Parallelization of mortar spectral element method
Heithem ABBES, Nejmeddine CHORFI and Mohamed JEMNI
Pages: 155-167

Parallelization of Mortar Spectral Element Method

Heithem ABBES[(1)], Nejmeddine CHORFI[(2)] and Mohamed JEMNI[(1)]

[(1)] École Supérieure des Sciences et Techniques de Tunis, Unité de recherche UTIC
5, Av. Taha Hussein, B.P. 56, Bab Mnara, Tunis, TUNISIA
E-mail : heithem.abbes@esstt.rnu.tn mohamed.jemni@fst.rnu.tn

[(2)] Faculté des Sciences de Tunis
Département de Mathématiques
Campus Universitaire, Le belvédère, Tunis, TUNISIA
E-mail : nejmeddine.chorfi@fst.rnu.tn

Abstract: *Our work concerns the approximation of the 2D Stokes equations solution based on a variational formulation with primitive variables: velocity and pressure. We choose to use the mortar spectral element method which is a domain decomposition technique. Since, this method has a high spatial and temporal complexity, we opted to parallel processing. We present, in this paper, two versions of sequential algorithms of the resolution of 2D Stokes equations on one domain and then on two sub-domains using mortar spectral element method. Then, we present a parallelization of these algorithms based on exploiting data parallelism and using MPI directives and language C. The performance evaluation achieved through different experiments done on IBM SP2 parallel machine, shows a wide difference in execution time between the two versions.*

Keywords: *Stokes equations, mortar spectral method, tensorization, domain decomposition, data parallelism.*

Received: *September 12, 2005* | **Revised:** *July 11, 2006* | **Accepted:** *August, 31, 2006*

1. Introduction

The numeric simulation is characterized by a multidisciplinary process where intervene physical modeling, mathematical analysis and numeric resolution on computer. The interaction between these different stages is necessary to succeed any study in this field. The goal of the physical modeling is to get a set of equations which describes the intervening physical phenomena adequately. It often calls to several other disciplines as fluids and solid mechanics, electromagnetism, quantum mechanics, etc [15].
Several models are described by a system of partial derivatives equations (SPDE). Then, we ask some questions: can we prove, in an adequate theoretical setting, the existence and the singleness of the solution as well as its stability in relation to the data (limits conditions, initial conditions, and parameters of a model...)? When the answer to these questions is affirmative, we speak then about a very straightforward problem. But, if we do not have an analytic solution, the only way is to consider an approached resolution of the model equations system by a numeric method. Another point to underline is the regularity of the solutions of the model. Indeed, a numeric method can be adapted to the approximation of regular solutions, but it can be unsuitable in case of singularities [2].
The last step is the implementation of the numeric method; it is inescapable, because we rarely have the analytic solutions. The quality of the numeric method is defined, on one hand, according to the mathematical criteria's (stability, evaluation of error) and, on the other hand, according to its convenient performances (precision and cost). In general, according to the needs, we define a compromise between these two criterias [2].
Computing and numeric simulation intervene in various industrial applications such as in aerodynamics and hydrodynamics domains where we are interested in the correction of the phenomena turbulences while studying the perfect fluid flow [10, 11]. One of the big projects, that treat the three-dimensional geometric reconstitution of the blood vessels from the medical imagery, is the CALVI project, adopted by the INRIA laboratory [20]. A big part of this project consists of the simulation of the blood flow in the veins. The resolution of this type of

problems requires a big power of calculation as well as an important physical memory capacity, what shows the big interest of the resort to the parallelism.
In this setting, we are interested to the numeric simulation of the Stokes problem that consists on approaching the velocity and the pressure of fluid flow on a plate. The objective is to develop a parallel application based on mortar spectral element method after having developed a sequential application and studied different strategies of parallelization.
We start this paper by a brief illustration of the mathematical modeling of the physical problem. Then, in section 3, we describe the different steps of numeric simulation to converge toward a linear representation of the fluid flow. In section 4, we explain the tensorization approach, which is one of the most important distinctions between our work and any other one. Thereafter, in section 5, we illustrate parallel algorithm of different versions of resolution, on one domain and on sub-domains. In section 6, we present some numerical results to emphasize parallelism approach role. Finally, we end this paper by a conclusion and some perspectives that seem to be interesting.

2. The Continuous problem

The flow of an incompressible viscous fluid in a domain Ω of $\mathbb{R}^2$ is characterized by two variables: its velocity u and its pressure p. It is depicted by two following equations:
The equation of the conservation of the quantity of movement

$$-\nu \Delta u + grad\, p = f \quad in\ \Omega$$

The equation of incompressible that translates the conservation of mass

$$div\, u = 0 \quad in\ \ \Omega$$

Here ν, is a positive parameter, called mechanical viscosity, and the function f corresponds to the forces applied to the fluid. In general, we add to this system a limits condition of type Dirichlet, translating the fact that the fluid does not slip on the border of the domain:

$$u = 0 \quad in\ \partial\Omega$$

$\partial\Omega$ designates the border of the domain.
Thus, the problem gotten, known as the Stokes problem, is presented by the following system:

$$\begin{cases} -\nu \Delta u + \nabla p = f & in\ \Omega \\ div\, u = 0 & in\ \Omega \\ u = 0 & on\ \partial\Omega \end{cases} \qquad (S1)$$

To pass to the variational formulation, we intervene different spaces for the velocity and the pressure. It is about the spaces $H_0^1(\Omega)^2$ for the speed and $L_0^2(\Omega)$ for the pressure, with:

$$\begin{cases} \forall v \in H_0^1(\Omega)^2 : \\ \nu \int_\Omega \nabla u(x)\, \nabla v(x)\, dx - \int_\Omega div\, v(x)\, p(x)\, dx = \int_\Omega f(x)\, v(x)\, dx \\ \forall q \in L_0^2(\Omega) : \int_\Omega div\, u(x)\ q(x)\, dx = 0 \end{cases} \qquad (S2)$$

We demonstrate mathematically that the problem (S2) admits a unique solution. We are interested then to approaching this solution numerically by the spectral method.

3. The Discrete problem and algorithm of resolution

3.1. The Discrete problem

To discretise our problem, we use spectral method. This method, introduced by D. Gottlieb and S. Orszag [12, 17], is a technique to approximate the solutions for partial differential equations with a high-degree polynomial. In this sense, the precision of these methods is only limited by the regularity of the function to approach, contrarily to the other types of approximation as the finite-difference or the finite-element. The calculation of an approached solution essentially rests on the variational formulation of the continuous problem; the discretisation takes place by replacing the space of the functions tests by a space of polynomials and by evaluating the integrals using formulas of appropriated quadrature.
The spectral methods knew these last years a big development in the academic setting. They also begin to interest the industrial surroundings, essentially in the domains of aeronautics, of the meteorology, of the mechanics of non linear solid. A big part of their success comes of their elevated precision that, thanks to the huge development of the computer means, proves to be a more and more important asset in the numeric simulation [3] (see figure1).

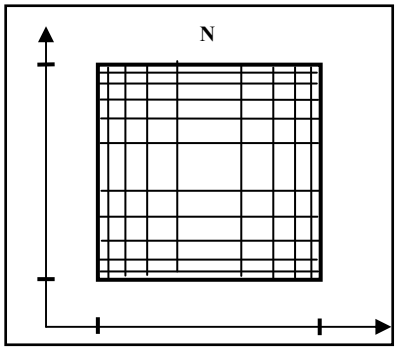

Figure 1. Spectral method

We note that the grid of Gauss-Legendre is defined by: $\Xi = \{(\xi_i, \xi_j), 0 \le i, j \le N\}$, in the domain $[-1,1]\times[-1,1]$ with N is the discretization parameter. We set $\xi_0 = -1$ and $\xi_n = 1$. We remind that there exists N-1 nodes ξ_j , $1 \le j \le N-1$, in $]-1,1[$, with $\xi_0 < \xi_1 < ..... < \xi_N$ and N+1 positive weights ρ_j , $0 \le j \le N$, such that:

$$\forall \phi \in \mathrm{P}_{2N-1}(-1,1), \int_{-1}^{1} \phi(\zeta) d\zeta = \sum_{j=0}^{N} \phi(\xi_j)\rho_j$$

Moreover the following property holds:

$$\forall \varphi \in \mathrm{P}_N(-1,1),$$

$$\|\varphi\|^2_{L^2(-1,1)} \le \sum_{j=0}^{N} \varphi^2(\xi_j)\rho_j \le 3\|\varphi\|^2_{L^2(-1,1)}$$

This leads to define discrete product: For any continuous functions u and v on Ω

$$(u,v)_N = \sum_{i=0}^{N}\sum_{j=0}^{N} u(\xi_i,\xi_j) v(\xi_i,\xi_j)\rho_i\rho_j$$

We start with defining some formulas and notations concerning the choice of basis functions and tests functions, what allows us to lead to the matrix system. We define the interpolation polynomial of Lagrange ℓ_i as follows:

$$\ell_i(x) = \frac{\prod_{j=0, j\neq i}^{N} (x - x_j)}{\prod_{j=0, j\neq i}^{N} (x_i - x_j)} \qquad \text{i = 0…N}$$

We define $\hbar_i$ the following polynomial

$$\hbar_i(x) = \frac{\ell_i(x)}{1-x^2} \quad \text{i=0..N}$$

Notice that the family $\ell_i\ell_j$, for $1 \le i, j \le N-1$, forms a $\mathrm{P}^0_N(\Omega)$ base. In the same way, the family $\hbar_i\hbar_j$ forms a $P_{N-2}(\Omega)$ base. The unknowns $U_N = (U^1_N, U^2_N)$ for the vector velocity are its values in $(P^0_N(\Omega))^2$, so we use the following decomposition:

$$U^k_N(x,y) = \sum_{i=1}^{N-1}\sum_{j=1}^{N-1} U^k_{ij}\,\ell_i(x)\ell_j(y) \qquad k = 1,2$$

Since the pressure is in $P_{N-2}(\Omega)$, we decompose it in the $\hbar_i\hbar_j$ base. The unknowns will be therefore the following quantities:

$$P_{ij} = P_N(\xi_i,\xi_j)\ (1-\xi_i^2)\ (1-\xi_j^2)$$

Hence, we have the following decomposition:

$$P_N(x,y) = \sum_{i=1}^{N-1}\sum_{j=1}^{N-1} P_N(\xi_i,\xi_j)\hbar_i(x)\hbar_j(y)$$

Notice that $U^k_N(x,y)$ (respectively $P_N(x,y)$) converge toward the exact velocity U (respectively the exact pressure P) when N tends toward $+\infty$.

The discrete problem is:

$$\begin{cases} a_N(U^1_N, \ell_r\ell_s) + b_{1N}(\ell_r\ell_s, P_N) = (f_1, \ell_r\ell_s)_N \\ a_N(U^2_N, \ell_r\ell_s) + b_{2N}(\ell_r\ell_s, P_N) = (f_2, \ell_r\ell_s)_N \\ b_{1N}(U^1_N, \hbar_r\hbar_s) + b_{2N}(U^2_N, \hbar_r\hbar_s) = 0 \end{cases} \qquad \text{(S3)}$$

With

$$a_N(U^k_N, \ell_r\ell_s) = (\nabla U^k_N, \ell_r\ell_s)_N$$
$$b_{kN}(U^k_N, \hbar_r\hbar_s) = (div U^k_N, \hbar_r\hbar_s)_N$$

3.2. Algorithm of resolution

When the integer r and s vary in the interval $[1..N-1]$, we get the following matrix system [18]:

$$\begin{cases} AU_1 + B_1^T P = F_1 \\ AU_2 + B_2^T P = F_2 \\ B_1U_1 + B_2U_2 = 0 \end{cases} \qquad \text{(S4)}$$

We notice that the size of the matrixes A, B_1 and B_2 is $(N-1)^2$.

We intend to calculate the general term, $a_N(\ell_i\ell_j, \ell_r\ell_s)$ of the matrix A, according to [3], we have:

$$a_N(\ell_i\ell_j, \ell_r\ell_s) = \alpha_{ir}\omega_j\delta_{js} + \alpha_{js}\omega_i\delta_{ir}$$

With

$$\alpha_{ir} = \begin{cases} \dfrac{4}{N(N+1)L_N(\xi_i)L_N(\xi_r)(\xi_i-\xi_r)^2} & \text{si i} \neq \text{r} \\ \dfrac{2}{3(1-\xi_i^2)L_N^2(\xi_i)} & \text{si i} = \text{r} \end{cases}$$

The matrix B1 is in the space $\mathrm{M}_{(N-1)^2}(\Re)$, whose general term is noted by:

If r ≠ i:

$$b_1(\ell_i\ell_j, \hbar_r\hbar_s) = -\frac{1}{2}\frac{\delta_{js}\omega_j}{(1-\xi_j^2)L_N(\xi_i)}\left(\frac{\omega_0}{L_N(\xi_r)(\xi_0-\xi_i)(\xi_0-\xi_r)} + \frac{2L_N(\xi_r)\omega_r}{(\xi_r-\xi_i)(1-\xi_r^2)} - \frac{\omega_N}{(\xi_N-\xi_i)(\xi_N-\xi_r)L_N(\xi_r)}\right)$$

If r = i:

$$b_1(\ell_i\ell_j,\hbar_r\hbar_s)=-\frac{1}{2}\frac{\delta_{js}\omega_j}{(1-\xi_j^2)L_N(\xi_i)L_N(\xi_r)}\left(\frac{\omega_0}{(\xi_0-\xi_i)(\xi_0-\xi_r)}-\frac{\omega_N}{(\xi_N-\xi_i)(\xi_N-\xi_r)}\right)$$

To determine the general term of the matrix B2, we remark that we have the following equality:

$$b_1(\ell_i\ell_j,\hbar_r\hbar_s)=-\sum_{m=1}^{N-1}\sum_{n=1}^{N-1}\ell_i(\xi_m)\ell'_j(\xi_n)\hbar_r(\xi_m)\hbar_s(\xi_n)\omega_m\omega_n = b_2(\ell_j\ell_i,\hbar_s\hbar_r)$$

Notice that the elements of the matrixes B_1 and B_2 have been determined from precise expressions which are complex to be resolved on only one domain. On the other hand, to pass to a resolution on two sub-domains, it was necessary to simplify these expressions but reducing the precision as follows:

$$b_1(\ell_i\ell_j,\ell_r\ell_s)=-\ell'_i(\xi_r)w_jw_r = b_2(\ell_j\ell_i,\ell_s\ell_r)$$

Based on the above, we can represent the matrix system (S4) under the following reduced shape [18]:

$$\begin{cases}\tilde{A}U+B^TP=F\\ BU=0\end{cases}\quad\text{(S5)}$$

With:

$$\tilde{A}=\begin{bmatrix}A & 0\\ 0 & A\end{bmatrix},\ B=\begin{bmatrix}B_1 & B_2\end{bmatrix}$$

$$U=\begin{pmatrix}U_1,\ U_2\end{pmatrix},\ F=\begin{pmatrix}F_1,\ F_2\end{pmatrix}$$

To solve the Stokes problem numerically, we use the algorithm of Uzawa. Indeed, this algorithm is particularly well adapted to this kind of problems since it takes their features into account [1].
This method is iterative. It is based on conjugate gradient algorithm. Therefore, it is useful to remain this one before presenting the Uzawa method.

Conjugate gradient algorithm (CGA)

Among the iterative methods, the most effective in the case of the spectral methods, is the conjugate gradient [6, 9]. The principle of this method is to construct a sequence $(U_0,U_1,...)$ of vectors that converges toward the exact solution U of the system $\mathrm{A\,U}=\mathrm{b}$, where A is a positive definite symmetric matrix.
The advantage of this method is that we know in advance the number of iterations necessary to converge toward the exact solution. Indeed, if the size of U is n, the algorithm executes a maximum of n iterations.
It suits to specify that the CGA is an algorithm of optimization that we apply to the problem $\min_{x\in\Re}\Phi(\mathrm{U})$ where $\Phi(\mathrm{U})$ is a quadratic shape and $\Phi(\mathrm{U})=\frac{1}{2}\mathrm{U^T A\,U}-\mathrm{b^T\,U}$.
The algorithm can be written as follows:

$$\begin{aligned}&Either\ any\ U_0\\ &R_0=b-AU_0\\ &Q_0=R_0\\ &For\ k=1,2,.....until\ verification\ of\ stop\ criteria\\ &\alpha_k=R_k^TR_k\Big/\left(Q_k,AQ_k\right)\\ &U_{k+1}=U_K+\alpha_kQ_k\\ &R_{k+1}=R_k-\alpha_kAQ_k\\ &\beta_k=R_{k+1}^TR_{k+1}\Big/R_k^TR_k\\ &Q_{k+1}=R_{k+1}+\beta_kQ_k\end{aligned}$$

Algorithm of Uzawa

This algorithm eliminates the velocity from the first line of the system (S5), what gives:

$$\begin{cases}U=\tilde{A}^{-1}\left(F-B^TP\right)\\ B\tilde{A}^{-1}\left(F-B^TP\right)=0\end{cases}\quad\text{(S6)}$$

Therefore, we are brought to solve the two uncoupled systems successively:

$$B\tilde{A}^{-1}B^T\,P=B\tilde{A}^{-1}F$$

$$AU=F-BP$$

The Uzawa method consists on solving this couple of systems by the conjugate gradient method since the matrix $B\tilde{A}^{-1}B^T$ is symmetric defined positive.

Indeed, this method requires much less memory and CPU time since, on the one hand, we do not save the two matrixes A and B and ,on the other hand, the resolution is done with coupled way on the velocity and the pressure; what minimizes the number of iterations.
The algorithm can be written as follows:

$$P_0 = 0$$
$$U_0 = \tilde{A}^{-1} F$$
$$R_0 = B U_0$$
$$D_0 = R_0$$
$$For\ k = 1,2,.....until\ verification\ of\ stop\ criteria$$
$$Z_k = \tilde{A}^{-1}\ B^T\ D_k$$
$$\alpha_k = R_k^T R_k \Big/ \left(D_k, B\, Z_k\right)$$
$$P_{k+1} = P_k + \alpha_k\, D_k$$
$$R_{k+1} = R_k - \alpha_k\, B\, Z_k$$
$$U_{k+1} = U_k - \alpha_k\, Z_k$$
$$\beta_{k+1} = R_{k+1}^T R_{k+1} \Big/ R_k^T R_k$$
$$D_{k+1} = R_{k+1} + \beta_{k+1}\, D_k$$

We notice that the equation $U_0 = \tilde{A}^{-1} F$ requires the inverse of A. However, a straightforward transformation brings us back to the resolution of a linear system what is less expensive: $X = A^{-1} F$ is put as $AX = F$ that we solve by the conjugate gradient method.

Numeric resolution by the mortar spectral element method

The mortar spectral element method, introduced in 1987 by C. Bernardi, Y. Maday and A.T Patera [5, 4], is a domain decomposition technique based on a partition of the domain calculation in sub-domains without recovery. It permits to use completely independent discretization on every sub-domain, thanks to splice condition on the interfaces (see figure 2).

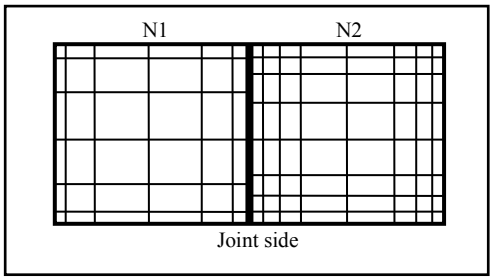

Figure 2. Mortar spectral element method

The splice condition for the level of the joined side has appeared a splice matrix Q which translates the transmission condition to the interfaces of the sub-domains. Thus, we get the linear system (S7) representative of the global problem [8]:

$$\begin{cases} Q^T \tilde{A} Q U + Q^T B^T P = Q^T F \\ \\ BQU = 0 \end{cases} \qquad \text{(S7)}$$

4. Tensorization

There are two main classes of numerical methods; the direct methods and the iterative methods. Each of them presents some advantages and disadvantages. However, the big difference between them consists of the fact that a direct method consumes a more memory, whereas an iterative method requires a big number of iterations to converge toward the exact solution.

Concerning our work, we tried to reduce the number of iterations as well as the necessary memory. Thus, the method that seems good is the tensorized iterative method.

4.1. Tensorization principle

Besides, the storage of the complete matrix is very expensive in memory. Therefore, it can be done only for small values of N. Although the matrixes generated by the spectral decomposition method present several zeros, the storage stays expensive since, contrarily to finite element decomposition methods, these elements do not respect a structure which enables their localization.

The iterative methods permit to converge toward the solution of the system by successive approximations and reduce the number of operations considerably; the principle is to construct a sequence of vectors U_n that converges toward the exact solution U. Indeed, in each iteration k, the longest part of the calculation consists of valuing the residual $R_k = F - A\,U_k$ and multiply the matrix A by a vector. Several advantages result from it. Thus, the matrix-vector multiplication can be calculated easily from the two following formulas:

$$a_N(\ell_i \ell_j, \ell_r \ell_s) = \alpha_{ir}\omega_j\delta_{js} + \alpha_{js}\omega_i\delta_{ir}$$

With:

$$\alpha_{ir} = \begin{cases} \dfrac{4}{N(N+1)L_N(\xi_i)L_N(\xi_r)(\xi_i - \xi_r)^2} & \text{si i} \neq \text{r} \\ \dfrac{2}{3\left(1-\xi_i^2\right)L_N^2(\xi_i)} & \text{si i} = \text{r} \end{cases}$$

In addition, the tensorization properties appear in these formulas permitting to calculate a matrix-vector multiplication with very low cost ($O(N)^3$ for the multiplication of the matrix A (of size $(N-1)^2$) by a vector V of size $(N-1)^2$. Indeed, for each $(N-1)^2$ of

the couples (i, j), $1 \le i, j \le N-1$, we must calculate the quantity:

$$\sum_{r=1}^{N-1}\sum_{s=1}^{N-1} a_N\left(\ell_i\ell_j, \ell_r\ell_s\right)V_{rs}$$

According to the first formula, this is equal to:

$$\sum_{r=1}^{N-1}\alpha_{ir}\rho_j V_{rj} + \sum_{s=1}^{N-1}\alpha_{js}\rho_i V_{is}$$

The calculation of both terms of the previous line requires $2(N-1)$ multiplications and $N-2$ additions. Therefore, the calculation of each coefficient of the vector, which is the result of the matrix-vector product, requires $4(N-1)$ multiplications and $2N-3$ additions, i.e. O(n), so O(n^3) in all.

5. Sequential algorithms

We observe that the matrix A does not need to be collected completely. Only the $(N-1)^2$ coefficients α_{ir} must be kept in memory, this number is lower than the total number of the coefficients of A (i.e. $(N-1)^4$).

This technique allowed us to find a compromise between the necessary number of operations and the consumed space memory. Indeed, for the calculation of the coefficients of the matrixes (A, B1, B2, B1T, B2T) and of the matrix-vector product, instead of stocking all matrixes (of size $(N-1)^2$), we stock only two matrixes $Alfa$ and $Beta$ with size is $(N-1)$. It allows us, therefore, to gain space memory and to reduce the calculation cost.

To emphasize this idea, two versions will be exposed in the rest of this paper, the first is an iterative version and the second is based on the tensorization principle.

5.1. Resolution without domain decomposition

We conceived and implemented, on the parallel machine IBM SP2 [13, 14], two versions of algorithms to solve Stokes problem on only one domain. The first is iterative without storage whereas the second, which presents an improvement of the first, uses the technique of tensorization. We start by presenting the first version, and then we focus on the improvement brought by the second one.

Non tensorized iterative version

The diagram of figure 6 illustrates the stages of the numeric resolution of the equation system (S1) presented in the section 2. Thus, the main task consists in four steps. The first is choosing the discretization parameter, as well as the domain of resolution to simulate the plate on which the fluid is flowing. The second consists on making a mesh of the domain using the spectral discretization. The third is the basis step of the resolution algorithm, it calls Uzawa procedure. The last step consists on calculating the exact solution in order to test the accuracy and the precision of results provided by our program.

Tensorized iterative version

We notice that in the Uzawa and conjugated gradient procedures, there is a lot of matrix-vector product. Since there is no storage of matrixes' elements in the iterative version, there will be a big redundancy in calculating these elements while multiplying a matrix by a vector. It is clear that storing five matrixes of $(N-1)^2 \times (N-1)^2$ elements does not solve the problem.

As we have already mentioned, the tensorization technique permits solving this problem by storing two matrixes noted $Alfa$ and $Beta$ sized $(N-1)$. Therefore, it is necessary to bring a little modification in the first diagram (see figure 3) in order to concretize this improvement (see figure 4).

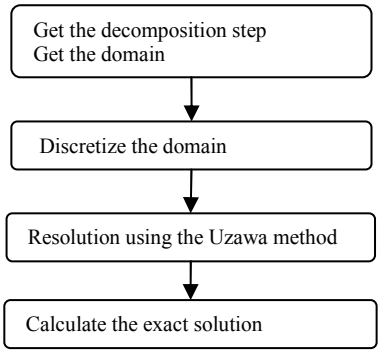

Figure 3. Main stages of the non tensorized iterative version

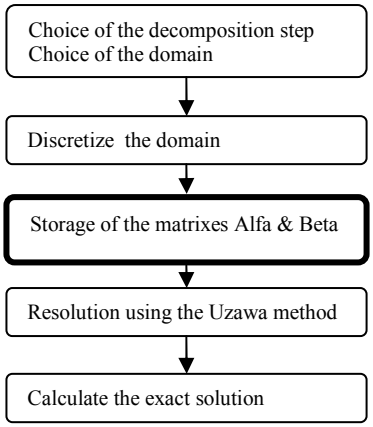

Figure 4. Main stages of the tensorized iterative version

5.2. Resolution on two sub-domains

As well as the resolution on one domain, for the resolution on two domains, two versions have been conceived and implemented on the SP2 IBM machine; one is iterative without storage and the other is iterative tensorized. To avoid burdening the article, we are not going to detail these two stages again. We will be limited to specifying differences brought by the transition on two domains.

Indeed, as we have already mentioned, the transition to two domains gave the mortar matrix which guarantees the data transfer between the two sub-domains. Therefore, we add a new stage in order to calculate the mortar matrix as well as its transposed matrix (see figure 5).

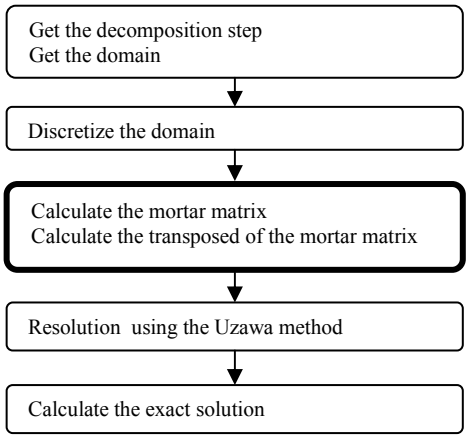

Figure 5. Main stages of the resolution on 2 domains

6. Parallelization approaches

The strategy adopted for the parallelization of sequential algorithms, called data parallelism, consists on distributing the calculation of unknowns on the different available processors (see figure 6). In this section, we are going to present a first parallelization of the resolution without domain decomposition, followed by a second parallelization for the resolution on two sub-domains.

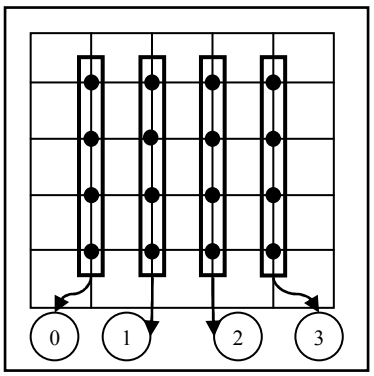

Figure 6. Distribution of data on 4 processors

6.1. Parallelization of the resolution without domain decomposition

In the section 5, we presented two sequential algorithms for the resolution on only one domain: the iterative version without storage and the tensorized iterative version. Since the main difference between these two versions appears in the addition of the two matrixes Alfa and Beta, the parallelizations of these two algorithms are nearly identical. Particularly, the parallelization of Uzawa algorithm is the same for the two versions.

Parallelization of Uzawa method

This stage is the most expensive while speaking about calculation time; it represents the basic section in the resolution algorithm. Let's notice that among the most frequent core in the algorithm, we find the matrix-vector product, the scalar product and the sum of two vectors. Thus, we had to parallelize these cores and minimize the communication between processors.

For the matrix-vector product, each processor calculates the result of multiplication between a vector and a lines' block known by its identity. Concerning the scalar product and the sum of two vectors, we used the global reductions operations MPI_Allreduces presented in the MPI library [21].

Parallelization of the tensorized iterative version

We keep the same principle of parallelization already presented for the first version. The only modification brought by this version, is the addition of a parallelization of the calculation of Alfa and Beta elements since the different processors are still free.

Let's specify that this parallelization is simpler than the one achieved for the resolution on two domains with joints. Indeed, the last one presents a high parallelism rate and requires an efficient communication process.

6.2. Parallelization of the resolution on two domains

We can allocate a set of processors to each sub-domain (see figure 7). Thus, we get two simultaneous local resolutions. We note that the two resolutions are not completely independent and require some intermediate calculations to finish the resolution and to lead to the final result. So, we need communication to exchange this information.

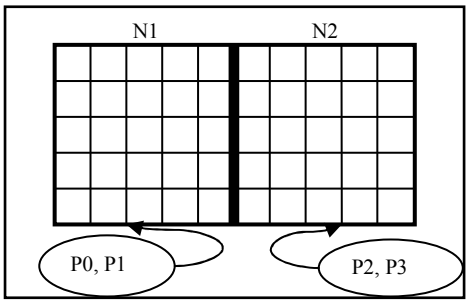

Figure 7. Resolution with four processors

Parallelization of calculation of the mortar matrix

Before continuing the resolution, there is a stage of calculation of the mortar matrix which guarantees the exchange of information between the two sub-domains. Since processors are still free, we can parallelize this step of computing by allocating to each processor one matrix block in accordance with its rank. At the end of this task, a general collection using the primitive MPI_Allgather is necessary to collect the matrix blocks on different processors.

Creation of communicators

In order to optimize the communication process, it was necessary to study the problem and to take account of its features. Indeed, the resolution on two domains requires an exchange of information between the two sub-domains and more the number of processors increases more this exchange is important. Thus, we need an efficient communication process in order to reduce the induced overhead.

Notably, to exchange information between sub-domains, a first idea consists of looping on the Send/Recv, what can be penalizing. Besides, a test would be obligatory in the loop to determine the rank of the destination processor. Therefore, the solution that seems to be interesting was to divide the set of processes (MPI_COMM_WORLD) into subgroups on which we can do some collective communication operations. Thus, each created subset will have its own communication space. We divided the global communicator into four subgroups via the command MPI_Comm_split (A command that create a new communicators based on colors and keys [21])

Notice that in Uzawa and conjugated gradient procedures, there are some global operations (such as the scalar product and the sum of two vectors) that cannot be done locally in every sub-domain. So, we took account of this constraint by using global reduction operations MPI_Allreduce of the MPI library with an adequate distribution of data followed by a general collection.

7. Experimental evaluations

We present in this section, the different experimental results gotten after applying several tests on programs of resolution of Stokes problem. Let's note that all measures presented in this section are given by the average of five tests on the parallel machine IBM SP2 at least. The implementation was done with C language and MPI library for communication.

7.1. Resolution without domain decomposition

Execution time

From curves of figure 8 and figure 9 presented below, we deduce that for the two versions, the execution time increases when we raise the discretization parameter N. we precise that for the non tensorized sequential version, we cannot specify the execution time for N beyond 21 because the machine IBM SP2 does not answer anymore. Therefore, the first iterative version can be executed for N beyond 21 only on two, four and eight processors.

We mention also that, for N=9, the execution time on eight processors (3,26 seconds) is superior to the one on four processors (2,61 seconds). We explain this by the fact that for this parameter, the number of operations done by every processor is too small in comparison with the communication time spent for exchanging information between the eight processors. Thus, it is necessary to raise the discretization parameter when we increase the number of processors so that the execution time dominates the communication time. In fact, for N=17, the execution time on four processors is 726,95 seconds whereas on eight processors it is 448,71 seconds.

Concerning the second tensorized iterative version, we did several tests for different values of the discretization parameter and on a number of processors that varies until four. The same interpretations made for the first version remain valid here, but we must specify the fact that this version is more robust since it enables solving the problem for N=21 (in the sequential resolution).

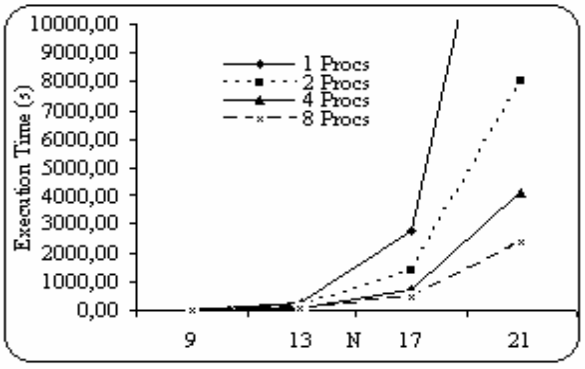

Figure 8. Curves of execution times of the non tensorized version of the resolution without domain decomposition

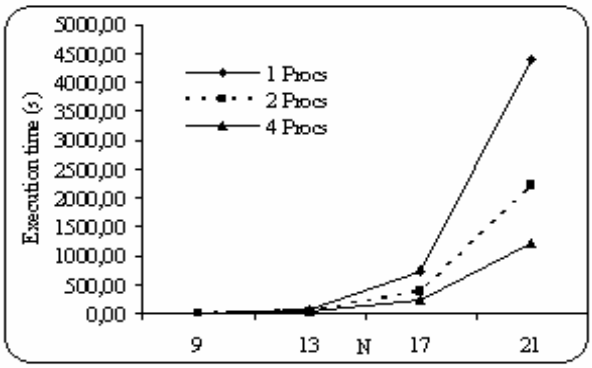

Figure 9. Curves of execution times of the tensorized version of the resolution without domain decomposition

Speed-up and efficiency

We succeeded to reach good speed-up. Indeed, we show in figure 10 and figure 11 that the speed-up rises with the increase of the discretization parameter N. Particularly, for N=19, the speed-up (=1,98) nearly reaches its optimal value on two processors (=2).
Concerning speed-up gotten with four processors, we notice that for a small discretization parameter (N=9), we cannot reach a good speed-up (=2,51 for the first version and 1,31 for the second one). Indeed, the calculation volume is less then the communication volume between the four processors. Therefore, the communication cost degrades the performances of the parallel program. But, when we increase N until 21, we can reach speed-up near to its optimal value on four processors (3,87 for the first version and 3,62 for the second one).

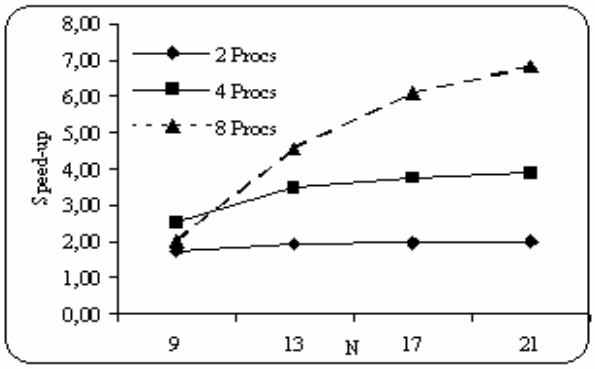

Figure 10. Curves of speed-up of the non tensorized version of the resolution without domain decomposition

After the study done about the speed-up, we analyze the efficiency of the parallelization of the non-tensorized iterative version. The histograms of figure 12 and figure 13 show that when we fix the number of processors and vary the discretization parameter value, the efficiency rises. Indeed, when we raise the problem size, the speed-up improves whatever is the number of processors, what entails an improvement of the efficiency. But, when we fix the discretization parameter (for example N=17) and continue increasing the number of processors, the efficiency is reduced. Indeed, if we increase the number of processors, the communication cost rises. Therefore, the speed-up and the efficiency decrease. Thus, the increase of the number of processors requires an increase of the problem size in order to guarantee the efficiency of the parallelization; this is called "scalability" [22].

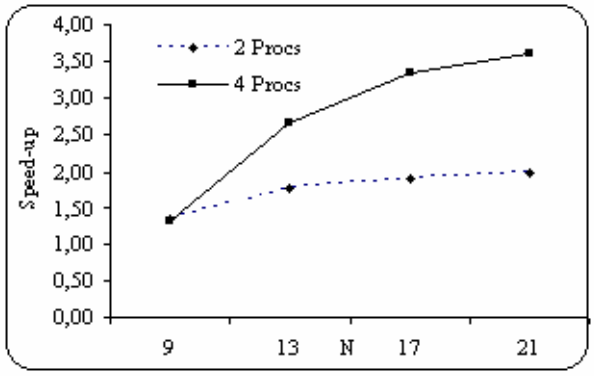

Figure 11. Curves of speed-up of the tensorized version of the resolution without domain decomposition

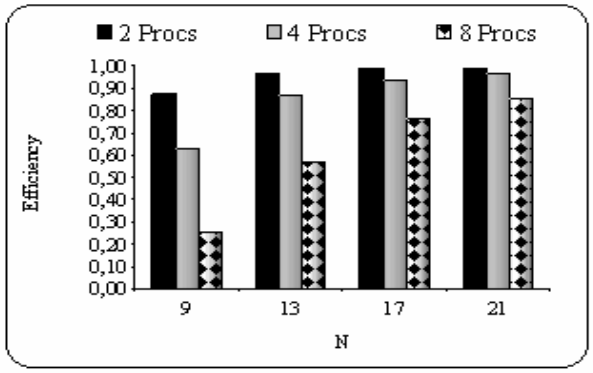

Figure 12. Histogram of efficiency of the non tensorized version of the resolution without domain decomposition

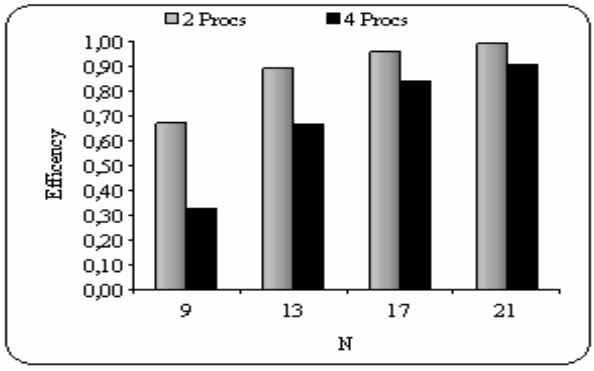

Figure 13. Histogram of efficiency of the tensorized version of the resolution without domain decomposition

Comparison of two versions

In order to put in evidence the improvement brought by the second iterative version, we present a comparative study between this version and the first one.

According to the curve of the figure 14, we can notice the big difference between times of execution gotten in the two versions for different values of discretization parameter. Indeed, we can deduce a constant result (≈3,7) while dividing times of execution of the two versions for each value of the parameter. Since the first version algorithm does not respond for N=21 and the execution of the program of the second tensorized version lasts 4385,55 seconds, we can deduce the execution time of the first version using the extrapolation by multiplying 4385,55 by 3,7.

These measures permit to confirm improvements brought by tensorized version concerning the gain of calculation cost and memory.

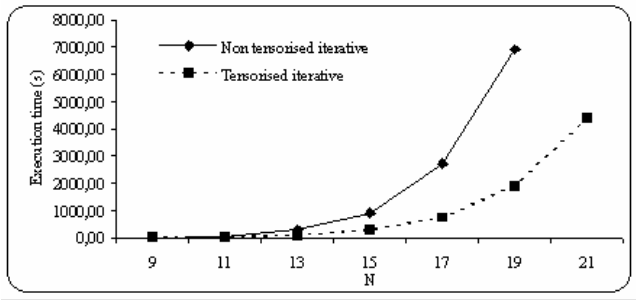

Figure 14. Curves of comparison of the two versions of the resolution without domain decomposition

7.2. Resolution on two domains with mortar spectral element method

This section is dedicated to the study of different results gotten while doing tests on the parallel versions of the resolution on two sub-domains.

Execution time

The curve of the figure 15 shows that the execution time grows when we the discretization parameter rises. Particularly, when N varies from 11 to 13, we notice a big raise of execution time. Let's note, that the sequential program does not respond anymore beyond N=13.

On the other hand, the improvement brought when applying the tensorization technique on the first version decreases the number of operations necessary in the resolution. Therefore, the parallelization of this version is efficient only for the resolution with a high discretization parameter. Indeed, figure 16 shows that the sequential execution, for N=5, gives an execution time (0,19) lower then the one found for the parallel execution on two (0,37) or on four processors (1,86). But when the discretization parameter increases until 21, we clearly see the contribution of the parallelization. In fact, the execution time decreases from 2291,33 to 1163,77 on two processors and 652,37 on four processors.

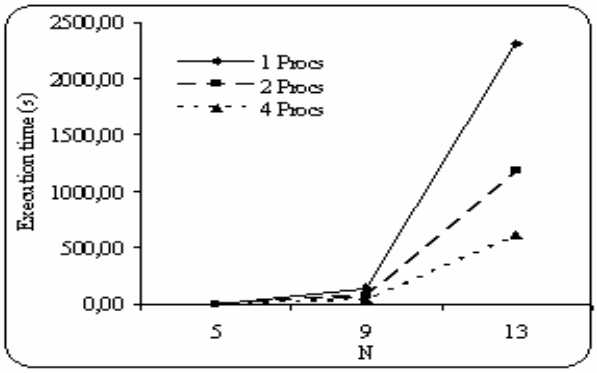

Figure 15. Execution time of the non tensorized version of the resolution on 2 domains

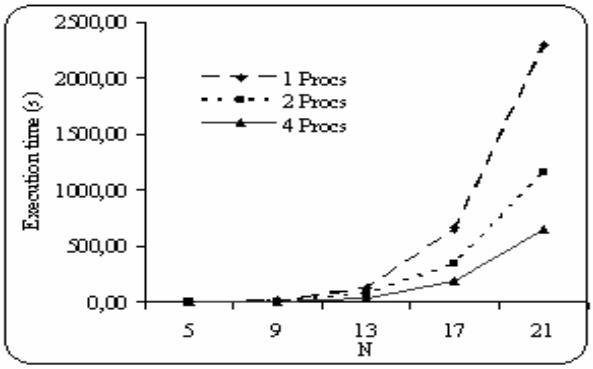

Figure 16. Execution time of the tensorized version of the resolution on 2 domains

Speed-up

For the non-tensorized iterative version, measures have gotten show that the speed-up for two processors is very low when the discretization parameter is small. On the other hand, we reach a

very good speed-up (≈1,96), when the parameter increases until 13 (see figure 17). On four processors, we note that the speed-up is mediocre (=1,73) for a small discretization parameter (=5).
Whereas, when this parameter increases until 13, we reach a speed-up near to its optimal value (=3,81). Therefore, we note that the speed-up improves when the calculation cost dominates in the total execution time.
For the second tensorized version, the results gotten show that the contribution of the parallelization of this version on two processors appears only when the resolution is done for a discretization parameter which is important enough.
Indeed, we note according to figure18 that the speed-up for two processors is very low when N=5 or N=7. That's due to the fact that the volume of data exchanged between the two processors generates a communication cost which is higher then the calculation one.
Therefore, the parallel execution time is dominated by the communication cost which reduces the speed-up. For four processors, the speed-up rises for important values of discretization parameter. On the other hand, the speed-up is low when this parameter is small. For example, for N=5, the speed-up is equal to 0,10. Thus, for small values of N, it is better to do the resolution on only one processor.
The two figures show that, for two or four processors, the speed-up is improved when the size of the problem rises.

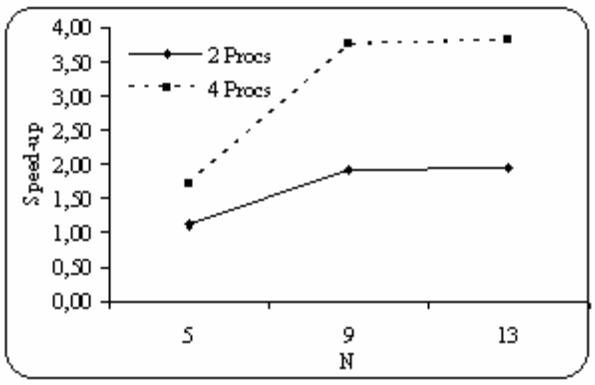

Figure 17. Speed-up of the non tensorized version of the resolution on two domains

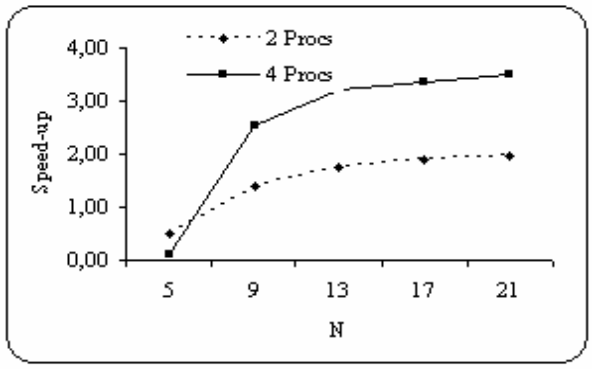

Figure 18. Speed-up of the tensorized version of the resolution on two domains

Comparison of two versions

This section is dedicated to the study of the difference between the two implementations of the two versions conceived for the resolution on two sub-domains by the mortar spectral element method. Indeed, we notice according to the gotten measures that the non-tensorized iterative version cannot solve the problem for a discretization parameter superior then 13 (see figure 19).
On the other hand, we could increase this parameter until 21 for the second tensorized iterative version. We notice a big difference between times of execution given by the first version and those given by the second one.
Thus, the speed-up (in sequential algorithm) gotten by tensorized version is rising while increasing N and varies from 3,74 (N=5) to 17,38 (N=13). Thus, we have succeeded to prove the improvement brought by the tensorization technique.

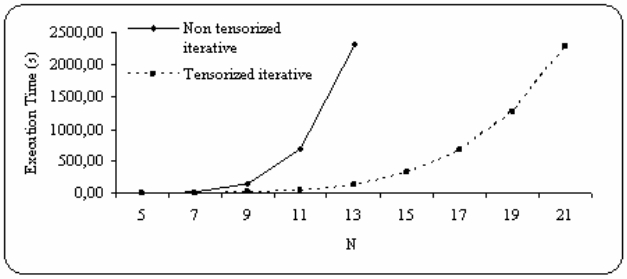

Figure 19. Comparison of the two versions of the resolution on two domains

7.3. Synthesis

We noted that in term of execution time, the iterative tensorized version is more interesting than the non-tensorized one. The second performs much more operations than the first and this is for the resolution on only one domain as well as on two sub-domains.
Concerning the speed-up, we note that it improves with the increase of the discretization parameter. This improvement is clearer when the cost of communication is small. Let's note that this factor

depends on the parallel machine performance. Let's note, besides, that we could reach a good speed-up on two, four and eight processors.
Concerning the efficiency, we saw that, on one hand, for a number of processors, it strongly depends on the raise of the discretization parameter value. On the other hand, for a value of this parameter, the increase of the number of processors degrades the efficiency. This confirms that the type of problem we landed is gluttonous in time of calculation and requires more processors when the problem size rises.
Let's note that for implementing the two versions of the resolution on two domains, we modified the two matrixes B1 and B2, what induced a reduction of the precision gotten in comparison with the resolution without domain decomposition.
Finally, we highlight the big interest of the parallelization of resolution of problems of this type. Indeed, on one hand, the adopted parallelization approach enabled us to study the resolution for high values of discretization parameter. On the other hand, we considerably reduced the execution time for the different landed resolutions.

8. Conclusion and perspectives

The parallel machines constitute a means permitting to improve performances of methods of resolution of various problems met in the engineer's sciences. Particularly, Stokes problem which deals with the calculation of the velocity and the pressure of a fluid flow. Our objective was to develop parallel applications for the resolution of this problem without domain decomposition, then with decomposition on two sub-domains.
In this work, we presented the sequential algorithms, as well as approaches of parallelization of the resolution on only one domain and the resolution on two sub-domains by the mortar spectral element method. Thus, we have conceived and implemented two versions for each resolution on a parallel machine IBM SP2: the first is an iterative non-tensorized version and the second is an iterative tensorized one.
The study of performances achieved through different tests done on this machine, showed the big difference, in term of execution time, between the iterative tensorized version and the non-tensorized one. We got some elevated speed-up for the different adopted parallelization approaches. Indeed, we could reach 1,98 on two processors, 3,87 on four processors and 6,82 on eight processors. Let's note that the speed-up improves while raising the discretization parameter value. Finally, we underline that the parallelization of this kind of sequential applications not only lead to time reduction but allowed solving large sized problems.
We mention some perspectives that seem to be interesting:
Conception and implementation of such type of approaches on grid computing. Thus, we will be able to study more complex forms with a mixed mesh which includes the decomposition by spectral element and finite element.
Study and simulation of the non-stationary fluid flow. This requires the resort to more numerically precise diagrams since an important range of physical ladders must be represented so that the calculation can be meaningful on times which could be long.

Acknowledgment

We would like to thank Professor Zaher MAHJOUB for his fruitful remarks and help.

References

[1] K.Arrow, L.Hurwicz, H.Uzawa, Studies in Nonlinear Programming, Standford University Press, (1958).
[2] E.Alexandre, Calcul Scientifique, Cours, Université de Liège, Octobre, (2000).
[3] C.Bernardi, Y.Maday, Approximations spectrales de problèmes aux limites elliptiques, Springer-Verlag France, Paris (1992).
[4] C. Bernardi, Y. Maday, A.T Patera, Domain decomposition by the mortar element method, Asymptotic and Numerical Mathods for Partial Differential Equations with Critical Pararmeters, ed. H.G. Kaper & M. Garbey, N.A.T.O. ASI Series C 384, Kluwer (1993), 269-286.
[5] C.Bernardi, Y.Maday, A.T.Patera, A new nonconforming approach to domain decomposition: the mortar element methods, Collège de France Seminar XI, ed. H. Brezis & J.-L. Lions, Pitman (1994), 13-51.
[6] D.Braess, Finite Elements.Theory, Fast Solvers and Applications in Solid Mechanics, Cambridge University Press (1997).
[7] J.Chaskalovic, Méthode des elements finis pour les sciences de l'ingénieur, ed. Tec et Doc, (2004).
[8] N.Chorfi, Traitement de singularités géométriques par la méthode des éléments spectraux avec joints, PHD Thesis, University of Paris VI, (1997).
[9] P.G.Ciarlet, Introduction à l'analyse numérique matricielle et à l'optimisation collection

" Mathématiques Appliquées pour la Maîtrise ", Masson, Paris (1982).
[10] A.J.Chorin, J.E.Marsden, A mathematical introduction to fluid mechanics, Springer-verlag, New York, (1993).
[11] D.Euvrard, Resolution numerique des equations au derivees partielles, Masson, (1994)
[12] D.Gottlieb, S.A.Orszag, Numerical Analysis of Spectral Methods, Theory and applications, SIAM Publications, Philadelphia, (1997).
[13] IBM guidebook, Volume 1, Hardware and Physical Environment, GA22-7280, (1999).
[14] IBM guidebook, Volume 2, Control Workstation and Software Environment, GA22-7281, (1999).
[15] V.Legat, Mathématique et Méthodes numériques ou les aspects facétieux du calcul sur ordinateur, University of Louvain, Canada, (2001).
[16] J.T.Oden, Finite Elements: an introduction of Numerical Analysis, Vol. II, ed. P.G.Ciarlet et J.L.Lions, North-Holland (1991), 3-15.
[17] S.A.Orszag, Comparaison of pseudospectral and spectral approximations, Stud. Appl. Math. 51 (1972), 253-259.
[18] M.Touihri, Discrétisation spectrale des équations de Navier-Stokes à densité variable, PHD Thesis, University of Paris VI, (1998).
[19] http://esm2.imt-mrs.fr/gar/efhtml
[20] http://www-irma.u-strasbg.fr
[21] http://www.idris.fr/data/cours/parallel
[22] http://mirage.imag.fr/Parallelisme/Main/SousParties/generalites.htm#scalabilite
[23] http://www.esstt.rnu.tn/utic/

Heithem Abbes received his Master degree in Computer Science in 2005, and his Engineer Diploma in Computer Science in 2003 from the Faculty of Sciences of Tunis. Since 2006, he is working towards his PhD at the research unit UTIC in Tunisia in collaboration with the laboratory LIPN in France. His research is focused on Grid Computing and Peer to Peer systems.

Nejmeddine Chorfi received his PhD degree from Pierre and Marie Curie University (Paris, France) in 1997. He is currently a full professor at the Department of Mathematics at the faculty of Science (Tunis, Tunisia). His research interests include Scientific computing for solving equations derived from physics. He runs several projects in cooperation with other universities and laboratories (Jaccques Louis Lions Laboratory – Paris, university of annaba – Algeria...).

Mohamed Jemni is a Professor at Ecole Supérieure des Sciences et Techniques de Tunis (ESSTT), University of Tunis in Tunisia. He obtained the HDR (Habilitation à Diriger des Recherches) in Computer Science from University of Versailles, France in 2004. He received his PhD in Computer Science from University of Tunis in 1997 and the Engineer Diploma in Computer Science from Faculty of Sciences of Tunis in 1991. He is the Head of the Research Unit of Technologies of Information and Communication of the University of Tunis and the director of research Master in Computer Science at ESSTT. He published more than 50 papers in international journals and conferences. His Research Projects Involvement are High performance Computing; Algorithmic and tools for Grid computing and Advanced research in e-learning.