Advanced numerical strategies for fast solution of transient problems

by Fabien Poulhaon

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in Computational Mechanics

at

Ecole Centrale de Nantes
GeM Laboratory, UMR CNRS 6183

Supervisors

Francisco Chinesta

Adrien Leygue

Acknowledgments

During the past semester, I have worked with a great number of people whose contribution in assorted ways to the research and the making of this thesis deserved special mention.

Firstly, I would like to record my gratitude to Francisco Chinesta for his supervision and guidance from the very early stage of this project. He offered me the opportunity to enrich my growth as a student and a researcher. This project was a great experience to me.

I gratefully acknowledge Adrien Leygue for his advices and his truly scientist intuition. He was a constant source of ideas and passions in science. Each time I was facing a difficulty, he was there to propose some alternatives and new horizons to explore.

I would like to thank also the PhD students and research engineers from the Model Reduction group within the GeM laboratory for their constant motivation. Many thank go in particular to Chady, Felipe, Brice and Hajer for their help and advices.

Special thanks to my fellows Feifei, Santiago and Shabeer for giving me such a pleasant time when working with them at the office, during lunch times and in many other circumstances. They contributed in their own way to the success of this project.

Finally, I would like to thank everybody who was important to the realization of this thesis and expressing my apology that I could not mention personally one by one.

Abstract

Real-time control of systems and processes has become a fundamental challenge during the past few years. Todays numerical simulation must not only be reliable but should also allow a dynamic interaction with a flux of information coming for instance from a measurement device. In a Dynamic Data Driven Application Sytem (DDDAS), the simulation is no more considered as an independent entity but included within a larger scope. The main issue concerns the speed of the computation: how can we reach real-time while ensuring a proper accuracy for the model ? In this work, we developed a new approach for the fast solution of transient equations based on the Proper Generalized Decomposition method, that could be subsequently included in a DDDAS. This project intends to propose in particular a first step towards a time parallelization of the computation based on a reduction of the model. The resulting speed-up proves to be quite promising for future realtime applications, especially for problems involving several characteristic times.

Contents

List of Figures

Introduction

Traditionally, when dealing with numerical models, one specifies static inputs : geometrical parameters, material parameters, boundary conditions, initial conditions...They are called static because they cannot be modified during the simulation. As soon as a parameter value is changed, the simulation has to be ran again. The associated computational cost can therefore become heavy. Moreover this approach is not always satisfying because simulation and reality are totally uncorrelated. This is a barreer to the efficiency and the reliability of the numerical model. In order to be representative of the physical phenomenom or at least of the image we have of it thanks to experimental measurements, most of the models require a dynamic response of the simulation.

In that context, Dynamic Data Driven Applications System (DDDAS) appears to be a very promising paradigm. A definition given by the National Science Foundation in 2006 presents DDDAS as a symbiotic feedback control system which can dynamically employ simulations to control and guide the experimental measurements, to determine when, where, and how it is best to gather additional data, and in reverse, can dynamically steer the application based on the experimental measurements.

The idea that lies under the DDDAS approach is the realtime control of systems and processes [7] [15]. Hence if we want to ensure a dynamic interaction between our simulation and data coming from sensors for instance, we need the computation to be fast enough to reach realtime. This is a challenge in itself because we have to ensure at the same time a proper accuracy for the model.

At this point we clearly understand that the computation cannot entirely be performed on-line. Part of the work has to be done off-line. The principle of the method we propose in this report consists in computing a general solution, valid for a wide range of problems. This latter is then particularized depending on the problem we are solving and finally the on-line workload reduces only to postprocessing which proves to be relatively light. The question arising then is: How can we compute this somehow "magical" general solution, this somehow numerical abaqus ?. Actually, the answer is quite simple: since we consider time/space as dimensions of the model, why not considering material parameters, boundary or initial conditions as dimensions too ?

We will see that classical mesh-based methods are not suited to the simulation of models defined in a space of high dimension. We have to employ another method in order to circumvent or at least alleviate the "Curse of Dimensionality". Among the available options, the Proper Generalized Decomposition (PGD), based on an a priori decomposition

of the unknown field, allows a significant reduction of the complexity of the model and proves to be a very interesting alternative to classical methods.

In the first part of this thesis report, we outline the limits of classical discretization methods when dealing with multidimensional models. We focus more precisely on the solution of the transient heat transfer equation. Both linear and non-linear cases are considered. After presenting the principle of the PGD method, we detail the different steps necessary to the construction of a solution under a separated form. Moreover, we propose an ambitious model in which the initial field of temperature is considered as an additional coordinate. This model opens new perpectives concerning Data Driven Inverse Identification.

We present in the second part a new kind of approach for solving transient problems based on a parallelization of the computation. We propose to split the time domain and to distribute the workload to several processors in order to take advantage of multicore architectures. We explain how the PGD makes it possible and what are the main issues associated to this concurrent approach: two options for time parallelization are detailed.

We expose then several methods in order to control and reduce the error caused by the reattachment of the local solutions at the interfaces of the time subdomains: minimization techniques (least squares, additional constraints...), classical and "optimized" overlapping. Because the primary objective of the parallelization is to increase the speed of the computation, we address results concerning the parallel speed-up in the chapter 7.

Finally, in order to illustrate the interest of this theoritical tool, we apply time parallelization to the simulation of an industrial process recently developed for composites manufacturing: the ultrasound welding.

Part I

A multidimensional model for the heat transfer equation

Chapter 1

The Proper Generalized Decomposition

1.1 The failure of classical mesh-based methods

Despite the impressive progress in computer performances during the last decade and the development of multiprocessors platforms and multicore processors, many problems encountered in computer science remain intractable because of their tremendeous complexity. Among these problems we can distinguish two main categories.

- On one hand, the transient problems involving several characteristic times. A classical example is the modeling of the degradation of a plastic material. A simple reaction-diffusion equation can be used to describe this phenomenon. Nevertheless, two characteristic times have to be taken into account. The first one, corresponding to the chemical reaction responsible for the degradation of the material at the microscopic scale, is of of the order of some microseconds, $\tau_{micro} \approx 10^{-5}s$. But in order to evaluate the consequences of this reaction on the mechanical characteristics of the material, the total simulation time should correspond to several years, $\tau_{macro} \approx 10\ years \approx 3.10^{8}s$. Obviously, the solution of such a problem with classical mesh-based methods like finite elements or finite differences is not achievable, because we would have to solve 3.10^{13} 1D, 2D or 3D problems!

- On the other hand, models defined in a space of high dimension. Many models encountered in computational science involve a tremendeous amount of dimensions as for example kinetic theory description of complex materials, quantum chemistry involving the solution of Schrödinger equation, financial mathematics for credit markets modeling. In those cases, the number of dimensions can easily reach 50, which corresponds to a complexity of 10^{100} in a classical mesh-based framework, if we consider only 100 elements per dimension. 10^{80} being the presumed number of particles in the whole universe, we will never be able to deal with such a quantity of degrees of freedom using computers: this is the so-called *Curse of Dimensionality*.

We could think however that the examples we just mentioned are not very common in mechanical engineering and that in most of the cases classical methods are sufficient. Actually, these methods show some limits even for very simple problems. In order to illustrate this idea let us have a look to the parametric heat transfer equation. For a unidimensional domain of length L this equation writes,

$$\frac{\partial u}{\partial t} - k\frac{\partial^2 u}{\partial x^2} = f \quad in \quad \Omega = \Omega_t \otimes \Omega_x = [0; t_{max}] \otimes [0; L] \tag{1.1}$$

$$\begin{cases} u(x,0) = u_0(x) \\ u(0,t) = u_a \qquad u(L,t) = u_b \end{cases} \tag{1.2}$$

The common approach is to solve this equation for a single value of the thermal conductivity k considering the temperature u as a function only of x and t. However, if we want to modify the value of k because we don't know precisely the characteristics of the material we are studying, we have to run several simulations for different values of k. This brute force approach can become really time consuming and doesn't guarantee a satisfying result.

A more interesting approach consists in incorporating k into the coordinates of the model, and finally compute a solution $u(t, x, k)$ defined in a 3D tensorial product space $\Omega = \Omega_t \otimes \Omega_x \otimes \Omega_k$, and valid for any thermal conductivity within the domain of variability Ω_k. Unfortunately, this idea relatively simple at first sight faces the issue already mentioned before. If we use a Finite Elements Method (or finite differences, finite volumes...), the number of degrees of freedom scales exponentially with the number of dimensions. Thus the addition of extra-dimensions is very penalizing. If for each dimension (x,t,k) we consider a mesh of 100 elements, which is relatively coarse, the complexity already reaches 10^6.

Mesh-based method are not suited to the calculate the solution of this kind of problems. Another method is required in order to deal with highly multidimensional models. We need a mathematical tool able to reduce the complexity of the model. Several options are available. Most of them are based on a finite sum decomposition of the unknown field. This decomposition can be done a posteriori and allows the extraction of a basis of functions, representative of the solution. We can mention the Proper Orthogonal Decomposition (POD), also called Karhunen-Loève decomposition, and the Singular Value Decomposition (SVD). However, we would like a method able to reduce the complexity of the solution without having to compute this latter. Several solutions to this problem have been developed. In particular, the Proper Generalized Decomposition (PGD) based on a separated representation of the solution appears to be a very promising alternative. We expose in the next section the main principle of this method and how it can be applied to heat transfer modeling.

1.2 PGD: A way to circumvent the Curse of Dimensionality

Pierre Ladevèze [11] interested in multiscale models, was looking for a method able to deal with a large number of linear evolution problems resulting from a nonlinear iterative strategy which is global in time. The objective was to reduce computational costs. Within the context of non-linear non-incremental solver LATIN, he developed the so-called radial approximation. This technique consists in approximating the solution by a sum of radial functions (called modes in the PGD). Each radial function is the product of a time function and a space function, so the solution of the space/time problem finally reads,

$$u\left(t,x\right) \approx \sum_{i=1}^{N} T_i\left(t\right) \cdot X_i\left(x\right) \tag{1.3}$$

where N is usually of some tens. The key of the method is it's ability to reduce the number of degrees of freedom: the complexity of the model scales linearly. If we consider 1000 elements for each dimension and 10 radial functions (N=10) the complexity is 10^4. With an FEM approach, the complexity scales exponentially and would be in that case 10^6.

The PGD is a generalization of the radial approximation for the solution of Partial Differential Equations defined in a tensorial product space $V = V_1 \otimes V_2 \otimes \cdots V_D$ where $\{V_i\}_{i=1}^{D}$ are separable Hilbert spaces. For instance, if the unknown field u we are trying to determine depends on x, t but also on Q parameters $(p^1, \cdots, p^Q)$, we can approximate it by a sum of modes, each mode being a product of functions, as follows,

$$u\left(t,x,p^1,...,p^Q\right) \approx \sum_{i=1}^{N} T_i\left(t\right) \cdot X_i\left(x\right) \cdot P_i^1\left(p^1\right) \cdots P_i^Q\left(p^Q\right) \tag{1.4}$$

The challenge is then to construct a separated representation of the solution without knowing a priori this latter, not even an approximation of it. This is not trivial because the basis functions used to approximate the solution are built on the fly. The idea is to introduce the separated representation of the solution into the weak form of the problem. Two approaches can then be implemented,

- Galerkin formulation.
- Minimization of the residual.

The mathematical anlysis of those methods is the subject of current research work, but the results provided up to now prove to be very satisfying. In both cases, an alternating directions fixed point algorithm is used. Let us illustrate this notion by coming back to the example of the 1D parametric heat transfer equation (1.1), where the source term f is constant for the sake of simplicity. If we consider the conductivity k as an extra-coordinate [4], the separated representation of the temperature field writes,

$$u(t,x,k) \approx \sum_{i=1}^{N} T_i(t) \cdot X_i(x) \cdot K_i(k) \tag{1.5}$$

The previous approximation will be constructed through an iterative process. If we suppose the approximation known at iteration n, at iteration n+1 we have,

$$u^{n+1}(t,x,k) \;=\; u^n(t,x,k) \;+\; T_{n+1}(t) \cdot X_{n+1}(x) \cdot K_{n+1}(k) \tag{1.6}$$

For alleviating the notation, let us replace $T_{n+1}(t) \cdot X_{n+1}(x) \cdot K_{n+1}(k)$ by $R(t) \cdot S(x) \cdot W(k)$. The alternating fixed point algorithm consists in considering successively 2 functions out of 3 as known and to compute the remaining one. Hence, we will first consider that S and W are known and we will obtain R. From the just computed R and the previous W, we will compute S and finally from the updated R and S we get W. This operation is repeated until reaching convergence. The mode n+1 corresponds finally to $T_{n+1}(t) = R(t)\,,\; X_{n+1}(x) = S(x)\,,\; K_{n+1}(k) = W(k)$.

Computing $R(t)$

The weak form of the 1D Heat transfer problem at iteration n+1 writes,

$$\int_\Omega u^* \left(\frac{\partial u^{n+1}}{\partial t} - k \frac{\partial^2 u^{n+1}}{\partial x^2} - f \right) d\Omega \;=\; 0 \tag{1.7}$$

where $\Omega = \Omega_t \otimes \Omega_x \otimes \Omega_k$ and the test function is chosen equal to,

$$u^* = R^*(t) \cdot S(x) \cdot W(k) + R(t) \cdot S^*(x) \cdot W(k) + R(t) \cdot S(x) \cdot W^*(k) \tag{1.8}$$

Since we assume that S and W are known, it finally simplifies to $u^* = R^*(t) \cdot S(x) \cdot W(k)$. Introducing the separated representation of the solution at iteration n+1 into 1.7, we obtain:

$$\int_\Omega R^* \cdot S \cdot W \left(\frac{dR}{dt} \cdot S \cdot W \;-\; k \cdot R \cdot \frac{d^2 S}{dx^2} \cdot W \right) d\Omega = -\int_\Omega R^* \cdot S \cdot W \cdot H^n d\Omega \tag{1.9}$$

with H^n, the residual at iteration n, writes,

$$H^n = \sum_{i=1}^{n} \left(\frac{dT_i}{dt} \cdot X_i \cdot K_i - k \cdot T_i \cdot \frac{d^2 X_i}{dx^2} \cdot K_i \right) - f \tag{1.10}$$

The weak form finally reduces to:

$$\begin{aligned}&\int_{\Omega_t} R^* \cdot \left((w_1 \cdot s_1 \cdot \frac{dR}{dt} - w_2 \cdot s_2 \cdot R \right) dt \\ &= - \int_{\Omega_t} R^* \cdot \left(\sum_{i=1}^{n} \left(w_4^i \cdot s_5^i \cdot \frac{dT_i}{dt} - w_5^i \cdot s_4^i \cdot T_i \right) - w_3 \cdot s_3 \cdot f \right) dt\end{aligned} \tag{1.11}$$

Using the following notation for the coeffcients r, s and w

$$\begin{bmatrix} r_1 = \int_{\Omega_t} R^2 dt & s_1 = \int_{\Omega_x} S^2 dx & w_1 = \int_{\Omega_k} W^2 dk \\ r_2 = \int_{\Omega_t} R \cdot \frac{dR}{dt} dt & s_2 = \int_{\Omega_x} S \cdot \frac{d^2 S}{dx^2} dx & w_2 = \int_{\Omega_k} k W^2 dk \\ r_3 = \int_{\Omega_t} R dt & s_3 = \int_{\Omega_x} S dx & w_3 = \int_{\Omega_k} W dk \\ r_4^i = \int_{\Omega_t} R \cdot \frac{dT_i}{dt} dt & s_4^i = \int_{\Omega_x} S \cdot \frac{d^2 X_i}{dx^2} dx & w_4^i = \int_{\Omega_k} k \cdot W \cdot K_i dk \\ r_5^i = \int_{\Omega_t} R \cdot T_i dt & s_5^i = \int_{\Omega_x} S \cdot X_i dx & w_5^i = \int_{\Omega_k} W \cdot K_i dk \end{bmatrix} \tag{1.12}$$

This equation is the weak form of an ODE defining the time evolution of R and we can solve it using for example Discontinuous Galerkin. If we come back to the strong form, it can be solved using finite differences among several other possibilities.

Computing $S(x)$

Here the test function takes the form $u^*(x,t,k) = S^*(t) \cdot R(t) \cdot W(k)$.
The weak form writes,

$$\int_\Omega S^* \cdot R \cdot W \left(\frac{dR}{dt} \cdot S \cdot W \; - \; k \cdot R \cdot \frac{d^2 S}{dx^2} \cdot W \right) d\Omega = - \int_\Omega S^* \cdot R \cdot W \cdot H^n d\Omega \tag{1.13}$$

Using the previous notation, it reduces to:

$$\begin{aligned}&\int_{\Omega_x} S^* \cdot \left((w_1 \cdot r_2 \cdot S - w_2 \cdot r_1 \cdot \frac{d^2 S}{dx^2} \right) dx \\ &= - \int_{\Omega_x} S^* \cdot \left(\sum_{i=1}^{n} \left(w_4^i \cdot r_4^i \cdot X_i - w_5^i \cdot r_5^i \cdot \frac{d^2 X_i}{dx^2} \right) - w_3 \cdot r_3 \cdot f \right) dx\end{aligned} \tag{1.14}$$

This corresponds to an elliptic steady state boundary value problem. In order to solve it, we can apply any discretization technique (FEM, FVM) on the weak form or we can come back to the strong form and use finite differences for instance.

$$w_1 \cdot r_2 \cdot S - w_2 \cdot r_1 \cdot \frac{d^2 S}{dx^2} = \sum_{i=1}^{n} \left(w_5^i \cdot r_5^i \cdot \frac{d^2 X_i}{dx^2} - w_4^i \cdot r_4^i \cdot X_i \right) + w_3 \cdot r_3 \cdot f \tag{1.15}$$

Computing $W(k)$

Finally, using the just computed $R(t)$ and $S(x)$ we can obtain the last field $W(k)$. In that case the test function to be used is $u^* = W^* \cdot R(t) \cdot S(x)$ and the weak form of the problem is,

$$\int_\Omega W^* \cdot R \cdot S \left(\frac{dR}{dt} \cdot S \cdot W \; - \; k \cdot R \cdot \frac{d^2 S}{dx^2} \cdot W \right) d\Omega = - \int_\Omega W^* \cdot S \cdot R \cdot H^n d\Omega \quad (1.16)$$

That reduces to:

$$\begin{aligned} &\int_{\Omega_k} W^* \cdot (s_1 \cdot r_2 \cdot W - r_2 \cdot s_1 \cdot k \cdot W)\, dk \\ &= - \int_{\Omega_k} W^* \cdot \left(\sum_{i=1}^{n} \left(s_5^i \cdot r_4^i \cdot K_i - r_5^i \cdot s_4^i \cdot k \cdot K_i \right) - s_3 \cdot r_3 \cdot f \right) dk \end{aligned} \quad (1.17)$$

The corresponding strong form being:

$$s_1 \cdot r_2 \cdot W - r_2 \cdot s_1 \cdot k \cdot W = \sum_{i=1}^{n} \left(r_5^i \cdot s_4^i \cdot k \cdot K_i - s_5^i \cdot r_4^i \cdot K_i \right) + s_3 \cdot r_3 \cdot f \quad (1.18)$$

We can notice that this equation doesn't involve any derivative with respect to k, and this is in accordance with the original equation. This algebraic equation is then very few time-consuming from a computational point of view. It confirms that the addition of extra coordinates is not too penalizing when using the PGD method.

Up to now we only focused on a linear heat transfer, the conductivity being considered as a constant. Nevertheless, many industrial applications involve conductivities that depend on the temperature in a more or less complex fashion and the problem becomes non-linear. The next chapter is devoted to the description of a multidimensional solution for a non linear heat transfer.

Chapter 2

Nonlinear Heat Transfer Modeling

In order to expose clearly the problem, let us consider a simple example of non-linearity. The conductivity k depends linearly on the temperature u, that is to say $k = a + b \cdot u$ where a and b are constant. The corresponding heat transfer equation writes,

$$\frac{\partial u}{\partial t} - \frac{\partial}{\partial x}\left((a + b \cdot u) \cdot \frac{\partial u}{\partial x}\right) = f \tag{2.1}$$

Keeping in mind that our final goal is to obtain a solution general enough that could be used for real-time control of experiments or manufacturing processes, we propose to solve this equation for any value of the parameters a and b within a domain of variability that we define a priori. To do so, we introduce a and b in the coordinates of the model, and we obtain the following separated representation :

$$u(t, x, a, b) \approx \sum_{i=1}^{N} T_i(t) \cdot X_i(x) \cdot A_i(a) \cdot B_i(b) \tag{2.2}$$

defined in a 4D space $\Omega = \Omega_t \otimes \Omega_x \otimes \Omega_a \otimes \Omega_b$.

This problem cannot be solved directly, a linearization step is required. Two incremental strategies for treating the presence of the non-linear term $u \cdot \frac{\partial u}{\partial x}$ are addressed in the following section.

2.1 Incremental linearizations

The first linearization strategy we employ consists in considering that at iteration n+1, the nonlinear term $u^{n+1} \cdot \frac{\partial u^{n+1}}{\partial x}$ can be evaluated using the solution at iteration n and then be replaced by the term $u^n \cdot \frac{\partial u^n}{\partial x}$. This term being actually known, it is moved to the right hand side of the equation and acts like a source term. Therefore, the equation to be solved at iteration n+1 reads,

$$\frac{\partial u^{n+1}}{\partial t} - a\frac{\partial^2 u^{n+1}}{\partial x^2} = f + b\frac{\partial}{\partial x}\left(u^n \cdot \frac{\partial u^n}{\partial x}\right) \tag{2.3}$$

Doing so we make a relatively strong assumption and as we demonstrate in the chapter 4 devoted to numerical results, the convergence of this method depends on the way the right hand side of the equation is updated.

Another option is to consider that the conductivity depends only on the temperature calculated at the previous iteration [5]. The term $u^{n+1} \cdot \frac{\partial u^{n+1}}{\partial x}$ is then partially evaluated using the solution at iteration n and replaced by the linear term $u^n \cdot \frac{\partial u^{n+1}}{\partial x}$. The linearized equation takes the form,

$$\frac{\partial u^{n+1}}{\partial t} - a\frac{\partial^2 u^{n+1}}{\partial x^2} - b\frac{\partial}{\partial x}\left(u^n \cdot \frac{\partial u^{n+1}}{\partial x}\right) = f \tag{2.4}$$

We can proceed in several ways in order to compute this solution. The first approach consists in performing the same kind of calculation we did in the linear case, except than this time we have a fourth dimension corresponding to coefficient b, characterizing somehow the importance of the non-linearity in the model. In that case, after introducing the separated representation of the temperature field, equation 2.2, an alternating directions fixed point algorithm is performed.

Let us detail now one iteration of the calculation of the function B_{n+1} considering that n iterations have already been performed. For the sake of clarity, the mode n+1 corresponding to the product $T_{n+1}(t) \cdot X_{n+1}(x) \cdot A_{n+1}(a) \cdot B_{n+1}(b)$ is denoted $R(t) \cdot S(x) \cdot W(a) \cdot V(b)$.

The weak form of the problem writes,

$$\int_\Omega u^* \left(\frac{\partial u^{n+1}}{\partial t} - a \cdot \frac{\partial^2 u^{n+1}}{\partial x^2} \; - \; b \cdot \frac{\partial u^n}{\partial x} \cdot \frac{\partial u^{n+1}}{\partial x} \; - \; b \cdot u^n \cdot \frac{\partial^2 u^{n+1}}{\partial x^2} - f\right) = 0 \tag{2.5}$$

R, S and W are fixed and the test function u^* is chosen equal to $V^* \cdot R \cdot S \cdot W$. Finally, the weak form becomes:

$$\begin{aligned}\int_\Omega V^* \cdot R \cdot S \cdot W \Bigg[\left(\frac{dR}{dt} \cdot S - a \cdot R \cdot \frac{d^2 S}{dx^2}\right) \cdot W \cdot V \; - \; b \cdot \left(\sum_{i=1}^{n} T_i \cdot \frac{dX_i}{dx} \cdot A_i \cdot B_i\right) R \cdot \frac{dS}{dx} \cdot W \cdot V \\ - \; b \cdot \left(\sum_{i=1}^{n} T_i \cdot X_i \cdot A_i \cdot B_i\right) R \cdot \frac{d^2 S}{dx^2} \cdot W \cdot V \Bigg] d\Omega = - \int_\Omega V^* \cdot R \cdot S \cdot W \cdot H^n d\Omega\end{aligned} \tag{2.6}$$

where H^n, the residual at iteration n writes,

$$H^n = \sum_{i=1}^{n} \frac{dT_i}{dt} \cdot X_i \cdot A_i \cdot B_i \; - \; a \sum_{i=1}^{n} T_i \cdot \frac{d^2 X_i}{dx^2} \cdot A_i \cdot B_i \; - \; b \left(\sum_{i=1}^{n} T_i \cdot \frac{dX_i}{dx} \cdot A_i \cdot B_i \right) \cdot \left(\sum_{j=1}^{n} T_j \cdot \frac{dX_j}{dx} \cdot A_j \cdot B_j \right) - \; b \left(\sum_{i=1}^{n} T_i \cdot X_i \cdot A_i \cdot B_i \right) \cdot \left(\sum_{j=1}^{n} T_j \cdot \frac{d^2 X_j}{dx^2} \cdot A_j \cdot B_j \right) \; - \; f \tag{2.7}$$

Let us introduce at this point some notations in addition to 1.12:

$$\left[\begin{array}{ccc} s_6^i = \int_{\Omega_x} S \cdot \frac{dS}{dx} \cdot \frac{dX_i}{dx} d\Omega_x & s_7^{ij} = \int_{\Omega_x} S \cdot \frac{dX_i}{dx} \cdot \frac{dX_j}{dx} d\Omega_x & \\ r_8^i = \int_{\Omega_t} R^2 \cdot T_i dt & s_8^i = \int_{\Omega_x} S \cdot \frac{d^2 S}{dx^2} \cdot X_i dx & w_8^i = \int_{\Omega_a} W^2 \cdot A_i da \\ r_9^{ij} = \int_{\Omega_t} R \cdot T_i \cdot T_j dt & s_9^{ij} = \int_{\Omega_x} S \cdot X_i \cdot \frac{d^2 X_j}{dx^2} dx & w_9^{ij} = \int_{\Omega_a} W \cdot A_i \cdot A_j da \end{array} \right] \tag{2.8}$$

Thereby the weak form simplifies to,

$$\begin{aligned} &\int_{\Omega_b} V^* \left(r_2 \cdot s_1 \cdot w_1 \; - \; r_1 \cdot s_2 \cdot w_2 \; - \; b \sum_{i=1}^{n} \left(s_6^i + s_8^i \right) \cdot r_8^i \cdot w_8^i \cdot B_i \right) V db \\ &= - \int_{\Omega_b} V^* \left(\sum_{i=1}^{n} \left(r_4^i \cdot s_5^i \cdot w_5^i - r_5^i \cdot s_4^i \cdot w_4^i - b \sum_{j=1}^{n} \left(s_7^{ij} + s_9^{ij} \right) \cdot r_9^{ij} \cdot w_9^{ij} \cdot B_j \right) B_i - r_3 \cdot s_3 \cdot w_3 \right) db \end{aligned} \tag{2.9}$$

The corresponding strong form is an algebraic equation, it doesnt' involve any derivative with respect to time or space. Therefore, the addition of coefficient b in the coordinates of the model has a moderate impact on the total computational cost. Nevertheless, we didn't take advantage of the work that has been done for the linear problem.

We already computed the solution of this problem in the linear case, that is to say for b=0.It corresponds actually to the 3D solution we exposed in the previous part, except that the coefficient k is denoted here a. We have a separated representation of this solution. It would be then very interesting from a computational point of view to use these modes for the calculation of the solution for the non linear problem.

We can build a 4D solution from the 3D solution by multiplying each mode computed upstream by a unit function that we denote 1_B. If for instance 50 modes have been necessary to obtain a converged solution for the linear problem, then the separated representation of the solution for the non-linear problem can be written as,

$$u(t,x,a,b) \approx \underbrace{\sum_{i=1}^{50} T_i \cdot X_i \cdot A_i \cdot 1_B}_{Solution\ of\ the\ linear\ problem} + \underbrace{\sum_{j=1}^{Q} T_j \cdot X_j \cdot A_j \cdot B_j}_{Corrective\ Modes} \tag{2.10}$$

The new modes that are calculated act then as a correction of the linear solution. This allows us to save computational time because less modes will be necessary to obtain a converged solution. The linear problem provides a good guess for the solution of the non-linear problem and it appears quite logical to use it as a starting point.

2.2 Another alternative: Newton's linearization

The Newton linearization assumes that the function value is close enough to some guess value [5]. Then we can assume that the solution at iteration n+1 is close to the solution at iteration n, and can be written,

$$u^{n+1} = u^n + \tilde{u} \tag{2.11}$$

where $\tilde{u}$ is small compared to u^n.

This method is theoritically more efficient because it exhibits a quadratic convergence rate if the initial guess is close enough to the solution we are looking for. The iteration n refers here to the Newton iterative procedure. It has no relation with the mode n of the separated representation of our solution, unlike what we saw in the incremental approach. In order to avoid any confusion, we will use the index k for Newton's iterations. The non-linear term $u \cdot \frac{\partial u}{\partial x}$ can be linearized using a two-variable Taylor expansion,

$$u^{k+1} \cdot \frac{\partial u^{k+1}}{\partial x} \approx u^k \cdot \frac{\partial u^k}{\partial x} \; + \; \tilde{u} \cdot \frac{\partial u^k}{\partial x} \; + \; u^k \cdot \frac{\partial \tilde{u}}{\partial x} \tag{2.12}$$

Using this new expression to replace the non-linear term in 2.1, we finally obtain that the increment $\tilde{u}$ is the solution of a linearized problem whose weak form writes,

$$\begin{aligned} &\int_\Omega \tilde{u}^* \left[\frac{\partial \tilde{u}}{\partial t} - a\frac{\partial^2 \tilde{u}}{\partial x^2} - b\left(\tilde{u} \cdot \frac{\partial^2 u^k}{\partial x^2} + 2\frac{\partial \tilde{u}}{\partial x} \cdot \frac{\partial u^k}{\partial x} + u^k \cdot \frac{\partial^2 \tilde{u}}{\partial x^2} \right) \right] d\Omega \\ &= \int_\Omega \tilde{u}^* \left[f - \frac{\partial u^k}{\partial t} + a\frac{\partial^2 u^k}{\partial x^2} + b\left(u^k \cdot \frac{\partial^2 u^k}{\partial x^2} + \left(\frac{\partial u^k}{\partial x} \right)^2 \right) \right] d\Omega \end{aligned} \tag{2.13}$$

With the Newton linearization is then slightly different. In the incremental approach, we compute our solution mode by mode, updating the left hand side of the equation by adding new terms corresponding to the just computed modes. Here, we are interested in the increment $\tilde{u}$ that doesn't correspond to one mode only but a group of modes. Indeed, assuming that $\tilde{u}(x, t, a, b)$ can be written in a separated form, the solution of 2.13 is computed using the PGD method. We are not going to detail how the modes are calculated but the process is exactly the same than described previously.

Once we reach convergence, we update the solution by adding the increment $\tilde{u}$. A new problem similar to 2.13 can then be defined and we proceed the same way to compute the next increment. The calculation stops when the norm of the increment is small enough.

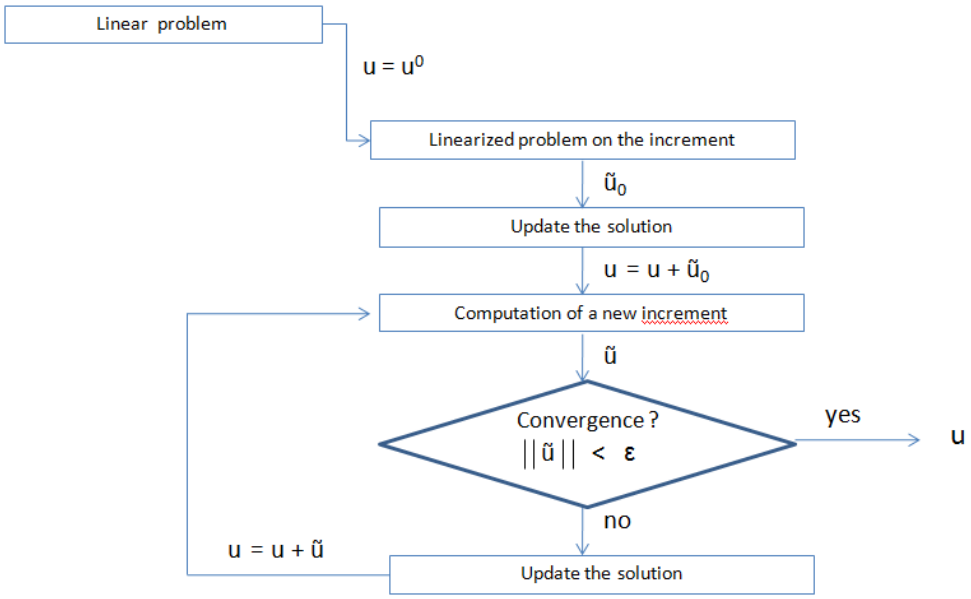


Figure 2.1: Newton's linearization

Thus we have two nested iterative procedures : Newton and PGD. For each iteration of the Newton's procedure, we have to apply the PGD method in order to compute the increment and store it in a separated fashion.

The key point of Newton's linearization is the choice of an initial guess. It has to be close enough to the final solution so the required number of iterations can be reduced. It's even more important for us because one Newton's iteration is quite costly from a computational point of view. Besides the cost associated to PGD, we have to take into account that the problem to be solved for computing the increment becomes more and more complex as we move forward. Indeed, the term u^k is updated at each iteration, and this leads to the introduction of numerous terms into the equation. Thus, this linearization method is much more demanding and the associated computational cost makes it less interesting than incremental approaches. We finally decided not to implement this strategy.

Chapter 3

Introducing fields in the coordinates of the model

Until now we only considered a multidimensional problem where the additional coordinates were scalar quantities: k, a or b. As we mentioned previously, our final objective is to compute a general solution of the heat transfer equation that could be incorporated for future applications into a DDDAS. The introduction of the thermal conductivity as a new coordinate was a first step. Nevertheless this model is not completely satisfying and can be improved by incorporating the initial condition, that is to say by introducing the initial field of temperature in the coordinates of the model.

3.1 Parametrization of the initial condition

The challenge is to compute a solution valid for *any initial condition*: $u\left(t, x, k, u_0(x)\right)$. In order to achieve this purpose, we must define one (or several) new dimensions which represent the initial field $u_0(x)$. In other words, we have to parametrize the initial condition. The first idea would be to use the nodal values corresponding to a linear finite element approximation on the calculation mesh. Indeed, if we use a classical finite element discretization in space the initial condition writes,

$$u_0\left(x\right) = \sum_{i=1}^{Nnodes} u_0^i \cdot \varphi^i\left(x\right) \tag{3.1}$$

where $\left\{\varphi^i\right\}_{i=1}^{Nnodes}$ are the delta Kronecker shape functions and $\left\{u_0^i\right\}_{i=1}^{Nnodes}$ are the corresponding nodal values.

This choice is not reasonnable because the mesh used for the computation has to be fine enough to guarantee a proper accuracy for the model. Then parametrizing the initial condition through those nodal values would lead to the addition of some hundreds of new coordinates. Even if the PGD based method is very good at dealing with highly multidimensional models, the associated computational cost would become unacceptable.

Actually, we don't need so many nodal values to represent the initial condition. In most of the cases, ten values or less are enough. Especially if we use higher order interpolation.

Thus an auxiliary grid, much coarser than the one used for the computation, is built and is only used for the construction of the initial field of temperature. The index **c** refers to the variables related to this coarse mesh, and the nodal values associated to this mesh are denoted p^j. The initial condition becomes,

$$u_0(x) = \sum_{j=1}^{Nc} p^j \cdot \varphi_c^j(x) \tag{3.2}$$

The choice of the number of nodes composing this second mesh has to be done in accordance to the interpolation that will be used to construct the initial temperature profile. Indeed if we use a linear interpolation, the required number of nodes to fit a curvilinear profile is obviously higher than for a polynomial interpolation. Moreover we will see that not only the number but the position of the nodes can be crucial. Let's just remark here that because we are dealing with a 1D problem, the first and the last node of the coarse mesh correspond to the boundaries. Then the associated nodal values are in fact used to parametrize the boundary conditions. This remains valid only if we are imposing constant Dirichlet Boundary conditions. Finally our magic solution, our numerical abaqus, will contain the solution of the heat transfer problem for any initial condition and any constant Dirichlet boundary conditions !

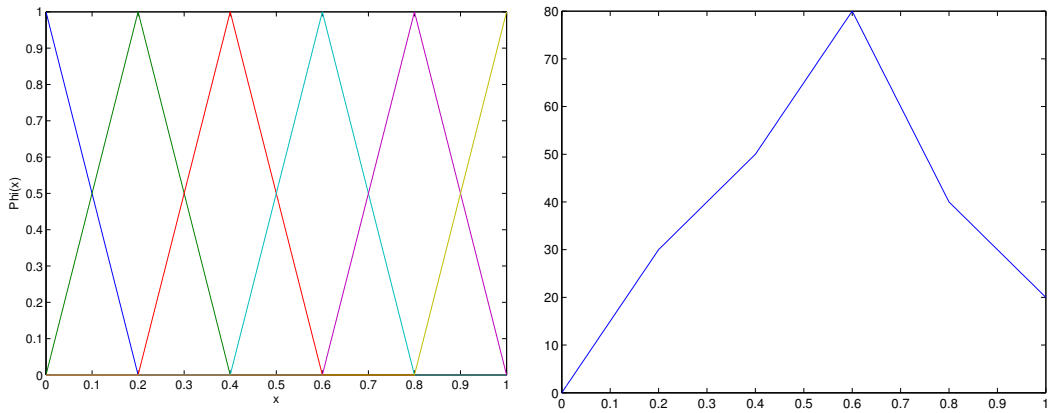
3.2 Interpolation

Among the several available options, we chose to focus only on the linear interpolation and an higher order interpolation coming from Newton's polynomial basis. The linear interpolation is the classical tool used in many discretization techniques: finite elements, finite differences. It is based on the use of piecewise linear functions defined with the delta Kronecker symbol.

$$\varphi_c^j\left(x_c^i\right) = \delta_i^j = \begin{cases} 1 \; if \; i = j \\ 0 \; else \end{cases} \tag{3.3}$$

However, linear interpolation fails to approximate accurately an initial field whose variations are smooth, unless we use a lot of nodes. But as we mentioned before, our objective is to use as few nodes as possible so the number of parameters for the initial condition and then the number of new dimensions of our model are compatible with the computational cost we can afford. Here appears the necessity of using another type of interpolation, more complex but much more efficient, for modeling realistic scenarios.

Newton polynomials represent an interesting alternative to the piecewise linear functions. This interpolation method is also called Newton's divided differences interpolation



(a) Linear shape functions (b) Linear approximation

Figure 3.1: Linear approximation

polynomial, because the coefficients of the polynomials are computed using divided differences. For a given set of points (x^i, p^i) corresponding respectively to the location of the nodes in the auxiliary coarse mesh and the associated nodal values, we are interested in finding a polynomial basis $\left(\psi_c^1, \psi_c^2, \ldots, \psi_c^{Nc}\right)$ such that our initial condition writes,

$$u_0(x) = \sum_{j=1}^{Nc} p^j \cdot \psi_c^j(x) \tag{3.4}$$

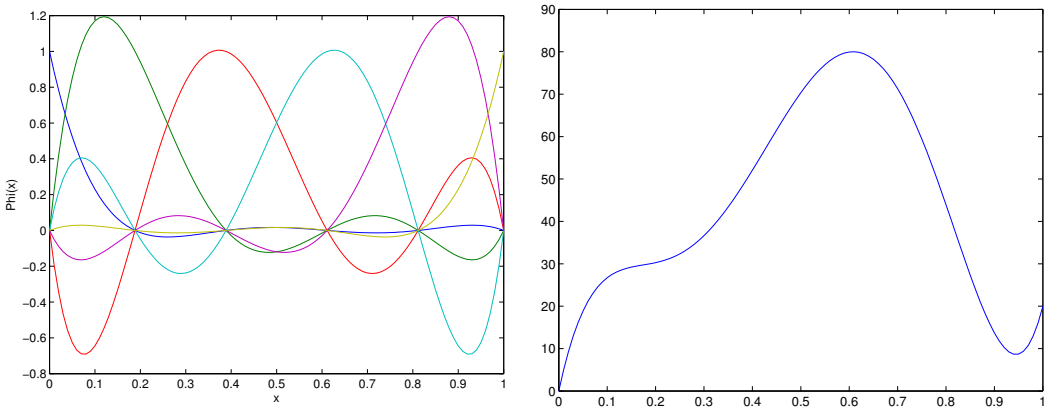
Newton's interpolation consists in finding the unique polynomial of least possible degree that passes through all the points from the set. But it doesn't provide directly the basis we are looking for. Some work has to be done. Initially, we know that the interpolation polynomial $I(x)$ reads,

$$I(x) = \sum_{i=1}^{Nc} c_i \cdot n_i(x) \quad where \quad n_i(x) = \prod_{j=0}^{i-1}(x - x_j) \tag{3.5}$$

and $c_i = [y_0, \cdots, y_i]$ is the divided difference defined as:

$$\begin{aligned} c_0 &= [y_0] = y_0 \\ c_1 &= [y_0, y_1] = \tfrac{y_1 - y_0}{x_1 - x_0} \\ c_j &= [y_0, y_1, \cdots, y_j] = \tfrac{[y_1, y_2, \cdots, y_j] - [y_0, y_1, \cdots, y_{j-1}]}{x_j - x_0} \end{aligned} \tag{3.6}$$

Then we can notice that each nodal value y_i (except the last one) is multiplied by several polynomials from the Newton basis when constructing $I(x)$, because they are present



(a) Polynomial functions (b) Polynomial approximation

Figure 3.2: Polynomial approximation

in several divided differences. For instance y_0 will be multiplied by a linear combination of all the polynomials in the basis. Hence we need to find a general expression for the polynomial multiplying each nodal values, in order to obtain the equality,

$$I(x) = \sum_{i=1}^{Nc} c_i \cdot n_i(x) = \sum_{j=1}^{Nc} p^j \cdot \psi_c^j(x) \tag{3.7}$$

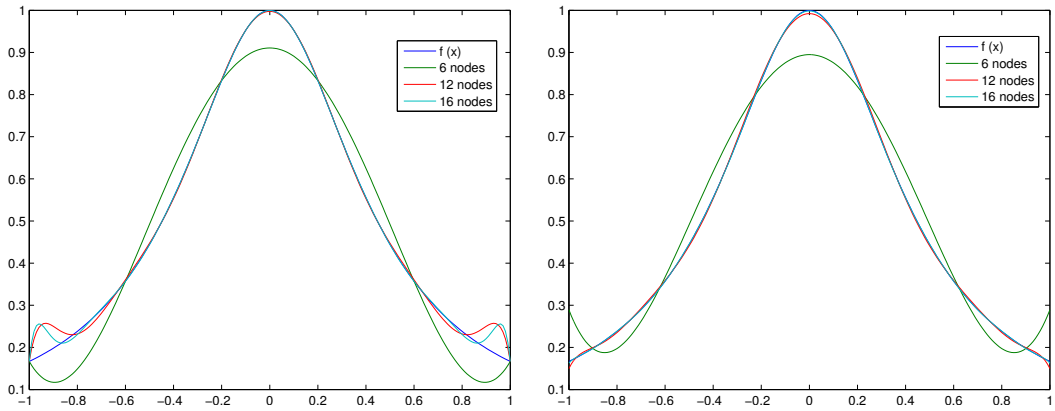
We propose the following expression,

$$\psi_c^j(x) = \frac{1}{\prod_{k=1}^{j-1}\left(\sum_{i=k}^{j-1}\Delta_i\right)}\left(n_j(x) + \sum_{l=j+1}^{Nc}\frac{n_l(x)(-1)^{l+j}}{\prod_{n=1}^{l-j+1}\left(\sum_{m=j}^{n+j-1}\Delta_m\right)}\right) \tag{3.8}$$

where $\Delta_i = x_i - x_{i-1}$ corresponds to the step length.
If the nodes are equidistant, then $\Delta_i = H \quad \forall i$, and the previous expression reduces to,

$$\psi_c^j(x) = \sum_{l=j}^{Nc}\frac{n_l(x)(-1)^{l+j}}{(j-1)!\,(l-j)!H^{l-1}} \tag{3.9}$$

The issue we face when using a polynomial basis like Newton's polynomial is the so-called Runge phenomenon. Indeed, some functions are very difficult to approximate using polynomials. The error increases with the number of nodes, which is quite paradoxal. One of the existing solution is to use a grid whose distribution of nodes is denser close to the edges. Such a set of nodes can be obtained by computing the roots of the Chebyshev polynomials of the first kind. They are called Chebyshev nodes and their general expression on an interval $[a, b]$ writes,



(a) Equidistant nodes (b) Chebyshev nodes

Figure 3.3: Runge phenomenon

$$x_i = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{2i-1}{2n}\pi\right) \tag{3.10}$$

where n is the number of nodes to be constructed. Let's illustrate the necessity of using Chebyshev nodes by taking the example of the function $f : x \to \frac{1}{1+5x^2}$. On a regular grid that is to say with equidistant nodes, Newton's polynomial are unable to approximate this function. We observe oscillations close to the edges as shown on figure 3.3 whereas the interpolation built from Chebyshev nodes gives much better results.

3.3 Construction of the multidimensional solution

We are finally able to represent the initial field of temperature with a reasonnable amount of parameters. This allows us to build a separated representation of the solution and to compute the modes on the fly thanks to the PGD. The strategy is then relatively simple. In order to calculate a solution valid for any initial condition or at least for a large range of configurations, we consider that each nodal value p^i has the same domain of variability $\Omega_p = [p_{min}; p_{max}]$, where the bounds p_{min} and p_{max} are fixed a priori. We use the variable p in the rest of this report for the nodal values. Finally, the separated representation of the temperature field writes,

$$u\left(t, x, k, p^1, p^2, \ldots, p^{Nc}\right) = \sum_{i=1}^{N} T_i(t) \cdot X_i(x) \cdot K_i(k) \cdot P_i^1(p) \cdot P_i^2(p) \cdots P_i^{Nc}(p) \tag{3.11}$$

which is defined in a space of dimension 3+Nc, $\Omega = \Omega_t \otimes \Omega_x \otimes \Omega_k \otimes \Omega_p \otimes \cdots \otimes \Omega_p$.

Each function composing a mode is computed using the alternating directions fixed point algorithm. Let's illustrate here one iteration for the calculation of the function P^1_{n+1}. At iteration n+1 the solution reads,

$$u^{n+1} = u^n + T_{n+1}(t) \cdot X_{n+1}(x) \cdot K_{n+1}(k) \cdot P^1_{n+1}(p) \cdots P^{Nc}_{n+1}(p) \tag{3.12}$$

For alleviating the notation, the second term of the right hand side of the previous equation is replaced by $R(t) \cdot S(x) \cdot W(k) \cdot V^1(p) \cdots V^{Nc}(p)$. The method consists then in introducing this separated representation in the weak form of the problem. In order to calculate V^1, all the other functions are fixed and the test function takes the form $u^* = V^{1*} \cdot R \cdot S \cdot W \cdot V^2 \cdots V^{Nc}$. The weak form of the problem writes,

$$\begin{aligned} &\int_\Omega V^{1*} \cdot R \cdot S \cdot W \cdot V^2 \cdots V^{Nc} \left(\frac{dR}{dt} \cdot S \; - \; k \cdot R \cdot \frac{d^2 S}{dx^2} \right) \cdot W \cdot V^1 \cdot V^2 \cdots V^{Nc} d\Omega \\ &= - \int_\Omega V^{1*} \cdot R \cdot S \cdot W \cdot V^2 \cdots V^{Nc} \cdot H^n d\Omega \end{aligned} \tag{3.13}$$

where H^n, the residual at iteration n, writes,

$$H^n = \sum_{i=1}^{n} \frac{dT_i}{dt} \cdot X_i \cdot K_i \cdot P^1_i \cdots P^{Nc}_i \; - \; k \sum_{i=1}^{n} T_i \cdot \frac{d^2 X_i}{dx^2} \cdot K_i \cdot P^1_i \cdots P^{Nc}_i \; - \; f \tag{3.14}$$

This equation reduces to:

$$\begin{aligned} &\int_{\Omega_p} V^{1*} (r_2 \cdot s_1 \cdot w_1 - r_1 \cdot s_2 \cdot w_2) \cdot v_1^2 \cdots v_1^{Nc} \cdot V^1 dp \\ &= - \int_{\Omega_p} V^{1*} \left(\sum_{i=1}^{n} \left(r_4^i \cdot s_5^i \cdot w_5^i + r_5^i \cdot s_4^i \cdot w_4^i \right) \cdot v_5^{2\;i} \cdots v_5^{Nc\;i} \cdot P_i^1 \; + \; r_3 \cdot s_3 \cdot w_3 \cdot v_3^2 \cdots v_3^{Nc} \cdot f \right) dp \end{aligned} \tag{3.15}$$

The coefficients r,s and w have already been introduced previously in 1.12. The same logic is used for the new coefficients appearing here, and then for $j \in [1, Nc]$ we have :

$$v_1^j = \textstyle\int_{\Omega_p} V^{j\,2} dp \quad v_3^j = \int_{\Omega_p} V^j dp \quad v_5^{j\;i} = \int_{\Omega_p} V^j \cdot P_i^j dp \tag{3.16}$$

Finally if we come back to the strong form of the problem, the evolution of the quantity V^1 is described by an algebraic equation of the form:

$$\alpha \cdot V^1 = \sum_{i=1}^{n} \beta_i \cdot P_i^1 \; + \; \gamma \cdot f \tag{3.17}$$

Depending on the precision we want to ensure, the introduction of the initial field of temperature in the coordinates of the model will lead to the addition of around 10 new scalar dimensions, each one corresponding to a nodal value on the coarse mesh. But the increase in the complexity of the model is not too penalizing. Indeed, as we saw in the previous section for the function $W(k)$ associated to the thermal conductivity, each equation describing the evolution of a modal function $V^i(p)$ doesn't involve any derivative with respect to time or space. All are algebraic equations and so they are very light from a computational viewpoint.

Chapter 4

Numerical experiments

4.1 Linear model

The equation to be solved is the transient heat transfer equation with time-dependent source term. First, we are interested in the 3D model involving the thermal conductivity k as a coordinate of the model. Thus the general solution is sought under the form,

$$u\left(x,t,k\right) \approx \sum_{i=1}^{N} T_i(t) \cdot X_i(x) \cdot K_i(k) \tag{4.1}$$

In our experiment the domain of variabilities are : $\Omega_t = [0,1]$, $\Omega_x = [0,1]$, $\Omega_k = [0.1,1]$. The approximation of the modal functions $X_i(x)$ is performed by using 1D linear finite element shape functions on a uniform mesh consisting of 100 nodes. Finite differences are used for the functions $T_i(t)$ and $K_i(k)$, and 100 nodes are considered for each dimension. The source term is $f(x,t) = \sin(\pi x) \cdot exp(-10t)$ and the initial and boundary conditions are defined as follows,

$$\begin{cases} u_0(x) = 5x \cdot (1-x) \\ u\left(t,0\right) = u\left(t,1\right) = 0 \end{cases} \tag{4.2}$$

Since we don't know a priori the number of terms required in the finite sum, that is to say the number of modes, that have to be computed, we solve the same problem using finite differences for several values of the thermal conductivity k. The solutions we obtain are used as references and compared to the PGD solution. We define the error at each time step as,

$$Error(t = t_i) = \frac{\int_{\Omega_x} \left(U_{FD}^{k0}(t_i,x) - U_{PGD}(t_i,x,k0)\right)^2 dx}{\int_{\Omega_x} \left(U_{FD}^{k0}(t_i,x)\right)^2 dx} \tag{4.3}$$

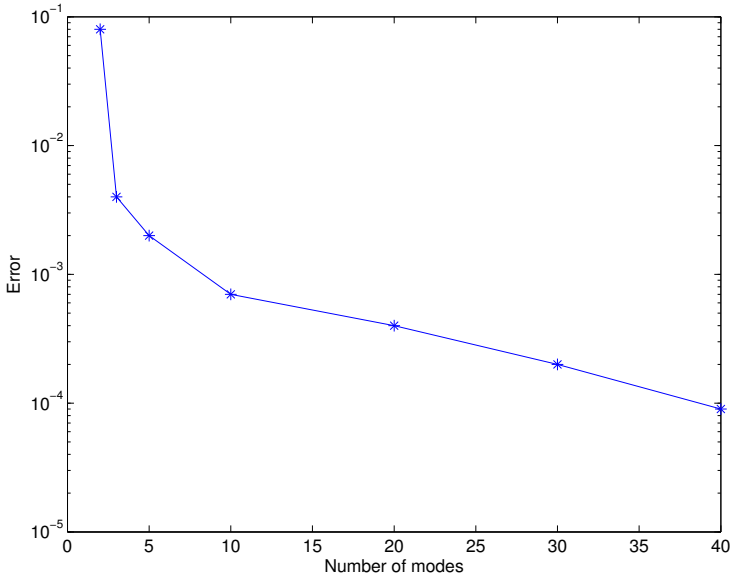


Figure 4.1: Convergence of the PGD solution

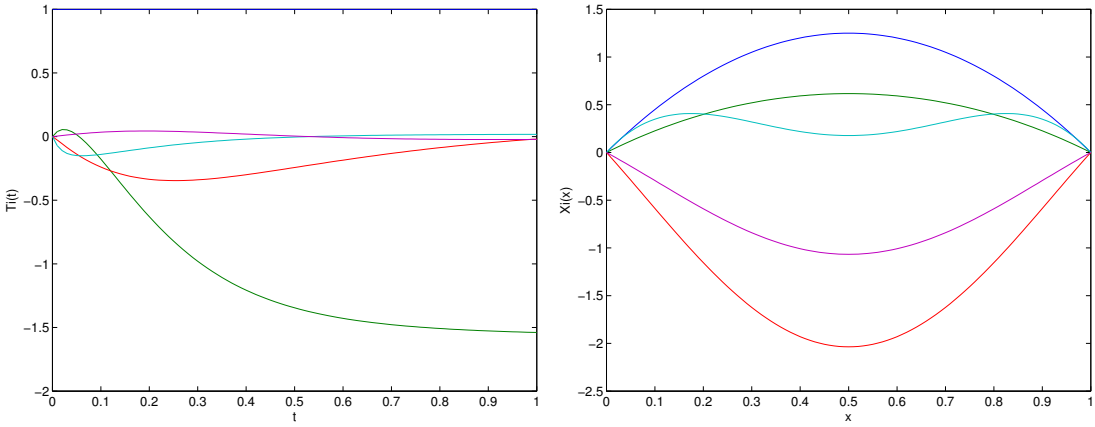
Mode	$\|T_i(t) \cdot X_i(x) \cdot K_i(k)\|_2$
1	$7,50.10^{-1}$
2	$1,73.10^{-1}$
3	$2,51.10^{-3}$
4	$2,27.10^{-6}$
5	$4,80.10^{-7}$

Table 4.1: Comparison of the norm of the modes

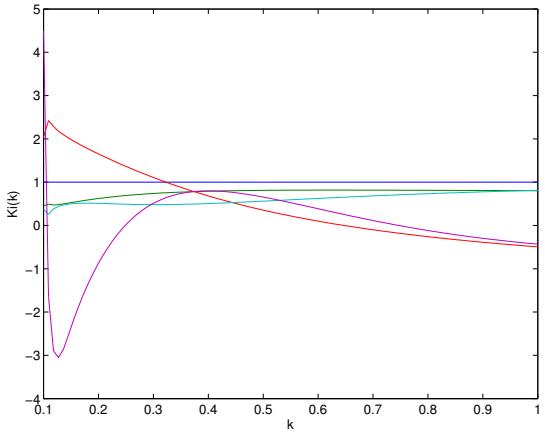
Then using this relation it's possible to get the evolution of the error with respect to the number of terms in the finite sum decomposition. On figure 4.1, we illustrate the convergence of the PGD method for a conductivity k=0.2. As we can observe, the required number of modes is relatively small: with only 10 modes the error reduces to 5.10^{-4}. The notion of "model reduction" makes perfect sense here. The complexity of the model has been reduced. Indeed on one hand ,the classical mesh based method, finite differences in this case, involves 10^4 degrees of freedom and is only valid for one value of k. On the other hand the PGD solution with 10 modes is valid for any thermal conductivity between 0.1 and 1 and exhibits a complexity of 3.10^3.

As we can see on figure 4.2, the norm of the modes decreases quickly. For the sake of clarity, only the five first modes are represented here but we can see how fast it converges. Indeed, the influence of mode #5 on the global solution is almost negligible compared to modes #1 and #2. In table 4.1, we detail the L_2 norm of the first modes and we can notice that the difference between the first and the fifth is more than 6 orders of magnitude.

The other very interesting characteristic of this 3D PGD solution is that it entails

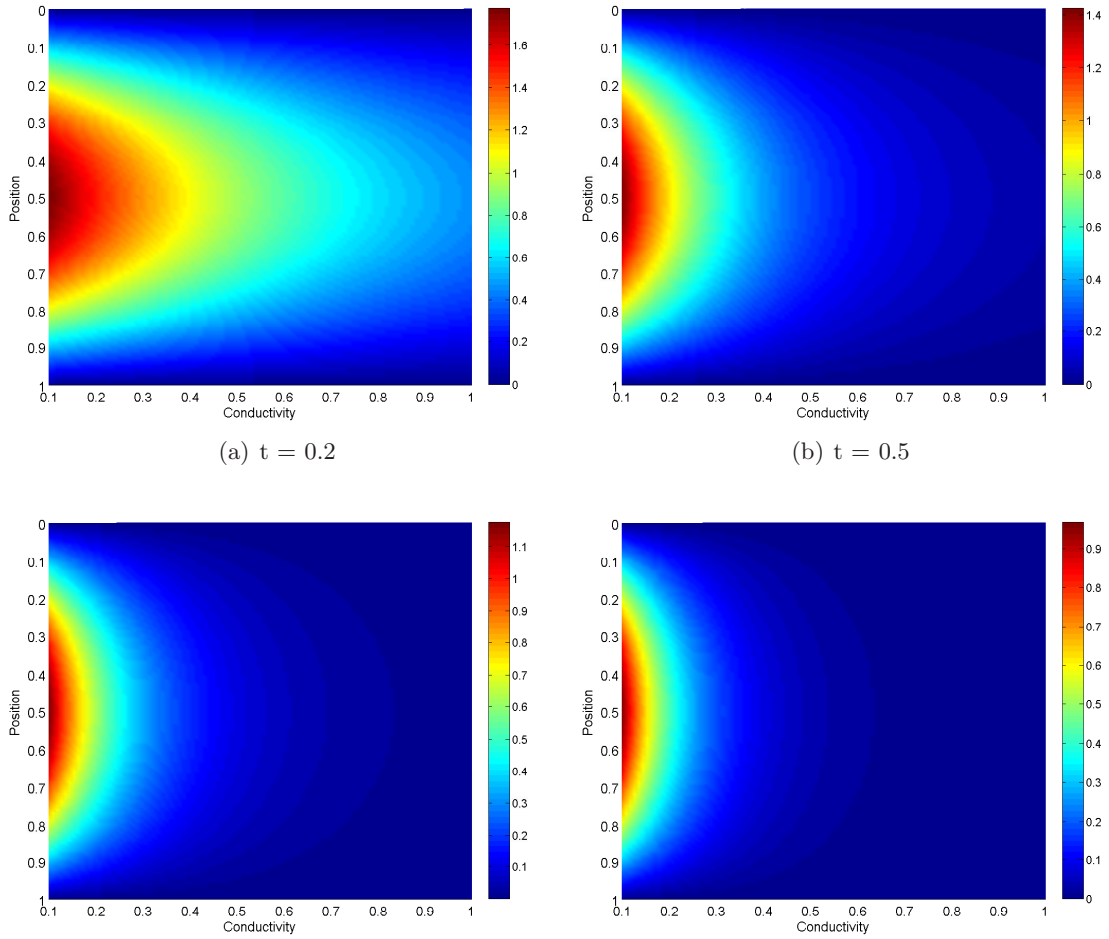


(a) $T_i(t)$ (b) $X_i(x)$



(c) $K_i(k)$

Figure 4.2: Five first modes of the decomposition



(a) t = 0.2

(b) t = 0.5

(c) t = 0.7

(d) t = 0.9

Figure 4.3: Snapshots of the temperature field vs the thermal conductivity

the possibility to represent the field of temperature with respect to the conductivity k. This is important if we want to optimize the value of the conductivity to be used or if we want to do some inverse analysis. Let's imagine that we don't know the conductivity of the material but we have access to the temperature at some points (actually one point is enough if it is not on the boundary) of our 1D domain (a bar for instance). Then our solution constitutes somekind of numerical abaqus in which it's possible to find the conductivity according to the experimental values we have. Thus, crossing the value of the temperature at a point for different times and comparing those datas to several snapshots of our numerical model, allows us to determine accurately the conductivity of the material, figure 4.3.

4.2 Nonlinear model

The case we considered in the previous section is quite academic, and is not really representative of the majority of the problems we encounter in an industrial framework. Actually, most of the time we need to take into account the evolution of the thermal conductivity with respect to the temperature. Indeed for a solid material the thermal conductivity is given by the relation,

$$k = \lambda_0 \left(1 + a\theta\right) \tag{4.4}$$

where λ_0 is the conductivity for a temperature of 0K, a is a specific coefficient for the material and θ is the temperature in Kelvin. Thus, as presented in chapter 2, we introduce two coefficients a and b, such that the conductivity reads now $k = a + bu$ and we are willing to compute a solution

$$u\left(t, x, a, b\right) \approx \sum_{i=1}^{N} T_i(t) \cdot X_i(x) \cdot A_i(a) \cdot B_i(b) \tag{4.5}$$

The main consequence of this more evolved expression for the conductivity is the introduction of the non-linear term $u\frac{\partial u}{\partial x}$ in the equation. A linearization strategy has to be used. Among the several possibilities, we chose to focus on two different incremental strategies. For more details concerning the associated methodology and algorithms, one can refer to chapter 2. In order to compare and evaluate the efficiency of these two strategies, especially their convergence rate, we study a manufactured problem. The principle is to impose a solution, to replace it in the heat transfer equation and to compute the source term that one should use.

We consider equation 1.1 where,

$$f_{manuf} = 20\pi \cos(2\pi t)(x - x^2) + \Big[20a + \Big(600b(x - x^2) - 100b\Big) \cdot (\sin(2\pi t) + 2)\Big] \cdot (\sin(2\pi t) + 2) \tag{4.6}$$

The exact solution of this problem, depicted on figure 4.4, is

$$u_{exact} = \sin\left((2\pi t) + 2\right) \cdot (10x \cdot (1 - x)) \tag{4.7}$$

The domains of variability we consider for the 4 coordinates are respectively $\Omega_t = [0; 2]$, $\Omega_x = [0; 1]$, $\Omega_a = [0.05; 0.5]$ and $\Omega_b = [0.005; 0.05]$. A finite element approximation is used for space functions $X_i(x)$ with a mesh of 500 nodes. The approximation of the modal functions $T_i(t)$, $A_i(a)$ and $B_i(b)$ is performed by using a finite element discretization with meshes of 1000 nodes for time and 20 nodes for the thermal coefficients. Again, it's important to underline that the PGD allows us to circumvent the

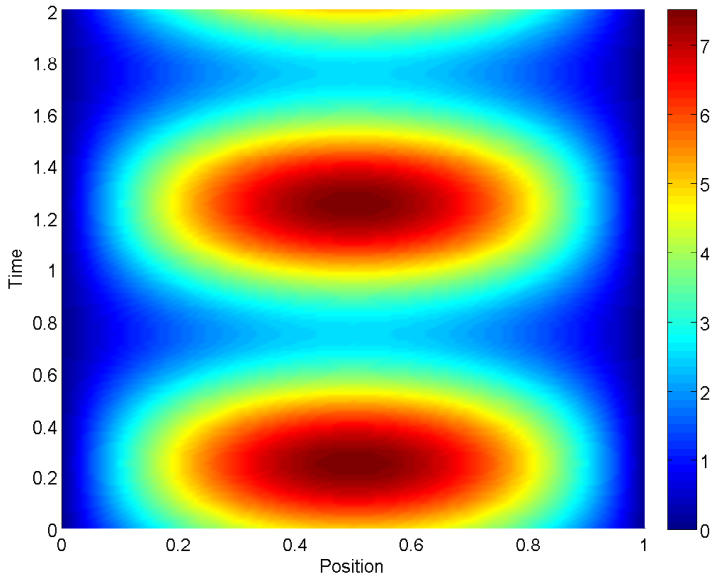


Figure 4.4: Exact solution

curse of dimensionality by alleviating the complexity of the model. Indeed in a classical mesh-based framework, the complexity of this 4D model would be $Ndofs_{classic} = 5.10^2.10^3.\left(2.10^1\right)^2 = 2.10^8$. Considering a pessimistic case for the PGD approach, where 100 modes would be necessary to reach convergence, the complexity reduces nevertheless to $Ndofs_{PGD} = \left(10^3 + 5.10^2 + 4.10^1\right) \cdot 10^2 = 1,54.10^5$. Therefore, the complexity is divided by 10^3 !

Nevertheless, two options can be considered. The first one consists in solving the nonlinear problem from scratch, by imposing only the initial and the boundary conditions from the manufactured problem. The second one, is based on the following idea: since we are able to compute a general solution of the linear problem as we did in the previous section, why not using this latter as a starting point for the solution of the nonlinear problem. This should allow us to reach convergence faster, and it appears to be more interesting from a computational point of view.

Our main conclusion concerning the convergence of the method when solving the linear problem was that about 30 to 40 modes were necessary to reach convergence as illustrated on figure 4.1. Given the fact that we are not solving exactly the same problem, and in order to get rid as much as possible of the error coming from the solution of the linear problem, we compute 50 modes. Then, the principle is to use those 50 modes as initial modes for the calculation of the solution for the nonlinear problem.

On figure 4.5 we compare the convergence of the model for both incremental strategies with a=0.1 and b=0.01. The first approach, for which the non linear term is computed using the solution obtained at the previous iteration, offers a faster convergence (blue curve). The error decreases quickly thanks to the first 5 modes, that are really repre-

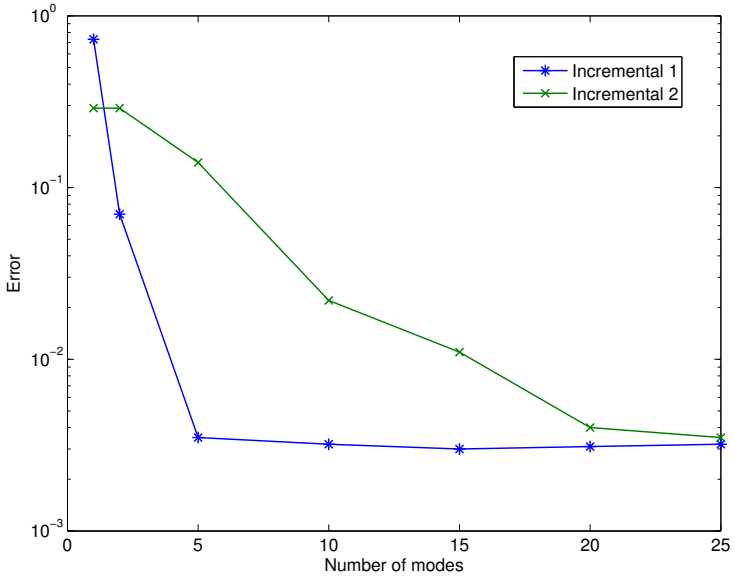


Figure 4.5: Convergence when using mode by mode iteration

sentative of the physics, and then the additional modes bring a small correction to this global tendency: finally with only 10 modes, the solution is converged. On the other hand the second linearization strategy consisting in a progressive update of the conductivity, requires more modes before reaching convergence (green curve). This is actually quite logical because the linearization doesn't affect the equation to be solved in the same way. With the first method, only the right hand side of the equation is modified from one iteration to the other: the source term is updated. But, the second method leads to an increase of the number of terms in the left hand side of the equation because the solution used to compute the conductivity is more and more rich (it contains more and more modes). Each time a new mode is computed a new term is added as follows :

$$\frac{\partial u^{n+1}}{\partial t} - \frac{\partial}{\partial x}\left(a + b \cdot u^n \frac{\partial u^{n+1}}{\partial x}\right) = f \Rightarrow u^{n+1} = u^n + X_{n+1} \cdot T_{n+1} \cdot A_{n+1} \cdot B_{n+1} \tag{4.8a}$$

$$\frac{\partial u^{n+2}}{\partial t} - \frac{\partial}{\partial x}\left(a + b \cdot u^n \frac{\partial u^{n+2}}{\partial x}\right) - \frac{\partial}{\partial x}\left(b \cdot \ X_{n+1} \cdot T_{n+1} \cdot A_{n+1} \cdot B_{n+1} \cdot \frac{\partial u^{n+2}}{\partial x}\right) = f \tag{4.8b}$$

At each iteration we compute only one mode, we update the linearized term with this latter, and we obtain finally a new problem to solve: the number of increments is equal to the number of modes.

Another possibility is to compute a group of modes at each iteration and update the linearized term only after calculating this group of modes. Each iteration of the loop is

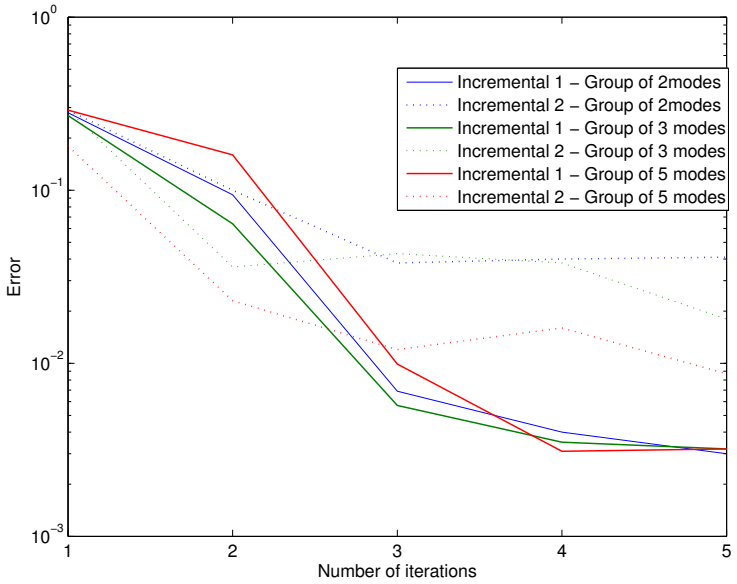


Figure 4.6: Convergence when computing group of modes

then more demanding from a computational point of view but the number of iterations required to reach convergence is logically smaller as we can observe on figure 4.6.

The size of the mesh used for the discretization of the dimension corresponding to coefficient a is fixed during the solution of the linear (20 nodes). Thus, the only grid we can modify here is the one used for the approximation of the modal functions $B_i(b)$. On figure 4.7, we can see that the influence of the mesh is negligible. The evolution of the error is comparable for the 3 cases we studied : 5, 10 and 20 nodes.

The main outcome of the use of the linear solution is the significant reduction of the number of modes to be computed before convergence. With less than 10 modes the solution is converged and the error reduces to 0.003 . This can be explained by the fact that the coefficient b is much smaller than a. Therefore the terms of the linearized equation that are multiplied by this latter have much less influence on the global behaviour of the solution than the ones involving the coefficient a. And finally, the linear solution turns out to be a very good starting point for the nonlinear problem.

4.3 Dynamic Data Driven Inverse Identification: Cauchy problem

We are interested here in solving the transient heat equation on a 1D domain where the boundary conditions are only known on one side. On the other side, we have no information about the condition to be used. In order to solve this kind of problem, we impose the so-called Cauchy boundary condition which consists in fixing both Dirichlet (i.e. temperature) and Neumann (heat flux) conditions on one side of the domain. Our

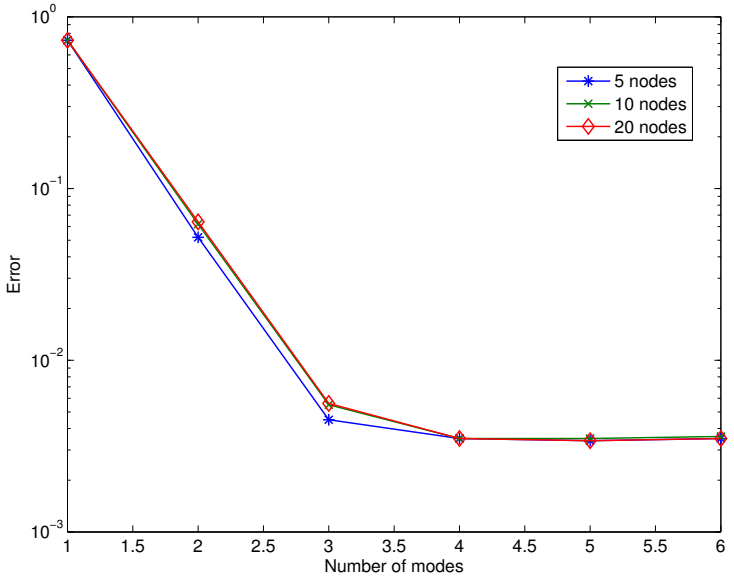


Figure 4.7: Influence of the discretization for coordinate b

objective is to identify the evolution of the temperature at the end of a bar where we cannot put any measurement device because of confinement issues (radioactivity for instance), only by measuring the temperature and the flux at the other end of the bar. The main difficulty occurs when the temperature to be identified has strong and sudden changes in time. *Are we able to describe correctly the amplitude of those changes and the moment it happens ?* This is particularly important to detect any dysfunction in the process we are monitoring. Our ability to react quickly and properly and then prevent any risk, relies on the accuracy of the method when identifying the time evolution of the unknown temperature.

$$\frac{\partial u}{\partial t} - k\frac{\partial^2 u}{x^2} = 0 \qquad in \quad [0; t_{max}] \otimes [0; L] \tag{4.9}$$

$$\begin{cases} u(x,0) = u_0(x) \\ u(0,t) = u_g(t) \\ \frac{\partial u}{\partial x}(0,t) = q_g(t) \end{cases} \tag{4.10}$$

Actually, we want to perform inverse identification. Given a set of temperature values $\left\{u_g^i\right\}_{i=1}^{Nb}$ and fluxes values $\left\{q_g^i\right\}_{i=1}^{Nb}$, coming from measurement devices put at the left end of the bar, we have to identify the time evolution of the temperature at the right end, denoted $\Theta(t)$. The problem is depicted on figure 4.8.

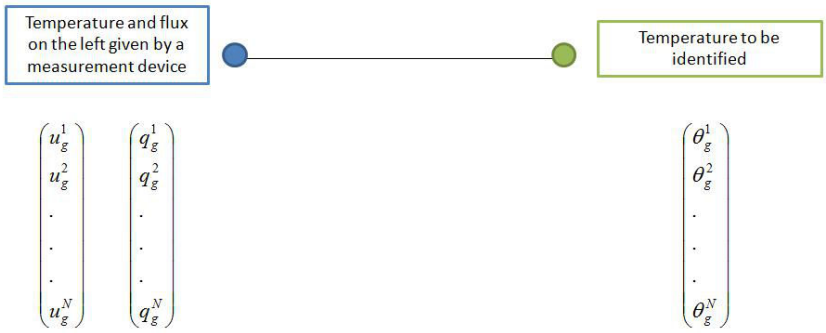


Figure 4.8: Cauchy problem

Let us consider that a measure of the temperature and the flux is performed every Δt seconds. As we explained in the chapter 3, thanks to the Proper Generalized Decomposition approach, we are able to compute a multidimensional solution of this problem, including the initial and the boundary conditions (if constant Dirichlet) in the coordinates of the model. Then, the idea here is to compute this general solution on a time interval of length Δt, parametrizing the initial and the boundary condition with a second mesh much coarser than the one used for the calculation. The first and the last node of this new mesh correspond respectively to the left and the right boundary, and the nodes in between are used to build the initial condition with a classical 1D linear finite element approximation. The solution is then sought under the following separated form,

$$u_{PGD}\left(t, x, p^1, \cdots, p^{Nc}\right) \approx \sum_{i=1}^{Nmodes} T_i(t) \cdot X_i(x) \cdot P_i^1(p) \cdots P_i^{Nc}(p) \tag{4.11}$$

where Nc is the number of nodes in the coarse mesh. Once we have this general solution, we can particularize it using the values provided by the measurements. We suppose here that the initial field of temperature $\left(u_0^2, \ldots, u_0^{Nc-1}\right)$ is known. The only value missing is the one on the right boundary corresponding to the coordinate p^{Nc}, but it can be identified solving a simple problem. Indeed, since we know the value of the flux at time $t_1 = \Delta t$, denoted here q_g^1, the temperature θ we wish to calculate is the solution of the equation,

$$\frac{\partial u_{PGD}}{\partial x}(x = 0, t = \Delta t, p^1 = u_g^1, p^2 = u_0^2, \cdots, p^{Nc-1} = u_0^{Nc-1}, \theta) = q_g^1 \tag{4.12}$$

Finally the problem is treated locally. The unknown temperature corresponds actually to the boundary condition that should be used for the local problem on the sampling interval $[0; \Delta t]$ in order to obtain the correct value for the flux on the left at $t = \Delta t$. More generally, if we denote θ_i the value of θ at time $t_i = i\Delta t$, we have the general algebraic equation,

$$\sum_{j=1}^{Nmodes} \frac{\partial X_j}{\partial x}(0) \cdot T_j(\Delta t) \cdot P_j^1(u_g^i) \cdot P_j^2(u_0^{2i}) \cdots P_j^{Nc-1}(u_0^{(Nc-1)i}) \cdot P_j^{Nc}(\theta_i) = q_g^i \quad (4.13)$$

At this point, it's important to understand that we are making an important assumption concerning the evolution of the temperature in between 2 measurement points. We consider that the temperature at the boundaries is constant. Indeed, in the equation we are solving, the general PGD solution is particularized using the boundary condition at the end of the interval, that is to say at $t = \Delta t$. In order to match the flux measured at time $t = t_i$, denoted q_g^i, we consider a boundary condition on the left which is constant and equal to u_g^i. This inevitably leads to the introduction of an error, as soon as the temperature profile is not constant. Actually we introduce a discontinuity at each time step, because the values we consider for the boundary condition on the left are successively $u_g^1, u_g^2, ...$

In order to set values for parameters $\{p^i\}_{i=2}^{Nc-1}$ we need an initial condition for each local problem. For the first iteration, we use the initial condition for the global problem, but then at the next iterations, a projection is required. Indeed, for the local problem on the time interval $[i\Delta t; (i+1)\Delta t]$ the initial condition to be used is the temperature field computed at the end of interval $[(i-1)\Delta t; i\Delta t]$. But this latter is defined on the fine calculation mesh, while the initial condition is defined on an auxiliary coarse mesh. We are not going to enter into too much details here because the projection step is explained chapter 5. Nevertheless, it's important to mention that this projection leads to the introduction of some error because the initial condition is approximated with a small number of piecewise linear functions. The error can be reduced by increasing the number of parameters for the initial condition, but the number of coordinates in the model and then the cost associated to the computation of the solution becomes more important.

We evaluate the reliability of this method by considering two test cases. For the first one, the temperature at the right boundary is a ramp function: constant at the beginning, followed by a sudden linear increase. In the second test, we consider a discontinuity.

4.3.1 First test: Ramp function

To set up those numerical experiments, we first solve a slightly different problem, using a finite differences approximation. Instead of imposing Cauchy boundary conditions, we impose mixed boundary conditions. On the left side, we choose to impose a natural convection flux that is to say:

$$\frac{\partial u}{\partial x}(0,t) = h\left(T(0,t) - T_{ext}\right) \quad (4.14)$$

where $h = 10$ is the coefficient of convection with the air, and $T_{ext} = 20$ is the ambient temperature. On the right we impose the profile of temperature we want to test, a ramp here. After solving the problem, we have access to the time evolution of the temperature

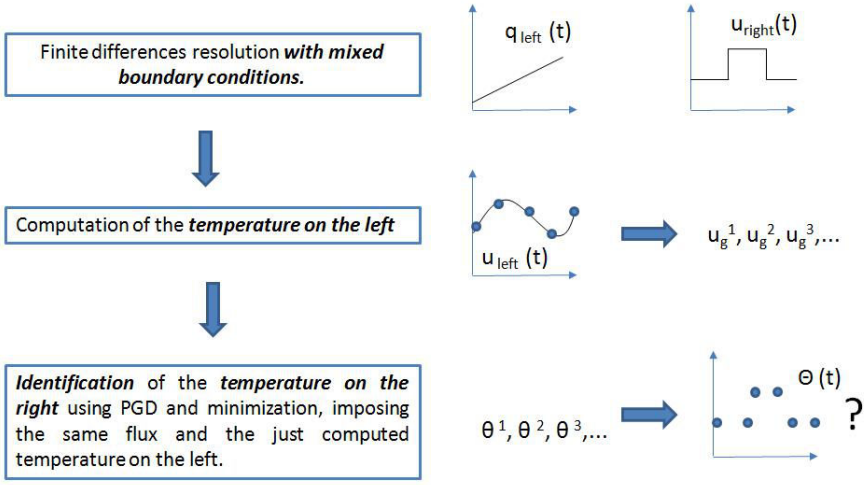


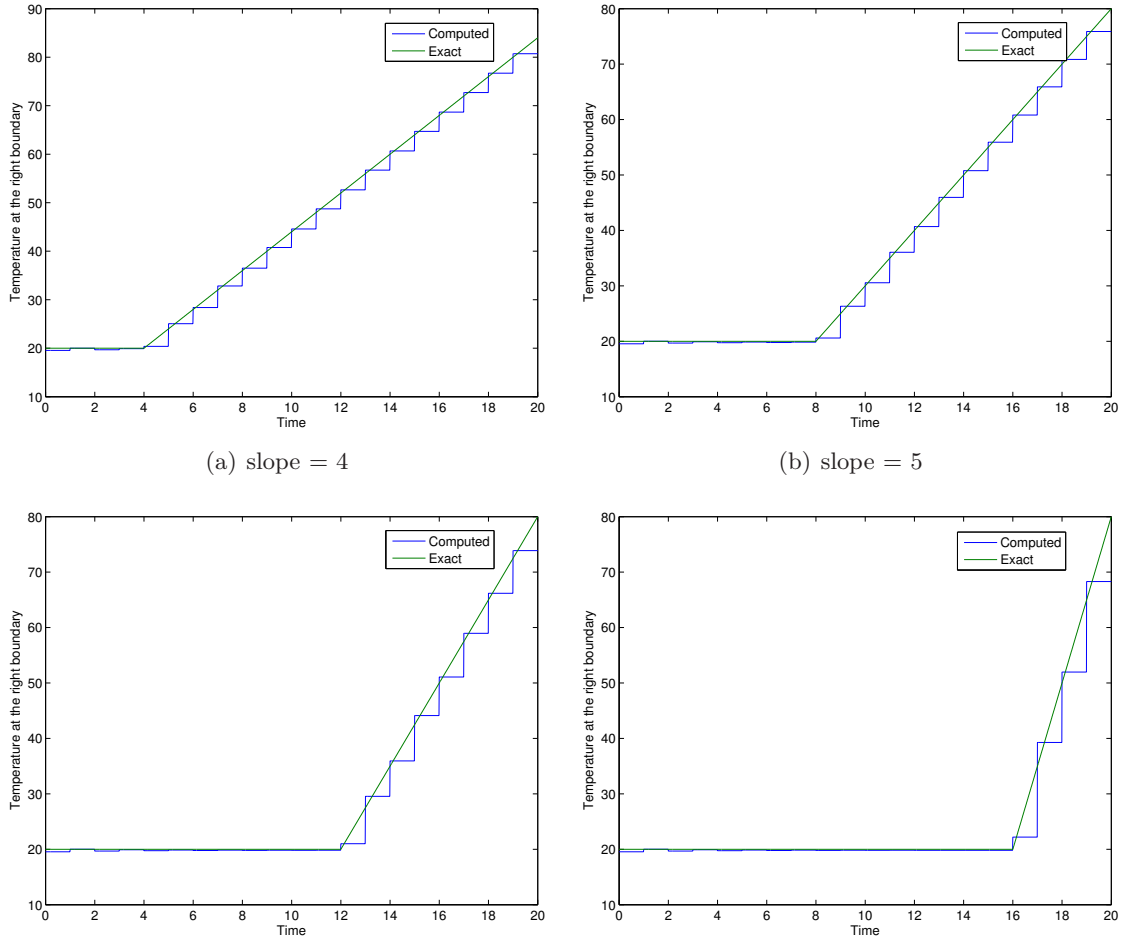
Figure 4.9: Principle of the test

and the flux on the left. A sampling is performed and the resulting set of values is then used as an input data for the Cauchy problem presented previously. If the method works properly, we should be able to recover the same time evolution for the temperature than the one we imposed, figure 4.9.

The first material we consider has a thermal conductivity $k = 0.1$ and the length of the domain is $L = 1$. Then we know a priori that the characteristic time for the diffusion of the heat can be approximated by,

$$\tau_{diff} \approx \frac{L^2}{k} = 10 \tag{4.15}$$

On figure 4.10, we simulate a scenario in which the measurement device provides a value of the temperature and the flux on the left periodically, with a period equal to 1. If the temperature on the right is constant, we are able to identify it with a satisfying accuracy: the delay in the information caused by the small diffusion has no impact. This is what we can observe on the first part of each profile. However, as soon as the unknown temperature increases the important characteristic time associated to the diffusion becomes a break to a correct identification. The more important and sudden the change is, the more difficult the identification becomes. In all cases, the increase of the temperature is underestimated because the information on which this identification is based, that is to say the values of the flux and the temperature on the left, is a delayed image of the real phenomenom. If a dysfunction occurs, we are not able to react immediately: its impact will only be measured a certain time after because of the slow diffusion. That's even more problematic when trying to identify a thermal choc as we demonstrate in the next section.



(a) slope = 4

(b) slope = 5

(c) slope = 7.5

(d) slope = 15

Figure 4.10: Influence of the slope of the ramp

4.3.2 Second test: Discontinuity

When considering a discontinuity in the temperature profile at the right boundary, we want to evaluate the ability of the method we propose to detect both amplitude and location of a thermal choc. Again, we consider a material with a low conductivity, $k = 0.1$, and we have a look at four different situations regarding the location of the discontinuity within the sampling interval, as shown on figure 4.11. The discontinuity occurs exactly at a sampling point (case a) or in between two sampling points (cases b, c and d).

Concerning the amplitude of the identified temperature profile compared to the exact one, we obtain similar results regardless when the thermal choc takes place. We get oscillations whose amplitude decrease in time and finally a convergence towards the exact solution. We can interpret this as a direct consequence of the slow diffusion of the heat in the material we are modeling. Since a certain time elapses between the moment the thermal choc occurs at the right boundary and the moment its impact on the left is actually measured, the convergence of the method is quite slow. Firstly, the value is much higher than expected, then it is compensated with a value much smaller than the exact one, and so on until stabilization. Finally the amplitude of the discontinuity is correctly identified but the time required to achieve this goal is not very satisfying.

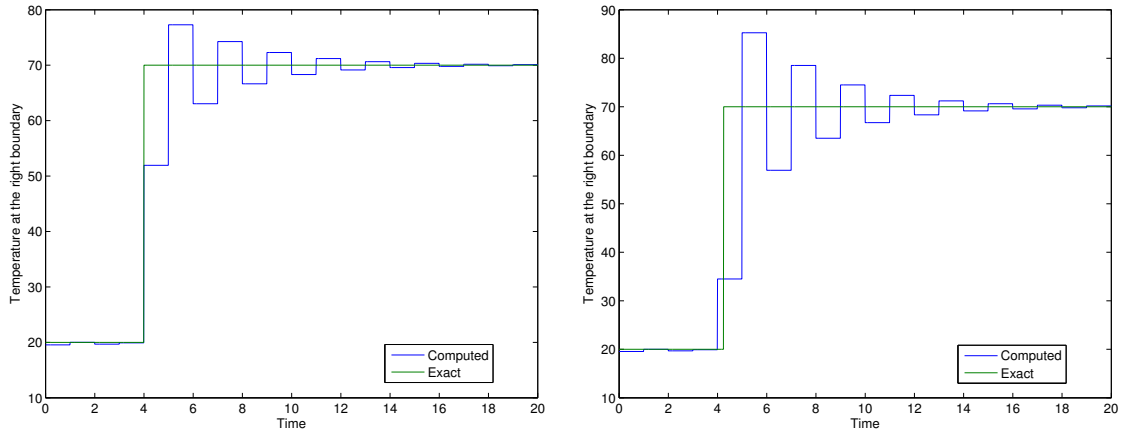
Moreover, the method shows some limits for determining the location of the discontinuity. If this latter occurs exactly at a sampling point, then the position is exactly identified. But as soon as the thermal choc occurs in between two sampling points, difficulties appear.

4.3.3 High conductivity material

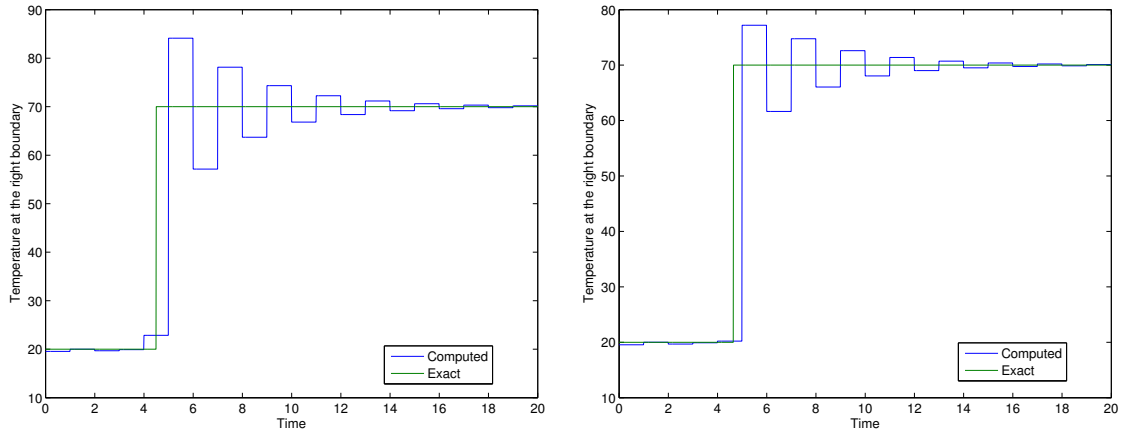
Up to now the results we presented were based on the hypothesis of a relatively low thermal conductivity, which is a critical case for all the reasons we mentioned previously. Now, we wish to evaluate the potential of this method when the conductivity of the material is much higher. Actually this is more representative of the industrial applications, where the material is very often metallic.

On figures 4.12 and 4.13, we can notice a significant improvement in the identification of the temperature at the right boundary. In the case of a discontinuity, both amplitude and location are well determined. We even recover the exact solution for a material of conductivity equal to 1 if the thermal choc occurs exactly at a sampling point. Even if we don't fit exactly a ramp profile because of the hypothesis we made on the evolution of the boundary conditions (i.e. constant during a sampling interval), the results are much more satisfying. This is quite logical because the increase of the thermal conductivity leads to a reduction of the characteristic time associated to the diffusion of the heat : the "transfer of information" from one boundary of the domain to the other is much faster.

Concerning the ramp evolution, we can improve the results by increasing the frequency of our sampling. Indeed, if we have access to the values of the temperature and the flux at the left boundary more often (2, 5 or 10 values/second instead of 1), then a sudden increase of the temperature at the right edge is identified more quickly. Moreover the choice of maintaining constant boundary conditions in between two sampling points is less penalyzing since the sampling interval is smaller. Therefore we obtain a very good

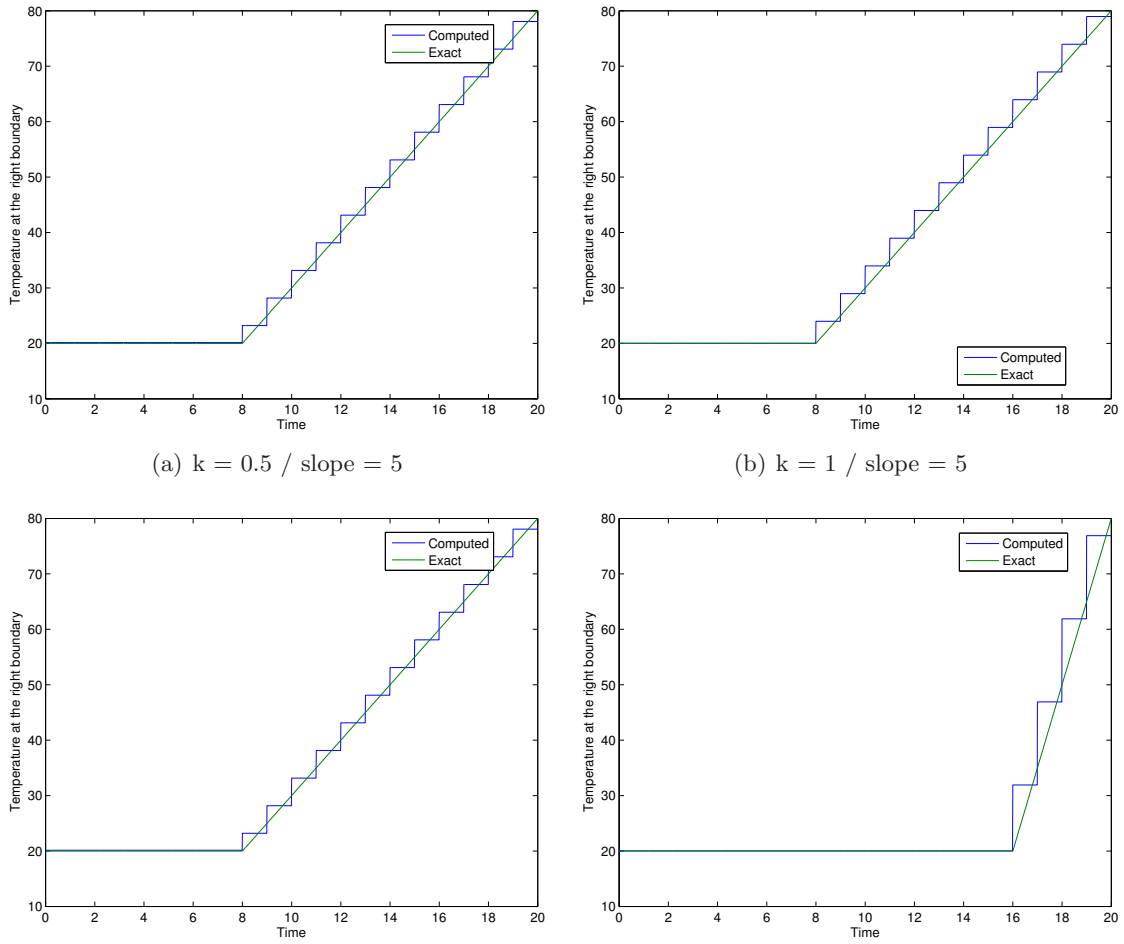


(a) 0 (b) $\frac{T}{4}$



(c) $\frac{T}{2}$ (d) $\frac{3T}{4}$

Figure 4.11: Influence of the position of the discontinuity for low conductivity materials



(a) k = 0.5 / slope = 5

(b) k = 1 / slope = 5

(c) k = 0.5 / slope = 15

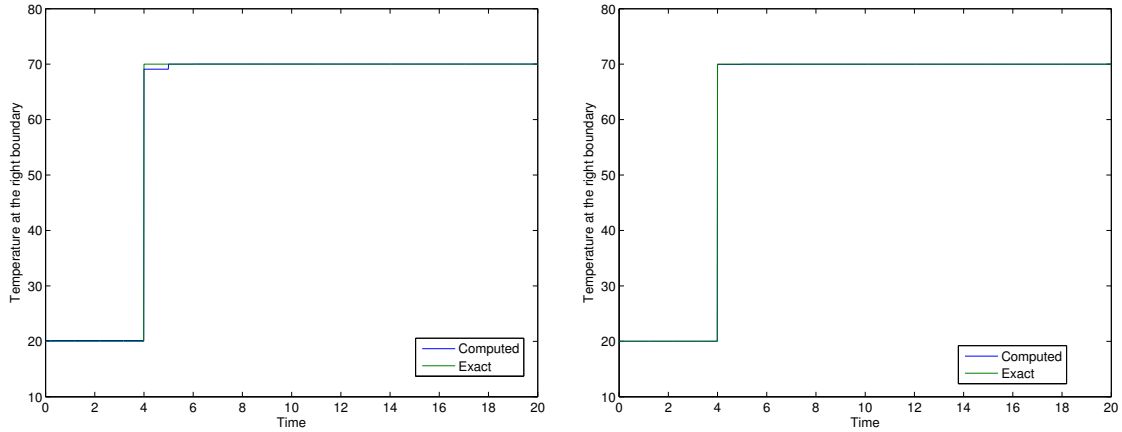
(d) k = 1 / slope = 15

Figure 4.12: Influence of the conductivity for a ramp profile

approximation of a ramp (figure 4.14), but we make use of a numerical trick, considering that the frequency of the measurement can be increased. However, that might not be the case in an industrial framework where the frequency of the sampling can be limited by the measurement device. We propose an alternative in the following section.

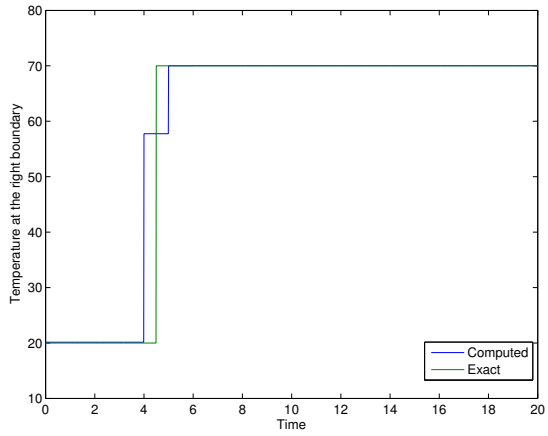
4.3.4 Exact-fitting for ramp evolution

Using the same kind of approach, it's possible to identify exactly the evolution of the unknown temperature if this latter is a ramp. Instead of considering a problem on the interval $[0; \Delta t]$ with constant boundary conditions, we could imagine that the temperature on both sides evolves linearly with time. This is quite a strong assumption, but it entails the ability to describe much more precisely a ramp with a relatively small number of data concerning the temperature and the flux on the left: we don't need to increase the

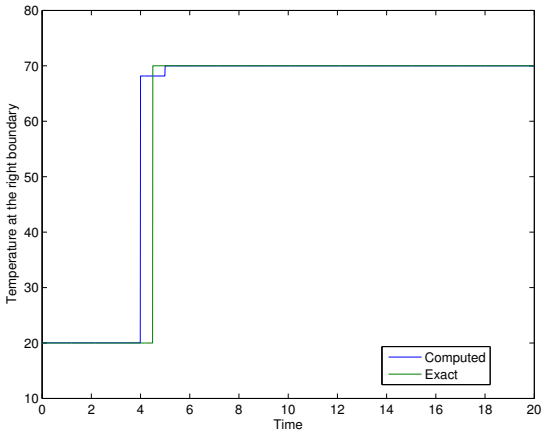


(a) k = 0.5 / 0

(b) k = 1 / 0

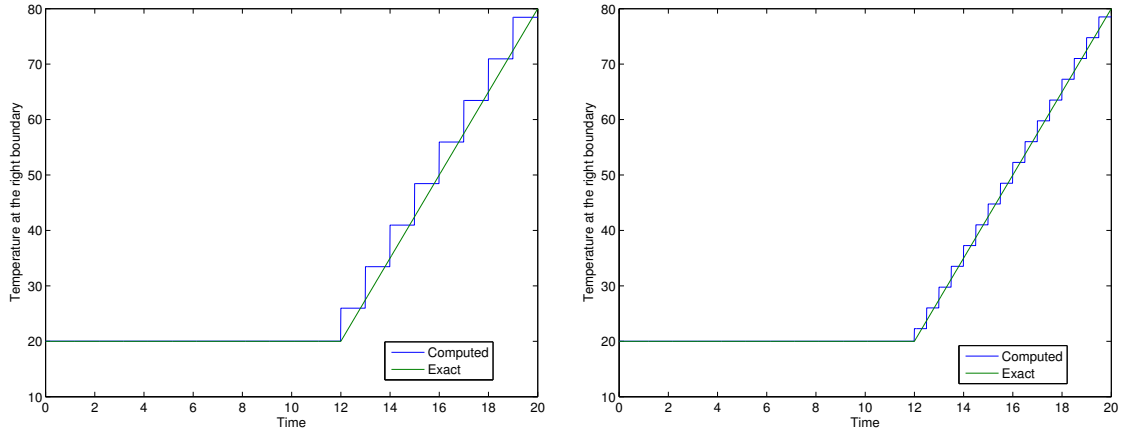


(c) k = 0.5 / $\frac{T}{2}$



(d) k = 1 / $\frac{T}{2}$

Figure 4.13: Influence of the conductivity for a discontinuity



(a) f = 1Hz

(b) f = 2Hz

(c) f = 5Hz

(d) f = 10Hz

Figure 4.14: Influence of the sampling

frequency of the sampling anymore.

The idea is to compute a solution of the problem valid for any initial condition, and also any linear time-dependent boundary condition. Actually, we could even compute a solution valid for any type of time-dependent Dirichlet boundary condition, because the method we are going to expose, can be extended to more complex evolutions. But, we have to keep in mind here that on the interval of calculation $[0; \Delta t]$, we only have 2 values for the temperature and the flux at the left boundary: one at the beginning of the interval and one at the end. Hence the only interpolation that can be done in between those values, is linear.

Just as we parametrize the initial condition by building a coarse mesh in space, why not using the same technique, but for the time discretization ? Indeed, we can consider that the boundary conditions can be approximated as follows,

$$u(0,t) = u_{left}(t) = \sum_{i=1}^{N_{ct}} \phi_c^i(t)\tilde{p}^i \tag{4.16a}$$

$$u(L,t) = u_{right}(t) = \sum_{j=1}^{N_{ct}} \phi_c^j(t)\tilde{p}^j \tag{4.16b}$$

where ϕ_c^i are shape functions defined on a auxiliary coarse mesh, $\tilde{p}^i$ are the corresponding nodal values. Actually, here we only need to consider 2 nodes for the mesh on each boundary and this leads to the introduction of only 2 extra-dimensions compared to the model presented before because the boundary condition at time $t = 0$ is already parametrized with the initial condition, figure 4.15. We use a 1D linear finite element approximation, so the shape functions are the classical delta Kronecker functions. For the sake of simplicity, we consider that the domain of variability for the coordinates parametrizing the boundary conditions is the same than the coordinates for the initial condition. The solution is then searched under the following separated form,

$$u\left(t, x, p^1, \cdots, p^{Nc}, \tilde{p}^1, \tilde{p}^2\right) = \sum_{i=1}^{N} T_i(t) \cdot X_i(x) \cdot P_i^1(p) \cdots P_i^{Nc}(p) \cdot \tilde{P}_i^1(p) \cdot \tilde{P}_i^2(p) \tag{4.17}$$

defined in $\Omega = \Omega_t \otimes \Omega_x \otimes \Omega_p \cdots \otimes \Omega_p = [0; \Delta t] \otimes [0; 1] \otimes [0; 100] \otimes \cdots \otimes [0; 100]$.

On figure 4.16, we can observe that even if the ramp presents an important slope, we are able to identify the temperature profile very precisely. Using the same kind of approach, we could for example try to find an optimal approach for identifying discontinuities. One of the possible options would consist in introducing a discontinuity in the parametrization of the boundary conditions. Indeed, instead of using linear shape functions for the approximation of the boundary condition, we could use rectangular functions. This has not been implemented yet but it could be the subject of future works.

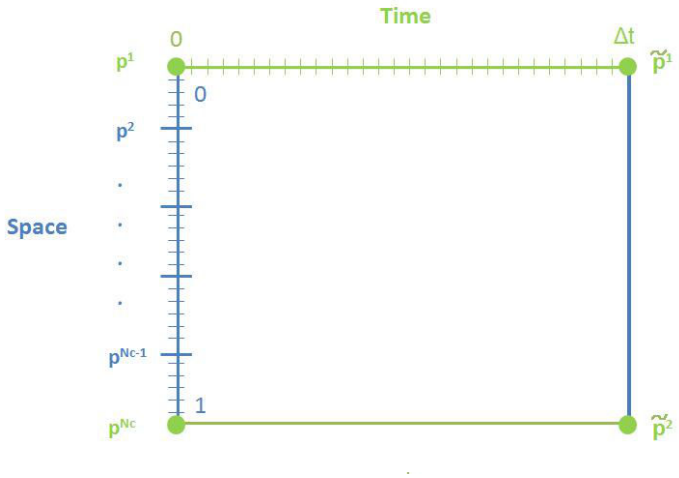
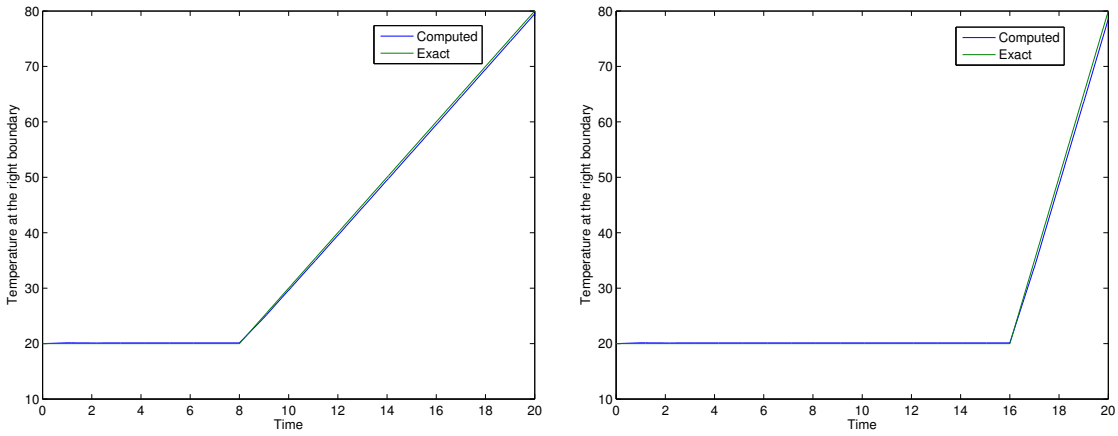


Figure 4.15: Parametrization of the initial and the boundary condition



(a) k = 1 / slope = 5

(b) k = 1 / slope = 15

Figure 4.16: Improvement of the results with a linear time evolution on the boundaries

Part II

A first step towards Time Parallelization

Chapter 5

Time parallelization

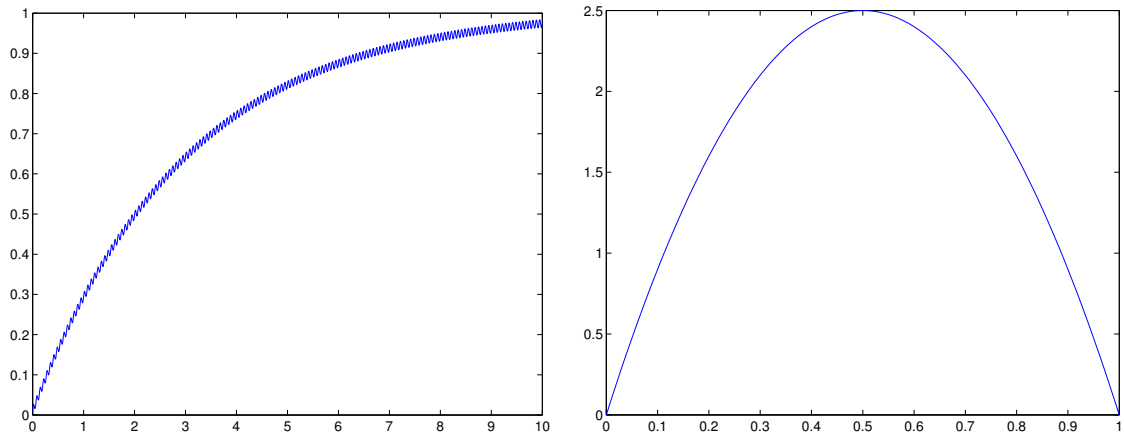
When dealing with transient problems, several time scales corresponding to different physical phenomenon have sometimes to be taken into account. For example, when studying the degradation of plastic materials which can be described by a simple reaction-diffusion equation, the characteristic time associated to the chemical reactions responsible for the degradation is of some microseconds. Then the time increment to be used for the computation has to be equal or smaller. But, in order to determine the aging of the material, that is to say to evaluate the consequences of those reactions on the mechanical and thermal properties of the material, the simulation has to be ran for a long time period, several orders of magnitude bigger. For example, if we consider a time period of a year, this corresponds to about $3 * 10^{13}$ time increments! This problem is then intractable given the computational resources available today. In order to illustrate this notion and point out the need to find alternative solutions to classical mesh-based methods, we consider the heat transfer equation with a source term that exhibits a large time variation coupled to very small oscillations.

$$\frac{\partial u}{\partial t} - k \cdot \frac{\partial^2 u}{\partial x^2} - f = 0$$

$$f(x,t) = 10 \cdot \left(1.01 - \exp\left(\frac{-t}{3}\right) + 0.01 \sin\left(30\pi t\right)\right) \cdot (x \cdot (1-x)) \tag{5.1}$$

The term $\sin(30\pi t)$ forces us to use a very small time step. Indeed if we consider that the discretization step should be of the order of $\frac{T}{10}$ where T is the period of the oscillations, we have to choose $\Delta t = \frac{1}{10}\frac{2\pi}{30\pi} \approx 6.10^{-3}s$. Let's say now that we are interested in the evolution of the temperature field for a period of 60 seconds. Using a classical finite element discretization for space and an implicit Euler method in time, we have to solve at each iteration k, the following linear system,

$$(M + k \cdot \Delta t \cdot K) \cdot U^k = F^k \tag{5.2}$$



(a) Time function (b) Space function

Figure 5.1: Source term

where M and K are respectively the mass matrix and the stiffness matrix. The required number of iteration being 10^5, we cannot afford to use a very fine mesh in space. Moreover, using this incremental approach, we cannot take advantage of parallel processing. The computation has to be done chronologically, because we need the **present solution** to compute the **future**.

What if we could get rid of this chronological imperative ? What if we could parallelize the computation and use multiprocessor platforms ? We describe in this chapter the construction of a numerical model allowing a parallelization in time of the computation.

5.1 Principle of time parallelization

Parallel computing has become a dominant paradigm in computer architecture. Clusters, multi-core processors have been widely developed during the past years. They offer a solution to speed-up the computation by carrying out many calculations simultaneously. Large problems can then be splitted into smaller ones and treated "in parallel". For engineering applications, the common approach is to perform a cutting on the spatial domain, called domain decomposition. Each region is then treated separately using the same algorithm developped for the sequencial approach, but the difficulty is then to ensure a transfer of information between the subdomains, and several continuity issues have to be taken into account on the interfaces.

Here we present another kind of parallelization based on the decomposition of the time interval. The principle is the same than for spatial decomposition except than this time we will solve the equation on the entire spatial domain but for different time intervals. And this will be done in parallel. At first sight this is quite difficult to imagine that we can solve a transient problem by computing what happens at the end without knowing the beginning. Actually, this is made possible by our ability to compute a solution valid

for any initial condition.

Let's consider indeed that we want to compute the temperature field in our 1D domain for a time interval $[0, t_{max}]$. We use a double core processor and we would like to take advantage of this architecture in order to save computation time. The basic idea is then to divide the time interval into two subintervals, not necessarily of same length that we denote $I_1 = [0, t_1]$ and $I_2 = [t_1, t_{max}]$. For each subinterval we compute the multidimensional solution including initial and boundary conditions (constant Dirichlet) and even the thermal conductivity depending whether it's known for the material we are modeling or not. This solution is searched under a separated form using an alternating directions fixed point algorithm, exactly as we detailed in the previous chapter. Finally we obtain the couple of solutions,

$$u_1(t, x, k, p^1, p^2, \ldots, p^{Nc}) \; for \; \left\{ \begin{array}{rcl} t & \in & I_1 \\ x & \in & \Omega_x \\ k & \in & \Omega_k \\ p & \in & \Omega_p \end{array} \right.$$

$$u_2(t, x, k, p^1, p^2, \ldots, p^{Nc}) \; for \; \left\{ \begin{array}{rcl} t & \in & I_2 \\ x & \in & \Omega_x \\ k & \in & \Omega_k \\ p & \in & \Omega_p \end{array} \right.$$

The next step consists in merging those local (from a temporal point of view) solutions into a unique global solution on the entire time interval. The difficulty comes here from the fact that we use two separate discretizations in space. The fine mesh is used for the calculation, whereas the coarse mesh is only necessary to parametrize the initial field of temperature and eventually the boundary conditions. In order to ensure the continuity of the global solution at the interface both local solutions must match. The following equality should be fulfilled,

$$u_1(t = t_1) = u_2(t = t_1) \tag{5.3}$$

The profile of temperature computed at the end of subinterval I_1 has to be used as initial temperature profile for subinterval I_2. In other words we will particularize the general solution u_2 by fixing the value of parameters $p_1, \ldots, p_{Nc}$ for the initial condition according to the temperatures obtained at the end of subinterval I_1. To do so, a projection from the fine to the coarse mesh is required in order to calculate the adequate nodal values to be used for the initial condition of the solution on subinterval I_2.

5.2 A decomposition approach or partial parallelization

Because the source term we are considering evolves in time, and actually that's the case for many industrial applications, we have to solve a different local problem on each

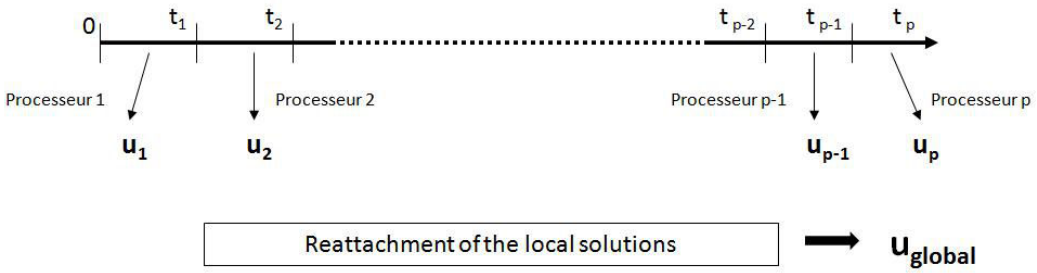


Figure 5.2: General principle of time parallelization

subinterval. Hence as detailed before we have to compute a multidimensional solution $u_i(t, x, k, p^1, \cdots, p^{N_v})$ using PGD, and this may become costly as the number of parameters for the initial field of temperature increases. A more interesting approach from a computational point of view, and then much more suited to real-time control considerations, would be a "partial parallelization". The global problem can be decomposed into an homogeneous problem (**h**) and a particular problem(**pa**), as follows:

$$\frac{\partial u^h}{\partial t} - k\frac{\partial^2 u^h}{\partial x^2} = 0 \qquad u^h(0, x) = u_0(x) \tag{5.4a}$$

$$\frac{\partial u^{pa}}{\partial t} - k\frac{\partial^2 u^{pa}}{\partial x^2} = f \qquad u^{pa}(0, x) = 0 \tag{5.4b}$$

Finally the solution is obtained by summing the homogeneous and the particular solution, that is to say $u = u^h + u^{pa}$. Since the homogeneous problem doesn't involve any time depending source term, we can solve it once and for all on a unique subinterval and use this solution for all the others subdomains. Indeed the only difference existing between the homogeneous problem to be solved on subinterval 1 and the ones on subinterval 2,3... is the initial condition. But, this latter is parametrized, included in the coordinates of the model and a multidimensional solution is computed thanks to the PGD method. We obtain a general homogeneous solution off-line and we particularize it to fit the initial condition at each interface on-line.

The computation of the remaining part of the solution, that is to say the contribution of the particular problem, is parallelized as exposed previously. Each local particular problem is solved concurrently on different processors using classical discretization methods as finite elements or finite differences.

The main advantage of the decomposition strategy is its adaptability or flexibility. It entails the ability to modify the source term on the fly. Since this latter only appears in the particular problem, solved on-line, it can be modified in real-time. This is particularly important for a simulation incorporated in a DDDAS. Indeed, let's imagine that as we are running the simulation, experimental data coming from sensors for instance show that the source term we are using is actually not exactly correct and should be updated in order

Nb of subintervals	Computation time (s)	Particular problem (%)	Projection (%)
2	0.081	17.3	82.7
4	0.12	6.1	93.9
10	0.14	2.2	97.8
25	0.18	0.7	99.3
50	0.24	0.3	99.7
100	0.38	0.15	99.85

Table 5.1: Repartition of the computation time

to fit the real phenomenon. This cannot be done using the first approach that consists in computing general solutions off-line and particularize them on-line because if the source term changes, then the general solution is not valid anymore. But with the decomposition technique, we just have to identify on which processors the particular problem with the new source term has to be solved and compute a new solution on-line. This is very cheap from a computational point of view because we use finite differences in 1D. And the work that has been done off-line, that is to say the homogeneous solution, is still usable. So finally, in order to build a real time simulation, the approach based on a decomposition of the problem is way better.

Concerning the on-line computation time, the cost of the partial parallelization is similar to the "full" parallelization because we are dealing with a 1D problem. If the number of time steps is relatively small (500 is our example) the cost associated to the solution of the particular problem is negligible compared to the time required for the projection as illustrated in table 5.1. But as we increase the number of increments, the solution of the particular problem becomes predominant whereas the cost of the projection remains constant. We expose what are the consequences on the parallel speed-up in the chapter 7.

Apart from the speed considerations, our ability to make the parallelization in time reliable is also closely related to our capacity to "reattach' accurately the local solutions. We present in the next section several alternatives to obtain the solution on the entire time interval from the local solutions.

5.3 From local to global solution

All along this section we will use index **f** to refer to the fine mesh used for the computation and index **c** for the coarse mesh. As we mentioned before, the work to be done to build the global solution from the local parts computed concurrently is a projection from the auxiliary coarse mesh to the fine calculation mesh. We consider here only 2 time subintervals but this can be extended to n subintervals without loss of generality. For the sake of clarity, we call u_f the solution $u_1\,(t = t_1, x, \dots)$ at the end of subinterval I_1 and u_c the solution $u_2\,(t = t_1, x, \dots)$ at the very beginning of subinterval I_2. We would like those two solutions to be equal, or at least as close as possible.

5.3.1 Least squares method

Let's consider a function θ: E $\rightarrow \Re$ where E = span$\{\varphi_c^i\}_{i=1}^{Nc}$ such that,

$$\forall\ u_c\ \in\ E,\ \theta(u_c) = \int_{\Omega_x} (u_c - u_f)^2 dx \tag{5.5}$$

We want to minimize this function in order to enforce the continuity of our global solution. Furthermore replacing u^c and u^f by their finite element approximation in equation 5.5, the minimization problem to be solved writes,

$$\arg\min_{u_c \in E} (\theta(u_c)) = \arg\min_{u_c \in E} \left(\int_{\Omega_x} \left(\sum_{i=1}^{Nc} \varphi_c^i \cdot u_c^i \ -\ \sum_{j=1}^{Nf} \varphi_f^j \cdot u_f^j \right)^2 dx \right) \tag{5.6}$$

where N_c and N_f are the number of nodes respectively in the coarse and in the fine mesh. Assuming that θ is differentiable, minimizing the function is equivalent to solving the problem,

$$\forall\ k\ \in [1; N_c],\ \frac{\partial \theta}{\partial u_c^k} = 2 \int_{\Omega_x} \varphi_c^k \left(\sum_{i=1}^{Nc} \varphi_c^i \cdot u_c^i \ -\ \sum_{j=1}^{Nf} \varphi_f^j \cdot u_f^j \right) dx = 0 \tag{5.7}$$

Finally this equation can be rewritten,

$$\sum_{i=1}^{Nc} \left(\int_{\Omega_x} \varphi_c^k \cdot \varphi_c^i dx \right) u_c^i = \sum_{j=1}^{Nf} \left(\int_{\Omega_x} \varphi_c^k \cdot \varphi_f^j dx \right) u_f^j \tag{5.8}$$

This corresponds the line number k of a linear system of equations whose matrix form reads,

$$M_{cc} \cdot \tilde{u_c} = M_{cf} \cdot \tilde{u_f} \quad \Leftrightarrow \quad \tilde{u_c} = M_{cc}^{-1} \cdot M_{cf} \cdot \tilde{u_f} \tag{5.9}$$

where M_{cc} is the $N_c \times N_c$ mass matrix for the coarse mesh, M_{cf} is a $N_c \times N_f$ matrix corresponding to the coupling between both meshes, $\tilde{u_c}$ and $\tilde{u_f}$ are vectors of nodal values.

$$M_{cc}(i,j) = \textstyle\int_{\Omega_x} \varphi_c^i \cdot \varphi_c^j dx \quad M_{cf}(i,j) = \textstyle\int_{\Omega_x} \varphi_c^i \cdot \varphi_f^j dx \quad \tilde{u_c}(i) = u_c^i \quad \tilde{u_f}(i) = u_f^i$$

The least squares method provides us the best choice, according to the L_2 norm, for the nodal values to be used for the initial condition on the second subinterval. Once we have those values, the method to obtain the global solution is simple. On the first

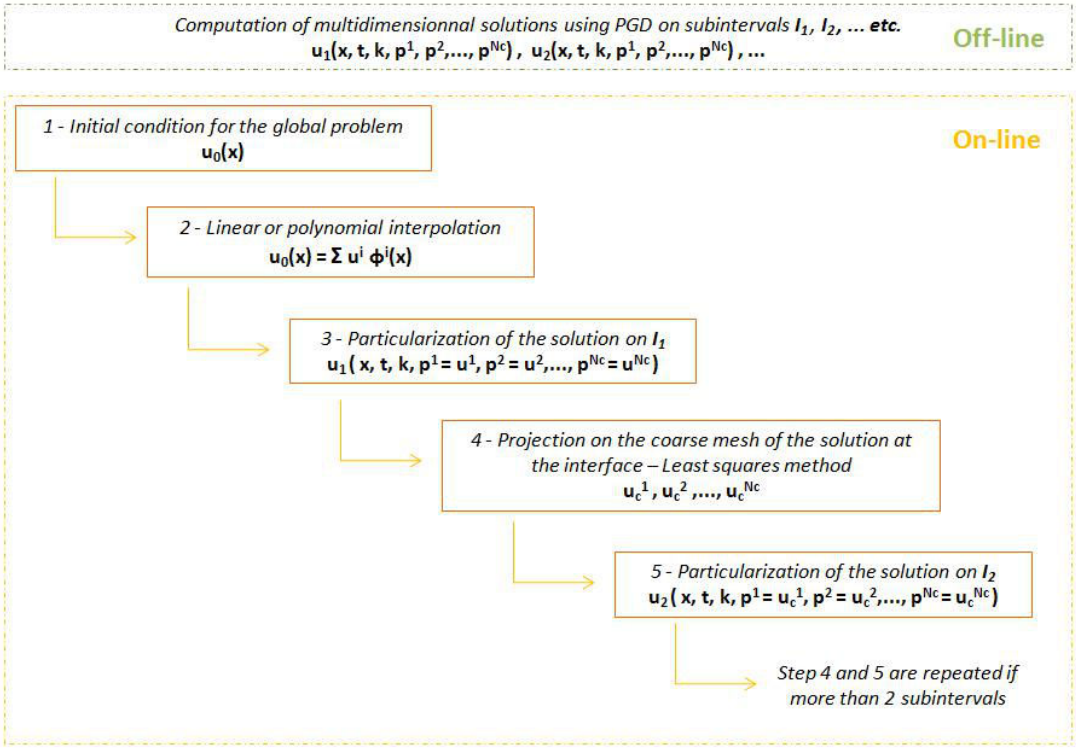


Figure 5.3: Time parallelization scheme

subinterval, I_1, the values of the parameters $\{p^i\}_{i=1}^{Nc}$ are fixed by the initial condition of the global problem. For the second subinterval, the general solution computed off-line is particularized using the values $\{u_c^i\}_{i=1}^{Nc}$ resulting from the projection based on the least squares method. If we use a decomposition approach based on the separation of the homogeneous and the particular problem, these values are used to particularize the solution of the homogeneous problem on subinterval 2 : $u_2^h\left(x,\ t,\ p^1 = u_c^1,\ \cdots,\ p^{Nc} = u_c^{Nc}\right)$.

Nevertheless, the least squares method is a purely mathematical approach. No physical consideration is taken into account. When dealing with thermal problems, one can be interested in ensuring the conservation of the internal energy or the heat flux at the interface. Indeed, we would like our model not to create energy from nothing or in reverse to dissipate energy. The method presented previously must be completed with a mathematical tool able to represent the physics of the phenomenon. We present now two alternatives to the classical least squares method, by introducing extra-constraints in the formulation of the problem. The new minimization problem is then solved using Lagrange multipliers.

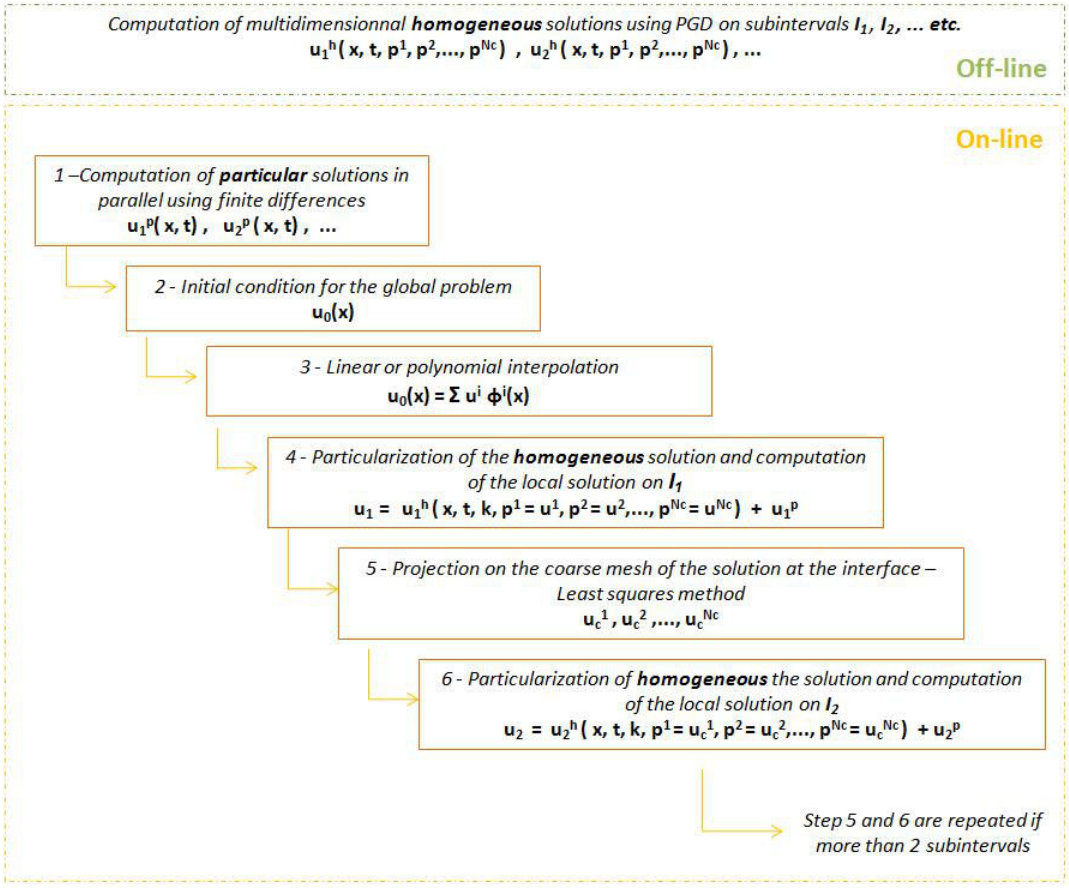


Figure 5.4: Time parallelization scheme for a decomposition approach

5.3.2 Conservation of internal energy

Assuming that the material we are modeling has a constant specific heat c, the variation of internal energy U at the interface writes,

$$\Delta U = c \int_{\Omega_x} (T_{right}(x) - T_{left}(x))dx \tag{5.10}$$

In our case, T_{right} will be replaced by u^c and T_{left} by u^f. We want to ensure the conservation of internal energy at the interface, that is to say $\Delta U = 0$. This latter condition acts as a constraint and a new minimization problem has to be defined. Considering the same vectorial space E and the same function θ already introduced, the problem to be solved is:

$$\begin{aligned} \arg\min_{u_c \in G} (\theta(u_c)) &= \arg\min_{u_c \in G} \int_{\Omega_x} (u_c - u_f)^2 \, dx \\ G = [u_c \in E, \Psi(u_c) &= \int_{\Omega_x} (u_c - u_f) \, dx = 0 \Big] \end{aligned} \tag{5.11}$$

The method of Lagrange multiplier provides a strategy for finding the minima of a function subject to constraints. To do so, we introduce a new function denoted L and defined as follows,

$$\begin{aligned} L(u_c, \lambda) = \theta(u_c) + \lambda\Psi(u_c) = \int_{\Omega_x} \left(\sum_{i=1}^{Nc} \varphi_c^i \cdot u_c^i \; - \; \sum_{j=1}^{Nf} \varphi_f^j \cdot u_f^j \right)^2 dx \\ + \lambda \int_{\Omega_x} \left(\sum_{i=1}^{Nc} \varphi_c^i \cdot u_c^i \; - \; \sum_{j=1}^{Nf} \varphi_f^j \cdot u_f^j \right) dx \end{aligned} \tag{5.12}$$

where λ is a coefficient called Lagrange multiplier. The objective now is to minimize L, that is to say to verify the conditions,

$$\frac{\partial L}{\partial \lambda} = 0 \quad \Leftrightarrow \quad \sum_{i=1}^{Nc} \left(\int_{\Omega_x} \varphi_c^i dx \right) u_c^i = \sum_{j=1}^{Nf} \left(\int_{\Omega_x} \varphi_f^j dx \right) u_f^j \tag{5.13a}$$

$$\frac{\partial L}{\partial u_c^k} = 0 \quad \Leftrightarrow \quad \sum_{i=1}^{Nc} \left(\int_{\Omega_x} \varphi_c^k \varphi_c^i dx \right) u_c^i + \frac{\lambda}{2} \int_{\Omega_x} \varphi_c^k dx = \sum_{j=1}^{Nf} \left(\int_{\Omega_x} \varphi_c^k \varphi_f^j dx \right) u_f^j \tag{5.13b}$$

This corresponds to a linear system of equations whose matrix form reads,

$$M_{cc}^* \cdot \tilde{u}_c^* = M_{cf}^* \cdot \tilde{u}_f \tag{5.14}$$

where $M^*_{cc} = \begin{bmatrix} M_{cc} & v_c^T \\ v_c & 0 \end{bmatrix}$ $M^*_{cf} = \begin{bmatrix} M_{cf} \\ v_f \end{bmatrix}$ $\tilde{u}^*_c = \begin{bmatrix} \tilde{u}_c \\ \frac{\lambda}{2} \end{bmatrix}$ v_c and v_f are 2 vectors defined as,

$$v_c = [\int_{\Omega_x} \varphi_c^1 dx \quad \int_{\Omega_x} \varphi_c^2 dx \ \cdots \ \int_{\Omega_x} \varphi_c^{Nc} dx] \quad v_f = [\int_{\Omega_x} \varphi_f^1 dx \quad \int_{\Omega_x} \varphi_f^2 dx \ \cdots \ \int_{\Omega_x} \varphi_f^{Nc} dx]$$

5.3.3 Conservation of flux

Instead of imposing a constraint on the internal energy we can be interested in ensuring the conservation of the thermal flux when projecting the solution from the fine mesh to the coarse one. We proceed the same way we did previously by expressing mathematically the constraint on the flux. A new minimization problem arises and writes,

$$\begin{aligned} \arg\min_{u_c \in H} (\theta(u_c)) = &\arg\min_{u_c \in H} \int_{\Omega_x} (u_c - u_f)^2 \, dx \\ H = [u_c \in E,\ \Gamma(u_c) = &\int_{\Omega_x} \left(\frac{\partial u_c}{\partial x} - \frac{\partial u_f}{\partial x} \right) dx = 0 \Big] \end{aligned} \tag{5.15}$$

Again we use the Lagrange multiplier method in order to compute this minimum. Let's introduce a new function Q which contains the constraint to be fulfilled.

$$\begin{aligned} Q(u_c, \lambda) = \theta(u_c) + \lambda\Gamma(u_c) = &\int_{\Omega_x} \left(\sum_{i=1}^{Nc} \varphi_c^i \cdot u_c^i \ - \ \sum_{j=1}^{Nf} \varphi_f^j \cdot u_f^j \right)^2 dx \\ &+ \lambda \int_{\Omega_x} \left(\sum_{i=1}^{Nc} \frac{d\varphi_c^i}{dx} \cdot u_c^i \ - \ \sum_{j=1}^{Nf} \frac{d\varphi_f^j}{dx} \cdot u_f^j \right) dx \end{aligned} \tag{5.16}$$

Minimizing Q is equivalent to solving the following system of equations,

$$\sum_{i=1}^{Nc} \left(\varphi_c^i(x_{max}) - \varphi_c^i(x_{min}) \right) u_c^i = \sum_{j=1}^{Nf} \left(\varphi_f^j(x_{max}) - \varphi_f^j(x_{min}) \right) u_f^j \tag{5.17a}$$

$$\sum_{i=1}^{Nc} \left(\int_{\Omega_x} \varphi_c^k \varphi_c^i dx \right) u_c^i + \frac{\lambda}{2} \left(\varphi_c^k(x_{max}) - \varphi_c^k(x_{min}) \right) = \sum_{j=1}^{Nf} \left(\int_{\Omega_x} \varphi_c^k \varphi_f^j dx \right) u_f^j \tag{5.17b}$$

Finally we can express this system into a matrix form as follows,

$$M^{**}_{cc} \cdot \tilde{u}^{**}_c = M^{**}_{cf} \cdot \tilde{u}_f \tag{5.18}$$

where $M_{cc}^{**} = \begin{bmatrix} M_{cc} & w_c^T \\ w_c & 0 \end{bmatrix}$ $M_{cf}^{**} = \begin{bmatrix} M_{cf} \\ w_f \end{bmatrix}$ $\tilde{u}_c^{**} = \begin{bmatrix} \tilde{u}_c \\ \frac{\lambda}{2} \end{bmatrix}$ w_c and w_f are 2 vectors defined as,

$$w_c = \left[(\varphi_c^1(x_{max}) - \varphi_c^1(x_{min})) \quad (\varphi_c^2(x_{max}) - \varphi_c^2(x_{min})) \ \cdots \ \left(\varphi_c^{Nc}(x_{max}) - \varphi_c^{Nc}(x_{min}) \right) \right]$$

$$w_f = \left[\left(\varphi_f^1(x_{max}) - \varphi_f^1(x_{min}) \right) \quad \left(\varphi_f^2(x_{max}) - \varphi_f^2(x_{min}) \right) \ \cdots \ \left(\varphi_f^{Nf}(x_{max}) - \varphi_f^{Nf}(x_{min}) \right) \right]$$

Whatever the method we choose, classical least-squares or least squares under a certain constraint, the projection at the interface of two time subintervals leads inevitably to the introduction of an error. Enforcing the continuity of the solution only at the nodes of the coarse mesh doesn't ensure a perfect continuity. Depending on the interpolation that is used and the number of nodes in the coarse mesh, this error can be controlled. The question arising here concerns the evolution of this error with respect to the number of time subtintervals. Indeed, if this latter increases with the number of subintervals we can wonder whether we will be able to recover a proper accuracy compared to the sequencial approach. This point is discussed in the following chapter and solutions for error control are proposed.

Chapter 6

Control of the error

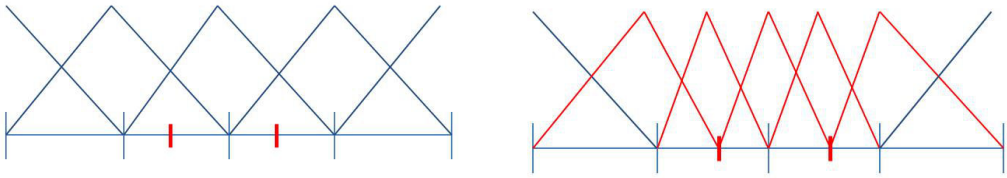
When considering a parallelization in time of the calculation, a new source of error due to the projection at the interface necessary to make the transition from one mesh to the other adds to the classical errors encountered in a numerical model: discretization error, approximations in the numerical scheme... As a result, in order to make this method reliable and accurate, strategies to control this new error, to bound it, have to be identified.

The first idea is to adapt the size of the coarse mesh, only used to parametrize the initial condition, to the profile we want to match. In the regions were the error is higher than a threshold value that has been fixed a priori, the coarse mesh is refined. However we have to keep in mind that by reducing the size of this mesh we increase the number of parameters representing the initial field and then the number of extra coordinates that have to be taken into consideration in the model. Consequently the time necessary to compute the solution increases because the number of modes in the PGD required to reach convergence becomes greater. A refinement strategy can be considered but the gain on the error should not be done at the expense of the computational cost. The parallelization, supposed to allow a speed-up of the calculation, would have the opposite effect.

Nevertheless, refinement is not the only way that can be explored to reduce the error. Because of the dissipative character of the heat transfer equation, which is parabolic, we know that the introduction of an error will affect the solution for a certain time but will be smoothed and will vanish as we move forward. The evaluation of this characteristic time for the dissipation of the error is then crucial and is a key point for building error control strategies. A non dimensional analysis of the equation leads to the following conclusion for the characteristic time denoted Δt:

$$\Delta t_{diff} \propto \frac{\Delta x^2}{k} \tag{6.1}$$

where Δx is the characteristic size of the auxiliary mesh and k is the thermal conductivity. If we want to reduce Δt we have to reduce Δx, but this actually corresponds to a refinement approach with the consequences already mentioned before. Actually, we can use this information on Δt in a different way. Indeed since we know an approximation for



(a) Coarse Mesh (b) Refinement

Figure 6.1: Classical refinement

the characteristic time for the diffusion of the error, the idea is to allow an *overlapping* of both solutions close to the interface. In other words the projection that was previously done at the end of subinterval I_1 will be done earlier, so the error introduced at this time vanishes at the interface. Several methods exist to perform overlapping, depending whether we ensure an optimal "reattachement" at the interface or not.

6.1 Refinement strategies

For the sake of simplicity and in order to compare several approaches for the refinement, we consider that a linear interpolation is used to approximate the initial field of temperature. At each interface a curvilinear profile has to be fitted by a linear piecewise function. Moreover we only treat the case of a 1D domain so we overcome difficulties related to global compatibility for the refined mesh.

Let's say that initially N_c^{init} nodes composed the coarse mesh or equivalently N_c^{init} parameters were used for the initial condition of the problem. We computed "in parallel" the general solution on each time subinterval using the PGD method. After projection at the interface of the subintervals, the error introduced is evaluated and appears to be higher than what we can afford. One option would be then to divide the size of the coarse mesh by two and start the computation again. But, this remeshing approach is not realistic for the parallelization because too expensive from a computational point of view. Refinement offers a much more adequate alternative.

6.1.1 Classical Refinement

The classical refinement strategy consists in adapting the size of the mesh locally. This is an element by element approach. Elements for which the error is higher than a given value are splitted into two smaller elements. Applying this technique to our coarse mesh results in the apparition of new parameters for the initial condition corresponding to the new nodes. We use the notation $\{\tilde{p}^i\}_{j=1}^{N_{ref}}$ where N_{ref} is the number of new nodes.

After determining the required number of extra nodes two options are available. We can start again the computation of the general solution on the subinterval, whose separated representation reads now,

$$u\left(x,t,k,p^1,\dots,p^{N_c^{init}},\tilde{p}^1,\cdots,\tilde{p}^{N_{ref}}\right)=\sum_{j=1}^{N}X_j\cdot T_j\cdot K_j\cdot P_j^1\cdots P_j^{N_c^{init}}\cdot\tilde{P}_j^1\cdots\tilde{P}_j^{N_{ref}} \quad (6.2)$$

Unless the initial coarse mesh is really unadapted, this option is less expensive than remeshing but is far from being the cleverest choice. Indeed before the projection and the refinement, we computed a general solution on the subinterval. Even if it finally appeared to be too poor, in the sense that the number of parameters for the initial condition were not sufficient, it is still a good starting point. All those modes we computed can be used as initial modes for the new problem. For example, let us consider that 100 modes have been necessary to reach convergence for the poor solution. Therefore, the second option consists in enriching those modes by multiplying them by unit functions $\{1_{\tilde{P}^i}\}_{i=1}^{N_{ref}}$. This trick has no impact on them but allows us to write the general solution under the form,

$$u=\underbrace{\sum_{i=1}^{100}X_i\cdot T_i\cdot K_i\cdot P_i^1\cdots P_i^{N_c^{init}}\cdot 1_{\tilde{P}^1}\cdots 1_{\tilde{P}^{N_{ref}}}}_{Poor\ solution}+\underbrace{\sum_{j=1}^{N}X_j\cdot T_j\cdot K_j\cdot P_j^1\cdots P_j^{N_c^{init}}\cdot\tilde{P}_j^1\cdots\tilde{P}_j^{N_{ref}}}_{Enrichment} \quad (6.3)$$

Proceeding so, we ensure a faster convergence of the method compared to the first option. The number of modes that have to be computed after refinement depend on the number of nodes that have been added.

6.1.2 Hierarchical refinement

Instead of reasoning on the elements trying to refine them, we can exploit refinement of basis functions that is to say the shape functions we use for the linear interpolation of the initial condition. A very interesting work has been done by Krysl, Grinspun and Schröder on natural hierarchical refinement. They developed a method called CHARMS [9][10]. This acronym stands for *Conservative Hierarchical Adaptative Refinement Method*.

We are not going to enter into too much details but the main idea is to use nested basis of functions. Let's consider a first basis $\{\varphi_i^1(x)\}$ with span $\aleph^1$ corresponding to the basis of linear shape functions for the coarsest mesh M_1. If we perform a uniform bissection of this latter, we obtain a finer mesh M_2. We can construct another set of functions $\{\varphi_i^2(x)\}$ with span $\aleph^2$.

We have the following inclusion:

$$\aleph^1\subset\aleph^2 \quad (6.4)$$

Hence there exists a set of coefficients $\{\alpha_{ij}\}$ such that any function on level 1 can be exactly represented by a set of functions on level 2,

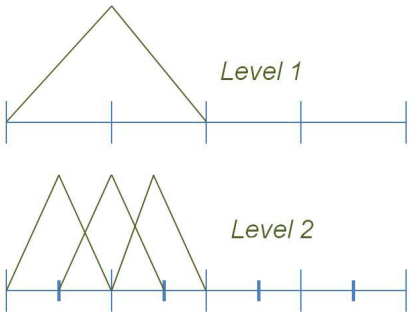
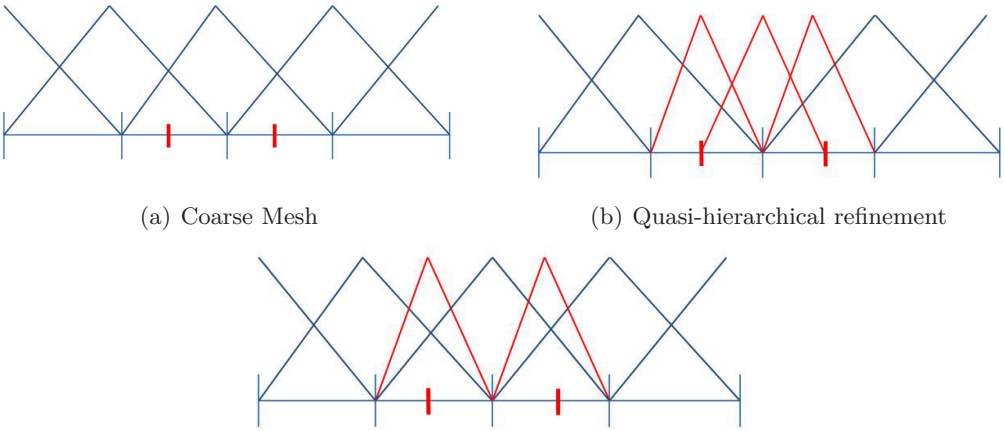


Figure 6.2: A "parent" function and its "childs"



(a) Coarse Mesh

(b) Quasi-hierarchical refinement

(c) Hierarchical refinement

Figure 6.3: Strategies based on the refinement of basis functions

$$\varphi_i^1(x) = \sum_j \alpha_{ij} \cdot \varphi_j^2(x) \tag{6.5}$$

This last relation is called refinement equation. We assume here that the sum in 6.5 has a finite number of terms. Let us introduce at this point some vocabulary widely employed when dealing with hierarchical refinement. We say that $\varphi_j^2(x)$ *is a child of the function* $\varphi_i^1(x)$ or equivalently that $\varphi_i^1(x)$ *is a parent of* $\varphi_j^2(x)$. In our simple 1D case, it's easy to understand that the childs of a function on level 1 are all the functions of level 2 whose support intersects the support of the function on level 1.

We outline here two types of refinement, but many others options can be considered. The first one called **quasi-hierarchical refinement** consists in creating a new set of functions by deactivating a parent function and activating its childs as illustrated in figure 6.3. This set constitutes an adaptative basis for the new approximation, because this

refinement preserves the linear independence requirement. In other words we are sure that proceeding so, the set of functions created are linearly independent. Here the refinement takes the form of a replacement of coarse-level functions φ_i^1 by finer-level functions φ_j^2.

The second alternative is called **hierarchical refinement**. In this approach the refinement is treated no more as a replacement but as the addition of finer-level "detail functions" to an unchanged set of coarse-level functions. By detail function we make a distinction inside the concept of "child functions". We say that the detail set of the coarse-level function φ_i^1 is the set of functions φ_j^2 such that,

$$\begin{cases} \varphi_j^2 \textit{ is a child of } \varphi_i^1 \\ \varphi_j^2(x_i) \;=\; 0 \end{cases}$$

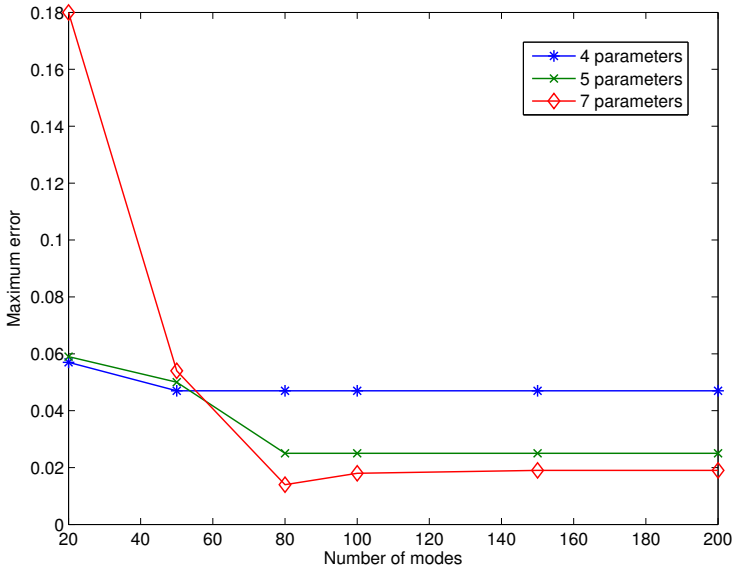
This method also preserves the linear independence requirement. We can notice that for both methods the number of shape functions that have to be modified during the refinement is small compared to the classical refinement approach.

6.1.3 Influence of the number of modes

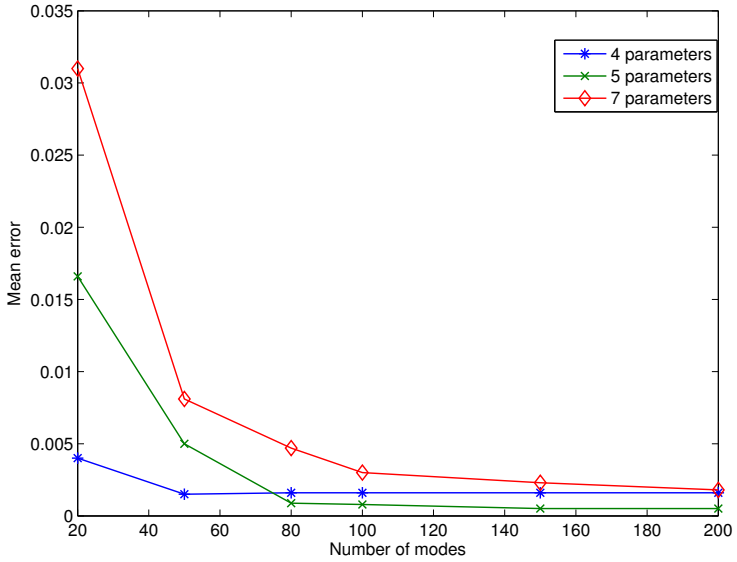
As we mentioned previously, the refinement of the coarse mesh used for the parametrization of the initial condition leads to the introduction of new coordinates in the model. This latter is enriched, and we can ensure a proper accuracy, but the price to be paid is an increase in the number of PGD modes required to reach convergence. Consequently, the associated computational cost becomes higher as we refine the mesh. The questions that arise at this point are : *What is the influence of the number of modes on the error ? What are the consequences on the computational time ? How many subintervals should be considered ?*

In order to answer to those key interrogations, we solve the 1D heat transfer problem with the same source term introduced in 5.1. Two time subintervals are treated separately and a linear interpolation is used for the initial condition. The projection is performed using classical least-squares method. The mean error and the maximun error, relatively to the L_2 norm, are our quantities of interest. The computation times that are given are the ones obtained with a laptop, Core 2 Duo processor and 4GB of RAM.

As we can notice on figure 6.4, the number of modes required to reach convergence increases significantly with the number of parameters used for the initial condition. Whereas 50 modes seem to be sufficient when considering 4 parameters, more than 200 modes have to be calculated in a model with 7 parameters. Moreover, as we increase the complexity of the model by adding new parameters, the computation of a single mode becomes heavier. Indeed, the number of iterations required to reach convergence in the alternating directions fixed point algorithm described in details in the first chapter is naturally higher when the number of dimensions of the model increases. So finally, this leads to a non negligible increase of the computation time.



(a) Error at the interface



(b) Mean error on one subinterval

Figure 6.4: Evolution of the error introduced by the projection

But, the error introduced by the projection has to be controlled and reduced in order to give some sense to the parallelization of the computation. As a consequence, an important number of parameters have to be taken into account. With 7 nodal values for the initial condition the error remains smaller than 2% and the mean error on the time subinterval is less than 0.2%. Compared to the 4 parameters model, the maximum error that is to say the error at the interface of the two the subintervals, is divided by 2.5.

Now considering that in order to ensure a proper accuracy for the "parallelized solution" we have to use at least 7 parameters for the initial field of temperature, the second step consists in determining an adequate number of subintervals. Let's imagine that the computation can be performed on an n-processors platform. *Does that mean that we should split the global problem into n local problems ?* At first sight, we could give the following argument: the higher the number of subintervals, the faster the computation on each subinterval and then the more interesting the parallelization becomes. But this is not necessarily the case because the speed is not the only factor to be evaluated. Actually the answer to this question is closely related to the phenomenom of diffusion we already mentioned before. Since the equation we are solving is parabolic, there exists a characteristic time Δt that completely fixes the minimum size of the subinterval we can afford. Hence using relation 6.1, we propose the following expression for the maximum number of time subintervals that should be considered :

$$N_{max}^{subintervals} \approx \frac{t_{max} \cdot k}{\Delta x^2} \tag{6.6}$$

where t_{max} is the final time, k is the thermal conductivity of the material and Δx is the characteristic size of the mesh built for the initial condition. If the number of subintervals is higher than this value, then the error introduced during the projection doesn't disspate entirely before the end of the subinterval. The consequence is an increase in the maximum error and the mean error because there is an accumulation of the error from one subinterval to the other. The gain in speed offered by this "concurrent" approach becomes then completely useless because the solution we obtain is not workable. This theoritical consideration is logically confirmed by the experiment as we can observe on figure 6.5. The total time here is 5, the thermal conductivity is 0.1 and 7 parameters are used for the initial condition. Thus in theory, the maximum number of time subintervals is $N_{max} \approx 25$.

If we consider less than 30 time subdomains, the error at the interface remains smaller than 2% and the mean error on the entire time domain is around 0.3%. This is similar to what we observed with only 2 subdomains. But the gain on the off-line computation time is significant: whereas about 1000 seconds were necessary with 2 subdomains, the computation is completed in less than 100 seconds for 25 subdomains. For a number of subintervals higher than this critical value, both maximum and mean error start increasing drastically and the model loses its reliability. If we split the global problem into 250 local problems we can reduce the off-line computation time to less than 2 seconds. Indeed, the number of modes to be calculated before convergence becomes small (less than 20) because each time subinterval corresponds to a very short period, and then the variations of the

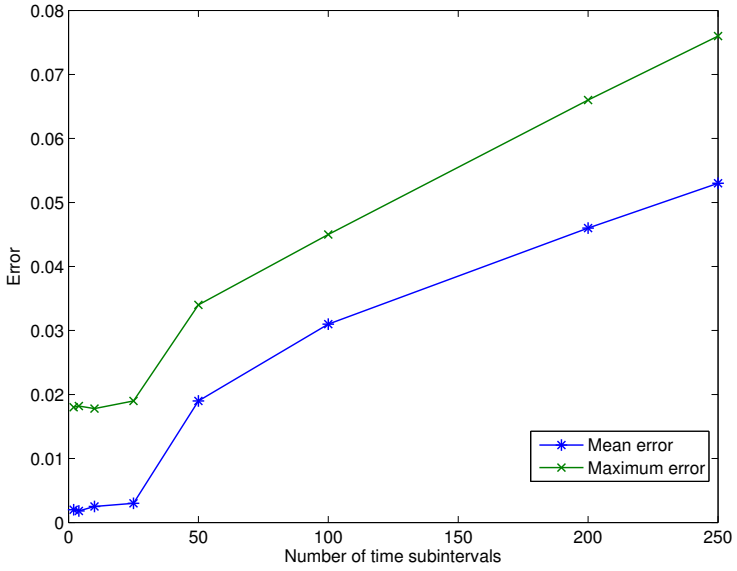


Figure 6.5: Evolution of the error with the number of subdomains

source term during this period are very smooth. But the error reaches more than 7% at the interface, which is clearly not acceptable.

This limited number of subintervals is a brake on the speed up of the computation we would like to provide and then a limit in the interest that time parallelization represents. Moreover, refinement strategies, both classical and hierarchical, are not well suited to a decomposition approach. Indeed, in this latter a multidimensional solution of the homogeneous problem is computed off-line and the particular solutions on each subinterval are computed on-line, simultaneously on different processors. But, if we need to modify the number of parameters for the initial condition because the solution is not accurate enough, then the homogeneous solution isn't valid anymore, and has to be enriched. So finally, all the work has to be done again and depending on the refinement that has to be performed, the computation time can become unacceptable for real-time contol.

Another option must be used in order to circumvent this limitation and the several gaps of the refinement strategy. Overlapping is an interesting alternative that allows us, as we present in the next section, to reduce the number of parameters for the initial field of temperature without loss of accuracy compared to the classical reattachment technique presented previously.

6.2 Overlapping

6.2.1 Principle

Up to now, after computing separately local multidimensional solutions on the subintervals, we reattached them by performing a projection from one mesh to the other at the

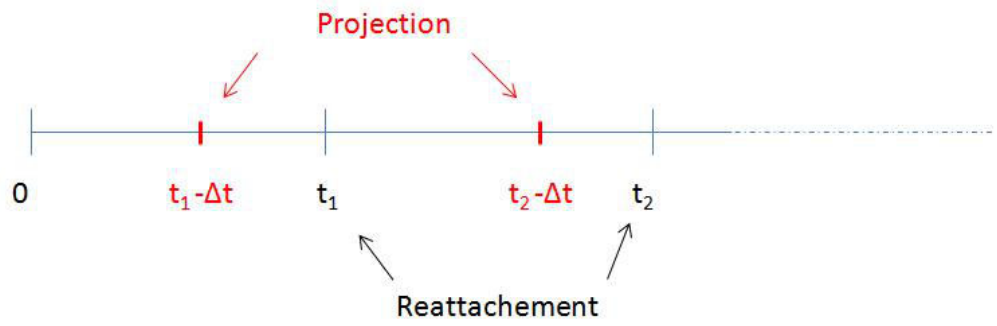


Figure 6.6: Overlapping principle

interfaces. The main drawback of this technique is the introduction of an error that can only be reduced by the use of an important number of parameters and a limited number of time subintervals. The idea now is to take advantage of our a priori knowledge of the behavior of the error. Since we can compute an approximation for the characteristic time of diffusion of the error, a smart choice would be to allow an overlapping of both solutions close to the interfaces. Then the projection should not be performed at the interface anymore but a certain time before. Proceeding so, we know that at the interface the error will be much smaller and then the jump from one solution to the other will be much smoother.

The method we employ consists in computing local solutions independently but now the subintervals are not distinct anymore: they have intersections corresponding to the overlapping. Consequently a part of the local solutions computed upstream is never used in the global solution. For instance on subinterval 1, the projection is performed at $t = t_1 - \Delta t$ but the local solution u_2 on subinterval 2 is actually incorporated in the global solution at $t = t1$. Therefore, this method seems to be less efficient in the sense that the computed information is not entirely used. But actually it proves to be much more powerful than the classical approach.

It opens a interesting perspective: we can avoid the use of an important number of parameters for the initial condition because the overlapping acts as a smoother. Thus the amount of error introduced during the projection has less impact on the global solution.

On figure 6.7 we can observe an interesting but quite logical characteristic of overlapping. Actually there exists a critical value for the size of the overlapping interval. Again this is related to the characteristic time Δt_{diff} for the diffusion widely mentioned before. We observe that the overlapping should not span an interval bigger than Δt_{diff}. Beyond this, the error stagnates and the remaining discrepancies, with respect to the exact solution, are not a consequence of the projection but more likely a result of approximations that are done in the PGD method. Using an appropriate overlapping period allows us to reduce very significantly the impact of the projection on the global solution, as shown on figure 6.8. The green curve corresponds to an overlapping of the order of Δt_{diff} and the blue one is obtained for a classical reattachment. The jump in the error at the interface is divided by 10 thanks to overlapping, when the computation time is almost the same.

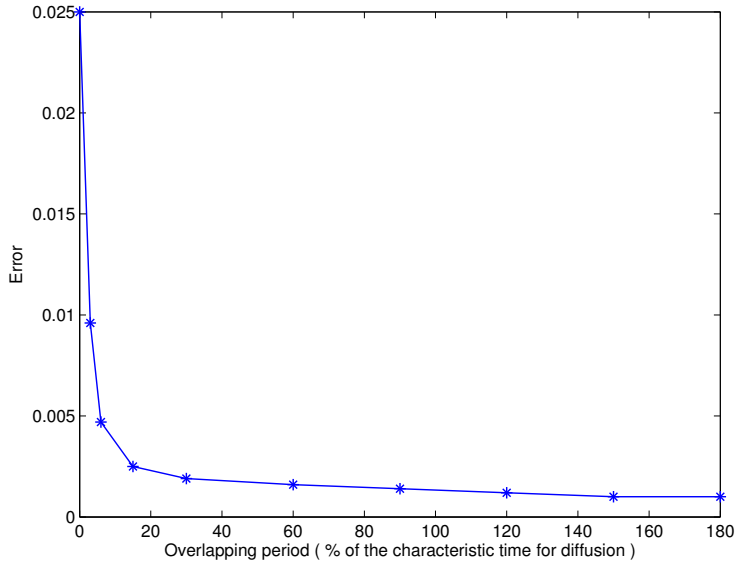


Figure 6.7: Influence of the overlapping on the error at the interface

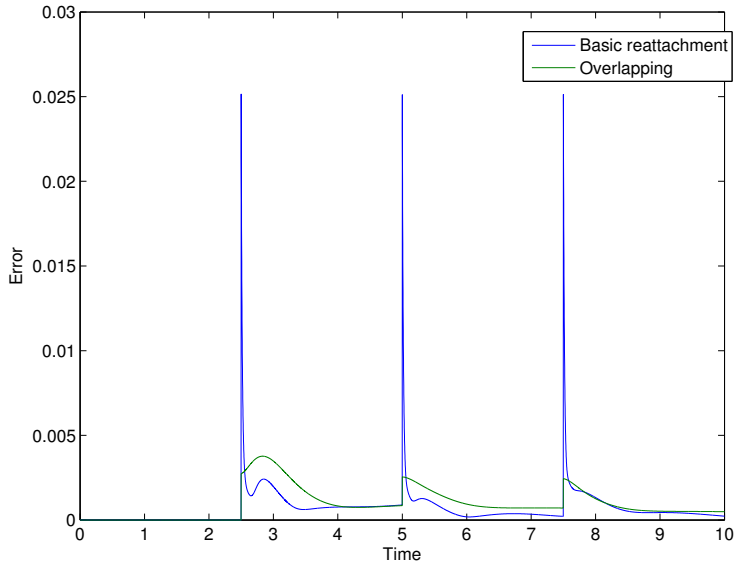


Figure 6.8: Evolution of the error for a 4 subdomains model

Nevertheless, this way of applying overlapping doesn't provide all the solutions to our issues. Indeed, it allows us to regulate the amount of error due to the projection from the fine mesh to the coarse one. This is crucial to ensure an appropriate accuracy. But we still have to tackle the problem concerning the speed of the computation. And the overlapping approach presented up to now is clearly not suited because we cannot increase the number of subdomains. Indeed we know that the optimum size for a subdomain is of the order of Δt_{diff}, but this is also the optimum overlapping ! This means that in order to reduce to its minimum the jump in the error at the interface 1/2 the projection would have to be done at the very beginning of the subinterval 1 ! Actually there exists several other alternatives to make the overlapping more efficient. One of them consists in adding a constraint on the values of the parameters to be used for the initial field of temperature.

6.2.2 Optimal overlapping

Instead of taking only advantage of the diffusive character of the equation, we can try to ensure an optimal reattachment of the solutions at the interface. Previously we tried to find the "best solution" according to the least squares method, at time $t_{projection} = t_{interface} - \Delta t_{diff}$ without taking into account any condition on the reattachment. The new approach that we present here consists in determining which values should be used for the initial condition at $t_{projection}$ in order to ensure an optimal reattachment at $t_{interface}$.

In order to formulate this problem in a proper way, we consider two time subdomains $I_1 = [0; t_1]$ and $I_2 = [t_1 - \Delta t; t_{max}]$. Multidimensional solutions u_1 and u_2 have been computed upstream and have to be reattached. Let us introduce the function L defined as,

$$L(p^1, p^2, \cdots, p^{Nc}) = \int_{\Omega_x} \left(u_1 \left(x, t = t_1 \right) - u_2 \left(x, t = t_1, p^1, p^2, \cdots, p^{Nc} \right) \right)^2 dx \tag{6.7}$$

Our objective is to find the values $\{p_i\}_{i=1}^{Nc}$ minimizing this function. Replacing u_2 by its separated representation, we obtain a non-linear optimization problem :

$$\arg \min_{p^1, \cdots, p^{Nc}} (L) = \int_{\Omega_x} \left(u_1 \left(x, t_1 \right) - \sum_{j=1}^{Nmodes} X_j \left(x \right) \cdot T_j \left(t_1 \right) \cdot P_j^1 \left(p^1 \right) \cdots P_j^{Nc} \left(p^{Nc} \right) \right)^2 dx \tag{6.8}$$

In order to solve this problem we can use for instance the Newton-Raphson (NR) method. Using the notation $p = \left[p^1, p^2, \cdots, p^{Nc} \right]$, the minimization corresponds to the solution of the equation $L\left(p\right) = 0$. The NR approach consists in finding in an iterative way the root p of the latter equation. Starting from a guess p_0, the value at iteration i is computed using the formula,

$$p_i = p_{i-1} - \frac{L\left(p_{i-1}\right)}{\nabla L\left(p_{i-1}\right)} \tag{6.9}$$

Thus, at each iteration the gradient has to be evaluated and inverted. These operations can be quite costly from a computational point of view if the size of p is large. However, p corresponds here to the coordinates used to parametrize the initial condition, and is usually smaller than 10.

The ability of this method to converge quickly towards the minimum value depends on the guess p^0 provided at the beginning. The method might not converge if the guess is very far from the solution. In order to prevent any risk of divergence, one of the possible option is to use the values resulting from the classical least square projection. Those values are the ones we used previously in the "classical overlapping" approach.

The Matlab routine called **fminsearch** is also a very good tool for solving non-linear optimization problem. This latter uses a simplex search method. Unlike Newton-Raphson's method, this is a direct method that doesn't use any numerical gradient. If we are trying to minimize a function involving n parameters, the simplex is characterized by n+1 distinct vectors that are its vertices. Then the simplex is a triangle for n=2, a tetrahedron for n=3... At each step of the search, a new point is generated and the value of the function at this point is compared with the value of the function at the vertices of the simplex. Usually one of the vertices is replaced by the new point and a new simplex is obtained. This step is repeated until the diameter of the simplex is less than a specified tolerance.

Another option is to use a different algorithm to solve this non-linear optimization problem. Among the several existing alternatives, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is particularly well suited to optimization problem with lack of constraints. This is a quasi-Newton's method involving the Hessian matrix. However, this latter is not evaluated directly but an approximation is preferred, and in that sense, it constitutes a generalization of the secant method. Consequently the algorithm is more complex compared to Newton-Raphson's but the gain on the error is significant.

Starting from an initial guess p_0 and an initial approximation for the Hessian H_0 the objective is to determine a direction of descent. The crucial part of the algorithm is the computation of an acceptable step size. Indeed once a search direction has been computed, we have to determine how far one should move along that direction. To do so we chose to implement a backtracking line search. It provides a value for α_k that gives a sufficient decrease in the function L in the sense of the Armijo-Goldstein condition. The corresponding algorithm is detailed in figure 6.10.

A comparison of the error with respect to the size of the overlapping interval for the methods just presented before is depicted on figure 6.11. We can notice that the improvement brought by both optimized strategy is very interesting. Concerning the size of the overlapping interval, the fminsearch routine proves to be the best approach. Thanks to this method, the error at the interface is reduced by 10 compared to the classical overlapping approach when considering an interval of $\Delta t = \frac{\Delta t_{diffusion}}{2}$. Therefore the number of subintervals we can afford is doubled and the accuracy of the reattachment is improved.

Finally, after exposing several techniques to control the error due to the reattachment of the local solutions at the interfaces, the remaining part of our work is devoted to the

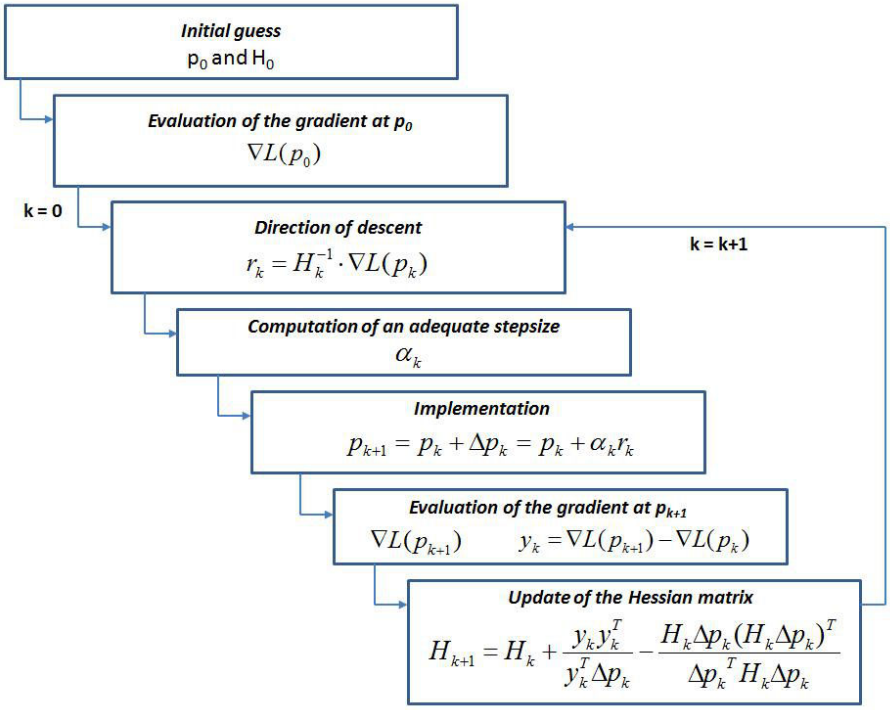


Figure 6.9: BFGS algorithm

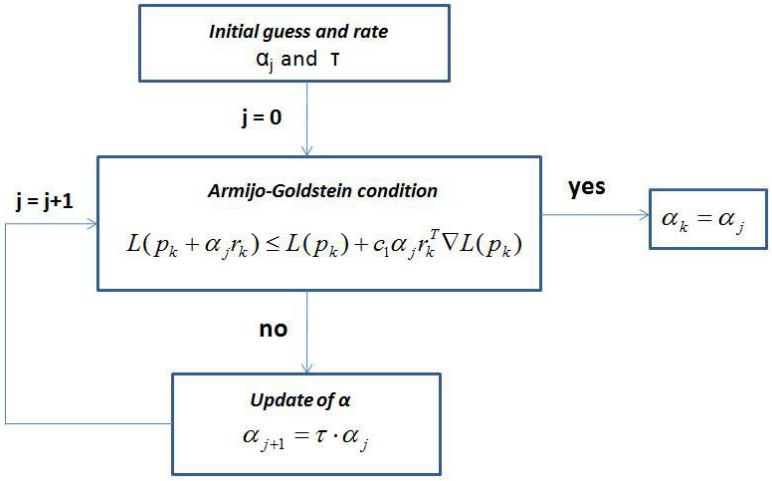


Figure 6.10: Backtracking line search algorithm

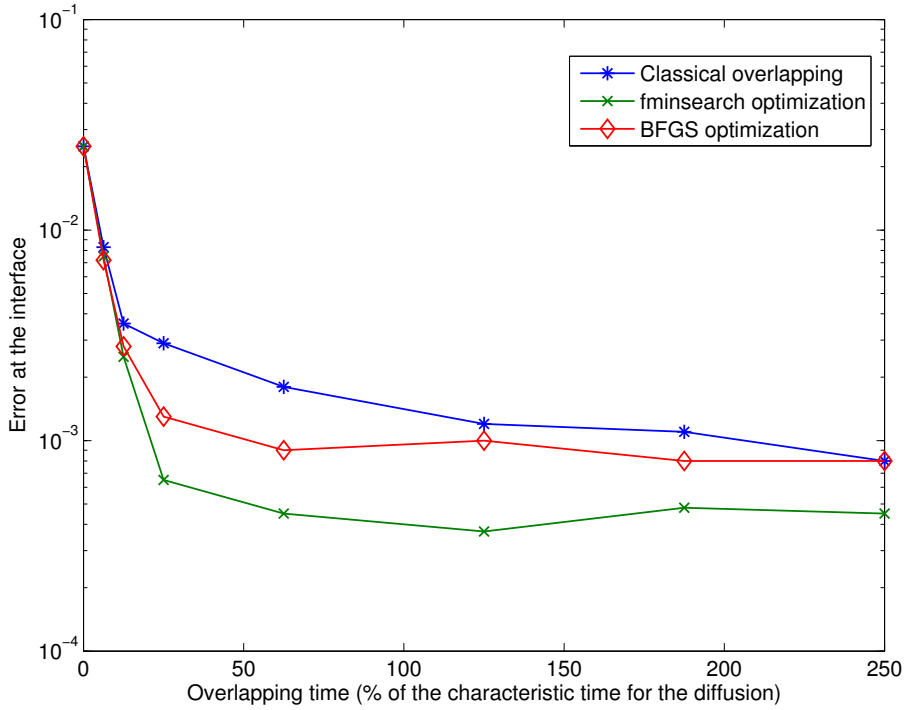


Figure 6.11: Comparison of the error for different overlapping techniques

evaluation of the speed-up supplied by the time parallelization of the computation. In the next section we explain the basic principle of parallel speed-up, the associated laws and we make a comparison between "partial" and "full" parallelization.

Chapter 7

Parallel speed-up

After studying how the accuracy of the solution can be affected by time parallelization and presenting several methods to control and reduce the error, we are now interested in determining how it affects the cost of one computation. The main interest of parallelization is of course to reduce to its minimum, according to the available ressources, the computation time by taking the best advantage of the computer architecture. During this section we will make the assumption that our parallelization is well balanced, that is to say that the workload assigned to each processor of the platform is exactly the same. In order to evaluate the efficiency of the parallelization, we want to calculate its speed-up. It corresponds to the gain in computation time made thanks to the parallelization of the algorithm compared to the sequential approach. In the ideal case, the algorithm can be fully parallelized and the speed-up is defined as,

$$S = \frac{T_s}{T_p} \tag{7.1}$$

where T_s and T_p are respectively the execution time of the **sequential** and **parallel** algorithm. When using n processors the ideal speed-up, also called linear speed-up is $S = n$, which means that the execution time can be divided by n. But this represents the limit and most of the time non-realistic case. Actually, as we will explain in this section, the parallel speed-up is a relative notion in the sense that it strongly depends on the size of the problem we are solving. For a given algorithm, we cannot provide a single value for the speed-up but more likely a range of values. Indeed, the equation 7.1 refers to a sequencial algorithm that can be fully parallelized. If this is not the case, and actually that's what happens for many algorithms, this relation doesn't hold anymore.

7.1 Speed-up laws for partially parallelizable algorithm

7.1.1 Amdahl's law

Let's consider that only a fraction P of the algorithm is parallelizable. In our algorithm, this corresponds to the computation of general solutions (or particular solutions if we

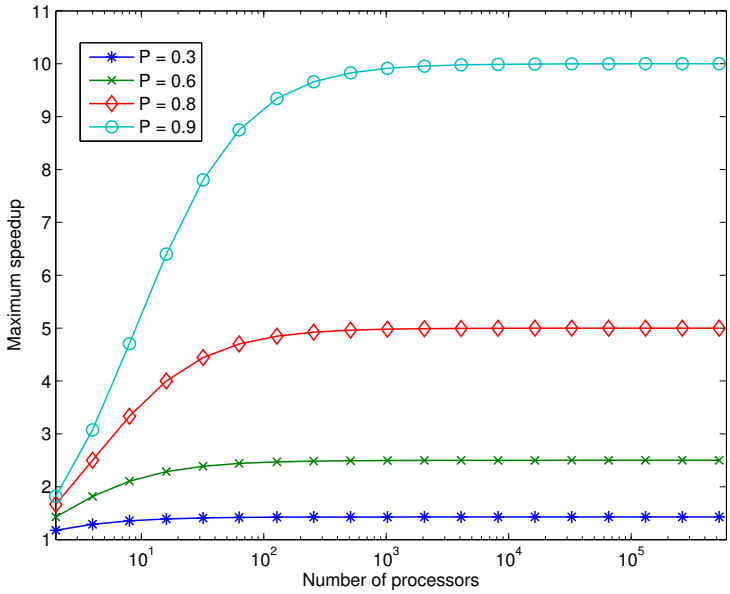


Figure 7.1: Amdahl's law

use a decomposition approach) on each subinterval. In both cases, the projection part of the algorithm involving least-squares minimization, overlapping (classical or optimized) cannot be parallelized and corresponds consequently to (1-P)% of the computation.

The Amdahl's law reads,

$$S(N) = \frac{1}{(1-P) + \frac{P}{N}} \tag{7.2}$$

where N is the number of processors. We can notice that for an infinite number of processors the maximum speed-up tends to $\frac{1}{1-P}$. This means that whatever the number of processors we use, the computation time will never be less than the execution time of the sequential part of the algorithm. But actually this relation doesn't take into account the evolution of the fraction P with respect to the size of the problem. Indeed, in Amdahl's law this fraction is considered as constant and then the maximum reachable speed-up is the same whatever the problem. But we can assume a priori that as the size of the problem increases, the sequential or "non-parallelizable" part of the algorithm decreases. For instance, for time parallelization, the computation time required to solve the particular problem on each subinterval using finite differences is much bigger if we consider very small time steps, whereas the workload associated to the "reattachement" of the local solutions remains constant. This latter only depends on the number of parameters used for the initial condition and of course the number of subintervals to be reattached. Thus, Amdahl's prevision is quite pessimistic and doesn't represent the best way to evaluate the efficiency of the parallelization. Gustafson's law is preferred.

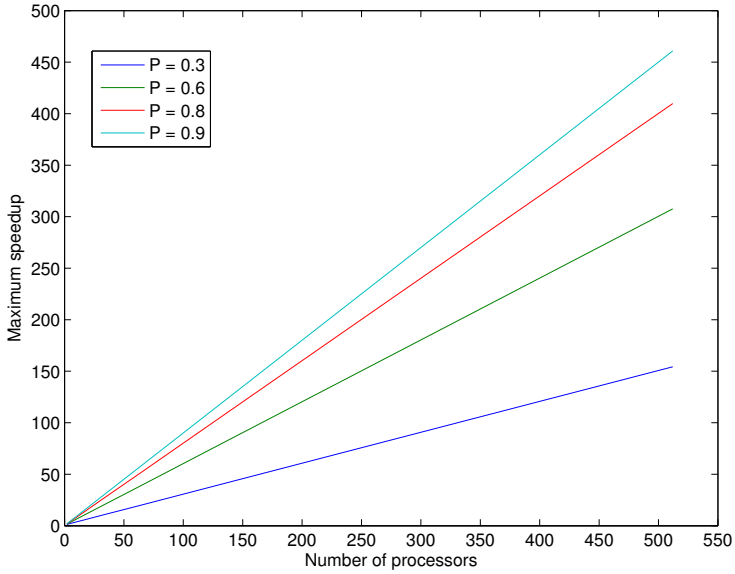


Figure 7.2: Gustafson's law

7.1.2 Gustafson's law

This law is based on the assumption that the parallelizable fraction P of an algorithm is not a fixed quantity but actually depends on the size of the problem. For a given problem, it's then possible to establish a relation between the potentially parallel part and the characteristic size of the problem. The question is : *What should we use as representative size of the problem ?* Actually, there is not a unique answer because it depends on the type of problem we are solving. For a transient problem, the number of time increments is crucial and then is chosen as a measure of the size of the problem. As we explained in our introduction to time parallelization, the superposition of several characteristic times for a multiphysics problem, is a limit to the use of classical incremental methods for real-time simulations: the number of time steps can be of several millions, or billions. Hence, time parallelization becomes really relevant for large problems because it provides an important speed-up of the computation as we demonstrate in the next section. If we denote q the size of the problem, Gustafson's law for the evolution of the speed-up writes,

$$S(N,q) = 1 + (N-1) \cdot P(N,q) \tag{7.3}$$

Then for a given computer architecture, that is to say for a given number of processors, the speed-up is not a fixed value. We obtain a range of values or a domain of variability for the speed-up.

7.2 Time parallelization speed-up

In order to evalutate the speed-up of time parallelization we consider the same transient heat transfer problem with the source term already introduced in equation 5.1. This problem is first solved incrementally with finite differences. In a real-time simulation context this corresponds to an on-line computation time, because we need to provide an initial condition and this can only be done "in live". Regarding the parallel approach we will then be only interested in the on-line part of the job.

7.2.1 Full parallelization approach

If we consider a full parallelization, the on-line part of the work consists in the reattachment of the solutions : projection from the fine mesh to the coarse one, and optimization of the parameters $\{p_i\}_{i=1}^{Nc}$ thanks to an overlapping strategy. Nevertheless according to the definition we gave in the previous section, we cannot use the word speed-up here because the computation times we are comparing are both related to sequential algorithms. Indeed, the reattachment is done subinterval by subinterval in a chronological order. The parallelization only concerns the off-line part of the work, when we compute multidimensional local solutions. Then we will preferably speak of gain in computation time provided by the upstream parallelization.

7.2.2 Decomposition approach

In this approach, an homogeneous multidimensional solution is computed off-line using the PGD method whereas the remaining part of the work is done on-line. Part of the on-line workload can be splitted and distributed to several processors. This parallelizable part of the algorithm corresponds to the calculation of the particular solutions on the time subdomains. The second part of the algorithm, during which the local solutions are reattached, has to be executed sequentially. Before doing any computation, we can make several assumptions on the evolution of the parallelizable fraction of the algorithm with respect to the size of the problem. Indeed, since we use finite differences in the sequential approach, we know that the time necessary to compute the solution of this transient problem scales more or less linearly with the number of time increments: at each time step a linear system of constant size has to be solved. But the increase in the number of time steps has no consequences on the projection part of the work.

For a given number of parameters for the initial field of temperature and a given number of subintervals, the computation time associated to the reattachment is constant with respect to the number of time increments. Then for a relatively small number of time steps, the time required for the projection is much bigger than the finite differences solution, the speed-up is close to 1. But as we increase the number of increments, the number of linear systems to be solved in the incremental approach becomes important and their solution requires more time than the reattachment itself.

On figure 7.3, we represented the evolution of the parallelizable fraction of the algorithm with respect to the size of the problem. As expected, this increases with the size

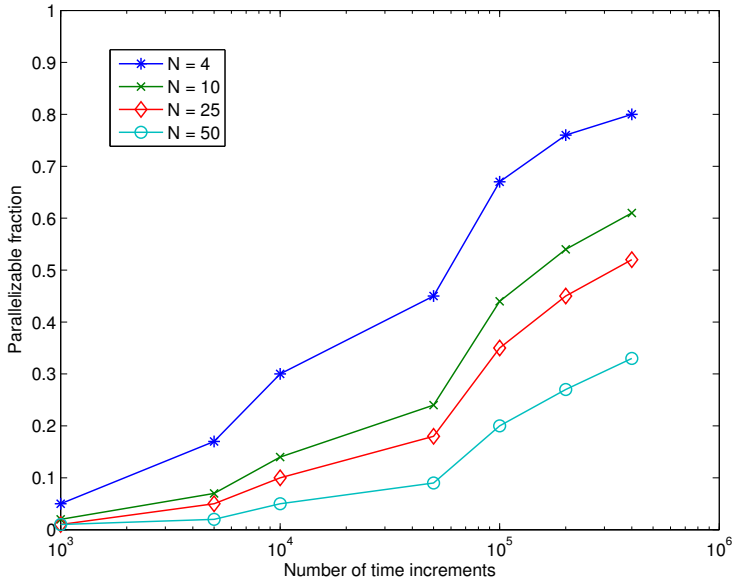


Figure 7.3: Influence of the size of the problem on the parallelizable fraction of the algorithm

of the problem. The second important characteristic we can notice on this graph is the influence of the number of processors. The higher the number of processors we consider, the smaller the parallelizable fraction is. Actually this is due to the fact that the time required to reattach the solutions scales linearly with the number of subintervals : the same operation of projection has to be performed at each interface.

Consequently, the speed-up provided by the partial parallelization of the algorithm is really significant in a real-time framework if the size of the problem is large enough. That is to say if the computation time associated to the solution of the global problem using finite differences is much higher than the time required to perform the projection at the interfaces of the subdomains. In that case, the decomposition approach appears to be a very interesting alternative as we illustrate on figure 7.4. The maximum speed-up we obtain when splitting the domain into 50 parts, is about 10 if the problem is solved on 10^5 time increments and reaches 17 for $4 \cdot 10^5$ increments. This opens new perspectives regarding the solution of problems involving multiple characteristic times. Indeed, the number of time increments is no more a brake or at least becomes less penalizing thanks to time parallelization. Nevertheless, in order to evaluate correctly the potential of this method, more complex problems should be solved.

To conclude this work on time parallelization, we present a more concrete application related to manufacturing processes of composite material. In the next section, we study the technique of ultrasound welding, mostly used in polymer industry and now for thermoplastic composites. After explaining its principle, we expose a simple model for representing the evolution of the temperature field in the welding region.

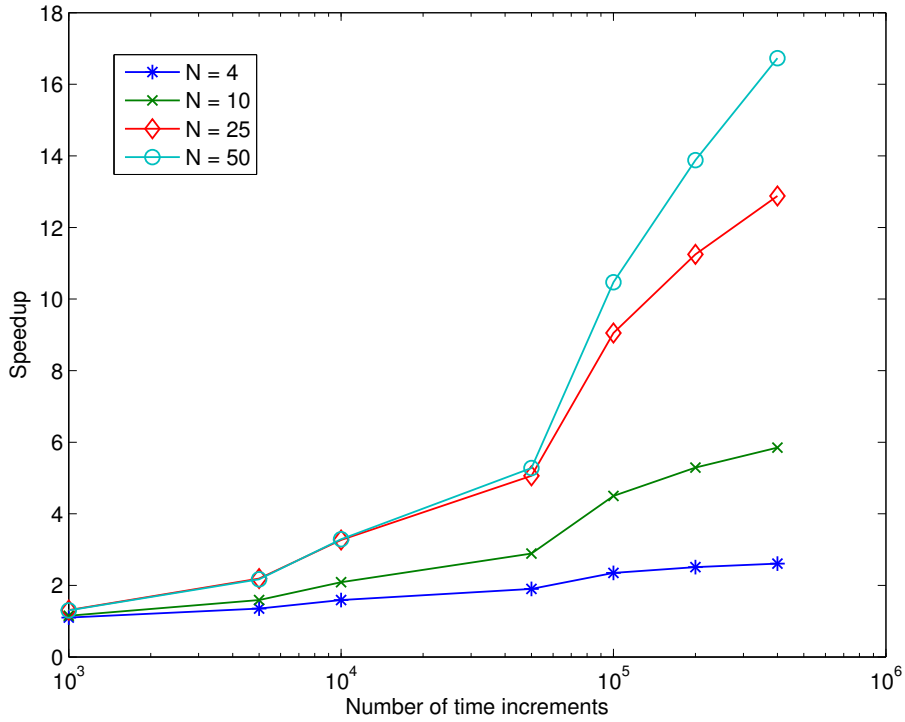


Figure 7.4: Speed-up for the decomposition approach

Chapter 8

Application to ultrasound welding for thermoplastic composites

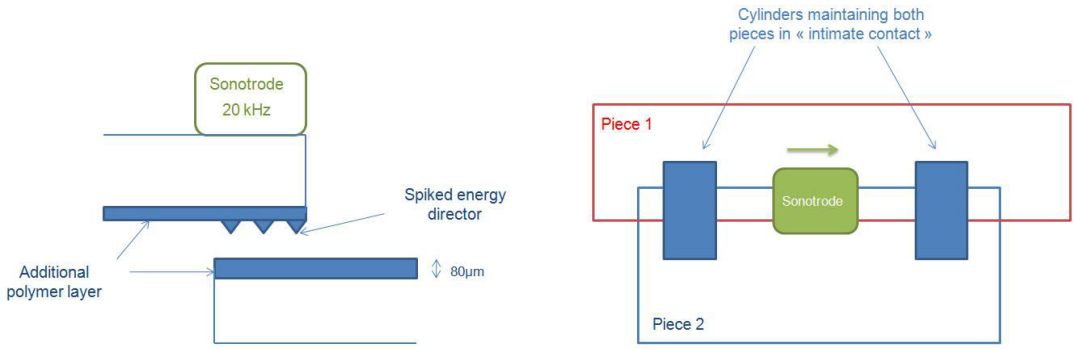
Composite materials are widely used in todays industry, and particularly in aeronautics. Thermoplastic composites are composites that use a thermoplastic polymer as a matrix. This category still represents a niche market because of the difficulties associated to its procesing. Indeed, polymerised thermoplastics tend to have very high melt viscosities, and consequently their injection into fibres requires high pressure and becomes more expensive. But, compared to thermoset composites, they exhibit superior impact and damage resistance properties. Another advantage is that thermoplastic composites can be readily recycled, which is an increasingly important issue in many markets.

In this final part, we chose to focus on a particular manufacturing process developed firstly for polymer industry: the ultrasound welding. This latter offers a very good alternative for assembling pieces made of thermoplastic composites. More simple than gluing, the mechanical characteristics of the assembly appears to be better than riveting because pieces don't have to be perforated. As we are going to explain in the next section the word "ultrasound" doesn't mean here that acoustic effects are playing any role but only refers to the high frequency of the load applied to the piece of composite.

8.1 Principle of ultrasound welding

For welding two thermoplastic composite pieces, specific physical conditions are required at the interface: a permanent contact, also called intimate contact, and a high temperature allowing diffusion of macromolecules. Because of the low diffusivity of composite materials, a high temperature at the interface can only be reached with a local heating. Several options like induction or resistance welding can be used, but a grid has to be positioned at the interface and remains there after welding: this can reduce the mechanical properties of the assembly.

Ultrasound welding is based on the dissipation of a mechanical work at the interface, allowing to increase very locally the temperature. EADS IW recently developed a new process called continuous welding, more suited to assembly for aeronautics compared to



(a) Energy directors (b) Intimate contact

Figure 8.1: Principle of the ultrasound welding

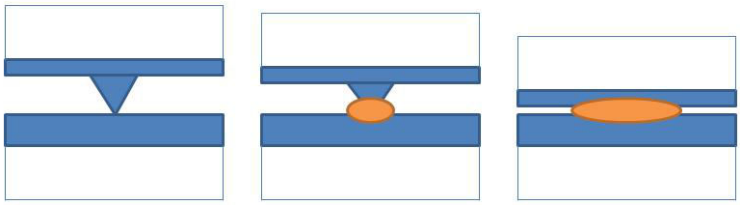


Figure 8.2: Fusion and flow of the energy director

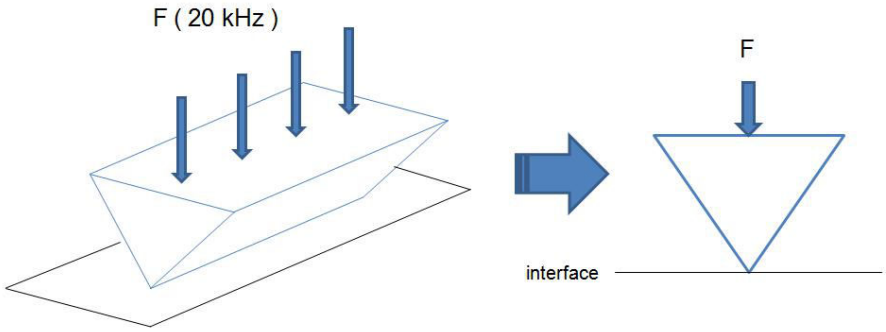


Figure 8.3: Plane strain model

the static ultrasound welding developped in the polymer industry. This process uses a tool which is a cylinder applying a periodic compressive force on the upper surface of the composite piece to be welded, as depicted on figure 8.1. Moreover additional cylinders are placed upstream and downstream of the welding region, in order to ensure a permanent contact of both pieces. Actually, an additional layer of polymer PEEK of width $80\mu m$ is added on both faces to be welded, and small prisms are molded on one of the face. The deformation is concentrated in those energy directors, which leads to a local heating by viscous dissipation. The prisms melt, flow at the interface and ensure the welding of both pieces, figure 8.2. As explained by A.Levy in his Phd thesis [12], the best compromise to study the welding is to choose a mesoscopic scale, focusing only on the mechanical and thermal consequences of the ultrasonic periodic compression on an energy director.

8.2 A simple 1D thermal model

The length of the prism (or more precisely of the contact zone) being much bigger (2mm) than the dimensions of its triangular basis (300 μm of height), and the load applied to the prism being constant along that direction, we can consider the strain in this direction as constrained and equal to zero: this corresponds to a plane strain approximation, as drawn on figure 8.3. Thus we only have to study a 2D problem. Moreover, the energy director is subjected to a compressive load, which is uniaxial and finally the corresponding stress writes,

$$\sigma = \sigma_x = \frac{F(t)}{S(x)} \tag{8.1}$$

where x is the vertical axis, whose origin is the welding interface and S is the section of the prism. According to the characteristics of the sonotrode, the sinusoïdal compressive load can be approximated by $F = F_0(\sin(wt)+1)$. Considering that the basis of the prism is an equilateral triangle, we obtain the following expression for the stress :

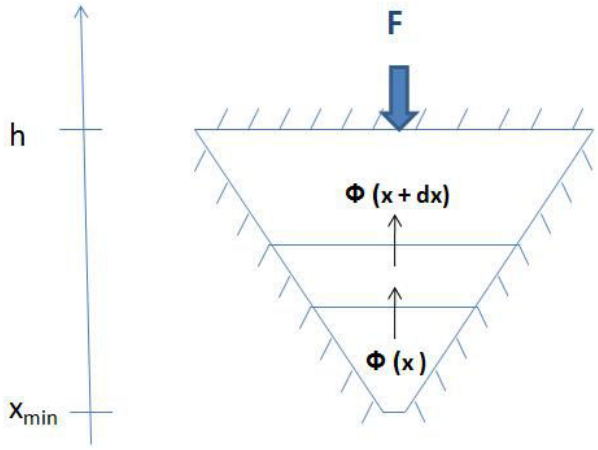


Figure 8.4: Conservation of internal energy

$$\sigma_x(x,t) = \frac{F_0\sqrt{3}\,(sin(wt)+1)}{2Lx} \tag{8.2}$$

where $F_0 = 6.3\ N$ is the magnitude of the load, $w = 4.10^4\pi\ rad.s^{-1}$ is the angular velocity and $L = 2.10^{-3}\ m$ is the length of the contact zone along the prism direction. From a numerical point of view this last expression is not acceptable because the surface becomes null at the interface and then the stress is infinite. Actually we assume that the extremity of the spiked energy director is flat, by slightly moving down the origin of the x axis.

In order to establish an equation for the evolution of the temperature in the energy director, we use the conservation of internal energy. We make the assumption that there is no heat transfer by convection or conduction at the boundaries of the triangular domain. If we consider a slice of thickness dx and volume V_e, as depicted on figure 8.4, we have:

$$\int_{V_e} \rho\frac{\partial e}{\partial t}dV + \int_{\partial V_e} \vec{q}.d\vec{S} - \int_{V_e} \sigma.\frac{d\epsilon}{dt}dV = 0 \tag{8.3}$$

If we denote $T(x,t)$ the average temperature in the section S(x) at time t, we can simplify the first two terms of the previous equation as follows,

$$\int_{V_e} \rho\frac{\partial e}{\partial t} = \rho c\frac{\partial}{\partial t}S(x)\int_x^{x+dx} T(x,t)dx = \rho cS(x)\int_x^{x+dx}\frac{\partial T(x,t)}{dt}dx = \rho c\frac{\partial T(x,t)}{dt}S(x)dx \tag{8.4}$$

$$\int_{\partial V_e} \vec{q}.d\vec{S} = -\int_{S(x)} (-k\frac{\partial T}{\partial x})dS + \int_{S(x+dx)} (-k\frac{\partial T}{\partial x})dS = \\ k\left(S(x)\frac{\partial T(x,t)}{\partial x} - S(x+dx)\frac{\partial T(x+dx,t)}{\partial x}\right) = -k\frac{\partial}{\partial x}\left(S(x)\frac{\partial T}{\partial x}\right)dx \tag{8.5}$$

Because of the configuration of the load, the stress and the strain depend only on x. Hence the last term of equation 8.3 can be also simplified,

$$\int_{V_e} \sigma\frac{d\epsilon}{dt}dV = S(x)\int_x^{x+dx} \sigma\frac{d\epsilon}{dt}dx = \sigma\frac{d\epsilon}{dt}S(x)dx \tag{8.6}$$

A Maxwell model is used to describe the behaviour of the PEEK considered as a viscoelastic material.

$$\frac{d\epsilon}{dt} = \frac{1}{E}\frac{d\sigma}{dt} + \frac{\sigma}{\eta} \tag{8.7}$$

where $E = 3.6\ GPa$ is the Young's Modulus and $\eta = 10^6$ is the characteristic viscosity. Regarding the source term, we can distinguish the work due to elastic deformation, $W_{el} = \frac{\sigma}{E}\frac{d\sigma}{dt}$ and the work associated to the viscous dissipation, $W_{visc} = \frac{\sigma^2}{\eta}$. Actually, we only keep this second part because it's the only one responsible for the production of heat and therefore the local increase of temperature at the interface. Finally, we obtain a local equation describing the evolution of the average temperature in a section of the prism.

$$\rho c\frac{\partial T}{\partial t} = \frac{k}{S(x)}\frac{\partial}{\partial x}\left(S(x)\frac{\partial T}{\partial x}\right) + \frac{\sigma^2}{\eta} \tag{8.8}$$

Concerning the boundary conditions, we made the assumption that there was no exchange of heat by conduction or by convection with the ambient air. Hence we impose a flux equal to zero at each boundary of the 1D domain, corresponding to the top and the bottom of the prism.

$$\frac{\partial T}{\partial x}(0,t) = \frac{\partial T}{\partial x}(h,t) = 0 \tag{8.9}$$

Because the frequency associated to the load is ultrasonic, the simulation of this problem requires the use of a very small time step. Indeed, in order to capture and simulate accurately the evolution of the temperature in the domain, we should consider a time step at least ten times smaller than the characteristic time of the problem. This latter corresponds here to the period $T = 5.10^{-5}\ s$ of the sinusoïdal load and then, the maximum time step we should use is $\Delta t = 5.10^{-6}\ s$. The sonotrode applies this load on the piece of composite for a period of 1s. An incremental approach, like finite differences would require 2.10^5 iterations. From a computational point of view, this proves to be very costly since a linear system has to be solved at each iteration. However, the problem being only 1D here, we could still afford an incremental approach.

8.3 Numerical results

8.3.1 A reliable approach...

We compute off-line a solution of the homogeneous problem, $u^h\left(t, x, p^1, \ldots, p^{Nc}\right)$ idefined in the multidimensional space $\Omega = \Omega_t \otimes \Omega_x \otimes \Omega_p \otimes \cdots \otimes \Omega_p$ where the respective domains of variability are,

$$\begin{cases} \Omega_t &= [0; 0.1] \ \ in \ s \\ \Omega_x &= \left[10^{-5}; 3.10^{-4}\right] \ \ in \ m \\ \Omega_p &= [250; 900] \ \ in \ K \end{cases}$$

Because of convergence problems, we have not been able to compute the solution on the entire period during which the load is applied, that is to say 1s. We only obtained a model valid on a period of 0.1s. The complete simulation is the subject of current work, but we couldn't finish it in time and therefore we cannot present the corresponding results in this manuscript.

A mesh of 20001 equidistant nodes is used for the time discretization, 1001 nodes in space and 11 nodes for each one of the coordinates p^j. In a classical mesh-based framework, this problem would be untractable since its complexity reaches $C_{FEM} \approx 2.10^4 * 10^3 * 11 * Nc \approx 1,5.10^9$ (we use Nc=7 parameters for the initial condition in this model). On the contrary the PGD based model has a complexity of $C_{PGD} \approx N * (2, 1.10^4)$ where N is the number of terms in the finite sum decomposition. Before reaching convergence 300 modes were necessary, which corresponds to a complexity $C_{PGD} \approx 6, 3.10^6 \approx \frac{C_{FEM}}{250}$!

A priori, we know that a particular attention has to be paid to the parametrization of the initial condition. Indeed, we want our model to guarantee the conservation of the thermal energy during the process. No creation nor dissipation of energy should be generated numerically. However, a small error on the temperature due to the projection of the solution at the time interfaces can lead to a non negligible jump in the energy of the system. Since our domain is adiabatic, this error will propagate and cumulate from one subdomain to the other until making the solution completely unrelevant. Therefore 7 parameters are used for the initial condition, but the nodes of the auxiliary coarse mesh are not equidistant. According to the temperature profile depicted on 8.5, this mesh should be finer in the vicinity of the spike because that's the region where the slope is more pronounced.

We solve concurrenly the particular problem on-line and we reattach the local solutions by enforcing the continuity using a least square method, already presented chapter 5. The evolution of the error in time is really representative of a dissipative model of evolution. Indeed, we can notice on figure 8.6 that this latter is very localized and vanishes quickly after the projection. Then if the size of the subdomain is large enough to allow a complete dissipation of the error before the next projection, the error is almost null (around 10^{-6}) except close to the interfaces. That's the case when considering 2, 4, 10 and 40 subdomains. On the contrary, as soon as the size of a subdomain is too small, the error doesn't vanish

entirely and adds from one subdomain to the other: that's what we obtain when splitting the time domain into 100 or 500 parts.

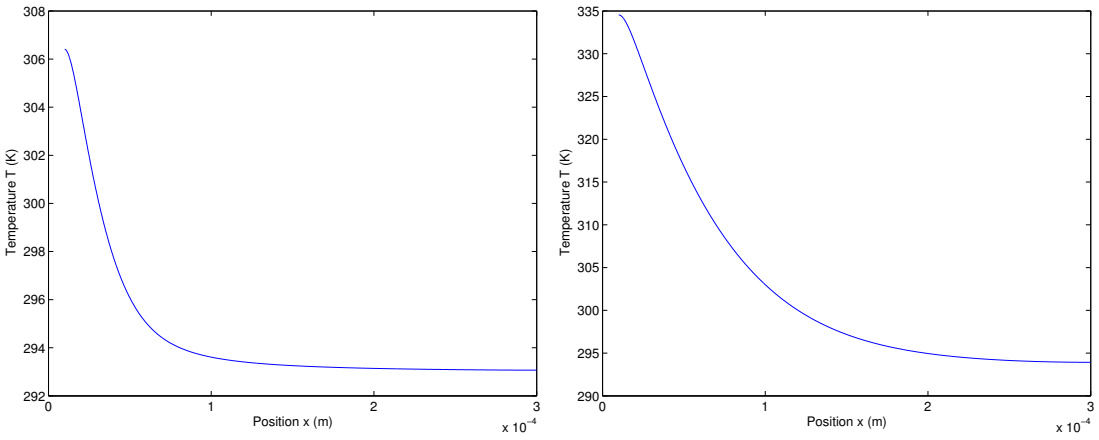
The second interesting comment we can make is related to the amplitude of the jump in the error. Indeed, whereas the maximum error is about 10^{-4} with 2 or 4 subdomains, it reaches 8.10^{-4} for 500 subdomains. This is actually due to the time evolution of the temperature profile in the domain. At the beginning of the process, the increase of temperature is very localized and therefore the profile of temperature exhibits a sharp evolution close to the extremity of the energy director. As we move forward in time, the diffusion of heat makes the distribution of temperature more homogeneous and then the profile smoother, 8.5. Because, we use a piecewise linear interpolation for the initial condition, the error introduced during the projection is smaller if the profile to be approximated is flat. That's the reason why when we split the time domain into a high number of parts, several interfaces are located in a zone where the temperature has strong variations, making the approximation more difficult.

The introduction of an optimized overlapping allows a significant reduction of both maximum and mean error, figure 8.7. The Matlab routine "fminsearch" is used to compute the optimum parameters $\{p^j\}_{j=1}^{7}$. As expected, the jump in error at the interfaces is smoothed (divided by almost 100). Finally the mean error is almost negligible (less than 10^{-5}) which makes the parallel approach a reliable method. We can assume that the remaining part of the error is not even due to the projections but more likely to approximations coming from the finite sum decomposition of the solution. Nevertheless, as we explain in the next section the improvement of the accuracy of the reattachment is done at the expense of the speed-up.

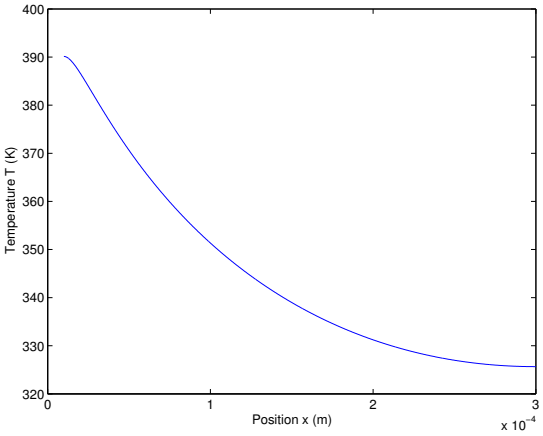
8.3.2 ..but a slow computation

Considering exactly the same discretization for x and t than previously, the solution of this problem with an incremental method (1D linear finite elements in space, and finite differences in time) requires 9 seconds, on the same computer we used during this project (2GHz doublecore processor, 4Gb of RAM). Unfortunately, the parallel approach cannot compete with the incremental one. Even if solving the particular problem is very fast, the computation time associated to the reattachment of the solutions is too penalizing, especially when we introduce some overlapping of the solutions, table 8.1. This is mainly due to the fact that at least 300 modes are required in order to reach convergence when computing the multidimensional solution of the homogeneous problem: the particularization of this latter at each reattachment proves to be really time-consuming.

Therefore, instead of allowing a speed-up of the calculation, the parallel execution of the program leads to an important increase of the computation time, which is quite paradoxal and meaningless. When extrapolating we could assume that the parallel approach would be equivalent to the sequential one for a high number of domains given the progressive reduction of the time necessary for the reattachment. But as we have shown previously, when we increase the number of subdomains the error cumulates and the model loses accuracy and reliability.



(a) t=0.001s (b) t=0.01s

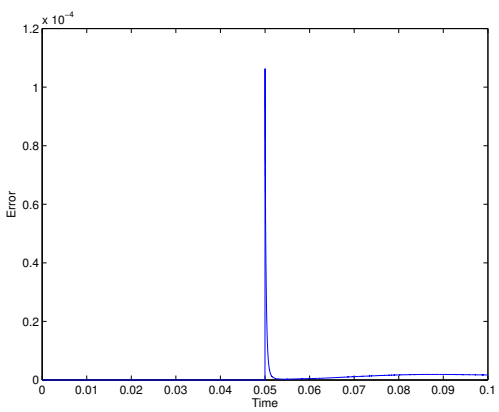


(c) t=0.1s

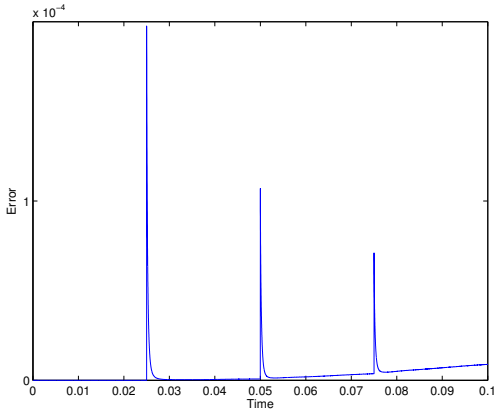
Figure 8.5: Evolution of the temperature profile in the domain

Number of subdomains	Particular problem(s)	Reattachment (s)
2	4.4	83
4	2.2	126
10	0.91	146
20	0.44	150
40	0.23	145
100	0.087	134
500	0.018	29

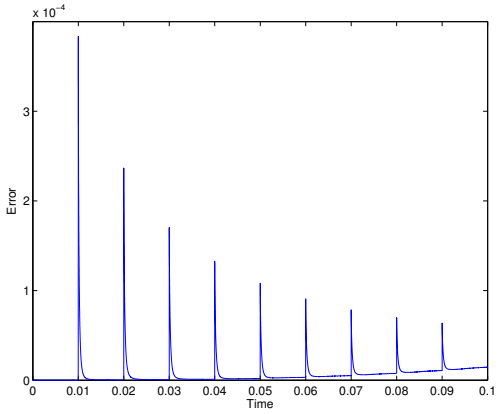
Table 8.1: Repartition of the computation time for a decomposition-based parallel approach



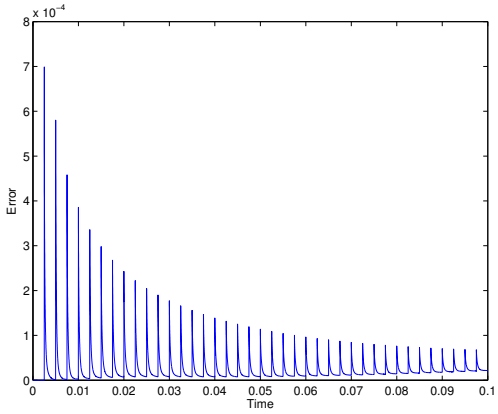
(a) 2 subdomains



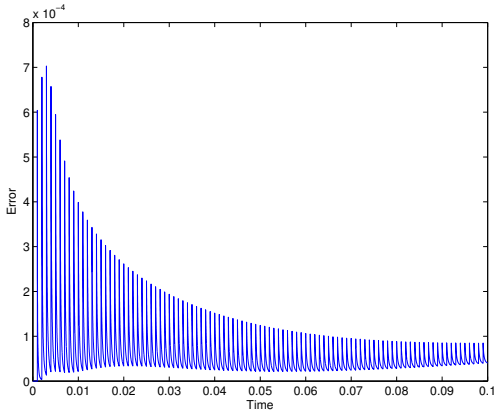
(b) 4 subdomains



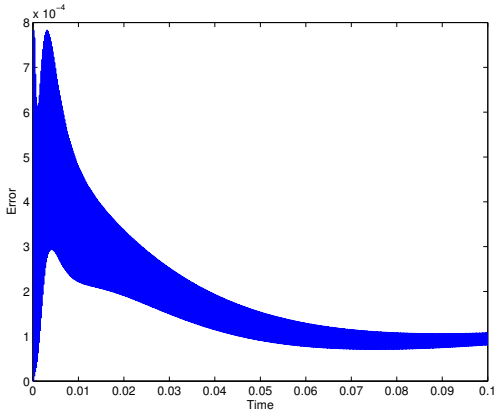
(c) 10 subdomains



(d) 40 subdomains

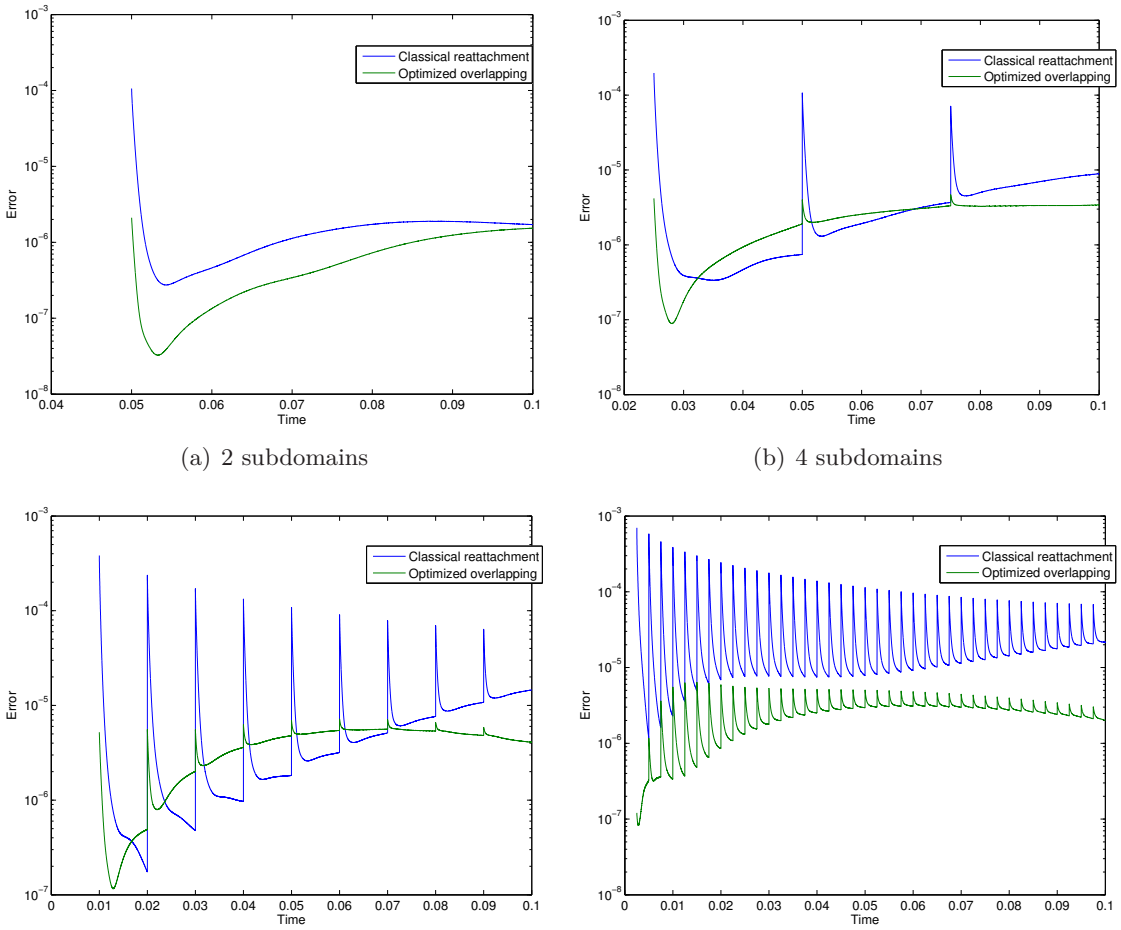


(e) 100 subdomains



(f) 500 subdomains

Figure 8.6: Evolution of the energy error with the number of time subdomains



(a) 2 subdomains

(b) 4 subdomains

(c) 10 subdomains

(d) 40 subdomains

Figure 8.7: Reduction of the error thanks to overlapping

Finally, the parallel approach has no real interest for the simulation of this manufacturing process. Even if it provides very good results, we rather use incremental approaches which allow a faster calculation. However, the solution we obtain with the parallel approach is way more adaptable. It is valid for a large range of initial conditions and the source term can be easily modified.

Conclusion

The Proper Generalized Decomposition based on a separated representation of the unknown fields involved in a partial differential equation, allows to compute cheap multidimensional solutions. The complexity scales linearly with the dimension of the model which makes possible the introduction of unknown or unprecisely known parameters in its coordinates. This method is young compared for instance to the FEM which has been studied and improved during several decades, but the perspectives it offers are numerous and probably still underestimated.

Our objective was to take advantage of this numerical tool to develop new strategies for a fast solution of transient problems that could be used in future real-time simulations. We proposed a methodology to introduce the initial condition of the problem in the coordinates of the model. Because this latter isn't a scalar quantity but a field, a parametrization is necessary. The construction of an auxiliary coarse mesh allows to approximate the initial field with a relatively small number of parameters and therefore to guarantee a reasonable computation cost for the resulting multidimensional solution. This approach is not only an exercise in style. It opens very interesting perspectives, especially for Data Driven Inverse Identification.

Actually, the quite promising application of this new strategy concerns the time parallelization. We presented two main approaches based on the splitting of the time domain. On one hand, the "full" parallelization consists in the off-line parallel solution of local problems. On the other hand, the "partial" parallelization uses a decomposition of the original problem. The solution of the homogeneous problem is computed off-line whereas the particular problem is parallelized on-line. This second option appears to be the best choice for real-time simulation thanks to its good flexibility: the source term can be modified on the fly with a moderate impact on the computation cost.

Nevertheless, parallelization leads inevitably to the introduction of an error at each interface of the time subdomains caused by the approximation on a coarse mesh of a profile defined on the fine calculation mesh. Adaptive refinement strategies for the auxiliary coarse mesh, like for instance hierarchical refinement, represent an appealing option but they result in the introduction of additional nodes and thus additional coordinates in the model and finally lead to an important increase of the computation cost.

However, the accuracy of the reattachment can be improved without modifying the complexity of the model. A natural overlapping based on the dissipative character of the heat transfer equation (parabolic) is easy to implement but restrains the number of time

subdomains we can afford and therefore the parallel speed-up. The optimization of the overlapping is less constraining regarding the number of subdomains, but requires the resolution of a nonlinear minimization problem at each interface.

Concerning the speed-up provided by this method, we pointed out however that a lot of work has still to be carried out. Indeed, the parallel approach can reduce significantly the computation time with the condition that the problem is large enough. But if the PGD based solution requires the calculation of an important number of modes, the results are not satisfying. In that case, the parallelization of the computation has a paradoxical impact on the computation time and proves to be slower than sequential approaches.

As a result, this non-incremental method offers new perspectives for the fast solution of transient problems but several aspects have still to be improved to make it a powerful tool.

Bibliography

[1] A. Ammar and F. Chinesta. Circumventing curse of dimensionality in the solution of highly multidimensional models encountered in quantum mechanics using meshfree finite sums decomposition. *Lecture notes Computational Science Engineering*, 65:1–17, 2008.

[2] A. Ammar, F. Chinesta, P. Diez, and A. Huerta. An error estimator for separated representations of highly multidimensional models. *Computer Methods in Applied Mechanics and Engineering*, 199:1872–1880, May 2010.

[3] A. Ammar, F. Chinesta, F. Lemarchand, P. Beauchene, and F. Boust. Alleviating mesh constraints: model reduction, parallel time integration and high resolution homogenization. *Computer Methods in Applied Mechanics and Engineering*, 197:400–413, 2008.

[4] A. Ammar, F. Chinesta, and E. Pruliere. On the deterministic solution of multidimensionnal parametric models using proper generalized decomposition. *Mathematics and Computers in Simulation*, 81:791–810, July 2010.

[5] A. Ammar, M. Normandin, F. Daim, D. Gonzalez, E. Cueto, and F. Chinesta. Non incremental strategies based on separated representations: Applications in computational rheology. *Computer Methods in Applied Mechanics and Engineering*, January 2010.

[6] S. Andrieux and T.N. Baranger. Energy methods for cauchy problems of evolution equations. In *6th International Conference on Inverse Problem in Engineering: Theory and Practice*.

[7] C.C. Douglas and Y. Efendiev. A dynamic data-driven application simulation framework for contaminant transport problems. *Computers and Mechanics with Applications*, 51, 2006.

[8] D. Gonzalez, F. Masson, F. Poulhaon, A. Leygue, E. Cueto, and F. Chinesta. Proper generalized decomposition based dynamic data driven inverse identification. *submitted to Journal of Mathematical analysis and applications*, May 2011.

[9] P. Krysl, E. Grinspun, and P. Schröder. Natural hierarchical refinement for finite elements methods. *International Journal for Numerical Methods in Engineering*, 56:1109–1124, 2002.

[10] P. Krysl, A. Trivedi, and B. Zhu. Object-oriented hierarchical mesh refinement with charms. *International Journal for Numerical Methods in Engineering*, 60:1401–1424, 2004.

[11] P. Ladeveze. *Nonlinear Computational Structural Mechanics*. Springer, 1999.

[12] A. Levy. *Modélisation et simulation d'un écoulement sous vibration. Application au soudage par ultrasons de composites à matrice thermoplastique*. PhD thesis, Ecole Centrale de Nantes, April 2010.

[13] F. Masson, E. Cueto, A. Ammar, and F. Chinesta. On the solution of partial differential equations with parametrized initial condition. *submitted to Elsevier Science*, September 2010.

[14] A. Nouy. A priori model reduction through proper generalized decomposition for solving time-dependent partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 199:1603–1626, April 2010.

[15] Y. Ouyang, J.E. Zhang, and S.M. Luo. Dynamic data driven application system: Recent development and future perspective. *Ecological Modelling*, 204:1–8, May 2007.