A Two-Dimensional Mathematical Model of Heat Propagation Equations and Their Significance for Soil Temperature

Anis Ben Dhahbi 1,2, Salah Boulaaras 3,4,*, Taha Radwan 3,5, Nadia Mezouar 6, Khaled Zennir 3, Mohamed Haiour 7, Ali Allahem 8 and Sewelem Ghanem 3

1 Physics Department, College of Science and Arts at ArRass, Qassim University, P.O. Box 53, ArRass 51921, Saudi Arabia; anis.dhahbi14@gmail.com
2 Unit of Research in Nuclear and High Energy Physics, Faculty of Sciences of Tunis, University of Tunis El Manar, Tunis 2092, Tunisia
3 Mathematics Department, College of Science and Arts at ArRass, Qassim University, P.O. Box 53, ArRass 51921, Saudi Arabia; taha_al2003@hotmail.com (T.R.); k.zennir@qu.edu.sa (K.Z.); sewalemg@yahoo.com (S.G.)
4 Laboratory of Fundamental and Applied Mathematics of Oran (LMFAO), University of Oran 1 Ahmed Benbella, Oran 31000, Algeria
5 Department of Mathematics and Statistics, Port Said University, Port Said 42511, Egypt
6 Faculty of Economics, University Mustapha Stambouli of Mascara, Mascara 29000, Algeria; nadia.mezouar@univ-mascara.dz
7 Department of Mathematics, College of Science, Annaba University, P.O. Box 12, Annaba 23000, Algeria; haiourm@yahoo.fr
8 Department of Mathematics, College of Science, Qassim University, P.O. Box 53, Buraydah 51921, Saudi Arabia; aallahem@qu.edu.sa
* Correspondence: S.Boulaaras@qu.edu.sa or saleh_boulaares@yahoo.fr; Tel.: +966-5596183

Received: 6 March 2019; Accepted: 28 March 2019; Published: 3 April 2019

Abstract: In the context of the development of the research project (Boulaaras et al.) which was supported by the Deanship of Scientific Research of Qassim University under the Project No. 1140 during the academic year 2016/1437, this research will introduce a two dimensional mathematical model based on the finite elements spatial method combined with a finite differences time scheme applied to the diffusion equations. This model will be applied in the hypothetical example where the obtained results will be compared with the real experimental data. Such a comparison will allow to predict the soil temperature for different depths and at different time periods according to certain conditions on the weather and take into account the stability conditions of the used numerical method. It’s worthy of note that some of the concreted mathematical problems will be addressed by the experimental scientific theories of functional analysis and then the numerical simulation of the theoretical study will be provided thanks to sophisticated methods with more convergence and stability. This mathematical model would be reduced by means of physical measurement and also in charge of the physical cost.

Keywords: mathematical model; heat propagation equations; soil temperature; finite elements; finite differences time scheme

1. Introduction

In the recent years, many alternative methods have been used to speed up solving partial differential equations explaining natural phenomena. At this regard, Galerkin methods are being used instead of finite difference style methods to resolve the horizontal and vertical fields in numerical
model. These methods can be used to solve partial differential systems and the problems of free border as presented in [1–7], but so far, such works do not use this method directly in the field of values of grid points after the scale partition. Therefore, no improvements was held concerning the accuracy of the solutions of the ordinary differential equation systems. The processes adopted in solving such systems basically consists of two methods: The finite element method, and the spectral method. Finite element method is a numerical method relies on principles inherited from the weak formulation, which implements a discreet mathematical algorithm aiming to find an approximate solution to the problem of free boundary (see [8–10]), and well known as Dirichlet conditions or Neumann data. The method of finite elements differs from the spectral method because it is not exhaustive; it allows instead to specify local values. However, it is a specific approximate function defined throughout the region and not just at distinct points (see [11–20]).

In fact, numerical methods can be used to solve complicated and unsolvable problems by analytical methods. The use of numerical solutions is limited in terms of the relative errors that must appear after doing a given numbers of iterations in the numerical calculations, and which must be estimated on an appropriate way to judge the acceptance or the rejection of the obtained results. Moreover, the numerical method raises questions about the proposed division such as the convergence and the stability of the method in order to ensure the existence and the uniqueness of the solutions of the studied problem. The numerical analysis relies on the description of the methods, which may allow obtaining numerical solutions to mathematical issues that are usually difficult to be solved by the usual analytical methods. On some issues, we cannot find solutions set by direct analytical methods and conventional theories. In such cases we look for algorithms of specific numerical methods to get the almost exact solutions in given points. Computational modeling can provide a very easy way to predict, and in many cases, can significantly reduce the required amount of field testing. The development of a two-dimensional and three-dimensional model which can be applied to some evolutionary partial equations during the second half of the twentieth century, drew on a continuous progress through automated calculation tools based on classical numerical methods such as finite differences.

In this context, and given that the exact analytic solutions are difficult to be obtained, or indeed do not exist, modern numerical methods, such as the finite elements method, are being applied to some diffusion equations describing the behavior of the soil temperature for different depths and at different day-times according to certain conditions on the weather. Thus, the finite element method, which is falling into a more general Galerkin family method, will be applied along this work to solve such diffusion equations in two-dimension case. The method should take into account the stability conditions of the used numerical method, where the expected values are determined from the initial known values through the observations of the temperature source. The computational model of the transport equations consists of two main steps: The first is to present the equation system that governs the behavior of the continuous temperatures in the soil. The second step is to supersede the equations on the continuous variables with equations involving a separate variable of its solution, obtained by an appropriate algorithm. Moreover, we aim into this research to highlight the application of the method to parabolic equation.

The outline of the paper is as follows: In the first section, some necessary notations of the problem by means of detailed finite element methods description are introduced, where the numerical analysis of the heat equations is studied in order to get the weak formulation of the presented problem using the finite deference time scheme combined with Galerkin method. Then, we establish the stability analysis of the given schemes. In the second section, a two-dimensional numerical prediction model that predicts the temperature field for different depths of homogeneous soil and for different times as well as days is presented. By using the finite element approximation of the Lagrange polynomial of order 1, it will be possible to check the effects of the different parameters governing the physics phenomena as well as the effects of the soil characteristics. And finally, we perform experimental measurements of the ground soil temperature in Qassim province in Kingdom of Saudi Arabia which will be used as real data in order to prove the accuracy of the presented numerical method.
2. Finite Element Methods

2.1. Finite Element Methods Description

The finite element method can be defined on \( \hat{T} \subset \mathbb{R}^p \) by a set of \( n_d \) nodes \( a^i \) and corresponding base functions

\[
\hat{\varphi}^i : \hat{T} \subset \mathbb{R}^p \to \mathbb{R}^Q
\]

Each base function corresponds to a degree of freedom (dof). Most of finite element methods are scalar, which means that \( Q = 1 \) but GETFEM++ support also intrinsic vectorial elements. The map between the reference element and the real element is called geometric transformation and is denoted by

\[
\tau : \hat{T} \to T
\]

This map is supposed to be polynomial (see [1,13]). The base functions \( \hat{\varphi}^i \) defined on the reference element define a set of base functions on the real element defined by

\[
\varphi^i (x) = \hat{\varphi}^i (\tau^{-1} (x))
\]

If the element is said to be equivalent through the geometric transformation \( \tau \) (or \( \tau \)-equivalent) then the base functions on the real element are just defined by

\[
\varphi^i (x) = \hat{\varphi}^i (x)
\]

This is generally the case for Lagrange elements, but not for Hermite elements (when some dof represent the gradient of the unknown). When the element is not equivalent through the geometric transformation then GETFEM++ allows to define a square matrix \( \tilde{M} \) depending on the real element (i.e., on the geometric transformation) such that base functions on the real element are defined by

\[
\varphi^i (x) = \sum_{j=0}^{n_d-1} \tilde{M}_{ij} \hat{\varphi}^j (x)
\]

We denote by

\[
[\hat{\varphi} (\hat{x})] = \begin{pmatrix}
\hat{\varphi}^0 (\hat{x}) \\
\hat{\varphi}^1 (\hat{x}) \\
\vdots \\
\hat{\varphi}^{n_d-1} (\hat{x})
\end{pmatrix}
\]

the \( n_d \times Q \) matrix, such that when a function is defined by

\[
f (x) = \sum_{i=0}^{n_d-1} a_i \varphi^i (x)
\]

One has

\[
f (\tau (\hat{x})) = \alpha^T \tilde{M} [\hat{\varphi} (\hat{x})]
\]

where \( \alpha \) is the vector of the components \( a_i \).

2.1.1. Different Types of d.o.f

To each base function of a finite element method corresponds a degree of freedom (d.o.f.) which is a linear form on this function. Table 1 gives the most significant types of d.o.f.
Table 1. Most significant types of d.o.f.

<table>
<thead>
<tr>
<th>Type</th>
<th>Expression</th>
<th>Commentary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lagrange type</td>
<td>$\Phi(a_i)$</td>
<td>Value of $\Phi$ on the node ($a_i$). The most simple d.o.f. allows the Lagrange interpolation and the value of some other base functions. This is generally the bubble functions type of d.o.f.</td>
</tr>
<tr>
<td>Hierarchical Lagrange Type</td>
<td>$\Phi(a_i) - ...$</td>
<td>Difference between the value of $\Phi$ on the node ($a_i$) and the value of some other base functions. This is generally the bubble functions type of d.o.f.</td>
</tr>
<tr>
<td>Mean type</td>
<td>$\frac{1}{</td>
<td>T</td>
</tr>
<tr>
<td>Derivative type</td>
<td>$\frac{\partial \Phi(a_i)}{\partial x}$ or $\frac{\partial \Phi(a_i)}{\partial \eta}$</td>
<td>Value of a derivative of $\Phi$ on the node ($a_i$). This kind of d.o.f. makes the element no to be $\tau$-equivalent. $\frac{\partial \Phi(a_i)}{\partial \eta}$ denotes the normal derivative with respect to a face</td>
</tr>
<tr>
<td>Second derivative</td>
<td>$\frac{\partial^2 \Phi(a_i)}{\partial x_1 \partial x_2}$</td>
<td>Value of a second derivative of $\Phi$ on the node ($a_i$). This kind of d.o.f. makes also the element no to be $\tau$-equivalent.</td>
</tr>
</tbody>
</table>

2.1.2. Graphical Codification of d.o.f

The graphical codification of d.o.f can be defined as shown in Figure 1.

- Value of the function at the node
- Value of the gradient along the first coordinate
- Value of the gradient along the second coordinate
- Value of the gradient along the third coordinate for 3D elements
- Value of the whole gradient at the node
- Value of the normal derivative to a face
- Value of the second derivative along the first coordinate (twice)
- Value of the second derivative along the second coordinate (twice)
- Value of the second cross derivative in 2D or second derivative along the third coordinate (twice) in 3D
- Value of the whole second derivative (hessian) at the node
- Scalar product with the normal to a face for a vectorial element
- Bubble function on an element or a face, to be specified
- Lagrange hierarchical d.o.f. Value at the node in a space of details

Figure 1. Symbols representing degree of freedom types.

2.2. Classical $P_k$ Lagrange Elements on Simplifies

We can define a classical $P_k$ Lagrange element of qualitative dimension and qualitative, degree. This element has only degrees of freedom which corresponds to the value of the function on a node. The grid of node is the so-called Lagrange grid. Figures 2–4 show examples of dimension 1, 2 and 3.
Figure 2. Examples of classical $P_K$ Lagrange elements on a segment.

Figure 3. Examples of classical $P_K$ Lagrange elements on a triangle.

The degree number of freedom for a classical $P_K$ Lagrange element of dimension $P$ and degree $K$ is $\frac{(P+K)!}{P!K!}$. For example, in dimension 2 ($P = 2$), this value is $\frac{(P+1)(P+2)}{2}$, in dimension 3 ($P = 3$), this value is $\frac{(P+1)(P+2)(P+3)}{6}$.

The method specified in GETFEM++ for the numbering of contracts is also illustrated in the Figures 2–4. Using another numeration, let

$$i_0, i_1, ..., i_p$$

be some indexes such that $0 \leq i_0, i_1, ..., i_p \leq K$ and $\sum_{n=0}^{p} i_n = K$. 
Thus, we can compute the coordinate of a node as:

\[ a_{i_0, i_1, \ldots, i_p} = \sum_{n=0}^{p} i_n S_n, \text{ for } K \neq 0 \]

where \( S_0, S_1, \ldots, S_n \) are the vertexes of the simplex (for \( K = 0 \) the particular choice \( a_{0,0,\ldots,0} = \frac{1}{\sum_{n=0}^{p} S_n} \) has been chosen). Then each base function, corresponding of each node \( a_{i_0, i_1, \ldots, i_p} \) is defined by

\[ \Phi_{i_0, i_1, \ldots, i_p} = \prod_{n=0}^{p} \prod_{j=0}^{i_n-1} \left( \frac{K\lambda_n - j}{j+1} \right) \]

where \( \lambda_n \) are the barycenter coordinates, that is to say the polynomials of degree 1 whose value is 1 on the vertex \( S_n \) and whose value is 0 on other vertices, we have

\[ \lambda_n = x_n, \quad 0 \leq n < P \]
\[ \lambda_p = 1 - x_0 - x_1 - \ldots - x_{p-1} \]

Between two elements of the same degrees, d.o.f. of the common face are linked, and the element is of class \( C^0 \). This means that the global polynomial is continuous. If we try to link elements of different degrees, we will have some problems with the unbound d.o.f. This is not automatically supported by GETFEM++, so we will have to support it.

For some applications (gradient calculation for example), we do not want d.o.f. of a common face to bind. That’s why there are two versions of the classical \( P_K \) Lagrange element (see Tables 2–4).
Table 2. Classical “$P_k$” Lagrange element.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Dimension</th>
<th>d.o.f. number</th>
<th>class</th>
<th>vectorial</th>
<th>$\tau$-equivalent</th>
<th>Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_i$</td>
<td>$P_i$</td>
<td>$(K_i + P_i)!$</td>
<td>$C^0$</td>
<td>$N_{0Q}$</td>
<td>$Q = 1$</td>
<td>$\tilde{M} = \text{Id}$</td>
</tr>
<tr>
<td>$0 \leq K \leq 255$</td>
<td>$0 \leq P \leq 255$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Discontinuous “$P_k$” Lagrange element.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Dimension</th>
<th>d.o.f. number</th>
<th>class</th>
<th>vectorial</th>
<th>$\tau$-equivalent</th>
<th>Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_i$</td>
<td>$P_i$</td>
<td>$(K_i + P_i)!$</td>
<td>Discontinuous</td>
<td>$N_{0Q}$</td>
<td>$Q = 1$</td>
<td>$\tilde{M} = \text{Id}$</td>
</tr>
<tr>
<td>$0 \leq K \leq 255$</td>
<td>$0 \leq P \leq 255$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Although Lagrange elements are set at random levels, a high degree selection can be a problem for a large number of applications. These elements are recommended for basic interpolation but for p.d.e. Applications, hierarchical elements are best (see corresponding section).

2.3. Other Geometries of Classical Lagrange

The classic Lagrange is obtained in parallel or prisms like a tensorial product of Lagrange elements on simplifies. When two elements are defined, we have on a dimension $P_1$ and the other in dimension $P_2$, we have the base functions of the tensorial product (on the reference element) as

$$
\hat{\phi}_{ij}(x, y) = \hat{\phi}^1_i(x) \hat{\phi}^2_j(x), \quad x \in \mathbb{R}^{P_1}, y \in \mathbb{R}^{P_2}
$$

where $\hat{\phi}^1_i$ and $\hat{\phi}^2_j$ are respectively the base functions of the first and second element (see Figure 5).

Figure 5. Examples of classical Lagrange elements on dimension 2.

The $Q_K$ element on a parallelepiped of dimension $P$ is obtained like the tensorial product of $P$ classical $P_K$ element on the segment. The examples in dimension 2 are shown in Figure 6 and in dimension 3 in Figure 7.

A prism in dimension $P$ is the direct product of a simplex of dimension $P - 1$ with a segment. The $P_K \otimes P_K$ element on this prism is the tensorial product of the classical $P_K$ element on a simplex of dimension $P - 1$ with the classical $P_K$ element on a segment. For $P = 2$ this coincide with a parallelepiped. Examples in dimension 3 are shown in Figure 7. This is also possible because we do not have the same degree in every dimension (see Figures A10 and 10). An example appears in Figures 8–10 and a recapitulation in Table 4.
Figure 6. Examples of classical Lagrange elements on dimension 3.

Figure 7. $P_2 \otimes P_1$ Lagrange element on a prism 12 d.o.f. $C^0$.

<table>
<thead>
<tr>
<th>$P_K \otimes P_K$ Lagrange element on prisms</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>FEM.PK.PRISM(P, K)</em></td>
</tr>
<tr>
<td>Degree</td>
</tr>
<tr>
<td>$2K$, $0 \leq K \leq 255$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$P_{K_1} \otimes P_{K_2}$ Lagrange element on prisms</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>FEM.PRODUCT(FEM.PK(P-1, K_1), FEM.PK(1, K_2))</em></td>
</tr>
<tr>
<td>Degree</td>
</tr>
<tr>
<td>$K_1 + K_2$, $k \leq K_1, K_2 \leq 255$</td>
</tr>
</tbody>
</table>

Figure 8. Lagrange element on prisms.
Table 4. $Q_K$ Lagrange element on parallelepipeds.

<table>
<thead>
<tr>
<th>Degree</th>
<th>Dimension</th>
<th>d.o.f. number</th>
<th>class</th>
<th>Vectorial</th>
<th>$\tau$-equivalent</th>
<th>Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>$KP$, $0 \leq K \leq 255$</td>
<td>$P$, $2 \leq P \leq 255$</td>
<td>$(K + 1)^P$</td>
<td>$C^0$</td>
<td>$N_0$</td>
<td>$\tilde{M} = \text{Id}$</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Figure 9. Incomplete $Q_2$ element, 8 d.o.f., $C^0$.

Figure 10. Incomplete $Q_2$ Lagrange on quadrilateral (Quad 8 serendipity element).

2.4. Elements with Hierarchical Basis

The idea behind the hierarchical base is to describe the solution at a different level: the rebound level and the most precise level. In the same estimation, some degrees of freedom represent a corrupt description, others more precise, etc. This corresponds to the precision domains. The hierarchical base contains a base for each of these spaces (this is not the case in classic Lagrange elements when optimizing the grid).

Among the advantages, the condition number of the stiffness matrices can be improved, which allows a local multigrid refinement approach.

Hierarchical Elements with Respect to the Degree

Hierarchical Elements with Respect to the Degree are illustrated in Figures 11 and 12.
2.5. Elements with Hierarchical Basis

Some particular choices: $P_4$ will be build with the basis of the $P_1$, the additional basis of the $P_2$ then the additional basis of the $P_4$.  
$P_6$ will be build with the basis of the $P_1$, the additional basis of the $P_2$ then the additional basis of the $P_6$ (not with the basis of the $P_1$, the additional basis of the $P_3$ then the additional basis of the $P_6$, this possible to build the latter with FEM-GEN-HIERARCHICAL(a,b) (see Figure 13).

2.5.1. Composite Elements

The main interest of composite elements is to build hierarchical elements. But this tool can also be used to build piecewise polynomials.

It is important to use a corresponding composite integration method. (see Figures 13 and 14)
2.5.2. Hierarchical Composite Elements

The method principle of Hierarchical composite element “FEM-PK-HIERARCHICAL-COMPOSITE (2,1,3)” and $P_K$ finite element method on a simplex with $S$ subdivisions are demonstrated in Figures 15–17.

<table>
<thead>
<tr>
<th>composition of a finite element method on a element with $S$ subdivisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{FEM}_i-\text{STRUCTURED-COMPOSITE}(\text{FEM}_i, S)$</td>
</tr>
<tr>
<td>Degree</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>$\text{FEM}_i$</td>
</tr>
</tbody>
</table>

Figure 13. Composite element “FEM-STRUCTURED-COMPOSITE (FEM-PK (2,1,3))”.

Figure 14. Composite of a finite element method on an element with $S$ subdivisions.

Figure 15. Hierarchical composite element “FEM-PK-HIERARCHICAL-COMPOSITE (2,1,3)”.

<table>
<thead>
<tr>
<th>hierarchical composition of a $P_K$ finite element method on a simplex with $S$ subdivisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{FEM}_i$-PK-HIERARCHICAL-COMPOSITE $(P, K, S)$</td>
</tr>
<tr>
<td>Degree</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>$K$</td>
</tr>
</tbody>
</table>

Figure 16. Hierarchical composition of a $P_K$ finite element method on a simplex with $S$ subdivisions.
2.6. Specific Elements in Dimension 1

2.6.1. GaussLobatto Element

The polynomials coefficients are pre-computed with Maple, hence they are only available for the following values of $K$: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 24, 32. We can remark that for $K = 1$ and $K = 2$, this is for the classical $P_1$ and $P_2$ (see Figure 18).

2.6.2. Hermite Element

Base functions on the reference element
\begin{align*}
\phi_0 &= (2x + 1)(x - 1)^2 \\
\phi_1 &= x^2(3 - 2x) \\
\phi_2 &= x(x - 1)^2 \\
\phi_3 &= x^2(x - 1)
\end{align*}

Note that, this element is close to be $\tau$-equivalent. The real value of the gradient on vertices are multiplied by the gradient of the geometric transformation. $\tilde{M}$ is not equal to identity but is diagonal (see Figures 19 and 20).

2.6.3. Lagrange Element with an Additional Bubble Function

In the following, Figure 21 is showing Lagrange Element with an Additional Bubble Function, while Figure 22 is showing Lagrange $P_1$ element with an additional internal bubble function.
Figure 21. $P_1$ Lagrange element on a segment with additional internal bubble function, 3 d.o.f., $C^0$.

<table>
<thead>
<tr>
<th>Lagrange $P_1$ element with an additional internal bubble function</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;FEM_PK.WITH.CUBIC.BUBBLE(1, 1)&quot;</td>
</tr>
<tr>
<td>Degree</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

Figure 22. Lagrange $P_1$ element with an additional internal bubble function.

2.7. Specific Elements in Dimension 2

2.7.1. Elements with Additional Bubble Functions

Figures 23–25 are explaining the principle of Elements with Additional Bubble Functions.

Figure 23. Lagrange element on a triangle with additional internal bubble function. (a) $P_1$ with additional internal bubble function, 4 d.o.f., $C^0$, (b) $P_2$ with additional internal bubble function, 7 d.o.f., $C^0$.

<table>
<thead>
<tr>
<th>Lagrange $P_1$ or $P_2$ element with an additional internal bubble function</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;FEM_PK.WITH.CUBIC.BUBBLE(2, 1)&quot;</td>
</tr>
<tr>
<td>Degree</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Figure 24. $P_1$ Lagrange element on a triangle with additional internal bubble function on face 0, 4 d.o.f., $C^0$.
2.7.2. Non Conforming $P_1$ Element

Figures 26 and 27 are explaining the treatment of Non Conforming $P_1$ Element.

Figure 25. $P_1$ Lagrange element on a triangle with additional d.o.f on face 0, 4 d.o.f., $C^0$.

Figure 26. $P_1$ non-confirming element on a triangle 3 d.o.f., discontinuous.
2.7.3. Hermite Element

We can give the following base functions:

\[
\hat{\phi}_0 = (1 - x - y) \left(1 + x + y - 2x^2 - 2y^2 - 11xy\right), \quad (\hat{\phi}_0 (0,0) = 1)
\]
\[
\hat{\phi}_1 = -2x^3 + 7x^2y + 7xy^2 + 3x^2 - 7xy, \quad (\hat{\phi}_1 (1,0) = 1)
\]
\[
\hat{\phi}_2 = 7x^2y + 7xy^2 - 2y^3 + 3y^2 - 7xy, \quad (\hat{\phi}_2 (0,1) = 1)
\]
\[
\hat{\phi}_3 = 27xy (1 - x - y), \quad (\hat{\phi}_3 (1/3,1/3) = 1)
\]
\[
\hat{\phi}_4 = x (1 - x - y) (1 - x - 2y), \quad (\partial_x \hat{\phi}_4 (0,0) = 1)
\]
\[
\hat{\phi}_5 = x^3 - 2x^2y - 2xy^2 - x^2 + 2xy, \quad (\partial_x \hat{\phi}_5 (1,0) = 1)
\]
\[
\hat{\phi}_6 = xy (x + 2y - 1), \quad (\partial_x \hat{\phi}_6 (0,1) = 1)
\]
\[
\hat{\phi}_7 = y (1 - x - y) (1 - 2x - y), \quad (\partial_y \hat{\phi}_7 (0,0) = 1)
\]
\[
\hat{\phi}_8 = xy (y + 2x - 1), \quad (\partial_y \hat{\phi}_8 (1,0) = 1)
\]
\[
\hat{\phi}_9 = y^3 - 2x^2y - 2xy^2 - y^2 + 2xy, \quad (\partial_y \hat{\phi}_9 (0,1) = 1)
\]

This element is not \(\tau\)-equivalent (the matrix \(\tilde{M}\) is not equal to identity). On the real element linear combinations of \(\hat{\phi}_4\) and \(\hat{\phi}_7\) are not used to match the gradient on the corresponding vertex. Idem for the two couples \((\hat{\phi}_5, \hat{\phi}_8)\) and \((\hat{\phi}_6, \hat{\phi}_9)\) for the two other vertices (see Figure 28).

<table>
<thead>
<tr>
<th>Hermite element on a triangle</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;FEM_HERMITE(2)&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Degree</th>
<th>dimension</th>
<th>d.o.f. number</th>
<th>class</th>
<th>vectorial</th>
<th>(\tau)-equivalent</th>
<th>Polynomial</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>10</td>
<td>(C^0)</td>
<td>No ((Q = 1))</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

2.7.4. Argyric Element

Figure 29 is defining the Argyric Element.
We can give the following base functions:

\[
\begin{align*}
\phi_0 &= 1 - 10x^3 - 10y^3 + 15x^4 - 30x^2y^2 + 15y^4 - 6x^5 + 30x^3y^2 + 30x^2y^3 - 6y^5, & (\phi_0(0,0) &= 1) \\
\phi_1 &= 10x^3 - 15x^4 + 15x^2y^2 + 6x^5 - 15x^3y^2 - 15x^2y^3, & (\phi_1(1,0) &= 1) \\
\phi_2 &= 10y^3 - 15y^4 + 15x^2y^2 + 6y^5 - 15x^3y^2 - 15x^2y^3, & (\phi_2(0,1) &= 1) \\
\phi_3 &= x - 6x^3 - 11xy^2 + 8x^4 + 10x^2y^2 + 18xy^4 - 3x^5 + x^3y^2 - 10x^2y^3 - 8xy^4, & (\partial_x\phi_3(0,0) &= 1) \\
\phi_4 &= -4x^3 + 7x^4 - 3.5x^2y^2 - 3x^5 + 3.5x^4y^2 + 3.5x^3y^3, & (\partial_x\phi_4(1,0) &= 1) \\
\phi_5 &= -5xy^2 + 18.5x^2y^2 + 14xy^4 - 13.5x^2y^3 + 18.5x^2y^4 - 8y^4, & (\partial_x\phi_5(0,1) &= 1) \\
\phi_6 &= y - 11x^2y - 6y^3 + 18x^3y + 10x^2y^2 + 8y^4 - 8x^4y - 10x^3y^2 + x^2y^3 - 3y^5, & (\partial_y\phi_6(0,0) &= 1) \\
\phi_7 &= -5x^2y + 14x^3y + 18.5x^2y^2 - 8x^4y - 18.5x^3y^2 - 13.5x^2y^3, & (\partial_y\phi_7(1,0) &= 1) \\
\phi_8 &= -4y^3 - 3.5x^3y^2 + 7y^4 + 3.5x^2y^3 + 3.5x^2y^4 - 3y^5, & (\partial_y\phi_8(0,0) &= 1) \\
\phi_9 &= 0.5x^2 - 1.5x^3 + 1.5x^2y - 0.5x^3y + 1.5x^2y^2 + x^2y^3, & (\partial_y\phi_9(0,0) &= 1) \\
\phi_{10} &= 0.5x^3 - x^4 + 0.25x^2y^2 + 0.5x^2 - 0.25x^2y^2 - 0.25x^2y^4, & (\partial_y\phi_{10}(1,0) &= 1) \\
\phi_{11} &= 1.25x^2y^2 - 1.25x^3y^2 - 0.75x^2y^4, & (\partial_y\phi_{11}(1,0) &= 1) \\
\phi_{12} &= xy - 4x^2y - 4xy^2 + 5x^3y + 10x^2y^2 + 5xy^3 - 2x^4y - 6x^3y^2 - 6x^2y^3 - 2xy^4, & (\partial_y\phi_{12}(0,1) &= 1) \\
\phi_{13} &= x^2y - 3x^3y - 3.5x^2y^2 + 2x^4y + 3.5x^3y^2 + 2.5x^2y^3, & (\partial_y\phi_{13}(1,0) &= 1) \\
\phi_{14} &= xy^2 - 3.5x^2y^2 - 3xy^3 + 2.5x^3y^3 + 3.5x^2y^3 + 2xy^4, & (\partial_y\phi_{14}(0,1) &= 1) \\
\phi_{15} &= 0.5y^2 - 1.5y^3 - 1.5x^2y^3 + 1.5y^4 + x^2y^3 + 1.5x^2y^3 - 0.5y^5, & (\partial_y\phi_{15}(0,0) &= 1) \\
\phi_{16} &= 1.25x^2y^2 - 0.75x^3y^2 - 1.25x^2y^4, & (\partial_y\phi_{16}(1,0) &= 1) \\
\phi_{17} &= 0.5y^2 + 0.25x^2y^2 - y^4 - 0.25x^2y^2 - 0.25x^2y^4 + 0.5y^5, & (\partial_y\phi_{17}(0,1) &= 1) \\
\phi_{18} &= \sqrt{2} \left( -8x^2y^2 + 8x^2y^2 + 8x^2y^2 \right) \cdot \left( \sqrt{23} (\partial_x\phi_0(0,0,0,0) + \partial_y\phi_1(0,0,0,0)) = 1 \right) \\
\phi_{19} &= -16xy^2 + 32x^2y^2 + 32xy^3 - 16x^2y^2 - 32x^2y^3 - 16xy^4, & (\partial_y\phi_{19}(0,0) &= 1) \\
\phi_{20} &= -16x^2y + 32x^3y + 32x^2y^2 - 16x^2y - 32x^2y^3 - 16x^2y^4, & (\partial_y\phi_{20}(0,0) &= 1)
\end{align*}
\]

On the real element linear combinations of the transformed base functions \( \phi_i \) are used to match the gradient, the normal derivatives and second derivatives and on the faces. We ca note that the use of the matrix \( \hat{M} \) (see also the documentation on the finite element kernel [1]) allows to define Argyre element even with nonlinear geometric transformations (see Figure 30).
2.8. Specific Elements in Dimension 3

2.8.1. Elements with additional Bubble Functions

The Elements with additional Bubble Functions principle is explained in Figures 31–33.

**Figure 30.** Argyric element on a triangle.

**Figure 31.** Lagrange element on tetrahedron with additional internal bubble function. (a) $P_1$ with additional bubble function, 5 d.o.f., $C^0$. (b) $P_2$ with additional bubble function, 11 d.o.f., $C^0$. (c) $P_3$ with additional bubble function, 21 d.o.f., $C^0$.

**Figure 32.** $P_1$ Lagrange element on tetrahedron with additional internal bubble function on face 0, 5 d.o.f., $C^0$. 

---

**Argyris element on a triangle**

<table>
<thead>
<tr>
<th>Degree</th>
<th>Dimension</th>
<th>d.o.f. number</th>
<th>Class</th>
<th>Vectorial</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>21</td>
<td>$C^1$</td>
<td>No ($Q = 1$)</td>
</tr>
</tbody>
</table>

| Elements with additional Bubble Functions principle |

<table>
<thead>
<tr>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C^0$</td>
<td>$C^0$</td>
<td>$C^0$</td>
</tr>
</tbody>
</table>

---

**Figure 31.** Lagrange element on tetrahedron with additional internal bubble function. (a) $P_1$ with additional bubble function, 5 d.o.f., $C^0$. (b) $P_2$ with additional bubble function, 11 d.o.f., $C^0$. (c) $P_3$ with additional bubble function, 21 d.o.f., $C^0$. 

---

**Figure 32.** $P_1$ Lagrange element on tetrahedron with additional internal bubble function on face 0, 5 d.o.f., $C^0$. 

Figure 33. Lagrange $P^1$ element with an additional internal bubble function on face 0.

2.8.2. Hermite Element

Figure 34 is showing the case of Hermite Element.

![Hermite element on a tetrahedron](image)

**Figure 34.** Hermite element on a tetrahedron, $P_3$, 20 d.o.f., $C^0$.

Base functions on the reference element:

$$
\begin{align*}
\phi_0 & = 1 - 3x^2 - 13xy - 13xz - 3y^2 - 13yz - 3z^2 + 2x^3 + 13x^2y + 13x^2z \\
& + 13xy^2 + 33xyz + 13xz^2 + 2y^3 + 13y^2z + 13yz^2 + 2z^3, \\
\phi_1 & = 3x^2 - 7xy - 7xz - 2x^3 + 7x^2y + 7x^2z + 7xyz + 7xz^2, \\
\phi_2 & = -7xy + 3y^2 - 7yz + 7x^2y + 7xy^2 + 7xyz - 2y^3 + 7y^2z + 7yz^2, \\
\phi_3 & = -7xz - 7yz + 3z^2 + 7x^2z + 7xyz + 7x^2z + 7y^2z + 7yz^2 - 2z^3, \\
\phi_4 & = 27xyz, \\
\phi_5 & = 27yz - 27xyz - 27y^2z - 27yz^2, \\
\phi_6 & = 27xz - 27x^2z - 27xyz - 27xz^2, \\
\phi_7 & = 27xy - 27x^2y - 27xyz - 27xyz, \\
\phi_8 & = x - 2x^2 - 3xy - 3xz + x^3 + 3x^2y + 3x^2z + 2xy^2 + 4xyz + 2xz^2, \\
\phi_9 & = -x^2 + 2xy + 2xz + x^3 - 2x^2y - 2x^2z - 2xy^2 - 2x^2z - 2xz^2, \\
\phi_{10} & = -xy + x^2y + 2x^2z, \\
\phi_{11} & = -xz + x^2z + 2xz^2, \\
\phi_{12} & = y - 3xy - 2y^2 - 3yz + 2x^2y + 3xy^2 + 4xyz + y^3 + 3y^2z + 2yz^2.
\end{align*}
$$

($\phi_0 (0,0,0) = 1,$ $\phi_1 (1,0,0) = 1,$ $\phi_2 (0,1,0) = 1,$ $\phi_3 (0,0,1) = 1,$ $\phi_4 (1/3,1/3,1/3) = 1,$ $\phi_5 (0,1/3,1/3) = 1,$ $\phi_6 (1/3,0,1/3) = 1,$ $\phi_7 (1/3,1/3,0) = 1,$ $\phi_8 (0,0,0) = 1,$ $\phi_9 (1,0,0) = 1,$ $\phi_{10} (0,1,0) = 1,$ $\phi_{11} (0,0,1) = 1,$ $\phi_{12} (0,0,0) = 1$)
\[
\hat{\phi}_{13} = -xy + 2x^2y + xy^2, \\
\hat{\phi}_{14} = 2xy - y^2 + 2yz - 2x^2y - 2xyz + y^3 - 2y^2z - 2yz^2, \\
\hat{\phi}_{15} = -yz + y^2z + 2yz^2, \\
\hat{\phi}_{16} = z - 3xz - 3yz - 2z^2 + 2x^2z + 4xyz + 3x^2z + 2y^2z + 3yz^2 + z^3, \\
\hat{\phi}_{17} = -xz + 2x^2z + xz^2, \\
\hat{\phi}_{18} = -yz + 2y^2z + yz^2, \\
\hat{\phi}_{19} = 2xz + 2yz - z^2 - 2x^2z - 2xyz - 2x^2z - 2y^2z - 2yz^2 + z^3,
\]

\[
(\partial_y \hat{\phi}_{13} (1,0,0) = 1) \\
(\partial_y \hat{\phi}_{14} (0,1,0) = 1) \\
(\partial_y \hat{\phi}_{15} (0,0,1) = 1) \\
(\partial_z \hat{\phi}_{16} (0,0,0) = 1) \\
(\partial_z \hat{\phi}_{17} (1,0,0) = 1) \\
(\partial_z \hat{\phi}_{18} (0,1,0) = 1) \\
(\partial_z \hat{\phi}_{19} (0,0,1) = 1)
\]

This element is not \(\tau\)-equivalent (the matrix \(\tilde{\mathbf{M}}\) is not equal to identity). On the real element linear combinations of \(\hat{\phi}_8, \hat{\phi}_{12}\) and \(\hat{\phi}_{16}\) are used to match the gradient on the corresponding vertex. Idem on the other vertices (see Figure 35).

### Hermitian element on a tetrahedron

<table>
<thead>
<tr>
<th>FEM</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;FEM.HERMITE(3)&quot;</td>
<td></td>
</tr>
<tr>
<td>Degree</td>
<td>dimension</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**Figure 35.** Hermite element on a tetrahedron.

### 2.9. Interpolation of Elements on Different Meshes

To increase the precision, it is not necessary to raise the order of the integration method. It is recommended to keep the normal order and use composite integration methods (see Figures 36 and 37).

**Figure 36.** Interpolation of elements on different meshes.

**Figure 37.** Use of the composite integration method to increase the precision.

### 3. Integration Methods

#### 3.1. Integration Methods Description

Integration methods are of two types. Exact integrations of polynomials and approximated integrations (cubature formulas) of any function. Exact integration can only be used if all elements are polynomial and the geometric transformation is linear.
A descriptor on an integration method is available thanks to the function
\( ppi = \text{getfem} :: \text{int-method-descriptor}("\text{name of method}"") \) where "name of method" is a string to
be chooser among existing methods.

3.2. Exact Integration Methods

The list of the available exact integration methods is given in (Figures 38 and 39)

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*IM_NONE() *</td>
<td>Dummy integration method (new in getfem++ 1.7).</td>
</tr>
<tr>
<td><em>IM_EXACT_SIMPLEX(n)</em></td>
<td>Description of the exact integration of polynomials on the simplex</td>
</tr>
<tr>
<td></td>
<td>of reference of dimension ( n ).</td>
</tr>
<tr>
<td><em>IM_PRODUCT(a, b)</em></td>
<td>Description of the exact integration on the convex which is the direct</td>
</tr>
<tr>
<td></td>
<td>product of the convex in ( a ) and in ( b ).</td>
</tr>
</tbody>
</table>

**Figure 38.** Newton cotes integration methods.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>IM_EXACT_PARALLELEPIPED(n)</em></td>
<td>Description of the exact integration of polynomials on the parallelepiped</td>
</tr>
<tr>
<td></td>
<td>of reference of dimension ( n ).</td>
</tr>
<tr>
<td><em>IM_EXACT_PRISM(n)</em></td>
<td>Description of the exact integration of polynomials on the prism</td>
</tr>
<tr>
<td></td>
<td>of reference of dimension ( n ).</td>
</tr>
</tbody>
</table>

**Figure 39.** Newton cotes integration methods.

Even though a description of an exact integration method exists on parallelepiped or prisms, most
of the time geometric transformations on such element are not linear and the exact integration can not
be used.

3.3. Newton Cotes Integration Methods

Use "IM_NC (N,K)", "IM_NC_PARALLELEPIPED (N,K)" and "IM_NC_PRISM (N,K)" to have
the Newton cotes integration of order \( K \) respectively on simplifies, parallelepiped and prisms.

3.4. Gauss Integration Methods on Dimension 1

Use "IM_GAUSS1D (K)" to have the Gauss-Legendre integration on the segment of order \( K \) (with \( K/2 + 1 \) points), and "IM_GAUSSLOBATTO1D (K)" to have Gauss-Lobatto-Legendre
integration on the segment of order \( K \) (with \( K/2 + 1 \) points). The latter integration method is
only available for odd values of \( K \). The gauss-Lobatto integration can be used in conjunction with
"FEM_PK_GAUSSLOBATTO1D (K/2)" to perform mass lumping.

3.5. Gauss Integration Methods on Dimension 2

The is illustrated in Table 5.
Table 5. Gauss integration methods on dimension 2.

<table>
<thead>
<tr>
<th>Graphic</th>
<th>Coordinates</th>
<th>Weights</th>
<th>Function to Call/Order</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/150" alt="Graph" /></td>
<td>x</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>1/3</td>
<td>1/3</td>
<td>1/2</td>
<td>&quot;IM-TRIANGLE(1)&quot;</td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/150" alt="Graph" /></td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
</tr>
<tr>
<td>2/3</td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/150" alt="Graph" /></td>
<td>1/3</td>
<td>1/5</td>
<td>3/5</td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/150" alt="Graph" /></td>
<td>a = 0.445948490915965, b = 0.091576213509771, c = 0.111690794839005, d = 0.054975871827661</td>
<td></td>
<td>&quot;IM-TRIANGLE(3)&quot;</td>
</tr>
</tbody>
</table>

4. Implementation of the Finite Element Method

This section discusses guidelines for implementing computer science of the finite element method. We start with the first step: the data of the mesh. We then discuss discretization, construction of equations and finally the resolution.

4.1. Data Structure of a Mesh

The data of a mesh consists at least by giving:

- The list of nodes
- The connectivity table

4.1.1. The List of Nodes

It is characterized by:

- $N_s$ number of nodes.
• The coordinates of each node $i = 1, ..., N_s$.

For example:

• Dimension 1: $x_0 := a < x_1 < x_2 < ... < x_n := b$

$N_s = n + 1$

Table with entry of length $N_s$ (see Figure 40).

![Figure 40. List of nodes, dimension 1.](image)

Do not always store this data structure. If the grid is uniform, it is given implicitly by

$$N_s = n + 1; x_0 := a; x_i := x_0 + ih, i = 1, ..., N_s - 1$$

• Dimension 2: table of coordinates \{(x_{1,i}, x_{2,i})\}_{i=1}^{N_s}

Number of nodes: $N_s$

Table with two lines and $N_s$ columns (see Figure 41).

![Figure 41. List of nodes, dimension 2.](image)

In the following example: $N_s = 5$ (see Figure 42).

![Figure 42. Distinction of vertices on the boundary of the inner vertex.](image)

4.1.2. Connectivity Table

It is characterized by:

• $N_E$: number of elements.

• For each element $e = 1, ..., N_E$ is given: $n_{e,j}$ number of the node $j$ of element $e$.

For example:
• Dimension 1

\[ N_E = N_s - 1 \]

For each element \( e = 1, ..., N_E \)

\[ n_{e,1} := e, \quad n_{e,2} := e + 1 \]

For the one-dimensional case, the connectivity table is implicit from the list of vertices. It is therefore useless to store.

• Dimension 2: For a mesh in triangles,

\( N_E: \) number of triangles.

\( n_{e,j}: e = 1, ..., N_E, j = 1, 2, 3 \)

The order in which the vertex numbers are given is not important (see Figure 43). If we can, we must respect a meaning as here the trigonometric sense. This can facilitate for some problems some programming points.

![Figure 43. Dimension 2: For a mesh in triangles.](image)

Often a reference number is added to introduce a characterization of the equations at the level of each element.

The table in Figure 44 indicates that elements 1 and 3 have the same characteristics, as well as elements 2 and 4.

![Figure 44. Meshes in triangle for 2 dimensions.](image)

4.2. Meshing and Boundary Conditions

We limit ourselves to the case where the dimension of the boundary problem is \( N \leq 2 \). For \( N = 2 \), we assume that \( \Omega \) is a polygonal domain. We give ourselves a meshing \( \tau^h \) with segments if \( N = 1 \) and with triangles if \( N = 2 \). We denote by \( I_D \) the vertex numbers of \( \tau^h \) that belong to the closure \( \Gamma_D \) of \( \Gamma_D \); \( I_D \) is usually given using vertex reference numbers.

Let examine the following examples (see Figure 45):

1. Dimension 1: \( a := x_0 < ... < x_n := b, \Gamma_D = \{a\}, I_D := \{1\} \)
2. Dimension 2 : $I_D := \{1, 4\}$

Knowing that $X^h$ is the space of finite elements, we pose

$$V^h := \{v^h \in X^h; v^h_i = 0; i \in I_D\}$$

(or else $V^h$ is the subspace of $X^h$ whose nodal values on $\Gamma_D$ are zero)

If in dimension two we suppose that the mesh is compatible with the change of type of boundary conditions, i.e., if the junction points of $\Gamma_D$ and $\Gamma_N$ are vertices of the mesh, we have the following proposition.

**Proposition 1.** Under the condition of previous compatibility on the mesh, we have

$$V^h = X^h \cap V$$

**Proof.** In dimension $N = 1$, there is nothing to prove. In dimension $N = 2$, just note that a function of $P_1^{(2)}$ is null on an edge of a triangle if and only if it vanishes at the ends of this edge. \qed

We will admit the following proposition whose result is intuitive.

**Proposition 2.** The spaces $X^h$ and $V^h$ respectively constitute internal approximations of $H^1(\Omega)$ and of $V$.

1. Without any restriction if $\Omega = [0, L]$.
2. Under the following angle condition if $\Omega$ is a polygonal domain of $\mathbb{R}^2$

$$\begin{cases} 
\theta_T is the smallest angle of $T \in \tau^h, \theta^h = \min_{T \in \tau^h} \theta_T \\
\theta^h \geq \theta_0 > 0 \text{ independently of } h
\end{cases}$$

(1)

The angle condition expresses that the mesh must not have flattened triangles as in Figure 46.

**Figure 46.** Unaccepted flattened triangles from the angle conditions.

This is why often, to refine the meshes, one operates in the following way: one divide each triangle into 4 by joining the middle of its edges (see Figure 47).
4.3. Assembling

The term assembling refers to the process of forming the equations of the linear system resulting from the application of a finite element method. The effectiveness of this process, which is generally not available for other Galerkin methods, participates for a large part to the popularity of the finite element method.

It is recalled to obtain the discrete problems associated with the variational formulations Constructed Using Finite Element Spaces $X^h$ and $V^h$. We therefore give $g^h$ an approximation of $g$ in $X^h$. Assuming that the function $g|_{\Gamma_D} \in C^0(\Gamma_D)$, one can construct the function $g^h$ by the following way (of interpolation) by giving its nodal values $g^h_i$ on the vertices $a^h_i$ of the mesh $\tau^h$

$$g^h_i = g(a^h_i), \quad i \in I_D, \quad g^h_i \text{ random if } i \notin I_D$$

The discrete problem is then written

$$\begin{cases}
  u^h \in X^h, \quad u^h - h^h \in V^h \\
  a(u^h, v^h) = L v^h, \quad \forall v^h \in V^h
\end{cases}$$

The discrete problem is then written

$$\begin{cases}
  u^h \in X^h, \quad u^h - h^h \in V^h \\
  a(u^h, v^h) = L v^h, \quad \forall v^h \in V^h
\end{cases}$$

Base functions $\{B^h_i\}_{j=1}^{N_s}$ were determined by their restriction to every element. They are defined by the conditions

$$B^h_i \in X^h, \quad B^h_j \left( a^h_i \right) = \delta_{ij}, \quad i, j = 1, ..., N_s$$

The variational system (2) is then equivalent to a linear system, we interest now to the formation of the coefficients. These coefficients are obtained by the contribution of each element of the mesh. Each of these contributions is described by small matrices: the elementary matrices. The assembling process makes it possible to accumulate all these contributions.

4.4. Elementary Matrices

The restriction of the bilinear form $a(., .)$ and the linear form $L(., .)$ to the subspace $X^h$ is described in the basis of $B^h_i$ by the Total matrices

$$\begin{pmatrix}
  v_1^h & \cdots & v_{N_s}^h
\end{pmatrix} A^t \begin{pmatrix}
  u^h_1 \\
  \vdots \\
  u^h_{N_s}
\end{pmatrix} = a(u^h, v^h); \quad u^h, v^h \in X^h$$

$$\begin{pmatrix}
  v_1^h & \cdots & v_{N_s}^h
\end{pmatrix} L^t = L v^h; \quad v^h \in X^h$$

where $u^h_i$ (resp. $v^h_i$) are the nodal values of $u^h$ (resp. $v^h$). The assembling process is based on the fact that the integrals.
\[ a\left(u^h, v^h\right) = \int_{\Omega} \sum_{i,j=1}^{N} a_{ij} \partial_i \mu^h \partial_j v^h \, d\Omega + \int_{\Omega} a_0 u^h v^h \, d\Omega + \int_{\Gamma_N} \lambda u v \, d\Gamma \]

\[ L v^h = \int_{\Omega} f v^h \, d\Omega + \int_{\Gamma_N} h v \, d\Gamma \]

With the appropriate adaptations for the one-dimensional case, can be decomposed in a sum of integrals on each element

\[ a\left(u^h, v^h\right) = \sum_{(T^e) \in \mathcal{I}^e} \int_{T^e} \sum_{i,j=1}^{N} a_{ij} \partial_i \mu^e \partial_j v^e \, dT^e + \sum_{T^e \in \mathcal{I}^e} \int_{T^e} a_0 \mu^e v^e \, dT^e + \sum_{T^e \in \mathcal{I}^e} \int_{T^e} \lambda \mu^e v^e \, dT^e \]

\[ L v^h = \sum_{(T^e) \in \mathcal{I}^e} \int_{T^e} \int_{T^e} f v^e \, dT^e + \sum_{T^e \in \mathcal{I}^e} \int_{T^e} h v \, dT^e \]

where \( \sum \) indicates the sum on the faces (edges in the case \( N = 2 \)) composing the boundary of \( T^e \) and \( \mu^e = u^h \mid_{T^e} \in \mathcal{P}_1 \) (respectively \( v^e = v^h \mid_{T^e} \in \mathcal{P}_1 \)) the polynomial expressions of \( u^h \) (resp. \( v^h \)) in the element \( T^e \). In the one-dimensional case, the last term of the sums above is given by \( \lambda u N_e v N_e \) and \( h v N_e \) respectively.

They are described globally in a direct way as we will see later. Therefore, each of the integrals above depends only on the nodal values of \( u^h \) and \( v^h \) (or only of \( v^h \)) relative to a node in \( T^e \). So we can describe them using matrices \( l \times l \) (or only \( l \times 1 \)) where \( l \) is the number of nodes per element: \( l = 3 \) in dimension \( N = 2 \), \( l = 2 \) in dimension \( N = 1 \). These are the elementary matrices. We gives the list below:

- **Elementary Stiffness Matrix**

\[ \int_{T^e} \sum_{i,j=1}^{N} a_{ij} \partial_i \mu^e \partial_j v^e \, dT^e = \begin{pmatrix} v^e_1 & \ldots & v^e_l \end{pmatrix} K^e \begin{pmatrix} u^e_1 \\ \vdots \\ u^e_l \end{pmatrix} \]  

(3)

- **Elementary Mass Matrix**

\[ \int_{T^e} a_0 \mu^e v^e \, dT^e = \begin{pmatrix} v^e_1 & \ldots & v^e_l \end{pmatrix} M^e \begin{pmatrix} u^e_1 \\ \vdots \\ u^e_l \end{pmatrix} \]  

(4)

- **Elementary Volume Loading Matrix**

\[ \int_{T^e} f v^e \, dT^e = \begin{pmatrix} v^e_1 & \ldots & v^e_l \end{pmatrix} F^e \]  

(5)

- **Elementary Stiffness Matrix Concentrated on the Edge**

\[ \int_{(G^e)} \lambda \mu^{(a)} v^{(a)} \, d \left( G^a \right) = \begin{pmatrix} v^{(a)}_1 & \ldots & v^{(a)}_l \end{pmatrix} K^{(a)} \begin{pmatrix} u^{(a)}_1 \\ \vdots \\ u^{(a)}_l \end{pmatrix} \]
where \((G^a)\)' is a face (edge in two dimensions) on \(\Gamma_N\) and \(l'\) is the number of nodal values on \((G^a)'\), \(u^{(a)}\) and \(v^{(a)}\) are the respective restrictions of \(u^h\) and \(v^h\) on \((G^a)'\).

- Elementary Loading Matrix concentrated on the Edge

\[
\int_{(G^a)'} h v^e \, d (G^a)' = \begin{pmatrix} v_1^{(a)} \\ \vdots \\ v_{l'}^{(a)} \end{pmatrix} F_e^{(a)}
\]

Matrices and forces concentrated in dimension \(N = 1\) are introduced at the initialization or the end of the assembling process.

4.5. Elementary Matrices in One Dimension

We have here \(T_e = [a^e_1, a^e_2] \). Taking a mesh compatible with the possible discontinuities of the data, that is, for which data discontinuities are always at the interfaces between two elements, we can assume that

\[
a^e = a|_{T_e}, a^e_0 = a_0|_{T_e}, f^e = f|_{T_e} \in C^\infty([a^e_1, a^e_2])
\]

For example, for the function shown in the Figure 48

![Figure 48. Assumptions for data discontinuity.](image)

we have

\[
a^{(1)} = L/4 \\
a^{(2)} = L/4 \\
a^{(3)} = x \\
a^{(4)} = x
\]

Then, the particularization of the general situation in the case of the dimension \(N = 1\) gives

1. Elementary stiffness matrix

\[
(u^e)' = \frac{u^e_2 - u^e_1}{|T_e|}, \quad (v^e)' = \frac{v^e_2 - v^e_1}{|T_e|}
\]
where \(|T^e|\) is the element length. Then we have

\[
\begin{pmatrix}
v_1^e \\ v_2^e
\end{pmatrix}K^e\begin{pmatrix}u_1^e \\ u_2^e\end{pmatrix} = \int_{a_i^e}^{a_2^e} a^e (u^e)' (v^e)' \, dx
\]

\[
= \frac{\bar{a}^e}{|T^e|} \left( u_2^e - u_1^e \right) \left( v_2^e - v_1^e \right)
\]

where \(\bar{a}^e\) is the mean value of the function \(a^e\) on the element \(T^e\)

\[
\bar{a}^e := \frac{1}{|T^e|} \int_{a_i^e}^{a_2^e} a^e \, dT^e
\]

that we can approach by the trapezes formula or the middle point one

\[
\bar{a}^e \approx \frac{a_2^e - a_1^e}{2} (a_1^e + a_2^e)
\]

2. Condensed mass matrix

Generally, it is sufficient to calculate the integral (4) in an approximate way by the method of trapezes

\[
\begin{pmatrix}
v_1^e \\ v_2^e
\end{pmatrix}M^e\begin{pmatrix}u_1^e \\ u_2^e\end{pmatrix} = \int_{a_i^e}^{a_2^e} a^e_0 u^e v^e \, dx
\]

\[
= \frac{|T^e|}{2} \left( (a_0^e)_1 u_1^e v_1^e + (a_0^e)_2 u_2^e v_2^e \right)
\]

with \((a_0^e)_j = a_0^e \left( a_j^e \right), j = 1, 2\). This gives a diagonal matrix

\[
M^e = \frac{|T^e|}{2} \begin{pmatrix} (a_0^e)_1 & 0 \\ 0 & (a_0^e)_2 \end{pmatrix}
\]

3. Elementary loading Matrix

Here again we can use the trapeze formula to approximate the integral (5)

\[
\begin{pmatrix}
v_1^e \\ v_2^e
\end{pmatrix}F^e = \int_{a_i^e}^{a_2^e} a^e_0 f^e v^e \, dx
\]

\[
\approx \frac{|T^e|}{2} \left( f_1^e v_1^e + f_2^e v_2^e \right)
\]

with \(f_j^e = f^e \left( a_j^e \right), j = 1, 2\). This gives the elementary matrix

\[
F^e = \frac{|T^e|}{2} \begin{pmatrix} f_1^e \\ f_2^e \end{pmatrix}
\]

4. Matrices concentrated on the edge
It is here matrices with a single non-zero coefficient which are given directly from

\[ \lambda u_{N_i}^h v_{N_i}^h \text{ and } h r_{N_i}^h \]

by

\[ (K_{c}^j)_{N_i N_i} = \lambda \text{ and } (F_{c}^j)_{N_i} = h \]

4.6. Elementary Matrices in Dimension Two

We note in the same way as in dimension on the expressions of the restrictions \( a_{ij} = a_{ij}|_{T^c} \), \( A^c \) the \( 2 \times 2 \) matrix whose coefficients are the \( a_{ij}^c, a_0^c = a_0|_{T^c} \) and \( f^c = f|_{T^c} \) that we can suppose in \( C^\infty (T^c) \) if the mesh is compatible with the discontinuities of the data.

We can then give the expression of the elementary matrices.

- Elementary stiffness matrix

The following method of determining the elementary matrix is in fact general. She can be used for the determination of any elementary matrix and in the framework of any finite element method.

We have by definition

\[
\begin{pmatrix}
 v_1^c & v_2^c & v_3^c
\end{pmatrix}
K^c
\begin{pmatrix}
 u_1^c \\
 u_2^c \\
 u_3^c
\end{pmatrix} = \int_{T^c} \left( \frac{\partial}{\partial x^1} v^c \frac{\partial}{\partial x^2} v^c \right) A^c \left( \frac{\partial}{\partial x^1} u^c \frac{\partial}{\partial x^2} u^c \right) dT^c
\]

Then if we denote by \( \overline{A}^c \) the mean value of \( A^c \) on \( T^c \)

\[
\begin{pmatrix}
 v_1^c & v_2^c & v_3^c
\end{pmatrix}
K^c
\begin{pmatrix}
 u_1^c \\
 u_2^c \\
 u_3^c
\end{pmatrix} = \left( \frac{\partial}{\partial x^1} v^c \frac{\partial}{\partial x^2} v^c \right) (\lvert T^c \rangle \langle \overline{A}^c \rangle) \left( \frac{\partial}{\partial x^1} u^c \frac{\partial}{\partial x^2} u^c \right)
\]

We now express \( u^c \) and \( v^c \) using their polynomial coefficients and a matrix writing

\[
u^c (x) = a_0^c + a_1^c x_1 + a_2^c x_2 = \begin{pmatrix} 1 & x_1 & x_2 \end{pmatrix} \begin{pmatrix} a_0^c \\ a_1^c \\ a_2^c \end{pmatrix}
\]

\[
u^c (x) = \beta_0^c + \beta_1^c x_1 + \beta_2^c x_2 = \begin{pmatrix} 1 & x_1 & x_2 \end{pmatrix} \begin{pmatrix} \beta_0^c \\ \beta_1^c \\ \beta_2^c \end{pmatrix}
\]

We can then calculate

\[
\begin{pmatrix}
 \frac{\partial}{\partial x^1} u^c \\
 \frac{\partial}{\partial x^2} u^c
\end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_0^c \\ a_1^c \\ a_2^c \end{pmatrix} = B^c \begin{pmatrix} a_0^c \\ a_1^c \\ a_2^c \end{pmatrix}
\]

\[
\begin{pmatrix}
 \frac{\partial}{\partial x^1} v^c \\
 \frac{\partial}{\partial x^2} v^c
\end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \beta_0^c \\ \beta_1^c \\ \beta_2^c \end{pmatrix} = B^c \begin{pmatrix} \beta_0^c \\ \beta_1^c \\ \beta_2^c \end{pmatrix}
\]
We then use the relation which makes it possible to connect the polynomial coefficients with the nodal values

\[
\begin{align*}
\alpha_0^e + \alpha_1^e a_{11} + \alpha_2^e a_{21} &= u_1^e \\
\alpha_0^e + \alpha_1^e a_{12} + \alpha_2^e a_{22} &= u_2^e \\
\alpha_0^e + \alpha_1^e a_{13} + \alpha_2^e a_{23} &= u_3^e
\end{align*}
\]

(6)

which can be written in matrix form

\[
P^e \begin{bmatrix} \alpha^e \end{bmatrix} = \begin{bmatrix} u^e \end{bmatrix}
\]

where

\[
P^e = \begin{bmatrix} 1 & a_{11} & a_{21} \\ 1 & a_{12} & a_{22} \\ 1 & a_{13} & a_{23} \end{bmatrix}, \quad \begin{bmatrix} \alpha^e \end{bmatrix} = \begin{bmatrix} \alpha_0^e \\ \alpha_1^e \\ \alpha_2^e \end{bmatrix}, \quad \begin{bmatrix} u^e \end{bmatrix} = \begin{bmatrix} u_0^e \\ u_1^e \\ u_2^e \end{bmatrix}
\]

and likewise

\[
P^e \begin{bmatrix} \beta^e \end{bmatrix} = \begin{bmatrix} v^e \end{bmatrix}, \quad \begin{bmatrix} \beta^e \end{bmatrix} = \begin{bmatrix} \beta_0^e \\ \beta_1^e \\ \beta_2^e \end{bmatrix}, \quad \begin{bmatrix} v^e \end{bmatrix} = \begin{bmatrix} v_0^e \\ v_1^e \\ v_2^e \end{bmatrix}
\]

To express the polynomial coefficients using the nodal values, we calculate the inverse

\[
H^e = (P^e)^{-1}
\]

\[
[\alpha^e] = H^e [u^e], \quad [\beta^e] = H^e [v^e]
\]

This inversion is in fact the determination of the basic functions on the element \(T^e\) of the finite element. As can be seen, this determination can be made (for this element and for any finite element) in a completely implicit way by programming.

Using all previous relationships, we get

\[
[v^e]^T K^e [u^e] = [v^e]^T (H^e)^T (B^e)^T (|T^e| \overrightarrow{A}) B^e H^e [u^e]
\]

Hence we obtain the elementary stiffness matrix by a simple calculation

\[
K^e := (H^e)^T (B^e)^T (|T^e| \overrightarrow{A}) B^e H^e
\]

The mean value of \(A^e\) can be obtained by an approximate integration formula as in dimension one

\[
\overrightarrow{A} \approx \frac{1}{3} (A_1^e + A_2^e + A_3^e) \approx A_0^e
\]

where \(A_j^e\) is the value of \(A^e\) at vertex \(a_j^e, j = 1, 2, 3\) and \(A_0^e\) value of \(A^e\) at the center of gravity of \(T^e\):

\[
\overrightarrow{A} \approx \frac{1}{3} \left( a_{1j}^e + a_{2j}^e + a_{3j}^e \right)
\]

- **Stiffness Matrix concentrated on the edge**

We usually have the data provided by the mesh (otherwise we can build by programming) from the list of edges on the border, each with a reference number that allows you, among other things, to know if the edge is contained in \(\Gamma_N\) or does not cover \(\Gamma_N\). We can therefore assume that we have the following data

\(N_e\): number of edges on the border

And for each edge \([a] = 1, ..., N_e\).
$n_{a1}$: number of the first vertex $a_1^{[a]}$

$n_{a2}$: number of the second vertex $a_2^{[a]}$

thus, We can traverse the edges and calculate the elementary matrix giving the contribution to stiffness concentrated on $\Gamma_N$

$\left(\begin{array}{c} \nu_1^{[a]} \\ \nu_2^{[a]} \end{array}\right) K_e^{[a]} \left(\begin{array}{c} u_1^{[a]} \\ u_2^{[a]} \end{array}\right) = \int_{(G^{[a]})'} \lambda^{[a]} u^{[a]} \nu^{[a]} d \left(\left(\begin{array}{c} \nu^{[a]} \end{array}\right)\right)'$

where $\lambda^{[a]} = \lambda|_{(G^{[a]})'}$. If we approach this integral by the trapeze formula, we get

$K_e^{[a]} = \frac{|(G^{[a]})'|}{2} \left(\begin{array}{cc} \lambda_1^{[a]} & 0 \\ 0 & \lambda_2^{[a]} \end{array}\right)$

where $|(G^{[a]})'|$ is the edge length and $\lambda_j^{[a]} = \lambda^{[a]} \left(\begin{array}{c} a_{e0}^{[a]} j \end{array}\right)$, $j = 1, 2$.

- **Condensed mass matrix.**

The condensed mass matrix without using the matrix $H$ and by using the integration formula approached from the vertices

$\int_{E^{[e]}} a_0^{[e]} u^{[e]} v^{[e]} dT^{[e]} = \frac{|T^{[e]}|}{3} ((a_0^{[e]})_1 u_1^{[e]} v_1^{[e]} + (a_0^{[e]})_2 u_2^{[e]} v_2^{[e]} + (a_0^{[e]})_3 u_3^{[e]} v_3^{[e]})$

where $(a_0^{[e]})_j := a_0^{[e]} \left(\begin{array}{c} a_j^{[e]} \end{array}\right)$, $j = 1, 2, 3$, are values of $a_0^{[e]}$ on the vertices of $T^{[e]}$, we have then:

$M^{[e]} := \frac{|T^{[e]}|}{3} \left(\begin{array}{ccc} (a_0^{[e]})_1 & 0 & 0 \\ 0 & (a_0^{[e]})_2 & 0 \\ 0 & 0 & (a_0^{[e]})_3 \end{array}\right)$

The fact that the conditional mass matrix is $\lambda E \lambda E$-diagonal will be decisive in some solving problems in dynamics, that is to say, depending also on the time $t$.

- **Elementary loading Matrix**

We have the same as for the elementary loading Matrix of volume in dimension one

$F^{[e]} := \frac{|T^{[e]}|}{3} \left(\begin{array}{c} f_1^{[e]} \\ f_2^{[e]} \\ f_3^{[e]} \end{array}\right)$

with $f_j^{[e]} = f^{[e]} \left(\begin{array}{c} a_j^{[e]} \end{array}\right)$, $j = 1, 2, 3$.

The elementary loading matrix concentrated on the edge is given in the same way by

$F_c^{[e]} := \frac{|(G^{[e]})'|}{2} \left(\begin{array}{c} h_1^{[e]} \\ h_2^{[e]} \end{array}\right)$

with $h_j^{[e]} = h^{[e]} \left(\begin{array}{c} a_j^{[e]} \end{array}\right)$, $j = 1, 2$. 


Assembling. The total stiffness matrix is defined by

\[
\left( v_h^1 \ldots v_h^{N_s} \right) K^t \left( \begin{array}{c} u_h^1 \\ \vdots \\ u_h^{N_s} \end{array} \right) = \int_{\Omega} \left( \frac{\partial_1 v_h}{\partial_2 v_h} \right) A \left( \frac{\partial_1 u_h}{\partial_2 u_h} \right) d\Omega, \quad u_h^i, v_h^i \in X_h
\]

Using the decomposition of the integral on \( \Omega \)

\[
\int_{\Omega} \{ \ldots \} d\Omega = \sum_{T_e \in \mathcal{T}_h} \int_{T_e} \{ \ldots \} dT_e
\]

and the definition (3) of the elementary stiffness matrix, we obtain the following relation between the total stiffness matrix and the elementary stiffness matrices

\[
\left( v_h^1 \ldots v_h^{N_s} \right) K^t \left( \begin{array}{c} u_h^1 \\ \vdots \\ u_h^{N_s} \end{array} \right) = \sum_{T_e \in \mathcal{T}_h} \left( v_e^1 \ldots v_e^l \right) K^e \left( \begin{array}{c} u_e^1 \\ \vdots \\ u_e^l \end{array} \right)
\]

We then identify the coefficient of \( u_h^j v_h^i \) in both members of this relation to get

\[
[K^t]_{ij} = \sum_{\{ T_e \in \mathcal{T}_h, n_{\epsilon}^{j,i} = i, n_{\epsilon}^{j,i} = j \}} [K^e]_{i,j}
\]

Consequences:

This last formula summarizes the assembling process. It has several consequences.

1. A coefficient \([K^t]_{ij}\) is null if \( a_h^j \) and \( a_h^i \) are not the vertices of a same element. In other words, degrees of freedom \( u_h^j \) and \( v_h^i \) are not coupled (i.e., \([K^t]_{ij} \neq 0\) or the equation relating to \( v_h^i \) depends on the unknown \( u_h^j \)) only if \( u_h^j \) and \( v_h^i \) belong to the same element.

2. To form \( K^t \), simply browse the elements by adding successively the contribution of each elementary matrix \([K^e]_{i,j}\) to the total matrix \([K^t]_{ij}\) with \( i = n_{\epsilon}^{i,j} \), \( j = n_{\epsilon}^{i,j} \).

5. Time Dependent Problems

In this section we limit ourselves to introducing the principles underlying the processes of numerical resolution of the problems modeling phenomena evolving during the time. This type of problems is called evolution problems.

5.1. Evolution Problems Models

We give below two typical examples of time-dependent problems. Most of the situations encountered in engineering have the characteristics of one or the other of these two examples.

5.1.1. Model Problems

We assume for simplification that \( g = 0 \) and \( h = 0 \) and we make the fundamental assumption that the matrix \( A(x) \) is symmetric (and thus positive definite) for almost all \( x \in \Omega \). It is further assumed that the data \( f \) depends not only on \( x \) but also on \( t \in [0, T] \) (i.e., one gives \( (x, y) \rightarrow f(x, y) \) defined on \( \Omega \times [0, T] \)). We give also a function \( \rho \in L^\infty(\Omega) \) verifying

\[
0 < \rho_m \leq \rho(x) \leq \rho_M, \text{ a.e. on } \Omega
\]

The two typical examples of time-dependent problems are put in this way.
1. Parabolic problem

\[
\begin{aligned}
\rho(x) \partial_t u(x,t) - \sum_{i=1}^{N} \sum_{j=1}^{N} \partial_{x_i} a_{ij}(x) \partial_{x_j} u(x,t) \\
+ a_0(x) u(x,t) = f(x,t), (x,t) \in \Omega \times ]0, T[ \\
u(x,t) = 0, (x,t) \in \Gamma_D \times ]0, T[ \\
\sum_{i=1}^{N} \sum_{j=1}^{N} n_i(x) \partial_{x_i} a_{ij}(x) \partial_{x_j} u(x,t) = 0, (x,t) \in \Omega \times ]0, T[ \\
u(x,0) = u^{(0)}(x), x \in \Omega
\end{aligned}
\]  

(7)

where \( u^{(0)} \) is the initial data of the problem.

2. Hyperbolic problem

\[
\begin{aligned}
\rho(x) \partial^2_t u(x,t) - \sum_{i=1}^{N} \sum_{j=1}^{N} \partial_{x_i} a_{ij}(x) \partial_{x_j} u(x,t) \\
+ a_0(x) u(x,t) = f(x,t), (x,t) \in \Omega \times ]0, T[ \\
u(x,t) = 0, (x,t) \in \Gamma_D \times ]0, T[ \\
\sum_{i=1}^{N} \sum_{j=1}^{N} n_i(x) \partial_{x_i} a_{ij}(x) \partial_{x_j} u(x,t) = 0, (x,t) \in \Omega \times ]0, T[ \\
u(x,0) = u^{(0)}(x), \partial_t u(x,0) = u^{(1)}(x), x \in \Omega
\end{aligned}
\]  

(8)

where now the problem requires two initial data \( u^{(0)} \) and \( u^{(1)} \).

5.1.2. Variational Formulation

By multiplying the PDE by a test function \( v \), independent of \( t \), in

\[
V := \{ v \in H^1(\Omega) ; v|_{\Gamma_D} = 0 \}
\]

we put problems (7) and (8) under the form

\[
\begin{aligned}
n \rightarrow u(., t) \text{ defined on } [0, T] \text{ with values on } V \\
\partial^n_t b(u(., t), v) + a(u(., t), v) = \int_{\Omega} f(x,t) v(t) dx, \forall v \in V \\
\partial_l^j u(., t) = u^{(l)}, 0 \leq l \leq m - 1
\end{aligned}
\]  

(9)
with \( m = 1 \) for the parabolic problem and \( m = 2 \) for the hyperbolic problem, and \( b \) is given by

\[
b(w, v) := \int_{\Omega} \varrho wvdx
\]

5.2. Semi-Discretization in Space

5.2.1. Finite Element Method

We use a mesh \( \tau^h \) of \( \Omega \):

- In segments if \( \Omega \) is a segment,
- In triangles if \( \Omega \) is a polygonal domain of the plane,
- In tetrahedrons if \( \Omega \) is a polyhedral domain of space.

Consistent with data discontinuities of given data and \( \Gamma_D \). We then use the \( P_1 \)-continuous finite element method to approach respectively \( H^1(\Omega) \) and \( V \) by \( X^h \) and \( V^h \).

The test functions do not depend on time \( t \). They are approached by \( \psi^h \) functions completely determined using their nodal values \( v_i \) where \( N_s \) is the number of vertices of the mesh (we omit the exponent \( h \) to lighten the notation). We note by \( [v^\sharp] \) the column vector formed by the nodal values of \( v^h \)

\[
[v^\sharp] := \begin{pmatrix} v_1 \\ \vdots \\ v_N \\ v_{N+1} \\ \vdots \\ v_{N_s} \end{pmatrix}
\]

If we suppose that the numberation of the vertices is such that \( a_i \in \Gamma_D \) is numbered by \( i \) varying from \( N + 1 \) to \( N_s \), the vector of nodal values not fixed at zeros of \( \psi^h \in V^h \) is given by

\[
[v] := \begin{pmatrix} v_1 \\ \vdots \\ v_N \end{pmatrix}
\]

The approximation of the unknown function \( u \) is carried out as follows: for all \( t \in [0, T] \), \( u(\cdot, t) \) is approximated by a function \( u^h(\cdot, t) \in V^h \). In particular, function \( u^h \) is completely determined by \( N_s \) functions \( t \in [0, T] \rightarrow u^h_j(t) \) giving the nodal values of \( u^h(\cdot, t) \). We also note the function of \( t \) defined in \( R^{N_s} \)

\[
[u^\sharp](t) := \begin{pmatrix} u_1(t) \\ \vdots \\ u_N(t) \\ u_{N+1}(t) \\ \vdots \\ u_{N_s}(t) \end{pmatrix}
\]

Similarly, the membership of \( u^h(\cdot, t) \) to \( V^h \) is characterized by the conditions

\[
u_{N+1}(t) = \ldots = u_{N_s}(t) = 0
\]
so that the only unknown functions to be determined are the components of the vector

\[
[u] (t) := \begin{pmatrix}
u_1 (t) \\
\vdots \\
u_N (t)
\end{pmatrix}
\]

Assuming that the initial data \( u^{(l)} \ (l \leq m - 1) \) are sufficiently regular (actually in \( C^0 (\Omega) \)), we can approach them from their values at nodes \( u^{(l)} (a_i) \) by a vector \([u^\ast]^{(l)}\) that we suppose to check \( u^{(l)} = 0 \ (i = N + 1, \ldots, N_s) \): condition compatibility on the initial data. These nodal values determine functions \( (u^{(l)})^h \in V_h \).

To approach \( u \), we are naturally led to solve the variational problem

\[
\begin{cases}
t \to u^h (., t) \text{ defined on } [0, T] \text{ with values on } V^h, \\
\partial_t [u^\ast] b (u^h (., t), v^h) + a (u^h (., t), v^h) = \int_\Omega f (x, t) v^h (t) \, dx, \quad \forall v^h \in V^h, \\
\partial_t [u^h (., t) = (u^{(l)})^h, \quad 0 \leq l \leq m - 1
\end{cases}
\]

whose unknowns are \( N \) unknown functions \( t \in [0, T] \to u_j (t), \ (j = 1, \ldots, N) \).

5.2.2. Reduction to a Differential System

In addition to the total stiffness matrix defined by the relation

\[
\begin{pmatrix}
\nu^\ast
\end{pmatrix}^T K^\ast \begin{pmatrix}
\nu^\ast
\end{pmatrix} = a \begin{pmatrix}
w^h, v^h
\end{pmatrix}, \quad w^h, v^h \in X^h
\]

we form the total mass matrix \( M^\ast \) defined by

\[
\begin{pmatrix}
\nu^\ast
\end{pmatrix}^T M^\ast \begin{pmatrix}
\nu^\ast
\end{pmatrix} = b \begin{pmatrix}
w^h, v^h
\end{pmatrix}, \quad w^h, v^h \in X^h
\]

where \( \begin{pmatrix}
\nu^\ast
\end{pmatrix} \) is as above with \( \begin{pmatrix}
\nu^\ast
\end{pmatrix} \) the column vector formed by nodal values \( \hat{a} \in \mathbb{R} \) of \( w^h \). We also define for all \( t \in [0, T] \) the total loading vector whose components depend on time \( t \) are given by the relation

\[
\begin{pmatrix}
\nu^\ast
\end{pmatrix} L^\ast (t) = \int_\Omega f (x, t) v^h (x) \, d\Omega, \quad v^h \in V^h
\]

The problem (11) can therefore be written using these notations

\[
\begin{cases}
\begin{pmatrix}
\nu^\ast
\end{pmatrix}^T M^\ast \begin{pmatrix}
\partial_t [u^\ast]
\end{pmatrix} (t) + \begin{pmatrix}
\nu^\ast
\end{pmatrix}^T K \begin{pmatrix}
[u^\ast]
\end{pmatrix} (t) = \begin{pmatrix}
\nu^\ast
\end{pmatrix}^T L^\ast (t), \quad t \in [0, T], \\
\begin{pmatrix}
\partial_t [u^\ast]
\end{pmatrix} (0) = \begin{pmatrix}
[u^\ast]
\end{pmatrix}^{(l)}, \quad l \leq m - 1
\end{cases}
\]

where \( \begin{pmatrix}
\partial_t [u^\ast]
\end{pmatrix} (t) \) is a vector with components \( \partial_t u_j (t) \), \( u_j (t) \) satisfies conditions (10) and \( v_i \) similar conditions

\[
v_{N+1} = \ldots = v_{N_s} = 0
\]

(13)
Exactly as in the case of problems not dependent on \( t \), removing the rows and columns of the matrices stiffness, mass and loading corresponding to degrees of freedom blocked at zero, the previous matrix variational system is rewritten

\[
\begin{cases}
M \left[ \partial^l u \right] (t) + K \left[ u \right] (t) = L (t), & t \in [0, T], \\
\left[ \partial^l u \right] (0) = \left[ u^{(l)} \right], & l \leq m - 1
\end{cases}
\]  

In this system, \( K, M \) and \( L (t) \) respectively denote the matrices: stiffness, mass and final loading.

5.2.3. Matrix Properties of the Differential System

We have assumed that the matrix \( A \) of coefficients of the PDE was symmetrical. This leads to the fact that the bilinear form \( a (., .) \) is symmetrical and hence the matrix stiffness \( K \) is symmetrical.

\[ K^T = K \]

The final matrix \( M \) is also symmetrical since the form \( b (., .) \) satisfies

\[ b (w, v) = \int_{\Omega} \rho w v d\Omega = \int_{\Omega} \rho v w d\Omega = b (v, w) \]

then we have the fundamental result for the resolution.

**Proposition 3.** Whatever its exact or condensed determination, the matrix mass \( M \) is positive definite.

**Proof.** We have first, for \( \left[ v^T \right] \) satisfying the conditions (13)

\[ \left[ v^T \right] M \left[ v \right] = \left[ v^T \right]^T M^T \left[ v^T \right] \]

then, we have

\[ \left[ v^T \right] M \left[ v \right] = \begin{cases}
\int_{\Omega} \rho v^2 d\Omega \geq 0, & \text{if } M \text{ is exact} \\
\sum_{e=1}^{N_e} |T_e| \sum_{e=1}^{N_e} \rho^h_i (v^e_i)^2 \geq 0, & \text{if } M \text{ is condensed}
\end{cases} \]

where \( \rho^h_i, v^e_i \) are the respective nodal values of \( \rho^e = \rho|_{T_e} \) and \( v^h = v|_{T_e} \) at the vertices of the element \( T_e \) and \( N^e \) vertices number of \( T_e \). The condition \( \rho \geq \rho_0 > 0 \) gives then directly that \( v^T M v > 0 \) as soon as \( v^h \neq 0 \).

**Remark 1.** The condensed mass matrix is a diagonal matrix

\[
M = \begin{pmatrix}
M_1 & 0 & \cdots & 0 \\
0 & \ddots & \cdots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
0 & \cdots & \cdots & M_N
\end{pmatrix}
\]

with \( M_i > 0 \ (i = 1, ..., N) \).

If \( \Gamma_D \) is empty, i.e., if we have a Neumann condition on the whole border \( \Gamma \), the form \( a (., .) \) is generally positive but undefined. In this case the matrix \( K \) is symmetrical and positive but not defined. This is assumed in all the following.
5.3. Time Diagrams

One is thus brought back to solve the differential system 14 which one can carry out using a numerical method of determining the solution at the points of a grid of the interval \([0, T]\)

\[
t_0 < t_1 < \ldots < t_m < t_{m+1} < t_M = T
\]

or at the points

\[
t_0 = 0, t_{m+1} = t_m + \Delta t_m \quad (m = 0, \ldots, M - 1)
\]

The system (14) is a differential system whose Lipschitz constant is in \(1/ h^2\). It is then stiff, it can not be solved by any scheme. We briefly describe below a family of diagrams for parabolic problems, the \(\theta\)-schemes, and a diagram for the hyperbolic equations, the Neumark scheme.

5.3.1. \(\theta\)-Scheme for the Parabolic Equations

These are one-step diagrams integration of the differential system (14). Recall that for the construction of this type of scheme we have to use a numerical integration formula. Indeed, if \([u]\) is differential solution to the system

\[
\begin{cases}
M [\dot{u}] (t) + K [u] (t) = L (t), & t \in [0, T], \\
[u] (0) = [u(0)]
\end{cases}
\] (15)

where \([\dot{u}] (t)\) designates the vector whose components are derived \(\dot{u}_j (t)\) from components of \([u]\), we have

\[
\int_{t_w}^{t_{w+\Delta t_w}} M [\dot{u}] (t) \, dt = \int_{t_{w+\Delta t_w}}^{t_{w+1}} (L (t) - K [u] (t)) \, dt
\]

or

\[
M [u] (t_{m+1}) - M [u] (t_{m}) = \int_{t_w}^{t_{w+1}} (L (t) - K [u] (t)) \, dt
\]

We then approach the integral of the second member by the numerical integration formula

\[
\int_{t_w}^{t_{w+1}} \varphi (t) \, dt \approx \Delta T (\varphi (t_{m}) + \theta \varphi (t_{m+1})) , \quad 0 \leq \theta \leq 1
\]

Then we get the scheme

\[
\begin{cases}
\frac{1}{\Delta t_m} (M [u]_{m+1} - M [u]_m) = (1 - \theta) ([L]_m - K [u]_m) + \theta ([L]_{m+1} - K [u]_{m+1}), \\
[u]_0 = [u(0)]
\end{cases}
\]

where

\[
[u]_m = \begin{pmatrix}
U_{1,m} \\
\vdots \\
U_{N,m}
\end{pmatrix}
\]
is the vector whose component $u_{jm}$ gives the approximation of the nodal value $u_j(t)$ at moment $t_m$
and with similar notation for $[L]_m$. This scheme is rewritten by putting all the known terms after $m$ no
integration in the second member

$$\begin{cases}
(\frac{1}{\Delta t_m} M + \theta K) [u]_{m+1} = (1 - \theta) [L]_m + \theta [L]_{m+1} + \left(\frac{1}{\Delta t_m} M - (1 - \theta) K\right) [u]_m \\
[u]_0 = [u^{(0)}]
\end{cases}$$

The schemes for $\theta < 1/2$ are conditionally stable. This makes them difficult to use, or even
practically useless. In fact, the only interesting scheme of this type is the explicit Euler scheme obtained
for $\theta = 0$

$$\begin{cases}
\frac{1}{\Delta t_m} M [u]_{m+1} = [L]_m + \left(\frac{1}{\Delta t_m} M - K\right) [u]_m \\
[u]_0 = [u^{(0)}]
\end{cases}$$

When the mass matrix $M$ is condensed, the calculation of $[u]_{m+1}$ is done by resolution of a
diagonal system.

The schemes for $\theta > 1/2$, are highly stable schemes. The scheme with the best stability property
is the implicit Euler scheme obtained for $\theta = 1$

$$\begin{cases}
\left(\frac{1}{\Delta t_m} M + \theta K\right) [u]_{m+1} = [L]_{m+1} + \frac{1}{\Delta t_m} M [u]_m \\
[u]_0 = [u^{(0)}]
\end{cases}$$

For $\theta = 1/2$, we obtain the Crank-Nicholson scheme. It is a weakly stable. But it’s the only
one that’s a 2nd order scheme. Note that for any scheme with $\theta > 0$, we must solve at each step of
integration a finite element problem

$$Ax = b$$

with

$$A = \left(\frac{1}{\Delta t_m} M + \theta K\right)$$

$$x = [u]_{m+1}$$

$$b = (1 - \theta) [L]_m + \theta [L]_{m+1} + \left(\frac{1}{\Delta t_m} M - (1 - \theta) K\right) [u]_m$$

It is clear that the direct methods, if the system size is not too large, are most indicated.

By decomposition of $A$ using the Gauss algorithm Under the form

$$A = LU$$

With each step of integration, one is thus brought to solve two triangular systems.
5.3.2. Neumark Scheme

It is now a question of solving the order 2 system in time

\[
\begin{cases}
    M \ddot{u}(t) + K [u](t) = L(t), \quad t \in [0,T], \\
    [u](0) = [u(0)], \quad \dot{[u]}(0) = [u(1)]
\end{cases}
\]  

(16)

We limit ourselves to the most used Neumark scheme. It is used on a uniform grid \( \Delta t_m = \Delta t, \ i = 0, \ldots, M - 1 \)

It is obtained by the following two approximations.

1. Order two approximation of the second derivative by a derivation in the sense of centered finite differences

\[
\begin{align*}
    [\ddot{u}](t_m) & \approx \frac{[u](t_{m+1}) - 2[u](t_m) + [u](t_{m-1})}{\Delta t^2} \\
    M [\ddot{u}](t_{m+1}) - 2 [u](t_m) + [u](t_{m-1}) & \approx L(t_m) - K [u](t_m)
\end{align*}
\]

2. Order two approximation for the second member to improve stability properties

\[
L(t_m) - K [u](t_m) \approx \frac{1}{4} ((L(t_{m-1}) - K [u](t_{m-1})) + 2 (L(t_m) - K [u](t_m)) + (L(t_{m+1}) - K [u](t_{m+1})))
\]

Then we get the scheme

\[
\left( \frac{1}{\Delta t^2} M + \frac{1}{4} K \right) [u]_{m+1} = b_{m+1}
\]

with

\[
b_{m+1} = \frac{1}{4} ([L]_{m-1} + 2 [L]_m + [L]_{m+1}) + \frac{1}{\Delta t^2} M (2 [u]_m - [u]_{m-1}) - \frac{1}{4} K (2 [u]_m - [u]_{m-1})
\]

We need two initial data: \([u]_0\) and \([u]_1\), we have

\[
[u]_0 = [u(0)]
\]

let calculate \([u]_1\) using Taylor development of order 2

\[
[u]_1 \approx [u](\Delta t) \approx [u](0) + \Delta t \dot{[u]}(0) + \left( \frac{\Delta t^2}{2} \right) [\ddot{u}](0)
\]

we have

\[
\dot{[u]}(0) = [u(1)]
\]

By going back to (16), we can calculate

\[
[u](0) = M^{-1} ([L]_0 - K [u]_0)
\]
5.4. Modal Analysis

5.4.1. Resolution of a Differential System by Diagonalization

Recall how to solve the differential system

\[
\begin{aligned}
\frac{d^m}{dt^m}w(t) + Aw(t) &= f(t), \\
\frac{d^l}{dt^l}w(0) &= v^{(l)}, \quad 0 \leq l \leq m - 1
\end{aligned}
\]  

by diagonalization. Suppose that \( A \) is diagonalizable, i.e., that there exists a matrix invertible \( V \) and a diagonal matrix \( \Lambda := \text{diag} (\lambda_1, ..., \lambda_N) \) such that

\[
V^{-1}AV = \Lambda
\]  

This leads to assuming that there exists a basis (of \( \mathbb{R}^N \) or of \( \mathbb{C}^N \)) depending on whether at least one eigenvalues be complex or not) which is actually the family of vectors \( \{V^*_j\}_{j=1}^N \) formed of column vectors of \( V \) which are the eigenvectors corresponding to eigenvalues \( \{\lambda_i\}_{i=1}^N \). Multiply on the left the system (18) by \( V^{-1} \) and make the changed function.

\[
v(t) = V^{-1}w(t)
\]

We are brought back to the system

\[
\begin{aligned}
\frac{d^m}{dt^m}v(t) + \Lambda v(t) &= F(t), \\
\frac{d^l}{dt^l}v(0) &= v^{(l)}, \quad 0 \leq l \leq m - 1
\end{aligned}
\]  

with \( F(t) = V^{-1}f(t) \), \( v^{(l)} = V^{-1}w^{(l)} \), \( 0 \leq l \leq m - 1 \).

This system is decoupled: it is equivalent to the resolution of \( N \) decoupled differential equations

\[
\begin{aligned}
\frac{d^m}{dt^m}v_j(t) + \lambda_j v_j(t) &= F_j(t), \\
\frac{d^l}{dt^l}v_j(0) &= v_j^{(l)}, \quad 0 \leq l \leq m - 1
\end{aligned}
\]

It is solved directly. We assume to simplify the discussion and because that will be checked for the case which interests us subsequently that the eigenvalues are real and positive. They are ordered as follows

\[
0 \leq \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_N
\]

1. System of order 1

\[
v_j(t) = v_j^{(0)} \exp(-\lambda_j t) + \int_0^t \exp(-\lambda_j (t-s))F_j(s) \, ds, \quad j = 1, ..., N
\]
2. System of order 2

\[ v_j(t) = v_j^{(0)} \cos(t \sqrt{\lambda_j}) + v_j^{(1)} \frac{\sin(\sqrt{\lambda_j})}{\sqrt{\lambda_j}} + \int_0^t \frac{\sin(\sqrt{\lambda_j} (t-s))}{\sqrt{\lambda_j}} F_j(s) \, ds, \quad j = 1, \ldots, N \]  

(23)

with

\[ v_j(t) = v_j^{(1)} t + \int_0^t (t-s) F_j(s) \, ds \]  

(24)

if \( \lambda_j = 0 \)

We come back to the solution \( w \) by the inverse transformation of (19)

\[ w(t) = Vv(t) \]

5.4.2. Problem with Generalized Eigenvalues

The system (14) is not in the form (18). Since \( M \) is symmetric definite positive, we can brought back to it. We decompose \( M \) by Cholesky’s algorithm

\[ M = R^T R \]

where \( R \) is upper triangular and invertible. Multiply the system (14) by \( (R^{-1})^T \) and by \( R \) the initial conditions. We write \( u(t) = [u] \) to simplify

\[
\begin{cases}
\frac{d^m}{dt^m} Ru(t) + (R^{-1})^T Ku(t) = (R^{-1})^T L(t), \\
\frac{d^l}{dt^l} Ru(0) = Ru^{(l)}, \quad 0 \leq l \leq m - 1
\end{cases}
\]

(25)

Then let’s

\[ w(t) := Ru(t) \]

and

\[ (R^{-1})^T L(t) = f(t) \]

We are thus brought back to the previous case with

\[ A := (R^{-1})^T K R^{-1} \]

However, since \( K \) is symmetrically positive, we have more precise results concerning its eigenvalues and its eigenvectors.

**Proposition 4.** There exists a matrix \( V \in \mathbb{R}^{N,N} \) orthogonal, i.e.,

\[ V^{-1} = V^T \]

such that

\[ V^T AV = \Lambda \]

with \( \Lambda = \text{diag}[\lambda_1, \ldots, \lambda_N] \) and

\[ 0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N \]  

(26)
**Proof.** Just make sure that $A$ is symmetrical and positive. We have

$$A^T = \left( \left( R^{-1} \right)^T K R^{-1} \right)^T = \left( R^{-1} \right)^T K^T R^{-1} = A$$

since $K^T = K$ and

$$v^T A v = v^T \left( R^{-1} \right)^T K^T R^{-1} v = z^T K z$$

with $z = R^{-1} v$, then we get

$$v^T A v = z^T K z \geq 0$$

\[\square\]

From this proposition we deduce the following theorem.

**Theorem 1.** There exists a matrix $U \in \mathbb{R}^{N,N}$ so that

$$U^T K T = \Lambda := \text{diag} \left( \lambda_1, ..., \lambda_N \right)$$

$$U^T M U = \mathbb{I}_N$$

with $\{ \lambda_j \}_{j=1}^N$ satisfying (26) and $\mathbb{I}_N$ is the identity matrix of $\mathbb{R}^{N,N}$.

**Proof.** We have using the previous proposition

$$V^T A V = \Lambda$$

Then let's

$$U = R^{-1} V$$

We deduce (27) Furthermore

$$U^T M U = U^T R^T R U = V^T V = \mathbb{I}_N$$

\[\square\]

**Remark 2.** It results from (27)

$$K U = \left( U^{-1} \right)^T \Lambda$$

and from (28)

$$M U = \left( U^{-1} \right)^T$$

$$K U = K U \Lambda$$

Multiply the two members of the previous equality by the $j$-th column vector of the matrix $\mathbb{I}_D$. The column vectors $U_{\ast,j}$ of the matrix $U$ are in fact a basis from vectors verifying

$$K U_{\ast,j} = \lambda_j K U_{\ast,j}$$

The couple $\left( \lambda_j, U_{\ast,j} \right)$ forms an eigenvalue and an eigenvector for the problem of generalized values

$$K w = \lambda M w$$
5.4.3. Modal Resolution

From matrices intervening in relations (27) and (28), we write the system (14) in the form

\[
\begin{align*}
\frac{dm}{dt}Mu(t) + U^T Ku(t) &= F(t), \\
\frac{dl}{dt}Uu(0) &= Uu(l), \quad 0 \leq l \leq m - 1
\end{align*}
\]

(29)

then, let

\[ u(t) = Uv(t) \] (30)

We get to a decoupled system

\[
\begin{align*}
\frac{dm}{dt}v(t) + \Lambda v(t) &= F(t), \\
\frac{dl}{dt}v(0) &= Uu(l), \quad 0 \leq l \leq m - 1
\end{align*}
\]

(31)

that we solve by formulas (22) or (23) or (24).

The relation (30) can also be written

\[ u(t) = \sum_{j=1}^{N} v_j(t) U_{s,j} \] (32)

each \( v_j(t) \) is solution of the differential equation

\[
\begin{align*}
\frac{dm}{dt}v_j(t) + \lambda_j v_j(t) &= F_j(t), \\
\frac{dl}{dt}v_j(0) &= v_j(l), \quad 0 \leq l \leq m - 1
\end{align*}
\]

(33)

Each term \( v_j(t) U_{s,j} \) is called a mode for the problem (14). It requires to be calculated, the determination of the eigenvalue \( \lambda_j \) and an eigenvector \( U_{s,j} \) normalized by the condition

\[ U_{s,j}^T MU_{s,j} = 1 \]

In fact, one does not need to calculate all the terms of the sum (32) to obtain a good approximation of \( u(t) \). It suffices to have the first eigenvalues and eigenvectors

\[ u(t) \approx \sum_{j=1}^{J} v_j(t) U_{s,j}, \quad J \ll N \]

To get an approximation of the solution with good accuracy.

6. Application

As an application for this work, experimental measurements of the temperature on the ground and in the soil are held. Such a data is in fact a useful tool to check the accuracy of the so-presented method using the error assessment in accordance with the weather temperatures data in Qassim province. It is worthy of note that Qassim province is one of the the most important region among
the thirteen administrative regions of Kingdom of Saudi Arabia, situated in the middle of the country, which has a population of about 1,370,727 and an area of 58,046 km², and known for its massive agricultural production.

6.1. Obtaining Initial Data

1. The experimental setup are based on the the weather data on 7–10 March 2017. Temperature measurements are taken in in agricultural ground soil at 10, 20, 30 and 50 cm levels, where the maximum and minimum values of the temperature are recorded.

2. It’s known that for any numerical prediction model, the data at the boundaries of the domain are indispensable. Hence, the $\alpha$ boundary designating the ground temperature is determined for each 10 min through a linear interpolation along the minimum and the maximum values for a given day, while the $\beta$ boundary is determined by referring to a previous work presented in [21]. The heat storage along the soil depth almost deals with the upper 60 cm layer. However, the experimental data show a deeper dependence of the temperature, i.e., depths exceeding 60 cm seem to be sensitive to climatic variations.

   It’s possible to estimate the temperature at 60 cm depth in the soil by studying the variation of the thermal gradient (the experimental data at 60 cm depth is not available) and considered to be a constant value for the whole experiment during 24 h (such an assumption is validated in a previous work), and the results are shown in Figure A1.

3. Taking $\Delta t = 10$ min and $h = \max h_j = 20$ cm.

4. The model will be run to get a value each 10 min along 12 h. The obtained results are then compared with the real data in order to estimate the mean squared error.

   $$EQM = \frac{1}{N} \sum_{n=1}^{n=N} \left( T_{\text{expected}} - T_{\text{real}} \right)^2$$

5. There are two modes of calculations: The first one so-called with initialization, means that the model can be turned while the data is initialized at 6 h, 12 h and 18 h. The second mode is called without initialization, where the model is executed for the whole deadline without any reset of the data.

6.2. Realistic Experience

The ground temperature can be predicted at different levels in the soil for different integration deadlines through the numerical code based on the finite element technique. Hence, two way of simulations are realized. The first way starts the the calculation without initialization and generates the predictions for different times. The second simulation launches the calculation with initialization for each 11 h. Thus, it would be possible to compare the predicted and the experimental values. The quadratic error and equally the mean squared error are calculated for each deadline along the whole simulation and the obtained results are as follows:

6.2.1. Without Initialization

A good agreement can be seen between the predicted and the experimental results for small depths, as shown in Figure A2. However, some differences between prediction and real data seem to appear when the depth increases. We note that the model gives exactly the real temperature for the 25 cm depth.

Taking Figure A3 in consideration, it’s clear that the agreement between the predictions and the experiment is quite good for the shallow depths, and the model regenerates almost the same values of the experiment for the levels between 25 and 30 cm. But on the other hand, Figures A2–A8 don’t show a good agreement between simulation and experimental data, especially for the too high deadlines,
namely 36 h, 48 h and 60 h for the different depths. In fact, the model has a weak overestimation for the temperature for small depths and underestimations for large depths. Thus, the best agreement are gotten for the depths included between 25 and 30 cm, which are still close to the observations, while the comparison for 72 h, 84 h and 96 h prove that the temperatures are not very well reproduced, though it persists to be close to the real data in the range of less than 0.6 °C.

Moreover, looking at Figure A9 allows to conclude that the smallest quadratic errors are fitting for the lowest deadlines, which is reasonable from a statistical point of view since the the model starts drifting away from the experimental value when the maturities increase. But if our case is considered, the simulation returns a quite well estimation for the deadlines of 84, 96 and 108 h (see Figure A10). It is worthy of note that the lowest error (0.19) corresponds to the best behavior of the model at the end of 24 h.

6.2.2. With Initialization

According to Figure A11 and Figure A12, the 12-h simulation with initialization does not report notable dissimilarity between the two predictions, where the same behavior of the model is realized with a small overestimation for the little depths and and a slight underestimation for larger depths. This would be checked by means of the the squared error, which reports exactly the same value for both methods. Figures A13–A15 equally allow the comparison between the simulation and the experimental results for the 24 h and 36 h, where the agreement seems to be good also for the 24 h deadline, with a very good behavior for the model. In spite of the slight overestimation with respect to the observations thereafter, we can not say that the model diverges completely since the differences are too small, the same previous comments for (Figures A16 and A17). Contrariwise, the predictions show an overestimation and end with a good similarity with the experiment for the 36 h deadline. While for the deadlines 96 and 108 h (Figures A18 and A19), a good correspondence is noticed between the simulation and the data for the deep depths, and small differences of less than 0.6 cm start to appear when the depth decreases for the 96 h deadline. While an unverified aspect is noticed for 108 h deadline where the simulation shows an underestimation for the temperature values for different depths. We note that the statistical results follow a positive evolution between the two simulation modes when comparing the mean squared errors, which varies from 0.46 for the without initialization case till 0.34 for the with initialization case. However, the accordance simulation-real data gets worse for different depths. In fact, the model shows a slight overestimation of the temperature value for the small and the large depths, and returns back realistic values for a depth of 10 cm, but it is still close to the observations, even though the simulation does not report a clear behavior with respect to the deadlines and depths. Both of overestimation and underestimation are seen, while the general behavior remains fairly realistic since the difference between the prediction and the experimental data is considered to be small.

7. Conclusions

The finite element method is proved, thanks to this work, to be a satisfactory tool allowing to describe the temperature variation as a function of the soil depth. The obtained results show that the method provides a highly precise prediction, where the difference between the experimental data and the model estimations does not exceed 0.6 degree, and the simulation code does not diverge even after 108 h of estimation. Eventually, the model will diverge from the realistic data as maturities increase, but far from the case considered into this study, and this should be attributed to the assumptions taken into account: Linear interpolation between minimum and maximum values of the soil temperature, which allows to calculate the temperatures every 10 min; the effects of clouds and dew that may be present during the morning time along the whole deadline and the presence of plants. Hence, the finite element method estimations showed very good agreements between the simulation predictions and the experimental data for the both simulation modes: With and without initialization, and the results were very crucial. It is worthy of note that the agreement between the simulation predictions and
the experimental data would be higher if the real measurement have been taken under a greenhouse, a more perfect case excluding the effect of the hydrometeor to the surface temperature variation.

Author Contributions: All authors contributed equally.

Funding: The authors gratefully acknowledges Qassim University, represented by the Deanship of Scientific Research, for the material support of this research under Number 3737-alrasscac-2018-1-14-S during the academic year 1439AH/2018.

Acknowledgments: The authors would like to thank the anonymous referees and the handling editor for their careful reading and for relevant remarks/suggestions which helped them to improve the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1. Without Initialization

![Figure A1. Estimation of the temperature at 60 cm depth in the soil.](image)

![Figure A2. Comparison after 24 h (Without Initialization).](image)
Figure A3. Comparison after 12 h (Without Initialization).

Figure A4. Comparison after 36 h (Without Initialization).

Figure A5. Comparison after 48 h (Without Initialization).
Figure A6. Comparison after 60 h (Without Initialization).

Figure A7. Comparison after 72 h (Without Initialization).

Figure A8. Comparison after 84 h (Without Initialization).
Figure A9. Comparison after 96 h (Without Initialization).

Figure A10. Comparison after 108 h (Without Initialization).

Appendix A.2. With Initialization

Figure A11. Comparison after 12 h (With Initialization).
Figure A12. Comparison after 24 h (With Initialization).

Figure A13. Comparison after 36 h (With Initialization).

Figure A14. Comparison after 48 h (With Initialization).
Figure A15. Comparison after 60 h (With Initialization).

Figure A16. Comparison after 72 h (With Initialization).

Figure A17. Comparison after 84 h (With Initialization).
Figure A18. Comparison after 96 h (With Initialization).

Figure A19. Comparison after 108 h (With Initialization).

References


© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).