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

Ming Chau\textsuperscript{a}, Didier El Baz\textsuperscript{b,∗}, Ronan Guivarch\textsuperscript{a}, Pierre Spiteri\textsuperscript{a}

\textsuperscript{a}ENSEEIHT-LIMA-IRIT, 2, rue Charles Camichel, 31071 Toulouse, France
\textsuperscript{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.

© 2007 Elsevier Inc. All rights reserved.

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
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
\[ \mathcal{A}(U^*) = 0, \]  
(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 \( \mathcal{A}(U^0) \geq 0 \), 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
\[ \tilde{\mathcal{A}}(\tilde{U}) = 0, \]  
(1.2)
where \( \tilde{\mathcal{A}} \) is the augmented system obtained from problem (1.1) (see [23]). In this case, \( \tilde{\mathcal{A}} \) 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:
\[
\begin{cases}
-\nu \Delta u + a \frac{\partial u}{\partial x} + b \frac{\partial u}{\partial y} + cu = f, & \text{everywhere in } \Omega, \\
u = 0 & \text{on } \partial \Omega,
\end{cases}
\]  
(2.1)
where \( \nu > 0, \nu > 0, \Omega \) is a bounded domain, \( f \) is a given function of \( L^2(\Omega) \) and \( \partial \Omega \) denotes the boundary of \( \Omega \). For the sake of simplicity, we assume that the discretization grid of the domain \( \Omega \) 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 \( b \):
\[
\frac{\partial u}{\partial x} = \begin{cases}
\frac{u(x,y) - u(x-h,y)}{h} + \mathcal{O}(h) & \text{if } a > 0, \\
\frac{u(x+h,y) - u(x,y)}{h} + \mathcal{O}(h) & \text{if } a < 0.
\end{cases}
\]  
(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 \( \hat{A} \) is irreducibly diagonally dominant. Thus, \( \hat{A} \) 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 \( A \) 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]).
2.2.1. Nonlinearities on the boundary

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

\[
\begin{aligned}
-\nu \Delta u + a \frac{\partial u}{\partial x} + b \frac{\partial u}{\partial y} + cu &= f \quad \text{everywhere in } \Omega, \\
\frac{\partial u}{\partial n} + \varphi(u) &= 0 \quad \text{on } \Gamma_d \quad \text{and} \quad u = 0 \quad \text{on } \partial \Omega - \Gamma_d, 
\end{aligned}
\]

(2.3)

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

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

\[
\frac{u_j - u_{j-1}}{h} + \varphi(u_j) = 0. 
\]

(2.4)

Thus, we have to solve the problem

\[
A(U) = AU + \phi(U) - g = 0, 
\]

(2.5)

where \( A \) is the discretization matrix associated with the linear part of the equations, \( \phi \) is a diagonal, non-decreasing operator and \( (g, U) \in \mathbb{R}^{\dim(A)} \times \mathbb{R}^{\dim(A)} \). It follows from (2.4) that the \( j \)-th component of \( \phi \) is equal to \( h \varphi(u_j) \) if \( j \) is the index of a point which belongs to \( \Gamma_d \) and is null if \( j \) corresponds to an interior point of \( \Omega \). 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 \( \phi \) is a continuous, non-decreasing, diagonal mapping, \( A \) is an \( M \)-function, according to Theorem 13.5.6 in [24], i.e. \( A \) 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 \( \Omega \). The general model can be given as follows:

\[
\begin{aligned}
-\nu \Delta u + a \frac{\partial u}{\partial x} + b \frac{\partial u}{\partial y} + cu + \varphi(u) &= f \quad \text{in } \Omega, \\
\text{B.C.} 
\end{aligned}
\]

(2.6)

where \( c \geq 0 \), \( f \in L^2(\Omega) \), \( \varphi : \mathbb{R} \to \mathbb{R} \) is a continuous, non-decreasing 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:

\[
\varphi(u) = e^{\beta u}, \quad \text{with } \alpha > 0, \quad \varphi(u) = \log(\beta + \delta u), \quad \text{with } \delta > 0 \text{ and a suitable sign for } \beta, 
\]

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 \( z \) subproblems as follows. For \( i = 1, \ldots, z \),

\[
\begin{aligned}
-\nu \Delta u_i + a \frac{\partial u_i}{\partial x} + b \frac{\partial u_i}{\partial y} + cu_i &= f_i \quad \text{everywhere in } \Omega_i, \\
u_{i,\Gamma_i} &= 0, \\
u_{i+1,\Gamma_i} &= u_{i-1,\Gamma_i} \quad \text{for } 2 \leq i \leq z, \\
u_{i+1,\Gamma_i} &= u_{i+1,\Gamma_i} \quad \text{for } 1 \leq i \leq z - 1, 
\end{aligned}
\]

(3.1)

where \( u_i \) and \( f_i \), respectively, are the restriction of \( u \) and \( f \), respectively, to \( \Omega_i \), \( \Omega_i = \bigcup_{j=1}^{i} \Omega_j \), \( \Omega_i \cap \Omega_{i+1} \neq \emptyset \), for \( i \in \{1, \ldots, z-1\} \), \( \gamma_i^1 = \partial \Omega_i \cap \Omega_{i-1} \), for \( i \in \{2, \ldots, z\} \), \( \gamma_i^1 = \partial \Omega_i \cap \Omega_{i+1} \), for \( i \in \{1, \ldots, z-1\} \) and \( \Gamma_i = \partial \Omega_i \cap \partial \Omega \), for \( i \in \{1, \ldots, z\} \) (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 \( \gamma_i^1 \) and \( \gamma_i^2 \), 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 \( A(U) = AU + \phi(U) \). Assume that \( A \) is an \( M \)-matrix.
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 $A(U) = 0$, via the Schwarz alternating method, then the augmentation process of the Schwarz alternating method transforms the $M$-matrix $A$ into an $M$-matrix $\tilde{A}$ and the monotone increasing mapping $\phi$ into the monotone increasing mapping $\tilde{\phi}$ (see [14,23]). Thus, the resulting nonlinear mapping $\tilde{A}$ 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 centered 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 $\beta$ processors without any order or synchronization, with $2 \leq \beta \leq x$ (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. parallelepipeds, 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
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
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.
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, $c = 10$, $v = 0.01$, $v = 0.1$ and $v = 1$, respectively, where $v$ 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).

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.
4. 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 \( \nu \). 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 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 \( \nu = 0.01, 0.1 \) 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 FRONTIER VALUES!

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, 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, 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 \( v = 1, 0.1 \) and 0.01, respectively.

Table C4 gives results for problem (2.1) with 92,837 discretization points, \( v = 1 \) 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, \( v = 1, 0.1 \) 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 \( v = 0.01 \) takes 314.1 s when the mesh is scanned according to the above

<table>
<thead>
<tr>
<th>Table C1</th>
<th>2D linear problem (2.1), ( v = 0.01 ), 130,305 nodes, eight subdomains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors</td>
<td>Time (s)</td>
</tr>
<tr>
<td>1</td>
<td>314.1</td>
</tr>
<tr>
<td>2</td>
<td>198.3</td>
</tr>
<tr>
<td>4</td>
<td>126.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table C2</th>
<th>2D linear problem (2.1), ( v = 0.1 ), 130,305 nodes, eight subdomains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors</td>
<td>Time (s)</td>
</tr>
<tr>
<td>1</td>
<td>2084.9</td>
</tr>
<tr>
<td>2</td>
<td>1245.2</td>
</tr>
<tr>
<td>4</td>
<td>797.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table C3</th>
<th>2D linear problem (2.1), ( v = 1 ), 130,305 nodes, eight subdomains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors</td>
<td>Time (s)</td>
</tr>
<tr>
<td>1</td>
<td>1986.3</td>
</tr>
<tr>
<td>2</td>
<td>1136.8</td>
</tr>
<tr>
<td>4</td>
<td>660.8</td>
</tr>
</tbody>
</table>
Fig. C1. 2D linear problem (2.1): $v = 0.01$ 130,305 nodes, eight subdomains, elapsed time of synchronous algorithms (solid) and asynchronous algorithms (dashed).

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

<table>
<thead>
<tr>
<th>Processors</th>
<th>Synchronous iterations</th>
<th>Asynchronous iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Efficiency</td>
</tr>
<tr>
<td>1</td>
<td>839.4</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>437.1</td>
<td>0.96</td>
</tr>
<tr>
<td>4</td>
<td>244.3</td>
<td>0.86</td>
</tr>
<tr>
<td>8</td>
<td>146.0</td>
<td>0.72</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Processors</th>
<th>Synchronous iterations</th>
<th>Asynchronous iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Efficiency</td>
</tr>
<tr>
<td>1</td>
<td>152.6</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>97.4</td>
<td>0.78</td>
</tr>
<tr>
<td>4</td>
<td>62.2</td>
<td>0.61</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Processors</th>
<th>Synchronous iterations</th>
<th>Asynchronous iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Efficiency</td>
</tr>
<tr>
<td>1</td>
<td>831.1</td>
<td>–</td>
</tr>
<tr>
<td>2</td>
<td>517.1</td>
<td>0.81</td>
</tr>
<tr>
<td>4</td>
<td>331.2</td>
<td>0.63</td>
</tr>
</tbody>
</table>

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


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, Massachusetts 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 value problems.