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

# 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].

# Bridging the gap between FEM and solar radiation formulations

## 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 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 axis must point to the Zenith and the global azimuth is measured with respect to the axis.

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

## 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

|  |  |  |
| --- | --- | --- |
|  |  | () |

where is the norm of the outward normal vector given in 2D problems by

|  |  |  |
| --- | --- | --- |
|  |  | () |

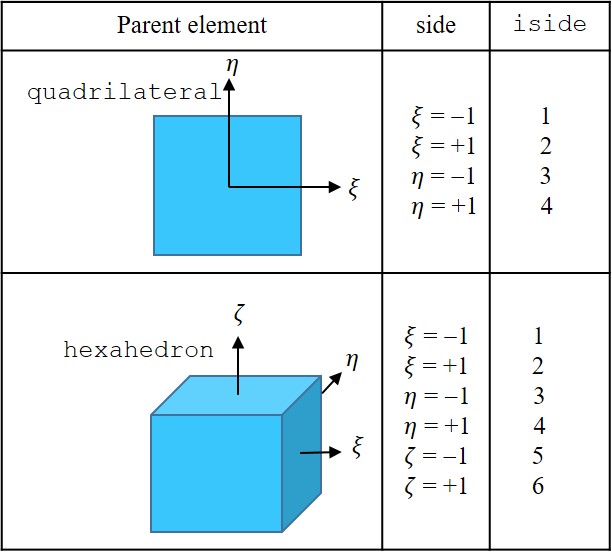


Figure - Boundary identification

and in 3D problems by

|  |  |  |
| --- | --- | --- |
|  |  | () |

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

|  |  |  |
| --- | --- | --- |
|  |  | () |

are used.

For the 2D case the tilt angle is obtained with the arc-cosine of the component and the azimuth is computed as the global azimuth plus if or if .

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 . The second term of the right hand side of beta allows to obtain the correct sign of the angle , Figure 2.

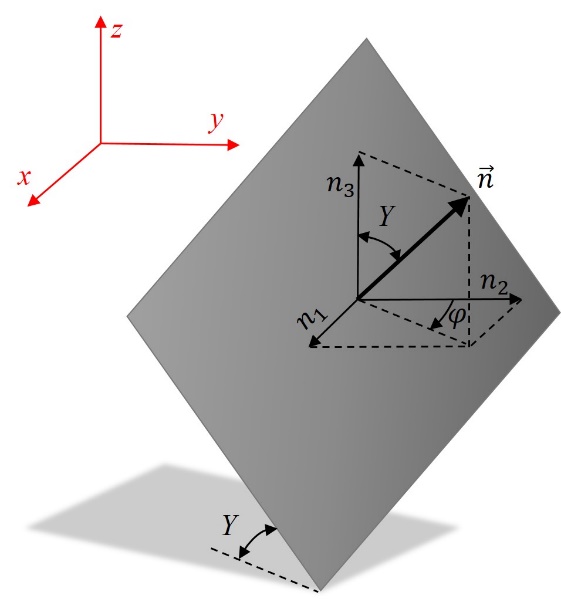


Figure – Integration point orientation

## 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 = year, = month and = day
2. If = 1 or 2 subtract 1 from and add 12 to
3. Dropping the fractional part of all results of all multiplications and divisions, let:

* =
* =
* = 2 – +
* = 365.25 (Y + 4716)
* = 30.6001 (M+1)
* = + + + – 1524.5

**Converting Julian day number to civil date (subroutine** civil\_date**)**

1. + 0.5
2. Set = integer part
3. Dropping the fractional part of all results of all multiplications and divisions, let:
   * =
   * = + 1 + – (
   * = + 1524
   * =
   * = 365.25
   * =
   * = 30.6001
4. Day of month = – – + ( – ) (including the decimal fraction of the day)
5. Month = – 1 if is less than 13.5, or = – 13 if is more than 13.5
6. Year = – 4716 if month is more than 2.5, or = – 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.

# 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 ();
3. If the point falls inside the domain of the element, that is , the element is the owner element and the local coordinates of the data point are ().

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 used to interpolate the coordinate vectors define a one-to-one mapping from the local coordinate system to the global coordinate system, such that

|  |  |  |
| --- | --- | --- |
|  |  | () |

where denotes the local coordinate vector and are the global coordinates of the element nodes (i.e., vertices of the element within which the point 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

|  |  |  |
| --- | --- | --- |
|  |  | () |

where denotes the entire vector of functions .

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

|  |  |  |
| --- | --- | --- |
|  |  | () |

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

|  |  |  |
| --- | --- | --- |
|  |  | () |

In matrix notation, equation (7) is

|  |  |  |
| --- | --- | --- |
|  |  | () |

By neglecting terms of order and higher and by setting , we obtain a set of linear equations for the corrections that move each function closer to zero simultaneously, namely

|  |  |  |
| --- | --- | --- |
|  |  | () |

The corrections are then added to the solution vector

|  |  |  |
| --- | --- | --- |
|  |  | () |

and the process is iterated to convergence.

# 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.

## 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.

## 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.

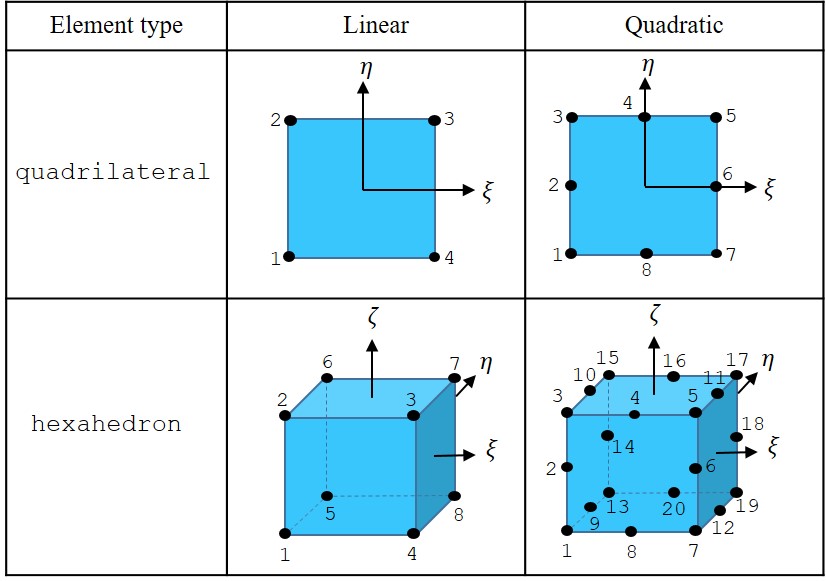


Figure – Type of elements

## 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 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 axis must point to the Zenith and the global azimuth is measured with respect to the axis.

## 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 and in 2D problems or , and for 3D problems, the material density , the specific heat , the total thermal transmission coefficient and the absorption coefficient . These parameters must be provided in consistent units.

## 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.

## Boundary conditions

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

### 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)

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

with

|  |  |  |
| --- | --- | --- |
|  |  | (13) |
|  |  | (14) |
|  |  | (15) |

where is the depth of the water, is the fractional day of the year, , and are the annual mean temperature, the amplitude of annual variation and the phase difference of water temperature at depth , is the time for maximum air temperature, and , , and to are obtained through the monitored temperatures.

The depth of the water 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 , , , and to , 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.

### 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.

Since the code works with clock time, a previous conversion from clock time to solar time must be performed. This transformation requires the knowledge of the location given by the longitude (long) in degrees (positive to the east of the Prime Meridian) and the local standards to which local clocks are set given by the time zone (timezone) in hours from UTC and, when applicable, the daylight saving time (saving=1), otherwise (saving=0).

To indicate that it will be considered the shaded zones, the logical variable shadow must to be set to .TRUE., otherwise it must be set to .FALSE.

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

|  |  |  |
| --- | --- | --- |
|  |  | (16) |

where is the solar constant (1367 W/m2), and and 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 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 = 1367 W/m2, 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.

In the case when shaded zones must be considered (shadow=.TRUE.), it will be necessary to indicate if the program must perform the shadow detection (shadowin=1) or if it was already done in a previous run (shadowin=2).

When shadowin=1, a shadow table containing zero and one values will be constructed and save as filename\_shadow.dat. Apart from the hfbc prescribed flux surfaces, another extra hfbc2 surfaces can be considered in the shadow formation. This surfaces must be indicated in similar way than the first hfbc ones.

When shadowin=2, the name of the previous computed shadow file must be given.

### 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, as a table of discrete values with indication of the total thermal transmission coefficient at each time, airin=2, or as a table of discrete values with constant total thermal transmission coefficient, airin=3.

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

|  |  |  |
| --- | --- | --- |
|  |  | (17) |

with

|  |  |  |
| --- | --- | --- |
|  |  | (18) |

where is the fractional day of the year, is the yearly mean temperature, is the amplitude of annual variation, is the yearly phase difference, is the yearly variation of the daily amplitude defined by the mean amplitude , amplitude and the phase difference , and is the daily phase difference.

The parameters must be given in the following order: , , , , , and 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.

If airin=3, the air temperature is introduced as a function of time through a table of discrete values. Each line of the table must contain date, time and air temperature. In this case, the code uses the total thermal transmission coefficient given in matrix prop.

## 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.

## 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.

# Verification examples

## 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 , constant thermal diffusivity and uniform initial temperature with the boundary conditions given by

|  |  |  |
| --- | --- | --- |
|  |  | () |
|  |  | () |

as shown in Figure 4.

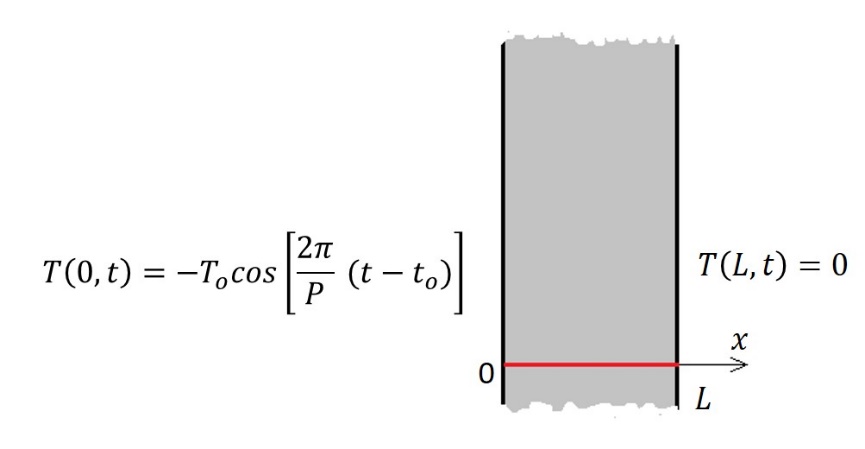


Figure - One dimensional heat conduction through a plane wall with periodic boundary conditions on one side and 0°C in the other

If the surface oscillation of temperature 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 can be calculated based on Puppini’s formula [8] and using trigonometric relationships

|  |  |  |
| --- | --- | --- |
|  |  | () |

with

|  |  |  |
| --- | --- | --- |
|  |  | () |
|  |  | () |

and

|  |  |  |
| --- | --- | --- |
|  |  | () |
|  |  | () |
|  |  | () |
|  |  | () |

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 was set to 1 m2/day, the period to 365 days, and the periodic parameters to 40°C and to 73 days.

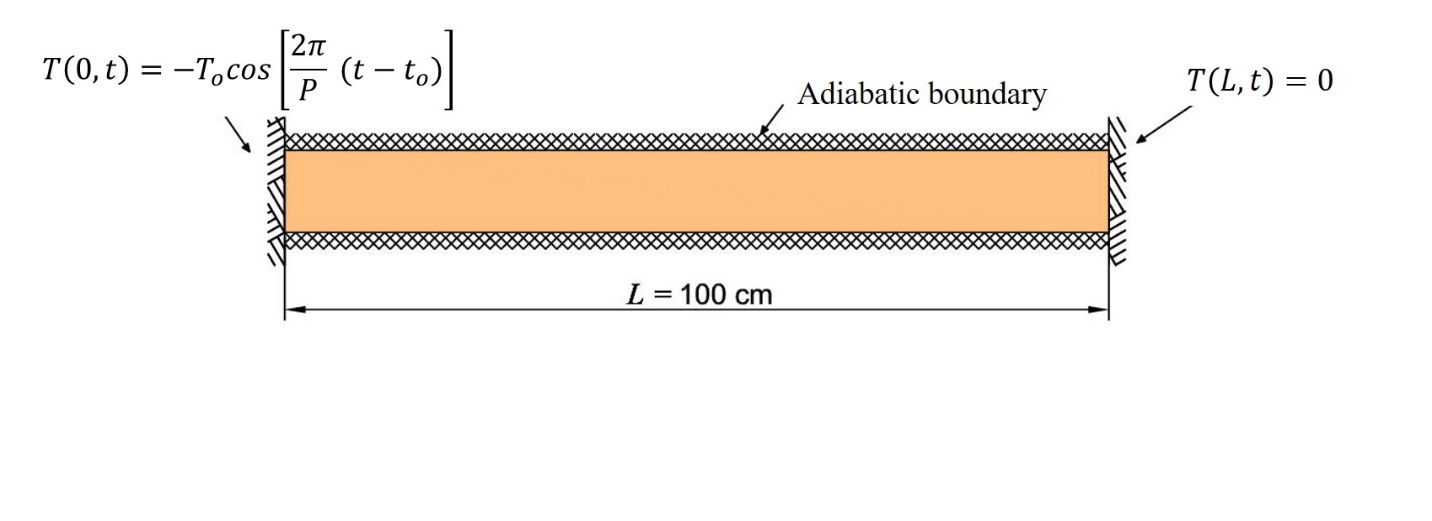


Figure – Idealization of the plane wall

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) was set to 40°C, 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.

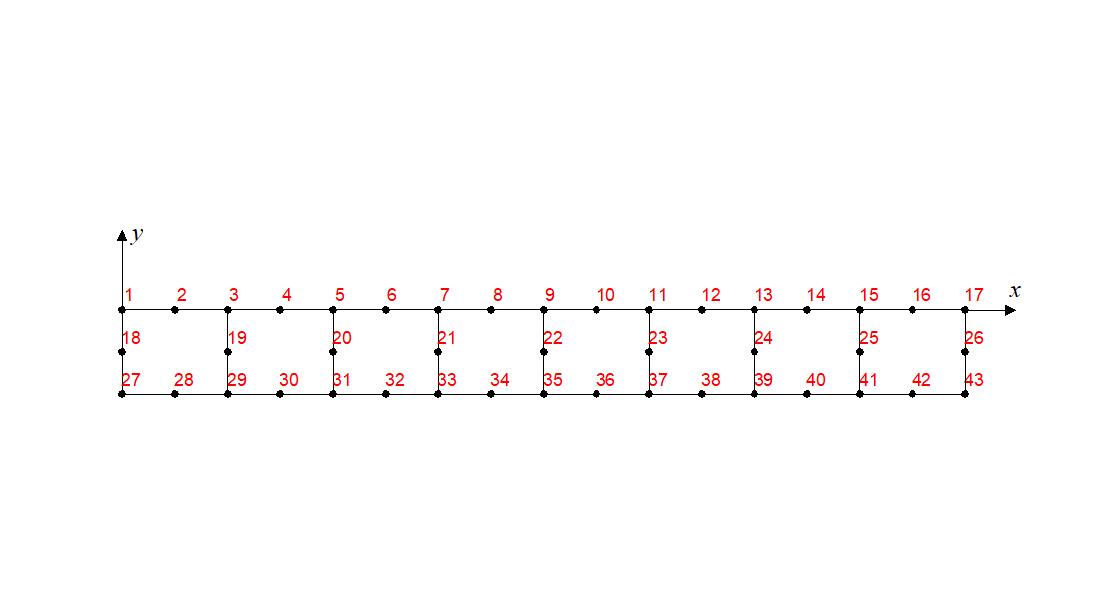
The adiabatic boundary condition is simulated as a prescribed flux boundary condition with . However, as does not contribute to the heat load vector, it is not necessary to specify this boundary conditions. In other words, the adiabatic boundary condition is the default boundary condition.

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(**

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 - 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 .

Table - Comparison of FEM with analytical solution

|  |  |  |  |
| --- | --- | --- | --- |
| Analytical | | FEM | |
|  | 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 |

## Steady-state temperature distribution along a rectangular fin

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

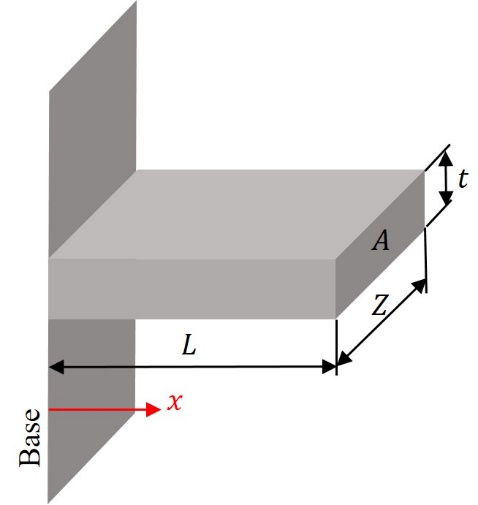


Figure 7 – One-dimensional conduction and convection through a rectangular fin

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

|  |  |  |
| --- | --- | --- |
|  |  | () |

with and where 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/(m2 °C). These data are similar to the verification problem given in ([10] §3.2.2.2). Figure 8 shows the idealisation of the problem.

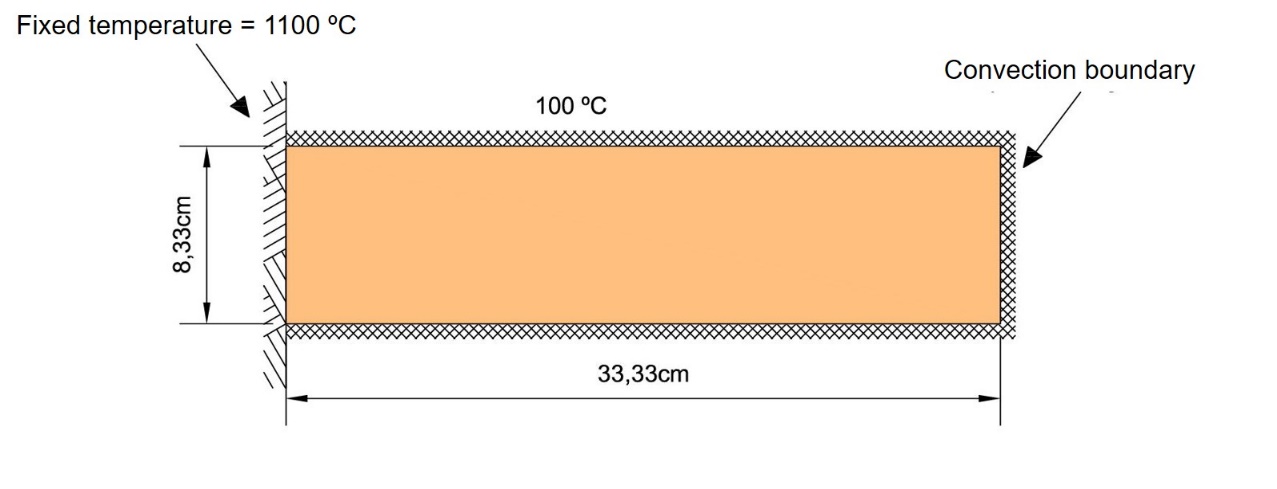


Figure – Idealization of the fin

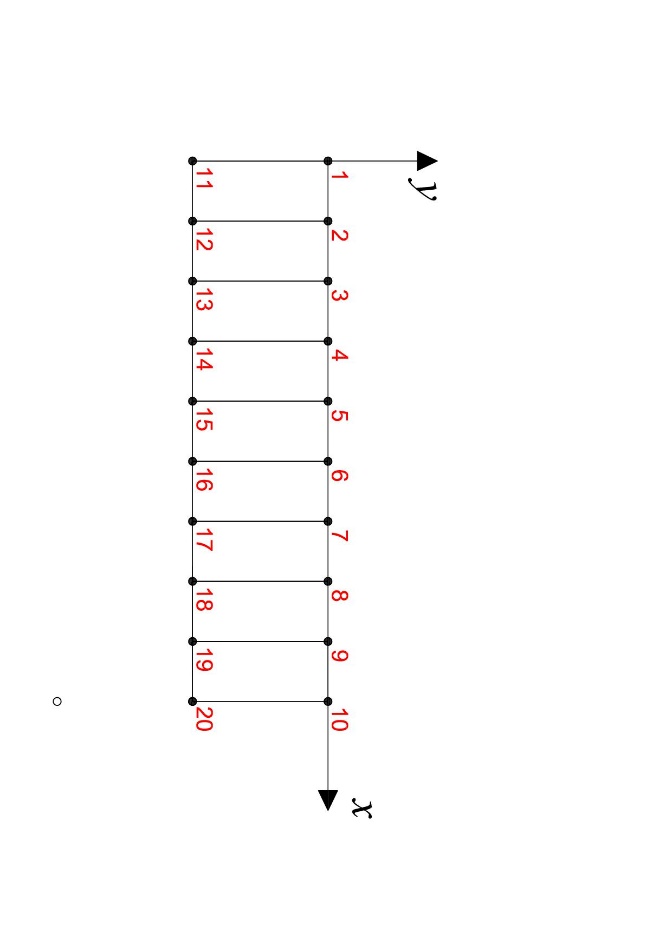
As the fin is infinitely long in the plane perpendicular to the analysed section, the problem can be solved as a 2D problem. In this case, goes to 24 and has a value of .

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 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(**

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 – Mesh and data for the fin

Table – Comparison of FEM with analytical solution

|  |  |  |
| --- | --- | --- |
|  | [°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 |

## 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 = 1.6 W/(m °C), = 0.2 W/(m °C), thickness = 25 cm, = 15 cm, convection coefficients = 100 W/(m2 °C), = 15 W/(m2 °C) and temperatures = 3000 °C, = 25 °C.

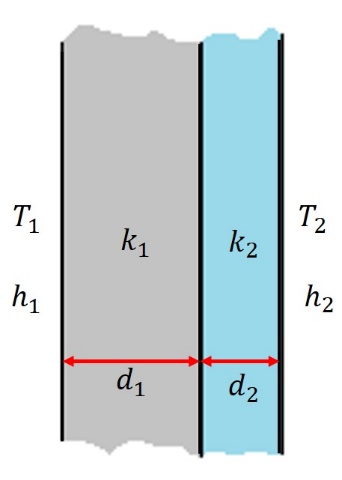


Figure – Composite wall

The solution to this problem is = 2970 °C, = 2497 °C and = 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.

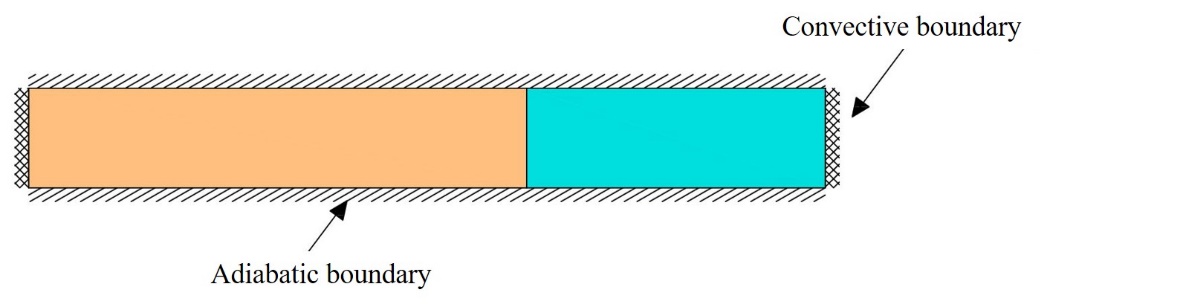
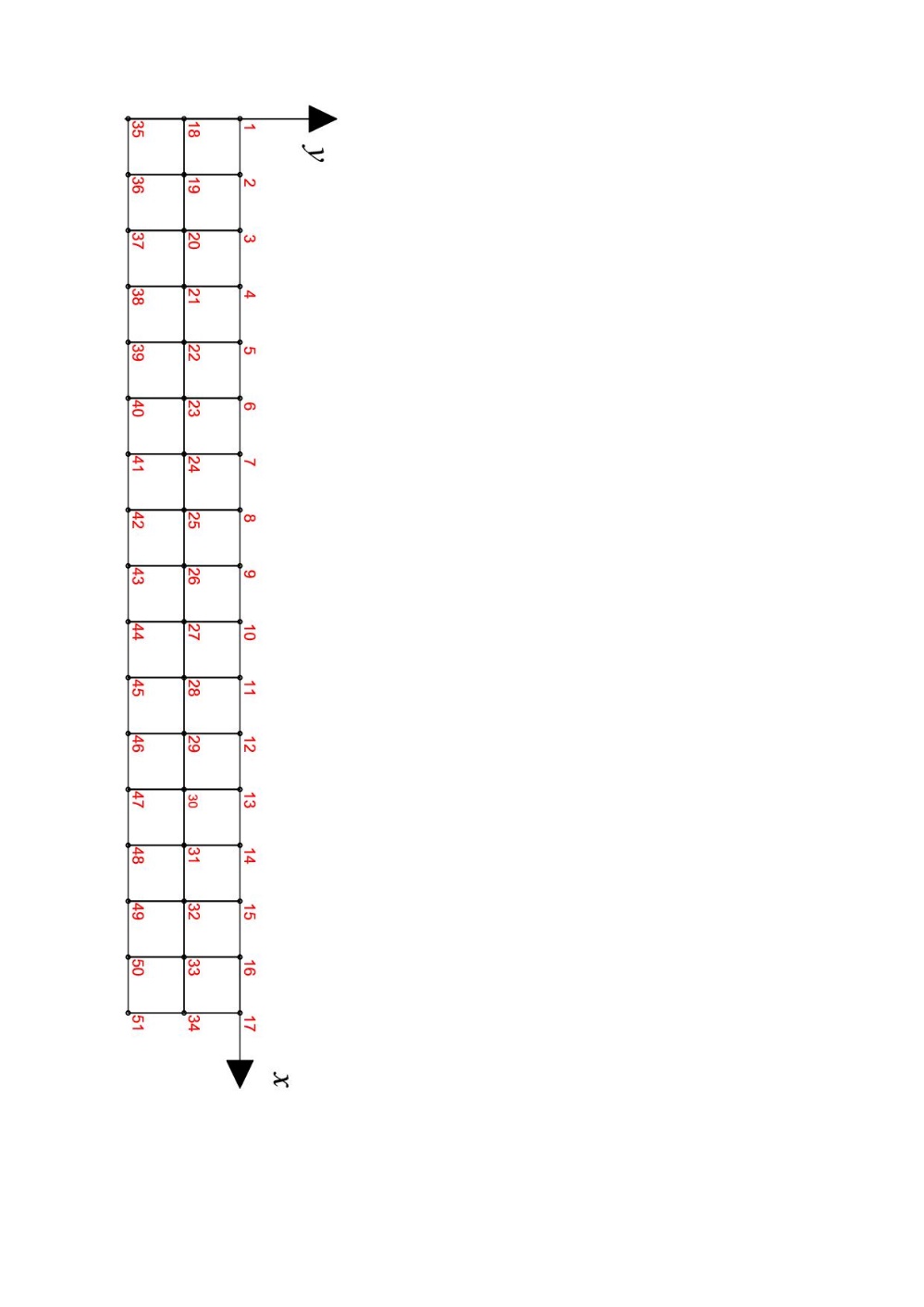


Figure – Idealization of the composite wall

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(**

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 – 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.

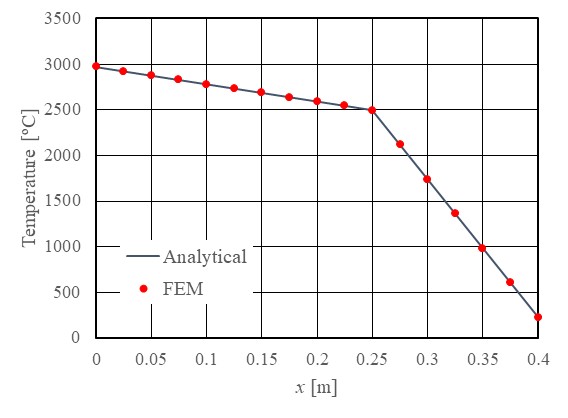


Figure – Composite wall: comparison between FEM and analytical solution

**References**

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

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

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

[] 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.

[] 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>

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

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