Node_22   0.5000E+00 -0.5000E-01
```

Figure 6 - Mesh and data for the plane wall

In Table 1 the comparison of the last 15 days are shown. As the FEM model computes the temperature at the beginning of each day, the corresponding analytical solution must be computed at day $(d-1)$.

Table 1 - Comparison of FEM with analytical solution

Analytical		FEM	
d	T [°C]	date	T [°C]
351	-1,419	18/12/2019 00:00	-1,419
352	-1,762	19/12/2019 00:00	-1,762
353	-2,105	20/12/2019 00:00	-2,105
354	-2,447	21/12/2019 00:00	-2,447
355	-2,788	22/12/2019 00:00	-2,788
356	-3,129	23/12/2019 00:00	-3,129
357	-3,468	24/12/2019 00:00	-3,468
358	-3,807	25/12/2019 00:00	-3,807
359	-4,144	26/12/2019 00:00	-4,144
360	-4,480	27/12/2019 00:00	-4,480
361	-4,815	28/12/2019 00:00	-4,815
362	-5,149	29/12/2019 00:00	-5,149
363	-5,481	30/12/2019 00:00	-5,481
364	-5,811	31/12/2019 00:00	-5,811
365	-6,139	01/01/2020 00:00	-6,139

5.2 Steady-state temperature distribution along a rectangular fin

This example corresponds to a one-dimensional fin exposed to a surrounding fluid at a temperature T_∞ as shown in Figure 7. The temperature of the base of the fin is T_o.

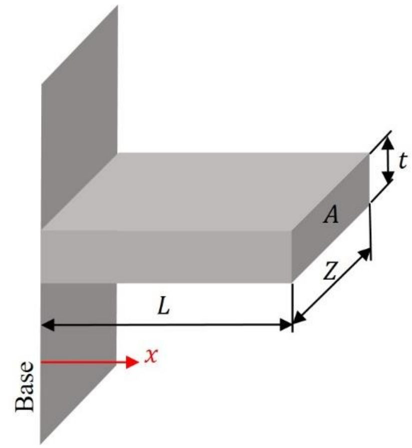


Figure 7 – One-dimensional conduction and convection through a rectangular fin

The solution is given by Holman ([9] §2.9)

$$\frac{T - T_\infty}{T_o - T_\infty} = \frac{\cosh[m(L-x)] + \left(\frac{h}{mk}\right)\sinh[m(L-x)]}{\cosh(mL) + \left(\frac{h}{mk}\right)\sinh(mL)} \tag{28}$$

with $m = \sqrt[2]{h_c P / kA}$ and where P is the perimeter.

It was assumed that the fin is 8.33 cm high, 33.33 cm wide and infinitely long. It is attached to a wall maintained at 1100°C, and is immersed in a fluid maintained at 100°C. The fin has thermal conductivity of 15 W/(m°C) and a heat convection coefficient of 15 W/(m^2 °C). These data are similar to the verification problem given in ([10] §3.2.2.2). Figure 8 shows the idealisation of the problem.

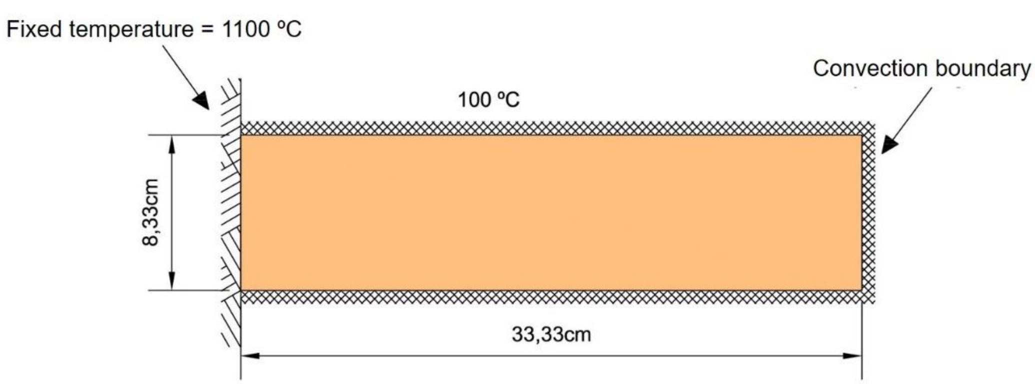


Figure 8 – Idealization of the fin

As the fin is infinitely long in the plane perpendicular to the analysed section, the problem can be solved as a 2D problem. In this case, P/A goes to 24 and M has a value of $\sqrt[2]{24}$.

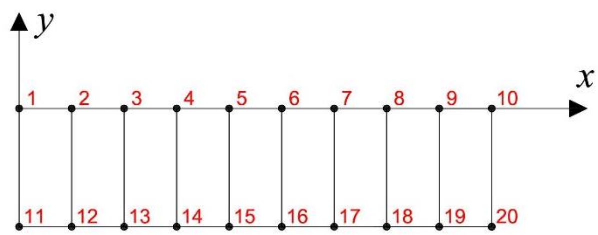
For the numerical model, the fin is divided into a row of 9 linear quadrilateral elements. Figure 9 shows the mesh and data used for the analysis.

The fluid temperature is introduced by expression (17), assigned the value of 100°C to T_m and zero to the rest of the parameters.

The problem must be run to reach a steady-state temperature. The history of the temperature of node 3 is used to control the convergence of the problem to a steady-state temperature, and the results are printed only once setting `npri`=60.

Table 2 shows the comparison between the obtained results and analytical solution.

```
element         nod  nels    nn   nip  ndim
quadrilateral     4     9    20     4     2

dtim         theta
10.0E+00   0.5E+00

jdate     jhour
20210705  0000
20210705  0010

utemp
s

phi     azimute
9999.    9999.

np_types
1

prop(kx,ky,ρ,c,h,a)
15.0E+00  15.0E+00  100.0E+00  300.0E+00  15.0E+00  0.0E+00

g_coord
  0.0000E+00  0.0000E+00
  0.3703E-01  0.0000E+00
  0.7407E-01  0.0000E+00
  ⋮
  0.2592E+00 -0.8330E-01
  0.2963E+00 -0.8330E-01
  0.3333E+00 -0.8330E-01

g_num
   11    1    2   12
   12    2    3   13
  ⋮
   18    8    9   19
   19    9   10   20

fixed_freedoms_1
0

fixed_freedoms_2
2
```



```
tempin
1
(node_2(i),value_2(i),i=1,fixed_freedoms_2)
1  1100.0E+00   11  1100.0E+00
hfbc
0
htbc
19
airin
1
t_m1          t_p1  delay_1     t_m2     t_p2  delay_2  delay_3
100.0E+00  0.0E+00  0.0E+00  0.0E+00  0.0E+00  0.0E+00  0.0E+00
((itrans(i,j),j=1,2),i=1,htbc)
1 4
2 4
3 4
  ⋮
8 3
9 3
9 2
indic
0
val0
0.0E+00
npri
60
nsensors
1
ifile   sensors(j,:)
Node_3  0.7407E-01 -0.4165E-01
```

Figure 9 – Mesh and data for the fin

Table 2 – Comparison of FEM with analytical solution

x/L	T [°C]	
	Analytical	FEM
0.00	1100.0	1100.0
0.11	943.1	942.8
0.22	813.9	813.5
0.33	708.4	707.8
0.44	622.9	622.3
0.56	554.7	554.0
0.67	501.5	500.8
0.78	461.6	460.9
0.89	433.5	432.8
1.00	416.5	415.8

5.3 Conduction through a composite plane wall

An infinite wall consisting of two distinct layers is exposed to an atmosphere at high temperature on one side and low temperature on the other. The wall eventually reaches an equilibrium where it passes a constant flux and the temperature distribution is unchanging ([10] §3.2.2.1).

Figure 10 shows the wall geometry and boundary conditions. The values adopted for the example were thermal conductivities k_1 = 1.6 W/(m °C), k_2 = 0.2 W/(m °C), thickness d_1 = 25 cm, d_2 = 15 cm, convection coefficients h_1 = 100 W/(m^2 °C), h_2 = 15 W/(m^2 °C) and temperatures T_1 = 3000 °C, T_1 = 25 °C.

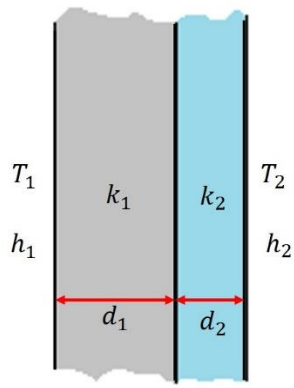


Figure 10 – Composite wall

The solution to this problem is $T(0)$ = 2970 °C, $T(d_1)$ = 2497 °C and $T(d_1 + d_2)$ = 226.7 °C with a linear variation between them.

Since the model is infinitely long in one direction, the model is essentially one-dimensional, and the horizontal boundaries may be represented as adiabatic boundaries. Figure 11 shows the idealization of the wall.

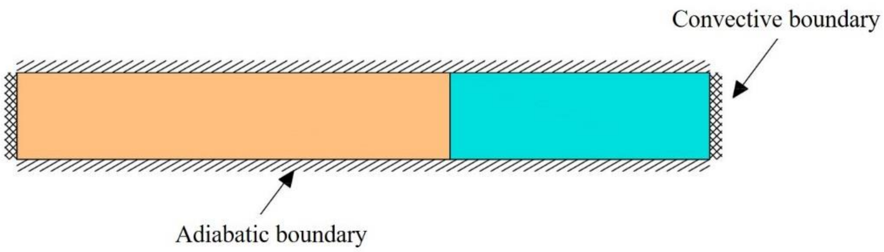


Figure 11 – Idealization of the composite wall

For the numerical model, the wall is divided into two rows of 16 linear quadrilateral elements. Figure 9 shows the mesh and data used for the analysis.

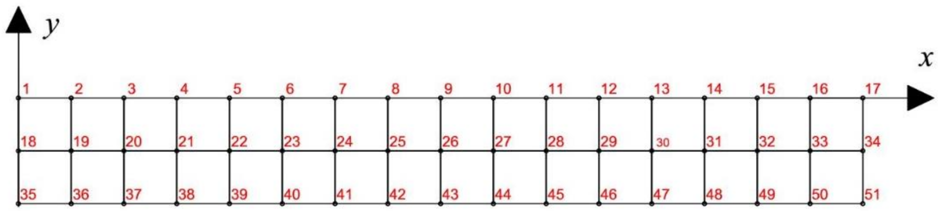
```
element          nod  nels    nn    nip  ndim
quadrilateral      4    32    51      4     2
```

dtim **theta**

```
1.0E+00   0.5E+00
```

jdate **jhour**

```
20200101  0000
20200101  0001
```



utemp

```
s
```

phi **azimute**

```
9999.    9999.
```

np_types

```
2
```

prop $(k_x, k_y, \rho, c, h, a)$

```
 1.6E+00  1.6E+00  1.0E+00  1.0E+00 100.0E+00  0.0E+00
 0.2E+00  0.2E+00  1.0E+00  1.0E+00  15.0E+00  0.0E+00
```

etype

```
 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
```

g_coord

```
  0.0000E+00  0.0000E+00
  0.2500E-01  0.0000E+00
  0.5000E-01  0.0000E+00
  ⋮
  0.3500E+00 -0.5000E-01
  0.3750E+00 -0.5000E-01
  0.4000E+00 -0.5000E-01
```

g_num

```
   18    1    2   19
   19    2    3   20
  ⋮
   48   31   32   49
   49   32   33   50
   50   33   34   51
```

fixed_freedoms_1

```
0
```

fixed_freedoms_2

```
 0
```

hfbc

```
0
```

htbc

```
 4
```

airin

```
 2
```

nfiles

```
 2
```

ifile

```
 left.dat
```

```
 right.dat
((itrans(i,j),j=1,3),i=1,htbc)
  1 1  1
 17 1  1
 16 2  2
 32 2  2
indic
0
val0
0.0E+00
npri
60
nsensors
1
ifile   sensors(j,:)
Node_25   0.2500E+00 -0.2500E-01
```

Figure 12 – Mesh and data for the composite wall

Since there are two different convection boundary conditions, the data is introduced through two files, `left.dat` and `right.dat`, shown in Figure 13 and Figure 14, respectively. The values of the temperature and convection coefficient for each time step are then obtained by means of a linear interpolation, therefore, only the extreme values were introduced.

```
20200101  0000  3000.0E+00  100.0E+00
20200101  0001  3000.0E+00  100.0E+00
```

Figure 13 – File `left.dat`

```
20200101  0000  25.0E+00  15.0E+00
20200101  0001  25.0E+00  15.0E+00
```

Figure 14 – File `right.dat`

Figure 15 shows a graphical representation of the steady-state temperature distribution.

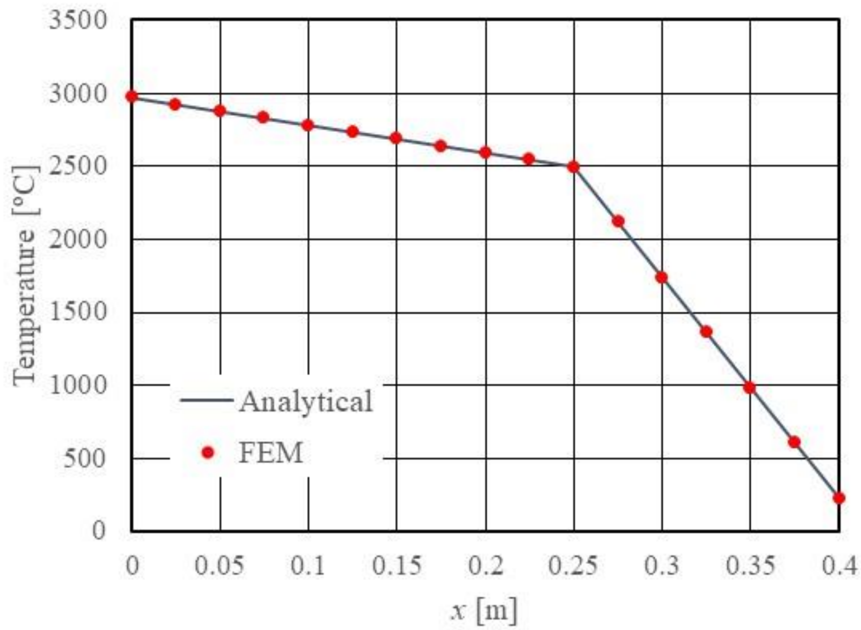


Figure 15 – Composite wall: comparison between FEM and analytical solution

References

[1] I.M. Smith, D.V. Griffiths. Programming the finite element method, 4th ed. John Wiley & Sons, Ltd, 2005.

[2] I.M. Smith, D.V. Griffiths. Programming the finite element method, 4th ed. Associated software. https://inside.mines.edu/~vgriffit/4th_ed/

[3] P. Duffett-Smith, J. Zwart. Practical Astronomy with your calculator or spreadsheet, 4th ed. Cambridge University Press, 2011.

[4] W.H. Jeffers. Julian day numbers. Last modified 26 January 1998. https://quasar.as.utexas.edu/BillInfo/JulianDatesG.html (accessed 17 September 2021)

[5] G. Silva, R. Le Riche. J. Molimard, A. Vautrin. Exact and efficient interpolation using finite elements shape functions. 2007. hal-00122640v2.

[6] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery. Numerical recipes in Fortran 77. The art of scientific computing, 2nd ed. Cambridge University Press, 1992.

[7] Z. Bofang. Thermal stresses and temperature control of mass concrete. Elsevier, 2014.

[8] U. Puppini. Variazioni di temperatura entro masse murarie. Bollettino della Unione Matematica Italiana. Anno 1 – N. 1, 1922, pp. 53-57. https://archive.org/details/bollettinodellau01unio

[9] J.P. Holman. Heat transfer, 10th ed. Mc Graw Hill, 2010.

[10] Itasca. FLAC Version 2.2, Verification, examples and benchmark problems. 1989.