Available online at www.sciencedirect.com
ScienceDirect
J. Parallel Distrib. Comput. 67 (2007) 581–591
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

MPI implementation of parallel subdomain methods for linear and nonlinear convection–diffusion problems

Ming Chau[a], Didier El Baz[b,*], Ronan Guivarch[a], Pierre Spiteri[a]

[a] *ENSEEIHT-LIMA-IRIT, 2, rue Charles Camichel, 31071 Toulouse, France*
[b] *LAAS-CNRS, 7, avenue du Colonel Roche, 31077 Toulouse Cedex 4, France*

Received 25 January 2006; received in revised form 28 December 2006; accepted 5 January 2007
Available online 31 January 2007

Abstract

The solution of linear and nonlinear convection–diffusion problems via parallel subdomain methods is considered. MPI implementation of parallel Schwarz alternating methods on distributed memory multiprocessors is discussed. Parallel synchronous and asynchronous iterative schemes of computation are studied. Experimental results obtained from IBM-SP series machines are displayed and analyzed. The benefits of using parallel asynchronous Schwarz alternating methods are clearly shown.

Keywords: IBM-SP series machines; MPI; Parallel computing; Asynchronous iterative algorithms; Schwarz alternating method; Subdomain methods; Convection–diffusion problems

1. Introduction

Convection–diffusion problems occur in many domains such as finance and hydraulics. The discretization of these problems leads to very large scale systems of algebraic equations. The introduction of parallelism via decomposition techniques can be very attractive and overlapping subdomain methods, as the Schwarz alternating method, can be very efficient when they are applied to the solution of these algebraic systems of equations (see [18]). This last remark is particularly true in the case of problems with nondifferentiable nonlinearities (see [3]). For more details on additive and multiplicative Schwarz alternating schemes, reference is made to [9–11,19,20].

The purpose of this paper is to show how we have implemented parallel overlapping subdomain methods via MPI on distributed memory multiprocessors. In particular, parallel asynchronous iterative schemes of computation are considered. In the case where several subdomains are assigned to each processor, the combination of parallel asynchronous iterative schemes of computation with the Schwarz alternating method permits one to obtain a behavior similar to the one of a multiplicative Schwarz alternating method.

We recall that the class of asynchronous iterations is a general class of parallel algorithms, whereby the components of the iterate vector can be updated in parallel via several processors, without any order nor synchronization (see [2,5,8,16,21,22]). The class of asynchronous iterations is well known for its practical interest (see, for example, [5,15]). This class has been extended in [23] (see also [13]). We have proposed in the above references a new class of parallel algorithms: asynchronous iterations with order intervals or with flexible communication. In the former class of parallel algorithms, computations make use of values of the components of the iterate vector which are generated in the end of updating phases. In the latter class, a partial update, i.e. the current value of any component of the iterate vector, which is not necessarily labelled by an outer iteration number, can be used at any time in the computations. Thus, flexible data exchange between processors are allowed. As a consequence, the coupling between communication and computation can be improved. In the partial ordering context, faster convergence can also be expected. The theory of asynchronous iterations with flexible communication has been considered in [12,13,23]. Mathematical models have been proposed and convergence results have been given in different theoretical

* Corresponding author. Fax: +33 5 61 33 69 69.
E-mail addresses: Ming.Chau@enseeiht.fr (M. Chau), elbaz@laas.fr (D. El Baz), Ronan.Guivarch@enseeiht.fr (R. Guivarch), Pierre.Spiteri@enseeiht.fr (P. Spiteri).

0743-7315/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2007.01.003

contexts. Convergence detection has also been studied in this series of papers. The convergence of asynchronous iterations with flexible communication was shown in the context of contracting operators (see [12]). Monotone convergence results for asynchronous iterations with flexible communication have also been given in the partial ordering framework (see [13,23]). We have considered in [23] the solution of the algebraic system

$$\mathcal{A}(U^*) = 0, \tag{1.1}$$

where $\mathcal{A}$ is a surjective M-function according to Rheinboldt (see [25]). It is shown in [23] that if asynchronous iterative algorithms with flexible communication have an initial guess U^0 satisfying $\mathcal{A}(U^0) \geqslant 0$, then under the weak assumption that recent values of the components of the iterate vector are increasingly used as the computation progresses, asynchronous iterative algorithms with flexible communication converge monotonically to the unique solution U^*, i.e. the algorithms generate monotone decreasing sequences of vectors $\{U^k\}$, which start from U^0 and converge to U^*,

Similarly, the solution of problem (1.1) via the Schwarz alternating method leads to the solution of the system

$$\widetilde{\mathcal{A}}(\widetilde{U}) = 0, \tag{1.2}$$

where $\widetilde{\mathcal{A}}$ is the augmented system obtained from problem (1.1) (see [23]). In this case, $\widetilde{\mathcal{A}}$ is also an M-function (see [23]). Thus, the theoretical results obtained in the case of problem (1.1) can be extended to problem (1.2). Finally, note that for nonlinear boundary value problems, as nonlinear convection–diffusion problems, suitable discretization schemes can also lead to M-function mappings.

Asynchronous iterations with flexible communication have been applied to the solution of several classes of nonlinear boundary value problems. Numerical results for the obstacle problem are displayed and analyzed in [26]. Numerical experiments on a shared memory machine have been carried out for the nonlinear diffusion problem (see [27]). Asynchronous iterations with flexible communication have also been applied to a class of convex optimization problems, i.e. network flow problems (see [13]).

In this paper, we concentrate on an efficient MPI implementation of parallel Schwarz alternating methods on distributed memory multiprocessors. Several parallel iterative schemes of computation are combined with the Schwarz alternating method. Parallel asynchronous iterative schemes of computation with flexible communication and synchronous iterative schemes of computation are compared. We consider in detail communication management. Finally, computational results obtained from IBM-SP series machines are reported and analyzed.

Section 2 deals with convection–diffusion problems. The Schwarz alternating method and parallel iterative schemes of computation are presented in Section 3. Section 4 deals with the implementation of parallel Schwarz alternating algorithms on distributed memory machines. Computational results are displayed and analyzed in Section 5. Appendices A and B display parallel codes. Computational results for 2D test problems are briefly shown in Appendix C.

2. Convection–diffusion problems

In this section, we present linear and nonlinear convection–diffusion problems. For the sake of clarity and simplicity, problem formulation is given in the 2D case.

2.1. Linear case

Consider the following linear convection–diffusion problem:

$$\begin{cases} -\nu\Delta u + a\dfrac{\partial u}{\partial x} + b\dfrac{\partial u}{\partial y} + cu = f, & \text{everywhere in } \Omega, \\ u = 0 \quad \text{on } \partial\Omega, \end{cases} \tag{2.1}$$

where $c \geqslant 0$, $\nu > 0$, Ω is a bounded domain, f is a given function of $\mathcal{L}^2(\Omega)$ and $\partial\Omega$ denotes the boundary of Ω. For the sake of simplicity, we assume that the discretization grid of the domain Ω is uniform. In the sequel h denotes the discretization step-size. We assume that the columns of the discretization grid are numbered naturally. The discretization of the operators which occur in problem (2.1) is made according to the following rules: the Laplacian is discretized via the classical five point scheme and the first derivatives are discretized as follows according to the sign of a and b

$$\frac{\partial u}{\partial x} = \begin{cases} \dfrac{u(x,y) - u(x-h,y)}{h} + \mathcal{O}(h) & \text{if } a > 0, \\ \dfrac{u(x+h,y) - u(x,y)}{h} + \mathcal{O}(h) & \text{if } a < 0. \end{cases} \tag{2.2}$$

Let A denote the discretization matrix of problem (2.1). If c is strictly positive, then regardless the sign of a and b, it follows from (2.2) that off-diagonal entries of matrix A are nonpositive and diagonal entries of A are positive. Moreover, the matrix A is strictly diagonally dominant; thus, A is an M-matrix.

If $c = 0$, then we can show that the matrix A is diagonally dominant. Moreover, by using the characterization of irreducible matrices (see [24]) we can verify that the matrix A is irreducibly diagonally dominant. Thus, A is an M-matrix.

Consider now a red-black ordering of the columns of the grid and let $\hat{A}$ be the corresponding discretization matrix derived from A by a permutation which preserves the sign of the entries. We consider the former discretization scheme; if c is strictly positive, then the matrix $\hat{A}$ is strictly diagonally dominant; if $c = 0$, then we can show analogously that the matrix $\hat{A}$ is irreducibly diagonally dominant. Thus, in both cases $\hat{A}$ is an M-matrix.

Finally, we note that under realistic hypotheses, the finite element discretization matrix occurring in some analogous linear partial differential equations is also an M-matrix (see [1]).

2.2. Nonlinear case

In this subsection, we consider different types of nonlinear convection–diffusion problems where nonlinearities arise on the boundary or in the domain (see [4]).

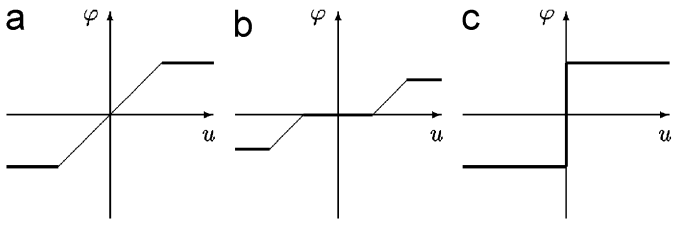


Fig. 1. Different graphs for φ.

2.2.1. Nonlinearities on the boundary

The first application is a boundary temperature control problem modelled as follows:

$$\begin{cases} -\nu\Delta u + a\dfrac{\partial u}{\partial x} + b\dfrac{\partial u}{\partial y} + cu = f \quad \text{everywhere in } \Omega, \\ \dfrac{\partial u}{\partial n} + \varphi(u) = 0 \ \text{ on } \quad \Gamma_d \quad \text{and} \quad u = 0 \ \text{ on } \partial\Omega - \Gamma_d, \end{cases} \tag{2.3}$$

where $\Omega \subset \mathbb{R}^2$, $c \geqslant 0$, $\Gamma_d \subset \partial\Omega$, $f \in \mathcal{L}^2(\Omega)$ and $\varphi : \mathbb{R} \to \mathbb{R}$ is a continuous, nondecreasing, nonlinear function. Fig. 1 displays some examples of graphs for function φ.

In particular, the graphs (a) and (b) model saturation phenomena and the graph (c) models a multivalued function corresponding to the boundary condition: $\frac{\partial u}{\partial n} + \varphi(u) \ni 0$. The discretization techniques presented in the previous subsection can be used for the interior points of domain Ω. For all points in Γ_d, the discretization of the Neumann condition leads to the solution of the following discrete equations:

$$\frac{u_j - u_{j-1}}{h} + \varphi(u_j) = 0. \tag{2.4}$$

Thus, we have to solve the problem

$$\mathcal{A}(U) = AU + \phi(U) - g = 0, \tag{2.5}$$

where A is the discretization matrix associated with the linear part of the equations, ϕ is a diagonal, nondecreasing operator and $(g, U) \in \mathbb{R}^{\dim(A)} \times \mathbb{R}^{\dim(A)}$. It follows from (2.4) that the jth component of ϕ is equal to $h\varphi(u_j)$ if j is the index of a point which belongs to Γ_d and is null if j corresponds to an interior points of Ω. If $c > 0$, then it follows from (2.4) and the Dirichlet condition defined on $\partial\Omega - \Gamma_d$ that the matrix A is a strictly diagonally dominant. Thus, A is an M-matrix. In the case where $c = 0$, we can verify by a similar argument that the matrix A is irreducibly diagonally dominant, regardless the sign of a and b, thus, A is an M-matrix. Since A is an M-matrix and ϕ is a continuous, nondecreasing, diagonal mapping, $\mathcal{A}$ is an M-function, according to Theorem 13.5.6 in [24], i.e. $\mathcal{A}$ is off-diagonally monotone decreasing and inverse monotone increasing. The results of this subsection can also be extended to the case where a red-black ordering is considered.

Note that the particular case of convection–diffusion problems with Neumann conditions defined everywhere on $\partial\Omega$ can also be considered. Then, the above analysis still holds when the condition $c > 0$ is satisfied.

2.2.2. Nonlinearities in the domain

We turn now to the case where nonlinearities are defined in the domain Ω. The general model can be given as follows:

$$\begin{cases} -\nu\Delta u + a\dfrac{\partial u}{\partial x} + b\dfrac{\partial u}{\partial y} + cu + \varphi(u) = f \quad \text{in } \Omega, \\ B.C., \end{cases} \tag{2.6}$$

where $c \geqslant 0$, $f \in \mathcal{L}^2(\Omega)$, $\varphi : \mathbb{R} \to \mathbb{R}$ is a continuous, nondecreasing function and B.C. represents a classical boundary condition, i.e. Dirichlet, Neumann, Robin or mixed. The discretization techniques quoted in the beginning of this section give rise to the same operator as in (2.5) and we are also in an M-function framework. The following nonlinear functions: $\varphi(u) = e^{\alpha u}$, with $\alpha > 0$, $\varphi(u) = \mathrm{Log}(\beta + \delta u)$, with $\delta > 0$ and a suitable sign for β, can be considered.

3. Algorithms

In this section, we present some background material on the Schwarz alternating method and parallel iterative schemes of computation. For the sake of clarity and simplicity, domain decomposition is presented in the 2D case.

3.1. The Schwarz alternating method

The effectiveness of domain decomposition methods is well known for boundary value problems. These methods are also well suited to parallel computing (see [18]). We concentrate here on parallel Schwarz alternating methods, which are based on overlapping subdomains.

Problem (2.1) can be decomposed into α subproblems as follows. For $i = 1, \ldots, \alpha$,

$$\begin{cases} -\nu\Delta u_i + a\dfrac{\partial u_i}{\partial x} + b\dfrac{\partial u_i}{\partial y} + cu_i = f_i \quad \text{everywhere in } \Omega_i, \\ u_{i_{/\Gamma_i}} = 0, \\ u_{i_{/\gamma_i^1}} = u_{i-1_{/\gamma_i^1}} \quad \text{for } 2 \leqslant i \leqslant \alpha, \\ u_{i_{/\gamma_i^2}} = u_{i+1_{/\gamma_i^2}} \quad \text{for } 1 \leqslant i \leqslant \alpha - 1, \end{cases} \tag{3.1}$$

where u_i and f_i, respectively, are the restriction of u and f, respectively, to Ω_i, $\Omega = \bigcup_{i=1}^{\alpha} \Omega_i$, $\Omega_i \bigcap \Omega_{i+1} \neq \emptyset$, for $i \in \{1, \ldots, \alpha - 1\}$, $\gamma_i^1 = \partial\Omega_i \bigcap \Omega_{i-1}$, for $i \in \{2, \ldots, \alpha\}$, $\gamma_i^2 = \partial\Omega_i \bigcap \Omega_{i+1}$, for $i \in \{1, \ldots, \alpha - 1\}$ and $\Gamma_i = \partial\Omega_i \bigcap \partial\Omega$, for $i \in \{1, \ldots, \alpha\}$ (see Fig. 2).

The decomposition (3.1) corresponds to an overlapping subdomain decomposition, whereby u_i is computed using the restriction of u_{i-1} and u_{i+1}, respectively, on γ_i^1 and γ_i^2, respectively. In the sequential case, the scheme of computation corresponds exactly to a multiplicative Schwarz scheme. In the parallel case, the Schwarz alternating method can be combined with an asynchronous iterative scheme of computation with flexible communication in order to be as close as possible to a multiplicative scheme.

Consider now the most general case, i.e. the nonlinear case. We have $\mathcal{A}(U) = AU + \phi(u)$. Assume that A is an M-matrix

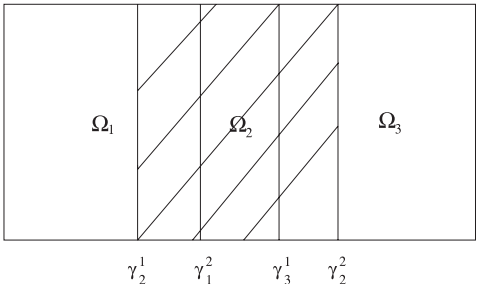


Fig. 2. An example with three subdomains.

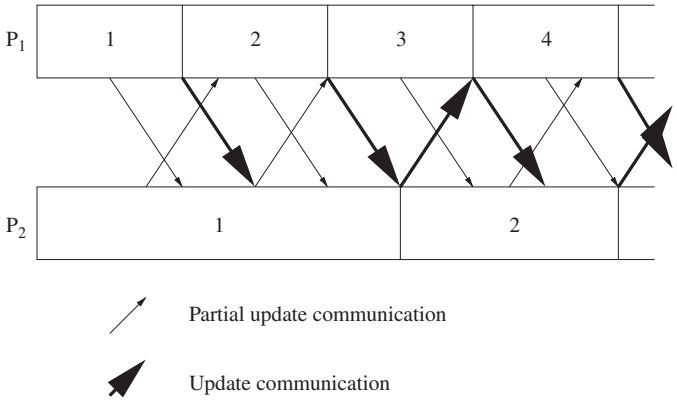


Fig. 3. A simple example of asynchronous algorithm with flexible communication.

and $\phi(u)$ is a monotone increasing mapping. Then, we are in a context similar to (2.5). If we solve the nonlinear simultaneous equations $\mathcal{A}(U) = 0$, via the Schwarz alternating method, then the augmentation process of the Schwarz alternating method transforms the M-matrix A into an M-matrix $\widetilde{A}$ and the monotone increasing mapping ϕ into the monotone increasing mapping $\widetilde{\phi}$ (see [14,23]). Thus, the resulting nonlinear mapping $\widetilde{\mathcal{A}}$ is a surjective M-function and we are in the convergence analysis framework considered in [23].

Note that any direct or iterative method can be used on each subdomain. In the case where $v = 0$, the discretization matrix is triangular. Thus, a relaxation method converges in only one iteration when the scanning order of the grid matches the triangular discretization matrix. In the case where v is small, the discretization matrix is nearly triangular. The entries associated with a triangular part of the matrix derived from the decentered discretization scheme have higher order of magnitude than other entries. A relaxation method can then be a quasi-direct method on each subdomain and its computational cost can be very low.

3.2. Parallel iterative schemes of computation

The Schwarz alternating method has been combined with two parallel iterative schemes of computation: an asynchronous iterative scheme with flexible communication and a synchronous one. We recall that parallel asynchronous iterative algorithms with flexible communication are general iterative methods whereby iterations are carried out in parallel by up to β processors without any order nor synchronization, with $2 \leqslant \beta \leqslant \alpha$ (see [13,12,23]). The main feature of this class of parallel iterative methods is to allow flexible data exchange between the processors. The value of the components of the iterate vector which is used in an updating phase may come from updates which are in progress and which are not necessarily labelled by an outer iteration number. Fig. 3 displays the typical behavior of parallel asynchronous iterations with flexible communication in the simple case where two processors, denoted by P_1 and P_2, respectively, exchange data. In Fig. 3, boxes and arrows, respectively, represent updating phases and communications, respectively.

4. Implementation

Parallel Schwarz alternating algorithms have been carried out on distributed memory machines via MPI. We have implemented both synchronous and asynchronous iterative schemes of computation in the 3D case.

4.1. Asynchronous algorithms with flexible communication

The SPMD code presented in Appendix A was carried out on several IBM-SP series machines. Convergence detection was performed via a snapshot algorithm (see [5, Section 8.2, 6]). Convergence occurs when a given predicate on a global state is true. An usual predicate corresponds to the fact that the iterate vector generated by the asynchronous iterative algorithm is sufficiently close to a solution of the problem (see [5, p. 580]). Several subdomains, i.e. parallepipeds, are assigned to each processor in order to implement a strategy of relaxation which is close to the multiplicative strategy. Each processor updates the components of the iterate vector associated with its subdomains and computes the residual norm corresponding to the subdomains in order to participate to the convergence detection.

The efficiency of parallel algorithms strongly depends on the communication frequency within the computations. In the code presented in Appendix A, communication frequency increases when the number of relaxations, denoted by `N`, decreases.

The Fortran code displayed in Appendix B shows how we have implemented communications. Point-to-point communications between two processes have been implemented using persistent communication request. Communication with the same argument list is repeatedly executed; it corresponds to data transmission of successive values of the components of the iterate vector associated with a subdomain frontier. That is the reason why persistent communication request has been used. Persistent communication request can be thought of as a communication port or a half-channel. This approach permits one to reduce the communication overhead between the process and the communication controller. A persistent communication request is created by MPI_SSEND_INIT or MPI_RECV_INIT, respectively, in the transmitter or the receiver, respectively. For the sake of robustness, we have used a synchronous mode send operation since ready mode is unsafe and buffered mode may

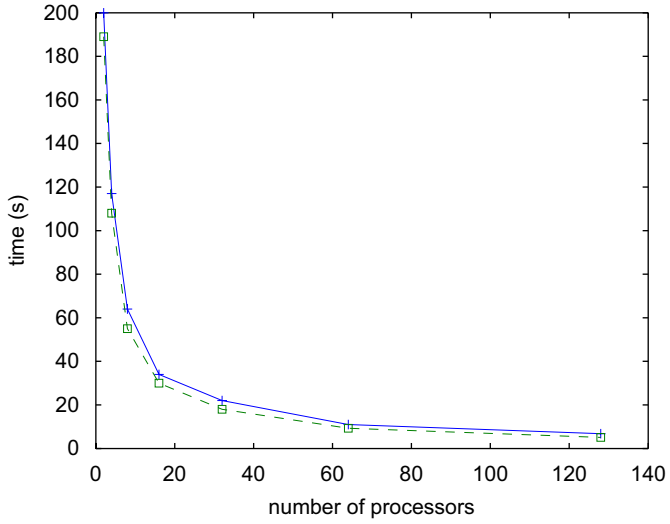


Fig. 4. 3D problem, $\nu = 0.01$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P690+, elapsed time of synchronous algorithms (solid) and asynchronous algorithms (dashed).

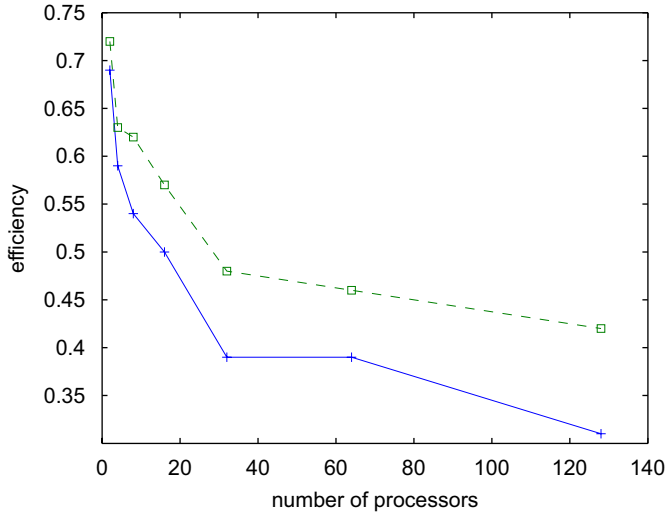


Fig. 5. 3D problem, $\nu = 0.01$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P690+, efficiency of synchronous algorithms (solid) and asynchronous algorithms (dashed).

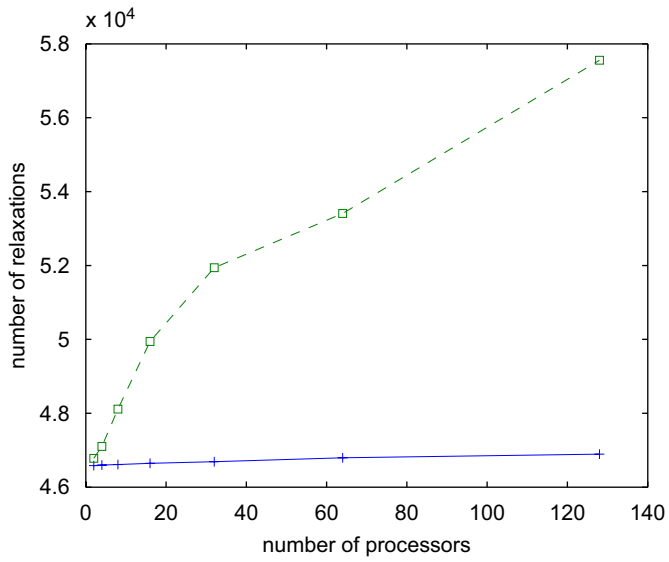


Fig. 6. 3D problem, $\nu = 0.01$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P690+, number of relaxations of synchronous algorithms (solid) and asynchronous algorithms (dashed).

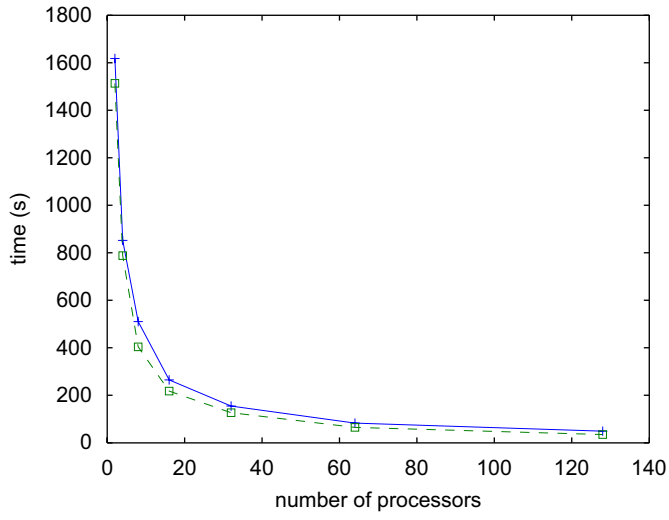


Fig. 7. 3D problem, $\nu = 0.1$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P690+, elapsed time of synchronous algorithms (solid) and asynchronous algorithms (dashed).

lead to overflow in the high communication frequency case. Note that the use of a synchronous mode send operation is not in contradiction with the implementation of asynchronous iterations since the implementation of communication layers and the type of computation scheme are independent. Persistent communication requests are activated by MPI_START. Reception tests have been made via MPI_TEST.

If global convergence is detected, then computations can be terminated and resources can be freed. All persistent communication requests are cancelled via the MPI_CANCEL function and finally suppressed via the MPI_REQUEST_FREE function. Note that cancellation of send requests must occur before cancellation of receive requests; otherwise data exchange based on rendezvous mechanism may fail. Thus, all processes are synchronized via the MPI_BARRIER function before the cancellation of receive requests. For more details on the implementation of asynchronous iterative schemes of computation, the reader is referred to [7] (see also [17]).

4.2. Synchronous algorithms

Implementation of parallel synchronous iterative schemes of computation was based on the blocking reception of boundary values. The MPI_WAITALL function was preferred to the MPI_WAIT function since programming is easier and overhead is reduced with the former function. The termination order of requests is totally handled by MPI with the MPI_WAITALL

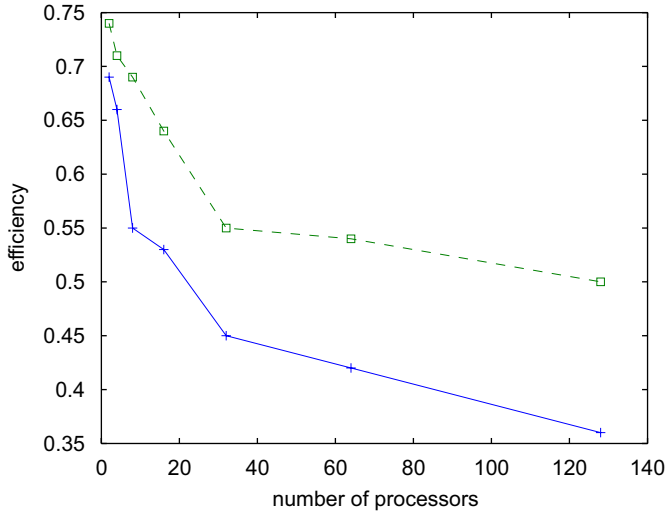


Fig. 8. 3D problem, $\nu = 0.1$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P690+, efficiency of synchronous algorithms (solid) and asynchronous algorithms (dashed).

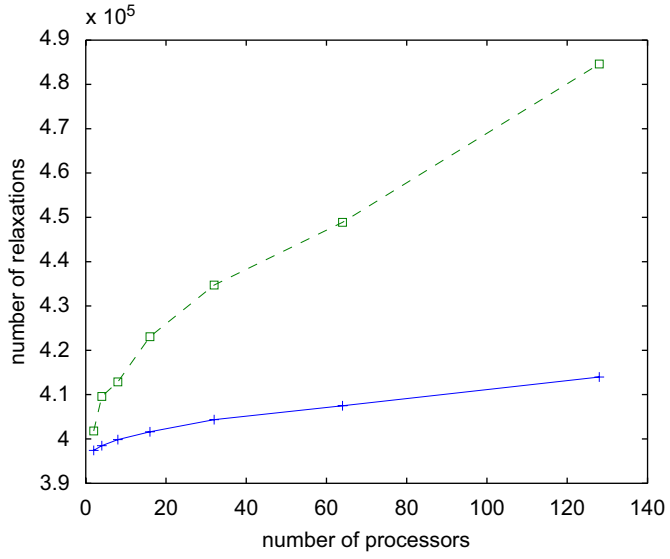


Fig. 9. 3D problem, $\nu = 0.1$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P690+, number of relaxations of synchronous algorithms (solid) and asynchronous algorithms (dashed).

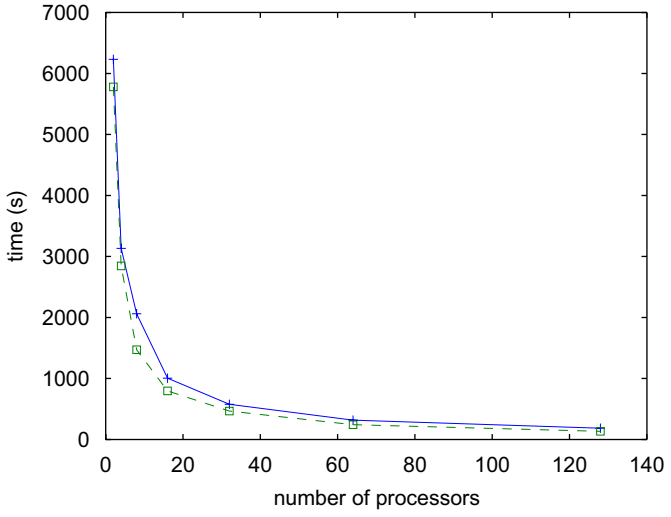


Fig. 10. 3D problem, $\nu = 1$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P690+, elapsed time of synchronous algorithms (solid) and asynchronous algorithms (dashed).

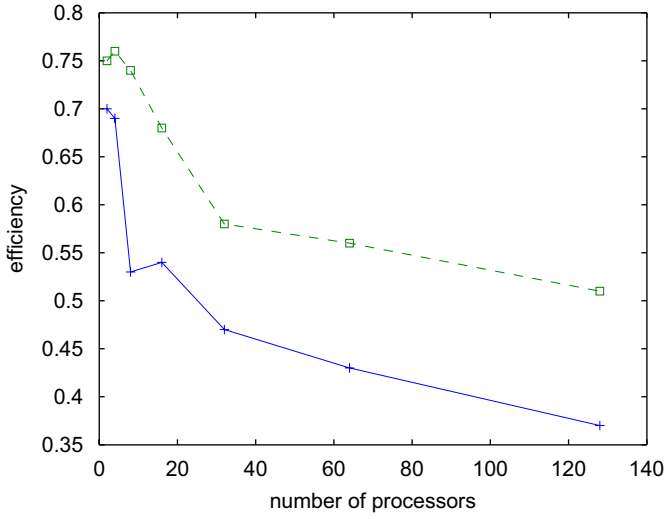


Fig. 11. 3D problem, $\nu = 1$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P690+, efficiency of synchronous algorithms (solid) and asynchronous algorithms (dashed).

function. On the other hand, the use of MPI_WAIT would require the programmer to handle the termination order. We do not present here the code we have implemented, since implementation is straightforward in this case (the code is less complex than the one quoted in the previous subsection devoted to asynchronous iterations). Reference is made to [7,17] for implementation details concerning parallel synchronous iterative algorithms.

5. Numerical experiments

Computational experiments were carried out on several IBM-SP series machines in CINES and IDRIS computing centers. More precisely, the main support of our experiments was an IBM-SP4 with 12 SMP nodes of 32 P690+ processors (1.3 GHz); nodes are connected via a Federation network (1.6 Gbits per seconds). In order to improve the effectiveness of parallel iterative algorithms, we have also used an IBM-SP4 machine with six sets of 16 nodes of four P655 processors (1.3 GHz). In the latter architecture, nodes are also connected via a Federation network with similar transfer rate, however bandwidth is better used since there are less processors per node. We have used up to 128 processors.

We present now the main computational results for 3D problems. The reader is referred to Appendix C for some test problems in the 2D case.

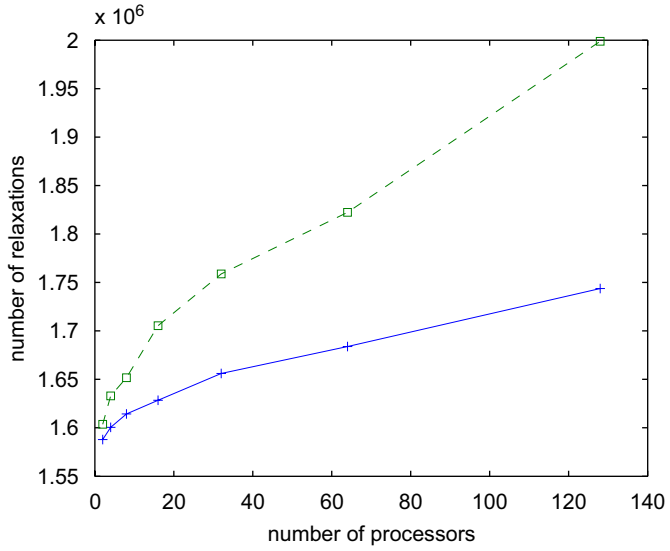


Fig. 12. 3D problem, $\nu = 1$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P690+, number of relaxations of synchronous algorithms (solid) and asynchronous algorithms (dashed).

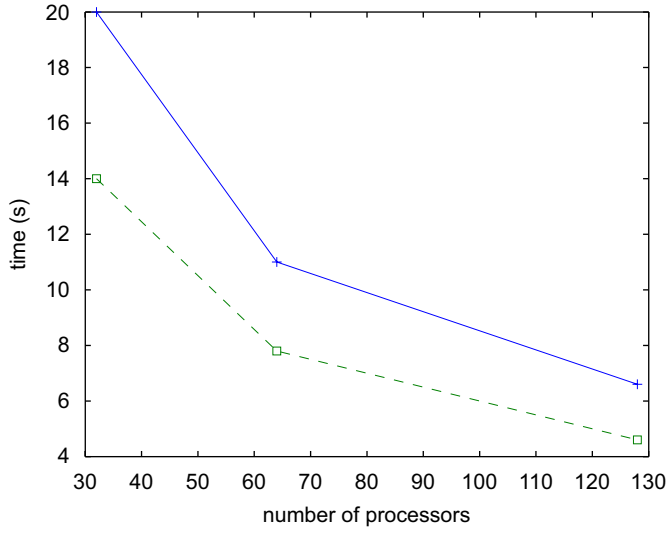


Fig. 13. 3D problem, $\nu = 0.01$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P655, elapsed time of synchronous algorithms (solid) and asynchronous algorithms (dashed).

Figs. 4, 7 and 10, respectively, display the elapsed time of parallel iterative algorithms for different values of the number of processors in the case of 3D linear problems with the following convection parameters: 0.5, 1.5 and -0.5, $c = 10$, $\nu = 0.01$, $\nu = 0.1$ and $\nu = 1$, respectively, where ν is the diffusion parameter. For all experiments, we have considered 3,750,000 discretization points and 256 well balanced, cubic subdomains. Figs. 5, 8 and 11 show the efficiency of parallel iterative algorithms in function of the number of processors for different values of the diffusion parameter. The number of relaxations is given in Figs. 6, 9 and 12. These computational results were obtained from an IBM-SP4 machine with 12 nodes of 32 P690+ processors (See Figs. 4–12).

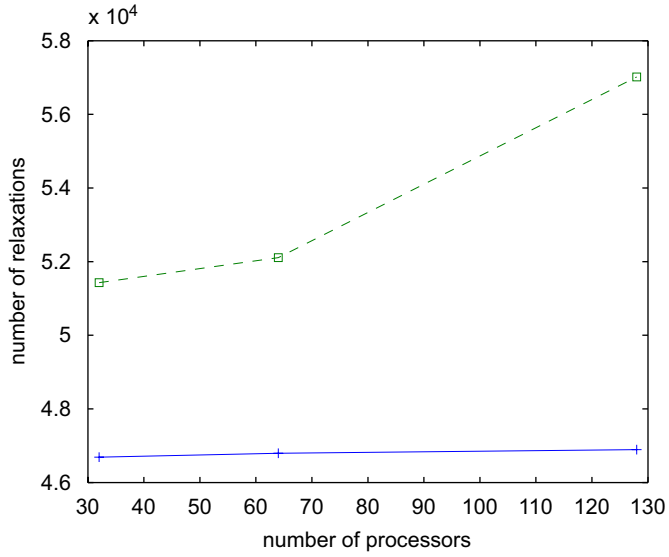


Fig. 14. 3D problem, $\nu = 0.01$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P655, number of relaxations of synchronous algorithms (solid) and asynchronous algorithms (dashed).

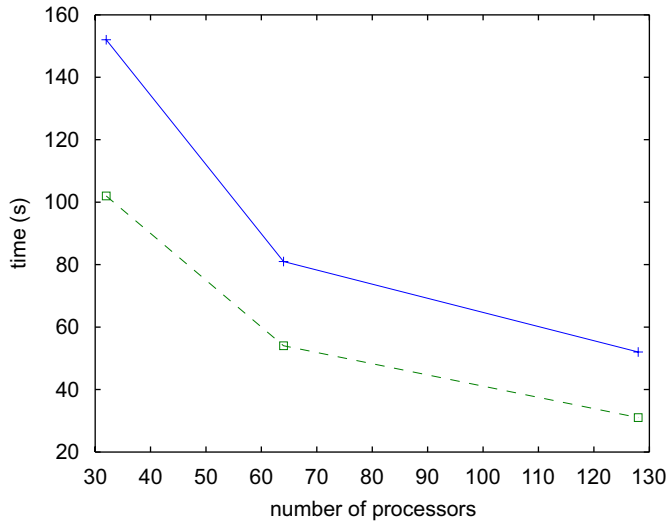


Fig. 15. 3D problem, $\nu = 0.1$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P655, elapsed time of synchronous algorithms (solid) and asynchronous algorithms (dashed).

We have tested several communication frequencies for data exchange. The tuning of the number of relaxations was made experimentally. We present here results in the case where data exchange occurs every two relaxations on each subdomain. All points of a subdomain are updated twice by the relaxation procedure first forward, then backward, as in SSOR scanning. Reception of boundary values occurs in the beginning of each updating phase. For sake of effectiveness, a different subdomain is considered after a communication. The subdomains assigned to a processor are treated cyclically according to a red-black ordering. Experimentally, this strategy turned out to be the most efficient one.

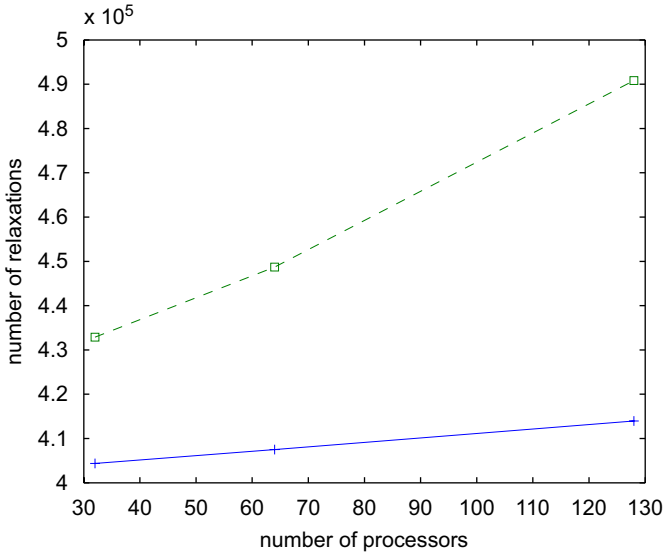


Fig. 16. 3D problem, $v = 0.1$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P655, number of relaxations of synchronous algorithms (solid) and asynchronous algorithms (dashed).

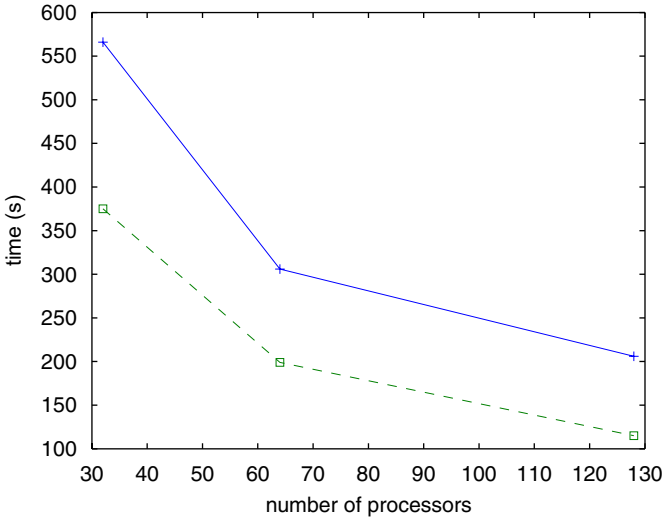


Fig. 17. 3D problem, $v = 1$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P655, elapsed time of synchronous algorithms (solid) and asynchronous algorithms (dashed).

We note that the elapsed time curves in Figs. 4, 7 and 10 present the same shape. We note also that the elapsed time decreases with the value of v. From Figs. 6, 9 and 12, we see that the number of relaxations increases with the number of processors. In the case of parallel synchronous schemes of computation, this phenomenon is mainly due to slight modifications in the order of treatment of the different subdomains; in the case of asynchronous schemes of computation, this fact is mainly due to the chaotic behavior of the algorithm. Finally, we note that asynchronous algorithms with flexible communication are more efficient than synchronous algorithms. It turns out that the overhead generated by additional relaxations in the case of asynchronous algorithms is smaller than the synchronization

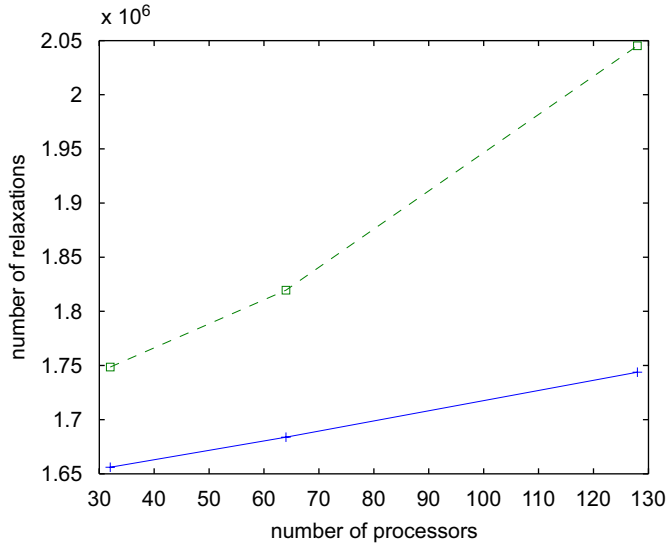


Fig. 18. 3D problem, $v = 1$, 3,750,000 nodes, 256 subdomains, IBM-SP4 P655, number of relaxations of synchronous algorithms (solid) and asynchronous algorithms (dashed).

overhead combined with processor idle time of parallel synchronous schemes of computation. Moreover, the efficiency of synchronous algorithms decreases faster than the efficiency of asynchronous algorithms when the number of processors increases.

Figs. 13–18 display experimental results obtained from an IBM-SP4 machine with six sets of 16 nodes of four P655 processors for the same problem as the one considered previously in this section, with the same parameters, number of discretization points and subdomains. Figs. 13, 15, and 17, respectively, show the elapsed time of parallel iterative algorithms for different values of the number of processors and $v = 0.01$, 0.1 and 1, respectively. Figs. 14, 16 and 18 give the number of relaxations for the different parallel iterative algorithms. Experiments with up to 128 processors were run. Note that it was not possible to perform experiments with less than 32 processors on the latter architecture, due to the enforced policy in the computing center. These results are particularly interesting in order to study the effect of the architecture on the effectiveness of the different parallel algorithms. In particular, we note that the elapsed time is reduced due to a better use of the network bandwidth.

6. Conclusions

In this paper, we have studied the solution of linear and nonlinear convection–diffusion problems via parallel subdomain methods. We have proposed efficient MPI implementations of parallel Schwarz alternating methods on distributed memory multiprocessors. Effective communication mechanisms that rely on persistent communication request have been used. Parallel synchronous and asynchronous iterative schemes of computation have been tested. Computational results obtained from IBM-SP series machines have clearly shown the benefits of using parallel algorithms. Experimental results have also shown that asynchronous iterative algorithms are more efficient

than synchronous algorithms and that a better use of network bandwidth can improve efficiency. In future work, experiments will be carried out on various parallel architectures including networks of processors and grids.

Acknowledgments

Part of this study was made possible by a grant of CINES, Montpellier and IDRIS-CNRS, Paris. The authors would also like to thank the referees for their helpful comments.

Appendix A. SPMD code of asynchronous iterative algorithms with flexible communication

```
do until global convergence
   for each subdomain assigned to the processor do
      if local convergence is not reached then
         for i in 1..N do
            receive the latest frontier values
            relaxation
         end do
         send the frontier values to the neighbors
      end if
   end do
end do
```

Appendix B. Implementation of persistent communication

```
! SEND THE FRONTIER VALUES TO THE NEIGHBORS
! SEND SOME FRONTIERVALUES!
Some messages may not be sent at this stage...
 do i = 1, NSEND
   SND_DELAY(i) = SND_DELAY(i) + 1
 end do
 call MPI_TESTSOME(NSEND,sndt,nout,outarray,starray,ierr)
 do i = 1, nout
   mpos = outarray(i)
   call MSGPK(... PACKING A MESSAGE ...)
   call MPI_START(sndt(mpos),ierr)
   SND_DELAY(mpos) = 0
 end do
! SEND THE LAST FRONTIER VALUES
! ... however, when local convergence is reached,
! we ensure that unsent messages are sent.
 do i = 1, NSEND
   if(SND_DELAY(i).gt.0)then
     call MPI_WAIT(sndt(i),starray(:, i),ierr)
     call MSGPK(... PACKING A MESSAGE ...)
     call MPI_START(sndt(i),ierr)
     SND_DELAY(i) = 0
   end if
 end do
! RECEIVE THE FRONTIER VALUES FROM THE NEIGHBORS

! RECEIVE SOME FRONTIER VALUES
call MPI_TESTSOME(NRECV,rcvt,nout,outarray,starray,ierr)
do i = 1, nout
 mpos = outarray(i)
 call MSGUPK(... UNPACKING A MESSAGE ...)
 call MPI_START(rcvt(mpos),ierr)
end do
```

Appendix C. Computational results: 2D case

In order to be exhaustive, we briefly present in this appendix a series of computational results for test problems in the 2D case. Results are given for linear and nonlinear convection–diffusion problems presented in Section 2.

Tables C1–C3 display numerical results obtained with sequential and parallel Schwarz alternating algorithms for linear problem (2.1) with 130,305 discretization points, eight subdomains and different values of the diffusion parameter. We have considered the cases where $\nu = 1$, 0.1 and 0.01, respectively. Table C4 gives results for problem (2.1) with 92,837 discretization points, $\nu = 1$ and 16 subdomains and Fig. C1 shows the elapsed time of parallel algorithms for up to eight processors. Tables C5–C7, respectively, display elapsed time and efficiency of parallel Schwarz alternating algorithms applied to nonlinear problem (2.3) with $a = 0.5$, $b = 1.5$ and $c = 10$, nonlinearity (a) in the graph of Fig. 1, 130,305 discretization points, eight subdomains, $\nu = 1$, 0.1 and 0.01, respectively.

A relaxation method was used on each subdomain. In the case of a nearly triangular discretization matrix, the mesh was always scanned according to an order suited to the shape of the matrix (see last paragraph of Section 3.1). We note that the sequential solution of linear problem (2.1) with $\nu = 0.01$ takes 314.1 s when the mesh is scanned according to the above

Table C1
2D linear problem (2.1), $\nu = 0.01$, 130,305 nodes, eight subdomains

	Synchronous iterations		Asynchronous iterations	
Processors	Time (s)	Efficiency	Time (s)	Efficiency
1	314.1	–	–	–
2	198.3	0.79	180.0	0.87
4	126.2	0.62	113.5	0.69

Table C2
2D linear problem (2.1), $\nu = 0.1$, 130,305 nodes, eight subdomains

	Synchronous iterations		Asynchronous iterations	
Processors	Time (s)	Efficiency	Time (s)	Efficiency
1	2084.9	–	–	–
2	1245.2	0.84	1175.2	0.89
4	797.1	0.66	666.0	0.78

Table C3
2D linear problem (2.1), $\nu = 1$, 130,305 nodes, eight subdomains

	Synchronous iterations		Asynchronous iterations	
Processors	Time (s)	Efficiency	Time (s)	Efficiency
1	1986.3	–	–	–
2	1136.8	0.87	1002.9	0.99
4	660.8	0.75	553.8	0.90

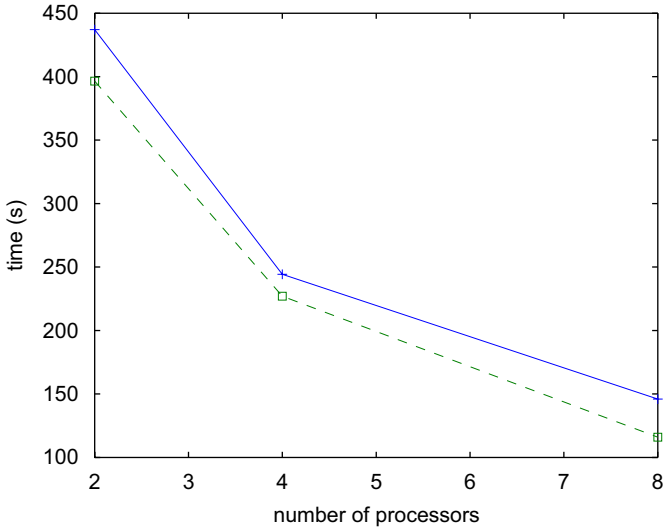


Fig. C1. 2D linear problem (2.1): $\nu = 0.01$ 130.305 nodes, eight subdomains, elapsed time of synchronous algorithms (solid) and asynchronous algorithms (dashed).

Table C4
2D linear problem (2.1), $\nu = 1$, 92,837 nodes, 16 subdomains

	Synchronous iterations		Asynchronous iterations	
Processors	Time (s)	Efficiency	Time (s)	Efficiency
1	839.4	–	–	–
2	437.1	0.96	396.5	1.06
4	244.3	0.86	227.0	0.93
8	146.0	0.72	115.9	0.91

Table C5
2D nonlinear problem (2.3), $\nu = 0.01$, 130,305 nodes, eight subdomains

	Synchronous iterations		Asynchronous iterations	
Processors	Time (s)	Efficiency	Time (s)	Efficiency
1	152.6	–	–	–
2	97.4	0.78	91.6	0.84
4	62.2	0.61	57.7	0.66

Table C6
2D nonlinear problem (2.3), $\nu = 0.1$, 130,305 nodes, eight subdomains

	Synchronous iterations		Asynchronous iterations	
Processors	Time (s)	Efficiency	Time (s)	Efficiency
1	831.1	–	–	–
2	517.1	0.81	473.3	0.88
4	331.2	0.63	278.4	0.75

quoted order (see Table C1); the solution of the same problem takes 398.3 s when the mesh is scanned according to the inverse order.

Table C7
2D nonlinear problem (2.3), $\nu = 0.1$, 130,305 nodes, eight subdomains

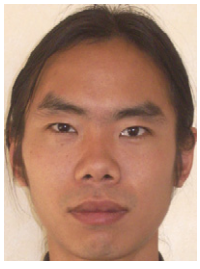
	Synchronous iterations		Asynchronous iterations	
Processors	Time (s)	Efficiency	Time (s)	Efficiency
1	2147.7	–	–	–
2	1247.4	0.80	1247.4	0.86
4	761.2	0.71	637.7	0.84

Tables C1–C7 show that parallel asynchronous algorithms with flexible communication are more efficient than synchronous algorithms. The lack of synchronization point and use of the current value of the components of the iterate vector lead to a better effectiveness for parallel Schwarz alternating methods. Note also that the efficiency of synchronous algorithms decreases faster than the efficiency of asynchronous algorithms when the number of processors increases. Moreover parallel algorithms become more efficient if two or more subdomains are assigned to each processor.

References

[1] O. Axelson, V. Barker, Finite Element Solution of Boundary Value Problems, Academic Press, Orlando, 1984.
[2] G.M. Baudet, Asynchronous iterative methods for multiprocessors, J. Appl. Comput. Math. 25 (1978) 226–244.
[3] A. Bensoussan, J.-L. Lions, Application des Inéquations Variationnelles en Contrôle Optimal Stochastique, Dunod, Paris, 1978.
[4] A. Bernudez, Contrôle par feedback a priori de systèmes régis par des équations aux dérivées partielles, Rapport de Recherche INRIA no. 288, 1978.
[5] D.P. Bertsekas, J. Tsitsiklis, Parallel and Distributed Computation, Numerical Methods, Athena Scientific, Englewood Cliffs, 1997.
[6] K.M. Chandy, L. Lamport, Distributed snapshots: determining global states of distributed systems, ACM Trans. Comput. Systems 3 (1) (1985) 63–75.
[7] M. Chau, Algorithmes Parallèles Asynchrones pour la Simulation Numérique, Ph.D. Thesis, INP Toulouse, 2005.
[8] D. Chazan, W. Miranker, Chaotic relaxation, Linear Algebra Appl. 2 (1969) 199–222.
[9] M. Dryja, An additive Schwarz algorithm for two and three dimensional finite element elliptic problems, in: T. Chan et al. (Eds.), Domain Decomposition Methods, SIAM, Philadelphia, 1989, pp. 147–153.
[10] M. Dryja, O.B. Widlund, Some domain decomposition algorithms for elliptic problems, in: L. Hager et al. (Eds.), Proceedings of the Conference on Iterative Methods for Large Linear Systems, Academic Press, San-Diego, CA, 1989, pp. 273–291.
[11] M. Dryja, O.B. Widlund, Toward a unified theory of domain decomposition algorithms for elliptic problems, in: T. Chan et al. (Eds.), Proceedings of the third international symposium on decomposition methods for partial differential equations, SIAM, Philadelphia, 1989, pp. 3–21.
[12] D. El Baz, A. Frommer, P. Spiteri, Asynchronous iterations with flexible communication contracting operators, J. Comput. Appl. Math. 176 (2005) 91–103.
[13] D. El Baz, P. Spiteri, J.C. Miellou, D. Gazen, Asynchronous iterative algorithms with flexible communication for nonlinear network flow problems, J. Parallel Distributed Comput. 38 (1996) 1–15.
[14] D.J. Evans, W. Deren, An asynchronous parallel algorithm for solving a class of nonlinear simultaneous equations, Parallel Comput. 17 (1991) 165–180.
[15] A. Frommer, D. Szyld, On asynchronous iterations, J. Comput. Appl. Math. 123 (2000) 201–216.

[16] L. Giraud, P. Spiteri, Résolution parallèle de problèmes aux limites non linéaires, M.2 A.N. 25 (1991) 73–100.
[17] R. Guivarch, Résolution Parallèle de Problèmes aux Limites Couplés par des Méthodes de Sous-Domaines Synchrones et Asynchrones, Ph.D. Thesis, INP Toulouse, 1997.
[18] K.H. Hoffmann, J. Zou, Parallel efficiency of domain decomposition methods, Parallel Comput. 19 (1993) 1375–1391.
[19] P.L. Lions, On the Schwarz alternating method I, in: R. Glowinski et al. (Eds.), Domain Decomposition Methods, SIAM, Philadelphia, PA, 1988, pp. 1–42.
[20] P.L. Lions, On the Schwarz alternating method II, in: T. Chan et al. (Eds.), Domain Decomposition Methods, SIAM, Philadelphia, PA, 1989, pp. 47–70.
[21] J.-C. Miellou, Itérations chaotiques à retards, étude de la convergence dans le cas d'espaces partiellement ordonnés, C.R.A.S., Paris 280 (1975) 233–236.
[22] J.-C. Miellou, Algorithmes de relaxation chaotique à retards, RAIRO R1, (1975) 55–82.
[23] J.C. Miellou, D. El Baz, P. Spiteri, A new class of asynchronous iterative algorithms with order interval, Math. Comput. 67 (1998) 237–255.
[24] J.M. Ortega, W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.
[25] W.C. Rheinboldt, On M-functions and their application to nonlinear Gauss–Seidel iterations and to network flows, J. Math. Anal. Appl. 32 (1970) 274–307.
[26] P. Spiteri, J.C. Miellou, D. El Baz, Asynchronous Schwarz alternating method with flexible communication for the obstacle problem, Calculateurs Parallèles, Réseaux et Systèmes Répartis 13 (2001) 47–66.
[27] P. Spiteri, J.C. Miellou, D. El Baz, Parallel Schwarz method for a nonlinear diffusion problem, Numer. Algorithms 33 (2003) 461–474.



Ming Chau was born in 1977; he graduated Engineer in Computer Sciences and Applied Mathematics from ENSEEIHT Toulouse, France in 2001 and received the Doctor degree in Computer Sciences and Applied Mathematics from Institut National Polytechnique de Toulouse (INPT) in 2005. Dr. Ming Chau has carried out researches in numerical simulation and parallel computing since 2000. He has taught C programming language and networking with Linux for four years at ENSEEIHT. Today, he works as an engineer and is involved in Social Economy.

Didier El Baz was born in Toulouse in 1958, he received the Doctor Engineer degree in Control Theory from INSA Toulouse in January 1984. Dr. El Baz was visiting scientist in the Laboratory for Information and Decision Systems, MIT, Cambridge, Massachussetts in 1984. He received the Habilitation à Diriger des Recherches degree in Computer Sciences from INPT in 1998. He is presently CNRS researcher in the LAAS-CNRS laboratory of Toulouse. His fields of interest are in parallel and distributed computing, peer to peer and grid computing, optimization, control theory, robotics and numerical analysis.

Ronan Guivarch is an Assistant Professor in the Computer Science and Applied Mathematics Laboratory of INPT. He received the Doctor degree in Computer Sciences in 1997. His thesis dealt with parallel asynchronous methods for the solution of coupled boundary problems. Dr. Guivarch has spent a postdoctoral year in CERFACS where he has worked on the development of a parallel library for Meso-NH, an atmospheric forecast code. In 1998, Dr. Ronan Guivarch joined INPT and he is now interested in numerical applications to be deployed on the Grid.

Pierre Spiteri was born in Bone, Algeria in 1943. He graduated in Mathematics at the University of Franche-Comté, Besançon, France in 1968. He received the Doctor degree in 1974 and the Docteur d'Etat es Sciences Mathématiques degree from the same university in 1984. He is now a full professor at ENSEEIHT in the Department of Applied Mathematics and Computer Sciences. He teaches numerical analysis, optimization and numerical solution of boundary value problems. His fields of interest are in numerical analysis, large scale nonlinear systems of evolution equations, optimal control, parallel computing and more particularly, domain decomposition methods for the solution of nonlinear boundary values problems.