Synthetic presentation of iterative asynchronous parallel algorithms.

Pierre Spiteri

Synthetic presentation of iterative asynchronous parallel algorithms

P. SPITERI
IRIT- INPT, University of Toulouse, Toulouse, France

Abstract

This paper deals with a synthetic presentation of parallel iterative asynchronous algorithms and also to the questions related to this kind of method such as : study of the behavior of parallel iterative asynchronous methods, stopping tests, implementation of these previous algorithms, applications and efficiency.

Keywords: asynchronous parallel algoritm, high performance computing, iterative method, subdomain method, multisplitting methods, discretized pseudo-linear problem, large scale systems, nonlinear boundary value problems, optimization

1 Introduction

Numerical simulation is an approach that has become essential in many scientific and industrial fields, thanks in particular to the current computing power and storage capacity of current computers. Thus, we are currently observing the development of studies of physical, chemical, biological, economic and medical phenomena, all of which have in common that they are mathematically modeled on the basis of the laws governing these various phenomena. Among the fields where numerical simulation is used, we can mention avionics, chemical engineering, fluid mechanics, nuclear applications, plasma physics, meteorology, oceanography, molecular biology and many other applications, this list being not exhaustive.

The mathematical models obtained can be extremely complex to solve. Solving the equations resulting from modeling can be a major source of difficulty due to the use of large data structures and calculation results to store and also due to prohibitive calculation times. For example, when phenomena are modeled by partial differential equations, discretization methods using classical finite differences methods, finite elements methods or finite volume methods lead to the resolution of very large algebraic systems; indeed, the complexity and number of unknowns are all the higher the more stringent the requirements on the accuracy of the results. To satisfy the growing need for computing resources, the joint development of parallel architectures and algorithms is a highly explored way of investigation at the present time.

In contrast to parallel applications such as data mining, the parallel resolution of large algebraic systems, mainly of an iterative nature, especially when they are sparse, involves al-

gorithms that require extensive data exchange between the different computing units. Efficient parallelization is then difficult to implement for the type of target problems to be solved, due to the latency time induced by remote data access. These accesses are an even greater handicap when the computing units have to be synchronized during the communication phases. Indeed, the fact of having to wait until a distant data becomes available before continuing a calculation necessarily causes a degradation of the performance of a parallel algorithm. The problem raised becomes all the more inextricable when the number of processors increases and the frequency of exchanges is high. Moreover, any mismatch between the computational loads assigned to the different processors contributes to reducing the efficiency of algorithm parallelization.

Powerful means of computation are necessary for large scale numerical simulation. For example, such simulations are particularly relevant for the solution of large scale algebraic systems derived from the discretization of coupled nonlinear boundary values problems arising in multi-physics. Nowaday, only massively parallel efficient computer architectures are able to offer this power of computation. In this context, grid computing, peer to peer computing and cloud computing appear to be a challenging paradigms in order to match the need in high speed computing. Concerning the efficient numerical algorithms to be implemented on such parallel architectures, it can be noted that parallel asynchronous iterative numerical methods deserve particular attention since it is difficult to synchronize large number of processors on heterogeneous and massively parallel architectures. Moreover this kind of particular methods are well adapted when the clusters are distant and heterogeneous. Nevertheless, among the many challenging issues related to the efficient use of asynchronous algorithms on new parallel or distributed architectures, such as convergence study, and performance analysis, the influence of roundoff errors and the derivation of stopping criteria is particularly hard to analyze and important to estimate.

In this presentation we will avoid as much as possible the use of too mathematical formalism to prefer the presentation of underlying ideas and, given the large number of scientific works published in the field, we refer to the main bibliographical references for additional information. Moreover, since the subject is large, and includes both numerical and computer aspects, the presentation of numerical methods will be preferred, without neglecting the computer aspects which will also be discussed.

The paper is organized as follows. In section 2 we describe the parallel iterative asynchronous algorithms accomplished first in pioneering works and then extended subsequently. Section 3 presents the difficult problem of termination of parallel iterative asynchronous algorithms viewed both from computer science approach and also from numerical analysis approach. Section 4 presents some principles of implementation of parallel iterative asynchronous algorithms; implementation on multi - CPU and on multi - GPU are described and this section is ended by some considerations for load balancing of parallel asynchronous methods. Section 5 allows to present various applications of parallel iterative asynchronous methods while section 6 gives some details to obtain a good efficiency for using such parallel iterative method. Finally the presented paper is ending by a short conclusion.

2 Parallel asynchronous algorithms

2.1 Pioneering works

It is in order to overcome these performance loss problems previously presented that asynchronous parallel algorithms have been developed. This kind of method is currently one of the more explored for the use of massive parallelism because it naturally makes it possible to consider a great number of processors and avoids a rigorous load balancing. These asynchronous parallel methods, applicable to solve fixed point problems, do not require necessarily synchronizations during the calculation process and aim to take full advantage of computing power by eliminating idle times due to blocking time waits. As a result, the calculation of the components of the unknown vector is performed in parallel iteratively and without any a priori order. In this kind of method, the values of the components of the iterate vector used in the calculations come at most from the values calculated just before; so the too old values of the components of the iterate vector must be definitively rejected as the calculations progress. Thus, the latest available values of the recently obtained components are used. In addition, each sub-vector of the iterate vector is always updated. Finally, it should be noted that synchronous parallel iterations correspond to a particular case of asynchronous parallel iterations; in addition, for particular choices of strategies for updating the components, we find in the synchronous case, the classical methods of relaxation such as Jacobi's method, Gauss – Seidel's method and the Alternating Direction Implicit method (A.D.I.). So when we analyze the behavior of asynchronous parallel methods, we also analyze the behavior of synchronous parallel methods as well as that of classical sequential iterative methods, which are then only particular cases.

These asynchronous iterative algorithms were therefore developed out of the desire to make maximum use of all the computing resources of new computer architectures and more particularly of Multiple Instructions Multiple Data (M.I.M.D.) architectures, or Single Program Multiple Data (S.P.M.D.), by eliminating the synchronization management time and the processor inactivity time resulting from synchronizations. This type of algorithm was tested in 1967 by J.L. Rosenfeld [1] and the convergence of these methods was studied first in 1969 by D. Chazan and W. Miranker [2] using contraction techniques to solve large linear systems. Subsequently, pioneering works was carried out mainly by J.C. Miellou [3] which in 1975 extended the work of D. Chazan and W. Miranker to certain classes of highly non-linear problems, also using contraction techniques, non-linearities being able to be either analytical in the mathematical model to be solved or due to constraints on the solution to be computed, which then leads to the solution of multi-valued problems. In the case of the solution of large linear systems J.C. Miellou and P. Spiteri [4] (see also [5]) were able to establish convergence criteria that are very easy to use [6]; indeed, for example for the solution of a linear system

$$AU = F,$$

if, on the one hand, the diagonal blocks of the A matrix involved in the linear system to be solved are either strongly defined positive or with a strong diagonal dominance and that, on the other hand, the matrix norms of the off-blocks subordinated by the classical norms are relatively weak modules, one could highlight a Lipschitz vector condition of the kind

$$q(V - U^\star) \leq Jq(W - U^\star)$$

where $q(.)$ is a vectorial norm that takes into account the decomposition of the problem into large blocks, J is a non-negative matrix, $U^\star$ is the solution to the fixed point problem associated to the problem to solve, V is the current iteration and W represents the interaction values calculated by the other processors. If in addition this last matrix J has a spectral radius smaller than one, it is a contraction matrix and the asynchronous parallel methods are convergent. Note that the value of the spectral radius of the matrix J allows to evaluate the asymptotic convergence rate of the iterative method. This result can be extended to the case of pseudo - linear problems such as

$$AU + \Phi(U) = F$$

where the matrix A involved in the affine application has the same properties as in the linear case, the latter affine application being perturbed by an increasing diagonal application $U \rightarrow \Phi(U)$; under these assumptions alone, the asynchronous parallel methods are convergent. This result can then be further extended in the case where A matrix is an M-matrix, i.e. having negative or null off-diagonal entries and admitting a non-negative inverse [7] to [9]; under these assumptions, using a fairly technical proof, J.C. Miellou and P. Spiteri in [4] have shown that asynchronous parallel methods are convergent whatever be the considered block decomposition (see [4]-[5]). It should be noted that when we discretize the boundary value problems by classical finite differences method, variational finite differences method, finite volume method and under certain assumptions, especially if the angle conditions are verified, by finite element method, the discretization matrices obtained are M-matrices; if, in addition, parabolic or second order hyperbolic evolution problems are considered, these problems being discretized by implicit or semi - implicit time marching schemes, the global discretization matrices are still M-matrices. Note that in [3] and [10], when the fixed point mapping is contracting with respect to a uniform weighted norm, a sufficient condition of convergence was stated; such condition is verified when the fixed point mapping is contracting with respect to a vectorial norm [3].

In addition, J.C. Miellou in 1975 [11] and also in 1986 [12], and also other authors [13] - [14] analyzed this kind of method using the discrete maximum principle which, if the iterative process is properly started, i.e. if an initial guess is properly chosen, has the effect of obtaining ordered iterations that converge monotonously towards to $U^\star$ solution of the problem to be solved, either by decreasing from the initial data which is called an over-solution, or by increasing from a sub-solution; the assumptions necessary for the reliability of such methods analyzed by partial order techniques are simple: the application of fixed point must be continuous, off-diagonal monotone decreasing and diagonal strictly increasing. Such properties are verified if in the initial problem to be solved, the function allowing to formulate the problem is an M-function introduced by J. Rheinbold [15] , i.e. a surjective function, off diagonal decreasing and of inverse monotone; in the linear case an M-function is an M-matrix. Note that a pseudo-linear operator built from a affine application where the matrix is an M-matrix and perturbed by a growing operator is an M-function.

Whether it's D. Chazan and W. Miranker or J.C. Miellou it should be noted that the methods studied were called chaotic methods insofar as the information communicated between the processors was delayed in relation to each other by finite delays. An extension of chaotic parallel iterations was proposed in 1978 by G. Baudet [16], who thus introduced asynchronous parallel iterations where the information communicated between the processors was no longer finite but could have infinite communication delays, which made it possible to take into account possible failures of the multiprocessors; chaotic parallel methods then became a special

case of asynchronous parallel methods. Convergence analysis could be carried out either by contraction techniques or by partial ordering techniques when the function defining the problem to solve is an M-function. Another distinct approach to analyze the behavior of asynchronous parallel algorithms was also considered by D. Bertsekas and J. Tsitsiklis in 1989 [17]; in the study proposed by D. Bertsekas and J. Tsitsiklis, the successive iterations of the method were located in nested sets centered on the solution. Thus, at the limit these nested sets belong to smaller and smaller sets and the parallel iterative method converges.

Note that B. Lubachewski and D. Mitra [18] have also established a sufficient condition for convergence of asynchronous parallel iterations when delays are bounded; this study was applicable to the solution of singular Markovian systems. In this case, the matrix used in the model is irreducible, nonnegative and of spectral radius equal to one; it is a stochastic matrix, i. e. the sum of the coefficients of each line is equal to one, and B. Lubachewski and D. Mitra further assume that one of its diagonal coefficients is positive.

The analysis of the behavior of these asynchronous parallel methods by contraction, partial ordering and nested techniques are the main techniques for studying the iterative behavior of these methods. Asynchronous parallel methods have been studied by many authors in many laboratories on all continents. The aim was on the one hand to highlight more general computation methods by using various methods to analyze the behavior of asynchronous parallel methods and on the other hand to obtain more flexibility in inter-processor exchanges.

2.2 Extensions of parallel asynchronous methods

2.2.1 Sub-domain methods

In terms of the efficiency of the implemented parallel algorithms, it is necessary that they have a sufficiently high granularity. Indeed, too small calculation process sizes will have a negative effect on the efficiency of methods, an increase in communication cost and synchronizations between computation processes, which will degrade the performance of parallel methods. Therefore developers gather computational tasks together in order to avoid this type of algorithmic behavior.

For example in the case of the numerical solution of partial differential equations, this type of grouping of computational tasks leads to the development of sub-domain methods. There are various kinds of sub-domain methods; on the one hand, sub-domain methods without overlapping and on the other hand, sub-domain methods with overlapping. To be complete, it would also be necessary to mention the sub-structuring methods that will be described in the following sub-paragraph.

Sub-domain methods without overlap consist in grouping several adjacent blocks of the vector to be calculated to obtain a block of larger size. The analysis of the behavior of the iterative process with high granularity thus obtained follows from the results obtained previously and mentioned in paragraph 2.1 in the context of convergence analysis both by contraction techniques and by partial ordering techniques. Indeed, if the discretization matrix is an M-matrix, a situation to which we can often reduce ourselves by choosing discretization patterns properly, we have seen that in this case the asynchronous parallel methods analyzed by contraction techniques were convergent whatever be the coarser decomposition (see [4]-[5]). This result can also be extended to situations where the discretization matrix is not an M-matrix, but has appropriate properties; this is particularly the case for a convection - diffusion problem

discretized by a centered scheme where this property of M-matrix is not necessarily verified, especially when convection is dominant [19]. On the other hand, for the convection - diffusion problem, if a forward or backward decentered scheme is used, depending on the sign of the convection coefficient, the M-matrix character of the discretization matrix of the problem is verified [20]. With regard to the analysis of the convergence of asynchronous parallel methods in the linear case, thanks to the properties of the M-matrices, similar convergence results can be obtained by using partial ordering techniques (see [11]-[12]); indeed, the sub-problems obtained by grouping the diagonal blocks of the discretization matrix are also M-matrices since any principal minor of an M-matrix is also an M-matrix (see [8]); so the results established in the contraction framework also extend to this case.

In addition, sub-domain methods with overlapping can also be used, such as Schwarz's alternating method. In this case, to analyze the behavior of these asynchronous methods, a result of D.J. Evans and J. Van Deeren [21] is used. These last authors established that, if A is an M-matrix, the matrix $\bar{A}$ obtained by the Schwarz augmentation process is also an M-matrix. Therefore, the behavior of the asynchronous Schwarz method can be immediately analyzed both by contraction techniques and by partial ordering techniques.

Due to the monotony properties of increasing diagonal operators, both when using non-overlapping sub-domain methods and sub-domain methods with overlapping, the convergence of these methods applied to solving single-valued and multi-valued pseudo-linear problems is trivially obtained, the latter case corresponding to the case of classical partial differential equations (P.D.E.) whose solution is subjected to constraints.

Finally, this analysis of sub-domain methods with or without overlapping is still valid for the block numberings usually used, whether it is the lexicographic numbering of the blocks or the red-black numbering of these latter [22].

2.2.2 Sub-structuring methods

F. Magoules [23] and F. Magoules and C. Vernet [24] extend asynchronous iterations models to the case of sub-structuring framework for the solution of linear algebraic systems derived from the classical discretization of linear boundary values problems. By considering an appropriate partitioning with non-overlapping sub-domains of the domain Ω where the partial differential equations (P.D.E.) is defined, F. Magoules et al. consider the matrix S obtained at the interface of the sub-domains, S corresponding to the classical Schur complement; by showing convergent splitting of S, the goal is then to yield an iterative mapping which does not require to compute explicitly the entries of S, contrarily to classical splitting such as Jacobi or Gauss-Seidel ones. Thanks to theoretical results based mainly on the properties of M-matrices and H-matrices, and also by stating new estimation between spectral radii and weighted uniform norms, F. Magoules et al. provide first practical splitting of the Schur complement and proposes a result allowing the possibility of a general unified framework for both asynchronous convergence analysis of Schwarz method and sub-structuring methods. Moreover F. Magoules et al. propose an original result corresponding to new convergent efficient asynchronous iterative sub-structuring methods requiring barely stronger assumptions than the ones required for the convergence of classical asynchronous algebraically non-overlapping additive Schwarz methods.

2.2.3 Multi-splitting methods

To highlight more general calculation methods, we can mention the joint work carried out by European, American and Asian laboratories, which has sometimes been carried out within the framework of collaborations to solve large scale problems, either linear or presenting non-linearities. The techniques used to study the behavior of these new parallel methods are essentially algebraic, based on the use of the Perron - Frobenius theorem. These studies performed by J. Arnal, R. Bru, A. Frommer, V. Migalon, D. P. O'Leary, D. P. O'Leary and R.E. White, J. Penades D. Szyld, and many other authors (see [25]-[45]) have lead to introduce the multi-spling methods where several contracting fixed point applications are considered simultaneously and a weighted average is made between all these iterates. A particular case of multi-splitting method is constituted by the two-level methods where an asynchronous parallel external iteration corresponding to a coordination of the sub-systems to be solved is considered, these latter sub-systems being themselves solved by an iterative method. A particular case of a two-level method is constituted by the coupling of a block relaxation method, each block being solved by a point relaxation method leading then to the two-stage method (see for example [38]); another example of a two-level method is constituted by an external iteration of Newton and an internal iteration of relaxation (see for example [25], [27], [30], [38], [42]).

This type of method has also been studied by Z.Z. Bai and also Chinese scientists (see [28]-[29]) who, to accelerate the speed of convergence, have introduced several relaxation parameters.

In addition, J. Bahi J.C. Miellou and K. Rhofir [27] have extended by contraction techniques the previous results for the resolution of pseudo - linear problems without or with constraints corresponding in the latter case to the solution of multi-valued problems.

Multi-splitting methods actually allow a unified presentation of sub-domain methods with or without overlap. Thus, after analysis, we can conclude on the behavior of both these two types of sub-domain methods. As for these latter sub-domain methods, the properties of M-matrices play an important roll; however, the study framework can be weakened in the case of the H-matrices, the latter context being studied, for example, by the works of [46].

2.2.4 Asynchronous parallel algorithms with flexible communications

In terms of flexibility of asynchronous parallel methods various studies was performed (see [47]-[50]); we can cite a paper presented in 1998 by J.C. Miellou, P. Spiteri and D. El Baz [47] where processors exchange data in a flexible way with the particularity of closely integrating the communication aspects into the calculation aspects; in this kind of method, communications occur during the re-updating of the components of the iterate vector and no longer at the end of each relaxation as was previously the case in the classical asynchronous parallel methods; in addition, component updates are performed using partial values of the component blocks of the iterate vector. The only constraint that is imposed is that:

- on one hand, no component of the iterate vector continues to be permanently updated when the problem is iteratively solved,

- on the other hand, the most recent values of the iterate vector must be used as the calculations progress.

This study was carried out using partial ordering techniques linked to the use of the discrete maximum principle. An application to financial mathematics or mechanics modeled by the obstacle problem has been performed (see [48]).

A similar study on flexible asynchronous parallel algorithms was conducted on one hand by A. Frommer and D.B. Szyld in 1998 for linear problems [49] and in 2004 by A. Frommer, P. Spiteri and D. El Baz for nonlinear problems [50]; in this new study the authors proposed a slightly different formulation of asynchronous parallel methods with flexible communication for which the analysis of their behavior is performed by contraction techniques using uniform norms with weights in accordance with the Perron - Frobenius theorem.

In addition, parallel asynchronous multi-splitting algorithms with flexible communications were also studied by P. Spiteri, J.C. Miellou and D. El Baz [42] by partial ordering techniques, for the numerical solution of pseudo - linear single-valued problems.

3 The difficult problem of stopping tests for asynchronous parallel iterations

If convergence detection for parallel synchronous iterations does not present any major difficulties, whereas the asynchronous case is a real challenge due to the nondeterministic behavior of such methods. Indeed the problem of termination of asynchronous parallel iterations is an extremely difficult problem to code since is concerned both in applied mathematics, when the iterate vector is sufficiently close to the solution of the problem and also in computer science. Nevertheless, in the computer science field, the specificity of multiprocessor architectures must currently be taken into account, particularly on distributed systems where communications are carried out by message passing and in this context the processors only have local information.

This kind of study is all the more an interesting contribution as parallel asynchronous methods are very interesting to use when there are very much synchronizations between the processors, expectations producing phases of inactivity of the processors. Moreover such use of parallel asynchronous methods present an additional interest when the interconnection network has a slow flow, situation that can be found for example in the use of large computing grids or cloud computing when distant and heterogeneous clusters are used.

3.1 Empirical methods

The most commonly termination methods used in preliminary works are often empirical. They are constituted by the observation by a particular processor of the local termination conditions on each processor. The iterative algorithm is then stopped when all local termination conditions are satisfied. This kind of termination will only give satisfactory results only if the degree of asynchronism is low. On the contrary, when delays between processors, due in particular to the imbalance of calculation tasks, are significant, this method can lead to early termination.

Another possible method of termination is obtained by sending termination and reboot messages by each processor and by using a special processor that collects and centralizes the termination messages (see [17]).

In a different approach [51], the termination scheme periodically samples the state of the processors and associates to each one a Boolean value depending on whether or not the local termination criterion is satisfi ed.This local value is then transmitted to the other processors.

The overall state is deduced by calculating the fixed point of a Boolean operator using an asynchronous iterative algorithm. However, this termination method requires that each processor has an estimate of the start and end times of the fixed point asynchronous algorithm using the Boolean operator.

Another termination method uses termination messages and acknowledgements of termination messages [52]. In this case, a processor completes its calculations provided that the local termination criterion is satisfied and that it has received termination and acknowledgement messages from all other processors for all its own termination messages.

Note that there is no formal proof of validity for the previous termination methods. Indeed, for a message passing machine architecture, all processors have local information; in particular, there is no global clock and moreover, messages can be arbitrarily delayed for long periods of time.

In what follows we will distinguish the computer science approach and the numerical analysis approach.

3.2 Computer sciences approach

Subsequently, the work carried out in this context has provided formal proof of validity. In the present sub-section we will distinguish a centralized approach and a decentralized approach.

3.2.1 Centralized approach

We present first the centralized approach. In the centralized approach of termination detection, a processor receives the state of the other processors and detects the global convergence when all processes have converged locally. Several studies have been undertaken by many authors.

D. Bertsekas and J. Tsitsiklis approach : D. Bertsekas and J. Tsitsiklis [53] consider that "each data communicated on a link is correctly received with a finite but unspecified delay". The method of terminating iterative asynchronous algorithms is then based on the decomposition of the problem into two distinct parts:

1. the iterative algorithm is modified so that it ends in a finite time and converges to a fixed point sufficiently close to the solution of the problem,
2. a computer procedure for detecting the termination is applied.

It should be noted that the asynchronous algorithm is modified and that it is different from the classical asynchronous algorithm since the value of one or more components of the iterate vector becomes fixed in a finite time, in contrast to what the calculation algorithm produces due to the fact that the components can change with each relaxation. D. Bertsekas and J. Tsitsiklis therefore introduce a new fixed point application defined either by the updated value deduced from the algorithm if the value of the component has changed, or by the old value if the update is not significant.

This context leads D. Bertsekas and J. Tsitsiklis to modify the iterative algorithm as follows: if the update of a component of the iterate vector does not significantly change its value then the new updated value is not modified and is not communicated to other processors; the termination of the modified iterative algorithm occurs when an update modifies the value of

the components of the iterate vector regardless of the processor and no message is in transit in the communication network. Thus, as far as the non-modification of the components of the iterate vector is concerned, all local termination conditions are satisfied.

Several procedures for detecting the termination of the modified asynchronous iterative algorithm can be used. These include the following

- the Dijkstra and Scholten procedure [54],
- the Chandy and Lamport snapshot algorithm [55].

The Dijkstra and Scholten procedure is based on the acknowledgement of all messages and the generation of a hierarchical activity graph. Processors communicate two kinds of information:

- the new value of the components of the iterate vector,
- the acknowledgment of messages containing a new value.

In an inactive state, a processor does not calculate and transmit any messages or acknowledgments. This inactive processor P changes state when a message is received containing a new value of a subset of components of the iterate vector from another processor that then becomes the father in the hierarchical activity graph. The message that has activated P has a particularly important role until the next inactivity phase; it is called a critical message.

In an active state, a processor updates the components of the iterate vector assigned to it, transmits the updated values to other processors, or at least, to save communications, those who need these values, and systematically sends an acknowledgment for any message received excluding the critical message that receives special processing. An active processor P changes state when its local termination condition is satisfied, acknowledgments have been sent for all received messages except the critical message, and an acknowledgment has been received for all sent messages. When the transition is complete, the processor P sends to its father the acknowledgment of the critical message.

In the initial phase, only one processor, called the root processor, and noted R, is active. In the following, all processors are gradually activated by the reception of messages. The activity graph evolves according to the critical messages received, the satisfaction of local termination conditions and the receipt of acknowledgements.

D. Bertsekas and J. Tsitsiklis have identified two main classes of methods for which the modified asynchronous iterative algorithm converges in a finite time:

1. the case where the asynchronous iterative algorithm is associated with a fixed point application contracting F for a uniform weighted norm,
2. the case where the asynchronous iterative algorithm is associated with an increasing monotonous fixed point application F.

The validity of the method of D. Bertsekas and J. Tsitsiklis can be formally proved. However, this method has drawbacks insofar as

- it requires the use of a complex protocol and twice as many communications as a traditional asynchronous iterative algorithm,

- in addition, more restrictive conditions must be satisfied in order to ensure the convergence of the modified asynchronous algorithm.

The other termination procedure, the Chandy and Lamport snapshot algorithm, is based on the production of snapshot obtained by tagging messages and the recording of the status of links and processors when tagging messages are delivered. The states recorded in a snapshot do not necessarily correspond to the real global state of the system at a given time. However, the information contained in a snapshot is sufficient to detect certain properties of the overall state of the system, in particular the termination. D. Bertsekas and J. Tsitsiklis have established a formal proof of validity of this termination method.

S.A. Savari and D. Bertsekas approach : Another particularly interesting termination method has been proposed by S.A. Savari and D. Bertsekas [56]. In this procedure, the model describing asynchronous iterations is slightly modified; the result of each new update of a component of the iterate vector is taken into account and communicated to the other processors if it is different from the last calculated value. In addition, requests are sent to all processors in the system whenever a non-local termination condition is not satisfied. A processor performs the calculations and forwards messages and requests to other processors as long as its local termination condition is not satisfied or it receives requests from other processors. Termination occurs when all processors have satisfied their local termination condition and no message relating to a request or a result of a refresh is in transit in the system. S.A. Savari and D. Bertsekas have given formal proof of the validity of this termination algorithm. The main advantage of this method is that it can be successfully applied to a larger class of iterative algorithms than the method of D. Bertsekas and J. Tsitsiklis. Its main drawback is that it requires a very large number of communication requests and require restrictive assumptions such as, for example, ordered communications.

J. Bahi et al. approach : Let us also quote the works of J. Bahi et al. in [57]; they have also studied a centralized termination of asynchronous parallel methods. In asynchronous context the convergence detection is even hardened by the difficulty to get a correct image of the global state at any time during the iterative process. The most common techniques used in distributed computing to recover such information are centralized and synchronous. These detection methods are not suited to large scale and / or distant distributed systems and also to asynchronous iterative algorithms. The convergence detection algorithm must also be asynchronous. Moreover the centralization of such detection method may generate the classical problem of bottlenecks. Indeed in a classical centralized algorithm, all processors directly communicate their information to the central one. However, such communication scheme, implying that one machine can directly be contacted by all the others, is not possible in all parallel systems, particularly in the distributed clusters in which each site may have restricted access policies for security reasons. In most cases, only one machine of a given cluster is reachable from the outside. In order to bypass that problem, an explicit forwarding of the message can be performed from any node in the system toward the central one. Such method presents the advantage of only involving communications between neighboring nodes and is well adapted to the hierarchical communication systems that can be found in distributed clusters. Unfortunately, that last scheme implies more communications, slowing down the network and indirectly the iterative process itself. Moreover, it also implies larger delays toward the central node.

3.2.2 Decentralized approach

The decentralized approach presented below is better suited to the termination of asynchronous parallel iterative algorithms

M. Chau et al. approach : For the parallel asynchronous methods, in [5]-[20] for stopping criterion M. Chau et al. used a decentralized algorithm where asynchronous convergence detection is achieved with a token circulation technique. Each processor updates the components of the iterate vector associated with each one's own block and computes the residual norm attached to this block in order to participate to the convergence detection. Convergence detection was performed via a snapshot algorithm (see [17], section 8.2 and [55]), corresponding to a variant of Lamport's method. Convergence occurs when a given predicate on a global state is true. A 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 [17], page 580); then, on every process, the norm of the local residue remains under a given threshold after two successive updates of the component (see [17], [58]). Due to the termination and the detection of the global state of the processes, the implementation of each variant of parallel asynchronous method is then more complex than the synchronous one. In the case of asynchronous iterations, point to point communications between two processes have been implemented using nonblocking MPI_SEND and receive subroutines. MPI_TEST is used in order to allow any processor to continue the computations without having to wait for the completion of any pending send or receive operations. Idle times due to synchronizations in message passing are suppressed in this way. On the other hand, the parallel synchronous iterative schemes are implemented by using MPI_WAIT. One has to take care about the deadlock issue when implementing synchronous communications using blocking MPI [1] subroutines. For more details concerning the implementation and the convergence detection of the considered parallel asynchronous methods, the reader is referred to [20].

J. Bahi et al. approach : Another approach more flexible is proposed by J. Bahi et al. in [57]; this approach do not use a token circulation. So, the most suitable detection algorithm must not only be asynchronous but also completely decentralized. Such decentralized algorithm for the global convergence detection works on all parallel iterative algorithms, either asynchronous or synchronous, with, in this last case, some minor modifications. The major difficulty with termination detection lies in the proof that the proposed algorithm does not detect convergence prematurely. Indeed, in asynchronous methods, the delays between iterations could lead to a false realization of the convergence criterion. This situation typically occurs in heterogeneous contexts, for example when a processor computes a new iteration whereas a slower processor computes a former iteration. Such difficulty is increased with distant processors where the communication / computation ratio may be important. As a consequence the principle of the decentralized detection algorithm is based on two distinct classical steps. The first one consists in detecting the local convergence on each processor while the second one properly consists in the global convergence detection.

The local convergence detection step is quite similar to the one used in the synchronous mode. Since there is no information between the current state of the iterative process and the fixed point to be found, so, in place, the norm of the residual is used to get an idea of the ter-

[1] Message Passing Interface

mination of the iterative process. So, the local iteration is ended when the norm of the residual is less than a given threshold. Nevertheless, it can be noted that, in all iterative algorithms and not especially in asynchronous algorithms, false detection of the global convergence can occur if no care is taken.

The common heuristic for the detection of a definitive local convergence is then to assume that such local convergence has performed a given number of successive iterations with a norm of residual less than the given threshold. Such required value of number of successive iterations exists and is finite since the iterative process converges; the result of [58] allows to give an estimate of such value.

Global convergence will occur when all local processes have converged. Unfortunately if the asynchronism is not responsible for the difficulty in evaluating the local convergence, it hardens the global convergence detection by making the building of a representative image of the global state of the system more difficult.

The decentralization of the detection of the termination algorithm is based upon a scheme quite similar to the leader election protocol. Such protocol consists in dynamically designating one processor to perform a given task, namely in our case, the global convergence detection. However, in the specific context of global convergence detection, the leader election process requires some specific adaptations which imply the use of a tree graph. Then a message informs the receiver that all the processors located in the subtree depending on the sender have reached local convergence. Hence, on each processor, the algorithm considers the number of neighbors in the tree from which no convergence message has already been received. So a node will detect the global convergence when it has received the convergence message from all its neighbors and is itself in local convergence. The correctness of this procedure of global convergence detection is proved in the context where contraction techniques are used (see [57]); in other cases, the process is still correct but an additional verification step is necessary after the global detection to ensure that the system was in the global context state at the detection.

F. Magoules et al. approach : In a recent study F. Magoules and Guillaume Gbikpi-Benissan [59]-[60] present original results concerning the problem of convergence detection for classical parallel asynchronous iterations. In this study convergence detection problem is classically presented in terms of a consistent estimation of the residue thanks to the use of sequences of single vectors generated by each processor. The snapshot algorithm introduced by K.M. Chandy and L. Lamport constitutes a way of building the global state of any distributed system by recording states of processes and communication channels; it allows in the sequel to propose two possible extensions for asynchronous iterations termination in First-In-First-Out (FIFO) communication environments. According to the point of view developed by S.A. Savari and D.P. Bertsekas where computation data are included into snapshot messages, the authors propose to extend the two protocols previously considered and to consider non-FIFO environments; then no coordination phase is introduced by the new two proposed protocols. Then, considering a communication model where at least computation messages are FIFO-ordered, the authors propose another method without snapshot messages, based only on computation data. The main idea is to record a channel state when two successively received messages are very close. Processes therefore need twice more memory for received messages. At last, the authors introduce a communication environment where the non-FIFO message delivering can be characterized, due to performance level requirements on the computation

platform. They therefore derive two new other protocols which build an approximate global iterations vector. With appropriate assumptions, a formal analysis is conducted when the space is normed by any norm, to bound the error between the exact and the computed approximate residual; particularly an analogous result is stated when the weighted uniform norm is used. Presentation of parallel experiments show the effectiveness and the efficiency of the study applied to a target problem solved on supercomputers with 48 to 504 processors cores on one hand and to 1024 to 5600 processors cores on the other hand. These experiments show clearly that the residue is accurately estimated. With respect to previous contributions developed by other authors, the present study show that only one reduction operation at each iteration (instead of two in other contributions) is sufficient to the computation of the residual error; thus, compared to existing supervised termination algorithms, the present study minimizes both delays of termination detection and communication overhead costs.

3.3 Numerical analysis approach

In fact, all the stopping criteria presented in sub-sections 3.1 and 3.2 do not really take into account the numerical aspects related to the mathematical analysis of the behavior of these parallel iterative methods. Several works related to the study of the convergence of asynchronous parallel methods has also highlighted numerical termination criteria. Thus a realistic implementation of termination of parallel asynchronous methods must combine both the aspects of computer science with the aspects derived from numerical analysis.

Use of a secondary error control algorithm : For parallel asynchronous methods J.C. Miellou proposed in [3] a termination detection method based on the use of a secondary error control algorithm derived from the secondary algorithm of F. Robert [61] and G. Schroeder [62]. In the framework of study of convergence by contraction techniques, this secondary algorithm allows to solve a system of small dimension $z = Jz$, where J is the contraction matrix with a spectral radius smaller than the unit. Obviously, for the solution of the problem $z = Jz$, whatever be the initial guess z^0, the secondary algorithm will converge to zero. So, by choosing at each step p for the principal algorithm and also for the secondary algorithm the same strategy for choosing the components to relax and the same delayed values, J.C. Miellou has established that if the secondary algorithm is initialized by a vector z^0 verifying

$$z^0 \geq q(U^0 - U^\star),$$

then at each step p of the principal algorithm the following inequality

$$z^p \geq q(U^p - U^\star)$$

is verified; so we can terminate the principal asynchronous parallel algorithm, when all the components of z^p are of modulus less than a given tolerance ϵ The secondary algorithm requires significantly fewer calculations than the principal algorithm and can very well be implemented with the latter at a lower cost; it therefore provides a reliable and economical method for terminating asynchronous parallel methods. We refer to [3] for a practical determination of z^0.

Use of the discrete maximum principle : Related to the works performed in [11]-[12] (and also [42], [47]-[48]) We have seen that convergence analysis can be carried out using partial

ordering techniques; we can then take advantage of the properties of a decrease of the iterate vector W^p calculated from an over-solution W^0 and of the growth of the iterate vector V^q calculated from a sub-solution V^0, to obtain at each step a control of the solution $U^\star$ such as

$$V^q \leq U^\star \leq W^p.$$

Thus, if the components of the vector $W^p - V^q$ are all less than a given tolerance ϵ, we can conclude that the asynchronous parallel methods are converging. However, it should be noted that this process, although very reliable, is not economical in terms of the amount of calculation and calculation time.

Use of nested sets : based on the consideration of perturbation by floating point errors [63], another approach to dynamically terminating parallel iterative asynchronous algorithms in a general situation of topological context was developed by J.C. Miellou, P. Spiteri et al. in [64] to [66]; this approach is based on considerations involved in the analysis of the numerical behavior of these iterative methods. This approach essentially uses the notion of nested sets studied by D. Bertsekas et al. (see [17]) to analyze the convergence of these methods. The principle of this approach is the same whether the problem is linear or non-linear. However, for linear problems the effect of rounding errors is taken into account by using an approximate contraction notion introduced by J. Wilkinson (see [67]-[68]) and extended in [69] in the context of parallel asynchronous iterations. In order to give some estimate of the distance between the exact solution and the current iterate and then to propose several on line original stopping criteria for parallel asynchronous iterations usable for general fixed point methods defined in a product space the previous authors consider asynchronous iterations with multiple initializations and develop a complete study allowing to obtain, when the considered stopping tests are satisfied, upper bounds of the absolute error and also sharp estimates of bounds of the residue very useful for efficient various stopping criteria. Note that these bounds depend on the chosen topology, and specifically on the equivalence norm constants between the chosen norm and a specific Perron-Frobenius norm. These methods of termination are relative to an estimation of the diameter of appropriate nested sets, centered on the solution required, and that contain the selected iterate vectors necessary to achieve the stopping criterion; in other words the considered stopping criteria are described with respect to the distance of the extremities of a *segment order* containing the selected iterates necessary to achieve the stopping criterion. Thus, if the size of this *segment order* is lower or equal to an appropriate threshold associated to the considered stopping test, then the iterations stop. This property of stopping the iterations is proved rigorously thanks to the convergence property of asynchronous iterations when contraction or approximate contraction properties of the fixed point mapping are considered. Moreover to obtain a correct estimation of the *segment order* considered to stop the iterative algorithm it is necessary to enclose several successive iterates which then allow to construct by a dynamical way the nested sets. This is made possible thanks to the use of the notion of *sliding macro-iteration*. More precisely:

1. a *sliding macro-iteration* corresponds to the consideration of successive iterate vectors including those with just refreshed components and the previous iterate vectors directly used for getting them; a *sliding macro-iteration* is distinct to a *macro-iteration* used for the study of convergence of asynchronous parallel iteration; indeed a *macro-iteration* corresponds to the fact that all components of the iterate vector are updated at least

one time using available values of the components associated to the considered *macro-iteration*,

2. the construction of such sets is possible when the convergence of the asynchronous parallel iterative methods results of contraction or approximate contraction techniques, corresponding to the fact that the fixed point mapping is contracting with respect to a vectorial norm and admits a contraction matrix J, i.e. a Lipschitz matrix with nonnegative entries and spectral radius strictly less than one.

The proposed stopping criteria correspond to the three following cases

1. a stopping criterion with respect to the absolute error,
2. a stopping criterion with respect to the relative error,
3. a mixed stopping criterion with respect to a vectorial norm error, i.e. combining the absolute error and the relative error and using a mixed weight uniform norm absolute value-relative value vectorial norm stopping test.

The previous three stopping criteria are described with respect to the general distances defined on the classical n-dimensional space. It can be also noted that the stopping criterion with respect to the absolute error and the stopping criterion with respect to the relative error can be formulated by using general norms defined in the classical n-dimensional space while the mixed stopping criterion with respect to a vectorial norm error is defined in a more particular context using weighted uniform norms. In all cases and in the considered theoretical and general framework, it has been shown that each termination test is coherent and bounds of the residue and of the error, for the absolute value stopping test and for the mixed stopping criterion with respect to a vectorial norm are established. Finally some criteria related to dynamic termination such as absolute errors and errors in vectorial norm are particularly well suited to specific contexts such as the use of distributed memory machines, grid computing, peer to peer and cloud computing.

Beside, note also, that in a previous study [58], we have considered an analogous study using the same theoretical framework in the case of linear fixed point parallel asynchronous iterations, taking into account the influence of round off errors; in this work the stopping test is not achieved by a dynamic way, but is predictive, i.e. allows to predict, taking into account the value of the parameters used to estimate the convergence rate, how many macro-iterations will be required to obtain a given accuracy. Lastly let us also mention a study concerning the termination of parallel asynchronous iteration for nonlinear problems [70].

4 Implementation of asynchronous parallel algorithms

Scientists who have carried out theoretical studies on asynchronous parallel methods have obviously been very interested in testing these methods in real scale. However, at the time these studies were started, there were practically no multi-processor machines.

The first experiments were therefore carried out by simulating parallel executions. The first simulations were carried out in the late 1960s by J.L. Rosenfeld [1]. Further asynchronous parallel execution simulations were performed in the late 1970s by P. Spiteri et al. (see [71]-[72]) where these authors simulated the actual running of parallel asynchronous executions on a

single-processor machine. At the same time G. Baudet was able to test these methods on CMMP at the Carnegie Mellon Institute [16]. Subsequently, other more appropriate and realistic methods of implementation were used, such as the use of Transputeurs [6], which made it possible to evaluate the performance of asynchronous parallel algorithms. The beginning of the 1980s also saw the emergence of multiprocessor machines, first with common memory, such as Aliant or Cray machines [73], and also with distributed memory machines [74]. It should be noted that currently parallel runtime simulations are still used to estimate at a low cost the computation cost of an application to be carried out later on a multiprocessor machine (see [75]). These studies then lead also the researchers to implement these parallel methods as efficiently as possible on peer-to-peer architectures, on a computing grid and on the cloud.

It would be cumbersome to make an exhaustive presentation of all the experiments carried out and the efforts made by programmers to use as efficiently as possible these new architectures.

We briefly present below several recent contributions on the implementation of asynchronous parallel algorithms and we refer the reader to the bibliography for other implementation works.

4.1 Principle of implementation

This section is devoted to the presentation of the principle of implementation of parallel algorithms. The implementation of parallel algorithms have been carried out on clusters in [20] and in heterogeneous and distant clusters constituting a Grid platform in [5]. The principle of implementation of parallel asynchronous and synchronous iterative methods can be summarized as follows:

Do until global convergence
  For each block **do**
    - Perform communications of block boundary values
    - Perform block relaxation
  End For
End Do

The method without overlapping corresponds to a straightforward block relaxation method, with communications of the values of the components associated to the boundaries of a block; so such method does not present any difficulty. Thus, sequences of smaller subproblems are solved on each processor of the parallel computer in order to compute a solution of the global problem; practically more accuracy is obtained. Several blocks are assigned to each processor in order to implement a strategy which is close to the multiplicative one. To obtain a faster convergence of the parallel computations, each processor handles contiguous blocks, numbered according to red-black or lexicographic ordering ; note that such red-black ordering is more appropriate for parallel computations and, in this case the convergence results of convergence still hold by a straightforward way [22].

In the case of asynchronous iterations, point to point communications between two processes have been implemented using nonblocking MPI_SEND and MPI_RECV subroutines. MPI_TEST is used in order to allow any processor to continue the computations without having to wait for the completion of any pending send or receive operations. Idle times due to synchronizations in message passing are suppressed in this way. On the other hand, the parallel synchronous iterative schemes are implemented by using MPI_WAIT. One has to take care

about the deadlock issue when implementing synchronous communications using blocking MPI subroutines.

For simplicity, with respect to the block decomposition considered the block partitioning retained for the assignation of the large blocks to the processors is very simple; in fact, in order to define a large block, we have gathered several adjacent blocks of the natural block decomposition. So, on each large block associated to each process, a classical and sequential block relaxation method is used in order to solve each subproblem ; this kind of method is implemented by solving each tri-diagonal block, associated to classical discretized boundary value problems, by the TDMA method, corresponding to the Gauss's elimination method applied for tri-diagonal block; in order to reduce the elapsed time of computation, note that the implementation of the block relaxation method is optimized by elimination of sequences of redundant code. Indeed in this implementation the LU decomposition of each diagonal block is performed only once in the initialization phase. Then, this optimization allows to decrease the elapsed time. For more details concerning the implementation and the convergence detection of the considered parallel asynchronous methods, the reader is referred to [5] and [20]. Note also that, instead of using block relaxation methods on each large block, it is also possible to implement various efficient numerical method for the solution of each large block; so, we refer to [38] for the use of the point relaxation method or also to [19] for the use of preconditioned conjugate gradient method.

The simulation algorithm is based on master/slave paradigm which is a commonly used approach for parallel and distributed applications. In our case, a master process controls the distribution of work to a set of slave processes.

Algorithm
  if (master)
    compute discretization matrix and right hand side
    send input data to slaves
    compute SPMD parallel algorithm to solve the algebraic system
    receive output data from slaves
  else
    receive input data from master
    compute SPMD parallel algorithm to solve the algebraic system
    send output data to master
  endif

Algorithm 1: General algorithm.

This process is used for the solution of the system that arises in the numerical simulation of the target problem. Matrix and right hand side creation can be implemented sequentially since this part of computation is not very intensive and except particular cases, are computed once. In this way, the master process can be seen as a matrix and vector filler that feeds a parallel solver. In other words, only intensive computations have been parallelized with MPI facilities.

In the case of parallel Schwarz alternating method, sub-domains may overlap each other. For the implementation of the Schwarz alternating method, the domain Ω, where the boundary value problem is defined, is split into overlapping parallelepiped or sub-domains of any form. Minimal values are chosen for overlapping between them (about one or two mesh points). The parallel synchronous and classical asynchronous Schwarz alternating methods have been implemented as follows

do until global convergence
  for each subdomain assigned to the processor **do**
   if local convergence is not reached **then**
    receive the latest boundary values
    solve the subsystems
    send the boundary values to the neighbours
   end if
  end for
end do

Algorithm 2: Schwarz alternating method.

Sub-domains are assigned to each processor following for example a red-black ordering. For the solution of the target problem, note that in each sub-domain is considered the implementation of the block Gauss-Seidel method which has the advantage of having a multiplicative behavior.

4.2 The library Jack 2

F. Magoules and Guillaume Gbikpi-Benissan present in [60] and [76] a new object-oriented communication library Jack2, developed to ease the implementation of both classical, synchronous and asynchronous iterative methods in this two last cases for distributed architectures. In the case of asynchronous iterative methods and in order to reduce the calculation times, the weight of idle times due to synchronizations between the processes is decreased. Then the goal is to allow a full overlapping of communication and computation phases during the parallel processing, in contrary of the parallel synchronous algorithms where, for example, a processor must wait for updates from another processor that has not completed its own calculations. In the case where many synchronizations are necessary, and if in addition the communication network is slow, such situation of synchronous methods could therefore considerably decrease the parallel computational efficiency. The implementation of asynchronous methods is not easy to achieve regarding particularly to the management of communication requests and the evaluation of stopping criterion; such implementation needs to be carried out carefully and optimally. For the realization of Jack2 a precise specification has been carefully drawn up; with respect to previous works, additional contributions concerning

1. the protocol termination,
2. flexibility and usability improvements in the architectures in the design of the library,
3. complete details of implementations including parts of code,
4. additional parallel experimental results obtained in an homogeneous computational environment, allowing to measure efficiency impacts of various configurations and accuracy.

For asynchronous iterations convergence detection, the termination of the iterative process is accomplished with the same principle than the one presented by the same authors at the end of subsection 3.2.2.

Following these principles of implementation, and using the MPI programming framework, the Jack2 programming framework is carefully and progressively presented. Various choice

of implementation, such as several levels of encapsulation of the different operations to be performed, are presented. In a distributed environment, in order to avoid memory overhead costs, for the design of Jack2, the authors opts for MPI persistent requests in a similar way than the one considered for example in [5] or [20], rather than using collective procedures; in such persistent communications message buffers and neighbor process are specified once for all before beginning the iterations. Then a complete initialization part for communication parameters is defined where variables of communication graph, communication buffers and residual evaluation are specified; this initialization phase allows non-blocking reception requests between neighbors. Such Application Programming Interface (A.P.I.) allows to handle communications requests during asynchronous computations. Correct termination is ensured by adding partial snapshot operation in accordance with the same principle than the one presented at the end of subsection 3.2.2. Finally, to produce a unique code for both overlapping and asynchronous schemes the top front-end of Jack2 provides two classes, which manage communication requests and convergence detection. The management of communications and detection of convergence, in both cases synchronous or asynchronous, is achieved following the same pattern. Then function pointers are used according to the communication mode without using " if "conditions. For point-to-point communications Jack2 is configured to activate several MPI reception requests on each same reception buffer defined by the user application; then processes can use more recent data for last updated local components, instead of only benefiting from data received at the beginning of the computation phase, allowing more flexibility during the computation. Then such a mechanism of reception of the messages, compared to a conventional put / get semantics, allows a finer management of the delayed relaxations. Moreover, the author shows that the inhibition of message transmissions on an already busy communication link does not degrade the performance than those obtained by a put mechanism, because such a mechanism is reproducible by means of synchronous non-blocking MPI. The FIFO delivering on communication links considered in subsection 3.2.2 is then assured by the MPI specification. For the communications Point-to-point messages sending and reception request are carefully handled in order to minimize delays of communication data and then obtain the more possible multiplicative behavior of the iterative distributed algorithm; such situation is realized when the updates are sent at the beginning of the iterations according to a solution proposed in [5] or [20], even if the destination process is still computing; then the Gauss principle is respected to obtain the fastest convergence. Finally, last updates are transmitted when only explicitly requested into computation reception buffer. This feature is realized in Jack2 by adding supplementary buffers.

4.3 Implementation in multi-GPU architecture

Some works has been also focused on the implementation of synchronous and asynchronous iterative parallel methods on multi-GPU architecture (see [77]-[80]). For example this implementation make possible to solve a problem of a highly non-linear partial differential equation intervening in finance or mechanics [78]; this last reference concerns the obstacle problem that corresponds to a boundary value problem where the solution is subjected to constraints. It is then necessary at each step to project the intermediate iterations onto the convex set defining the constraints. Therefore, the Richardson projected point method and the projected block relaxation method were used. It appears that asynchronous parallel implementation reduces the downtime of parallel processes due to blocking messages and synchronization barriers. It was necessary to consider two levels of parallelization: a synchronous multi-threaded paralleliza-

tion in a computing node and parallelization between the different nodes of the cluster that communicate with each other by passing messages in synchronous or asynchronous mode. A comparison of the performances of algorithms implemented on multi-CPU and multi-GPU architectures is achieved, and then one can measure the influence of the architecture on the performance of the computation methods. In particular, the Richardson projected method, although having a slow convergence rate compared to the projected block relaxation method, is more efficient in terms of restitution time on multi-GPU architecture; this is due to the vectorial nature of the Richardson method which allows good multi-threaded parallelization. On the other hand, on a multi-CPU architecture, the performances are reversed, since considering the good speed of convergence of this method, the projected block relaxation methods allow to obtain better elapsed times. It has also been found that, thanks to the computing power of GPUs, their use reduces the ratio of computing time to communication time.

Always using multi-GPU architectures, synchronous and asynchronous methods have been implemented for the solution of large scale linear systems. The approach adopted consists in considering coarse-grained algorithms based on the two-level multi-decomposition method using the iterative parallelized Krylov GMRES method well adapted to GPU clusters [77]. This combines the performance of synchronous parallel iterations to solve linear sub-systems by the GMRES method and, for data exchanges, the flexibility of asynchronous iterations between GPU clusters for the overall resolution of the large linear system. This type of method is well suited to the use of distributed architectures composed of geographically remote computing nodes interconnected by a high latency communication network.

Let us also refer to [79]-[80] for the determination of the roots of high degree polynomial using GPU.

4.4 Load balancing for parallel asynchronous algorithms

To take into account the heterogeneity of the machines, it is necessary to balance the processor load, especially in synchronous mode but also in asynchronous mode. In the asynchronous mode, a properly distributed load allows an efficient recovery of communications by calculations. Load balancing can be applied in two different ways: either static or dynamic. In the case of dynamic load balancing, the evolution of the computing power available on each machine during the execution of the algorithm is taken into account. We therefore iteratively balance the load of each processor with its neighbors until the overall balance of the entire system is achieved. Thus asynchronism appears both at the application level and at the balancing method level.

In addition, it becomes less and less interesting to design balancing algorithms specific to a given context. Therefore there is a strong demand for fully adaptive algorithms that can be used on any type of parallel architecture without significant modification. This is due to the fact that we are currently witnessing the emergence of increasingly dynamic systems. This dynamic aspect is typically found in communications where the links between the calculation units are only intermittent.

D. Bertsekas and J. Tsitsiklis [17] proposed an asynchronous model for decentralized balancing based on the fact that each processor has its own load estimate and that of its neighbors. Thus each processor searches if it has neighbors less loaded and in this case it distributes part of its load to them. The work of D. Bertsekas and J. Tsitsiklis then evolved; thus, one variant was to transfer the calculation load only to the least loaded neighbor; this point of view has

the advantage of generating only load transfers per pair of neighbors, which makes balancing easier to implement and above all much lighter because the network is much less congested by message transfers.

This point of view was also adopted by J. Bahi and S. Contassot - Vivier (see [57]). Thus the most suitable strategies for such asynchronous use contexts are neighborhood strategies based on diffusion algorithms. In this context, the method proposed by J. Bahi and S. Contassot - Vivier is based on an algorithm based taking into account an asynchronous dynamic diffusion of information over time on a network where the link between processors is intermittent. The original point of view of this approach, based on the use of dynamic and/or decentralized distributions, is therefore different from the traditional methods used in load balancing algorithms where static and centralized distributions are taken into account. This point of view therefore takes into account the emerging evolution of multiprocessor architectures, which are increasingly complex and often include heterogeneous computing units and various interconnection networks. This approach therefore makes it possible to design load balancing schemes that take full advantage of the ever-increasing computing power and takes also into account the modularity of systems due to the increase in the number of possible parallel processes. The load estimator chosen by these authors is the residue between two successive iterations because this latter estimator makes possible to take into account the progress of the iterative solution of the problem; indeed, when two neighboring processes have very different residues, it means that one progresses faster than the other and therefore has a lower contribution to the overall progress of the calculation. To balance the contributions, a transfer of elements from the processor with the largest residue to the one with the smallest residue is then performed. In this context of use J. Bahi and S. Contassot - Vivier have established a convergence theorem. Experimental results on SimGrid confirm the effectiveness of the algorithm.

5 Applications of asynchronous parallel algorithms

5.1 Application in high performance computing

Asynchronous parallel iterative algorithms have been successfully applied to a large number of problems; more generally numerical solution of algebraic systems in several references, numerical solution of ordinary differential equations as well as numerical resolution of differential equations at both ends, algebraic differential equations and waveform relaxation methods for the simulation of electronic circuits (see for example [36] and [81]), Markovian systems [18], and also linear asynchronous iterations when the spectral radius of the modulus matrix is one [82], optimization problems, in particular convex flow problems in networks that have many applications in distribution and information routing (see [83]-[84]), finance...etc ... , and also determination of the root of polynomials [85]. Other very varied problems studied at MIT [17]: dynamic programming, searching for minimum paths, assignment type optimization problems, flow problems in networks, optimal routing in communication networks. The list of applications is therefore large and it is difficult to be exhaustive.

Many studies have also been achieved concerning numerical solution of linear and non linear partial differential equations; let us cite for example non-linear diffusion problems involved in plasma physics or in the modelling of solar ovens [6] and [42], the Hamilton - Jacobi - Bellman problem involved in image processing [47] , the obstacle problem modelling financial mathematics problems or mechanical problems (see [78] and [86] to [90]), the Navier -

Stokes equation modeling hydrodynamic problems and also, on various architectures, various applications in biology such as protein separation by electrophoresis modelled by problems coupled from this equation with one or more convection - diffusion equations to calculate the various protein concentrations as well as a potential equation describing the electrical behavior of the process (see [91] to [94] or fluid interaction - structure problems where the Navier - Stokes equation is coupled to the Navier equation describing the behavior of the structure [19]. Other applications related to P.D.E. resolution have been solved in medical image processing to perform filtering of noisy images as well as segmentation to detect contours and thus highlight homogeneous areas in ultrasound or PET images [95]-[96]. Other application to porous medium equation and various classical applications have been also studied (see [97]-[98]); note also a recent study concerning asynchronous optimized Schwarz method in [99].

5.2 Application of chaotic iteration in security

Another application distinct from high-performance computing objectives has been developed by C. Guyeux et al. [100] - [106] to solve computer security problems. While most of the studies so far conducted are related to the analysis of their convergences, in this works, C. Guyeux et al. does not seek to study the convergence properties of chaotic iterations but divergence properties in relation to chaos theory for iterative discrete dynamic systems. The combination of these two concepts is well suited to various target applications such as computer security, where chaos is linked to information concealment techniques, mainly tattooing and steganography. This activity leads them to develop various theoretical and practical studies. Moreover, still based on chaotic iterations, C. Guyeux et al. define a new class of hash function. Subsequently, this work was continued in a similar direction, while refining the theoretical points of view already developed and broadening the application points of view, particularly towards applications such as neural networks, the generation of pseudo-random numbers that play an important role in encryption systems where messages are hidden and encryption keys are generated, the implementation having been carried out either physically with a laser or implemented on FPGA or GPU. Still in the same formal framework, the study of security by wireless sensor networks, in particular video surveillance, but also indistingability, non-malleability, routing, etc., has been developed. Finally, a final aspect of C. Guyeux's works concerns the study, by asynchronous parallel methods, of complex dynamics appearing in bioinformatics, with in particular the analysis of the spatial dynamics of protein folding and the study of the temporal dynamics of genome evolution.

6 The efficiency of asynchronous parallel algorithms

From the benchmarks achieved, it appears that asynchronous parallel methods are of real interest when parallel processing involves a large number of synchronizations. This occurs in particular when convergence is slow, and especially when the interconnection network between processors and/or clusters is slow; this aspect increases in the case of massive parallelism. On the other hand, when convergence is fast, this situation corresponding to a low number of synchronizations between processors, and the interconnection network between processors and/or clusters is fast, synchronous parallel methods are still of interest.

On the theoretical level, we have seen that the general formulation of asynchronous methods makes it possible to present in a unified way both asynchronous and synchronous methods

as well as classical sequential relaxation methods, which makes possible to analyze once for all these methods whatever be the synchronization mode in the case of sequential or parallel codes.

To illustrate these conclusions, we refer to [95]-[96] corresponding to applications for the treatment of medical images or [38] corresponding to the simulation of a continuous casting problem in steelworks where the authors theoretically show that convergence is rapid and experimentally observe better performance of synchronous parallel methods in the case where the simulation is achieved on a cluster with fast interconnection network between the processors. On the other hand, in the last case of application, because of the slowness of the interconnection network between the clusters, when simulations are carried out on a computing grid, asynchronous methods are interesting. On the other hand, in [5] corresponding to financial mathematical applications or unilateral constraint problems, convergence is very slow and correlatively the performance in term of elapsed time of asynchronous parallel methods is much higher than that of synchronous ones.

This phenomenon of superiority of asynchronous parallel methods is therefore accentuated when the processors are heterogeneous and distant, which is the case for example with computing grids or cloud computing; we can therefore see here the influence of machine architecture on algorithm performance.

In addition, for better performance of iterative parallel asynchronous algorithms, it is necessary to achieve a decomposition of the problem in large tasks, so that the numerical behavior of the iterative method is rather multiplicative like the Gauss - Seidel method and no additive like the Jacobi method.

7 Conclusion

In this paper we have briefly, synthetically and as fully as possible presented the asynchronous parallel iterative methods, trying to identify the essential points. References to numerous works will allow readers to explore this large subject in full evolution. However, it was difficult to cite all the works carried out in at least 50 laboratories around the world. Nevertheless, a bibliographical search on the Internet makes it possible to enrich the list of references mentioned in this study.

References

[1] J.L. Rosenfeld, ”A case study on programming for parallel processors”, I.B.M., Thomas J. Watson, Research center report RC-64, 1967.

[2] D. Chazan, W. Miranker, ”Chaotic relaxation”, Linear Algebra Appl., 2 , 199-222, 1969.

[3] J.-C. Miellou, ” Algorithmes de relaxation chaotique à retards”, RAIRO Analyse numérique, 1 , 55-82, 1975.

[4] J.-C. Miellou, P. Spiteri, ”Un critère de convergence pour des méthodes générales de point fixe”, M2AN, 19, 645-669, 1985.

[5] M. Chau, A. Laouar, T. Garcia, P. Spitéri. ”Grid solution of problem with unilateral constraints”, Numerical Algorithms, Springer-Verlag, 75 - 4, 879 - 908, 2017.

[6] L. Giraud, P. Spiteri, ”Résolution parallèle de problèmes aux limites non linéaires”, M2AN, 25 , 579-606, 1991.

[7] J. Ortega, W. Rheinboldt, ”Iterative Solution of Nonlinear Equations in Several Variables”, Academic Press, New York, 1970.
[8] R.S. Varga, ”Matrix iterative analysis”, Prentice - Hall, 1962.
[9] A. Berman, R.J. Plemons, ”Nonnegative matrices in the mathematical sciences”, SIAM, 1994.
[10] M. El Tarazi, ”Some convergence results for asynchronous algorithms”, Numerische Mathematik, 39 , 325-340, 1984.
[11] J.-C. Miellou, ”Itérations chaotiques à retards, étude de la convergence dans le cas d’espaces partiellement ordonnés”, CRAS Paris, 280 , 233-236, 1975.
[12] J.-C. Miellou, Asynchronous iterations and order intervals, *In Parallel Algorithms and Architectures,* eds. M. Cosnard et al., North Holland, 85-96, 1986.
[13] D. El Baz, ”M-functions and parallel asynchronous algorithms”, SIAM Journal on Numerical Analysis, 27, 136-140, 1990.
[14] A. Frommer, ”On asynchronous iteration in partially ordered spaces”, Numer. Funct. Anal. and Optimiz., 12 , 315-325, 1991.
[15] W. Rheinboldt, ”On M-functions and their application to nonlinear Gauss-Seidel iterations and to network flows”, J. Math. Anal. and Appl., 32 , 274–307, 1970.
[16] G. Baudet, ”Asynchronous iterative methods for multiprocessors”, Journal Assoc. Comput. Mach., 25 , 226-244, 1978.
[17] D. Bertsekas, J. Tsitsiklis, ”Parallel and Distributed Computation”, Numerical Methods, Prentice Hall Englewood Cliffs N.J., 1989.
[18] B. Lubachevski, D. Mitra, ”A chaotic asynchronous algorithm for computing the fixed point of nonnegative matrix of unit spectral radius”, J. Assoc. Comput. Mach, 33, 130 - 150, 1986.
[19] V. Partimbene, ”Calcul haute performance pour la simulation d’interactions fluide-structure”, Thèse de doctorat, University of Toulouse, 2018.
[20] M. Chau, D. El Baz, R. Guivarch, P. Spitéri. ”MPI implementation of parallel subdomain methods for linear and nonlinear convection-diffusion problems”, Journal of Parallel and Distributed Computing, Elsevier, 67 (5), 581-591, 2007.
[21] D.J. Evans, W. Deren, ”An asynchronous parallel algorithm for solving a class of nonlinear simultaneous equations”, Parallel Computing, 17, 165–180, 1991.
[22] M. Chau, T. Garcia, P. Spitéri, ”Asynchronous Schwarz methods applied to constrained mechanical structures in grid environment”, Advances in Engineering Software, Elsevier Science, 74, 1-15, 2014.
[23] F. Magoules, ”From synchronous to asynchronous substructuring methods”, Advances in Parallel, Distributed, Grid and Cloud Computing, P. Iványi, B.H.V. Topping and G. Várady (Editors), Saxe-Coburg Publication, 203-215, 2017.
[24] F. Magoules, C. Venet,” Asynchronous iterative sub-structuring methods”, Mathematics and Computers in Simulation”, 145, 34-49, 2018.
[25] J. Arnal, V. Migallón, J. Penadés, ”Parallel Newton two-stage multisplitting iterative methods for nonlinear systems”, BIT Numerical Mathematics, 43, 849 - 861, 2003.
[26] J. Arnal, V. Migallón, J. Penadés, ”Non-stationary parallel multisplitting algorithms for almost linear systems”, Numer. Linear Algebra Appl., 6, 79 - 92, 1999.
[27] J.M. Bahi, J.C. Miellou, K. Rhofir, ”Asynchronous multisplitting methods for nonlinear fixed point problems”, Numerical Algorithms, 15, 315 - 345, 1997.
[28] Z.Z. Bai, ”Asynchronous multisplitting AOR methods for a class of systems of weakly nonlinear equations”, Appl. Math. Comp., 98, 49 - 59, 1999.

[29] Z.Z. Bai, ”The monotone convergence rate of the parallel nonlinear AOR method”, Computers Math. Appl., 31(7), 21 - 31, 1996.
[30] ZZ. Bai, V. Migallón, J. Penadés, DB. Szyld, ”Block and asynchronous two-stage methods for mildly nonlinear systems”. Numerische Mathematik, 82(1), 1 - 20, 1999.
[31] Z. Z. Bai, D.R. Wang, ”Improved comparison theorem for the nonlinear multisplitting relaxation method”, Computers Math. Appl., 31-8, 23 - 30, 1996.
[32] R. Bru, V. Migallón, J. Penadés, D.B. Szyld, ”Parallel, synchronous and asynchronous two-stage multisplitting methods”, Electronic Transactions on Numerical Analysis, 3, 24 - 38, 1995.
[33] R. Couturier, C. Denis, F. Jézéquel, ”GREMLINS: a large sparse linear solver for grid environment”, Parallel Comput 34(6–8), 380 - 391, 2008.
[34] R. Couturier, L. Ziane Khodja, ”A scalable multisplitting algorithm to solve large sparse linear systems”, The Journal of Supercomputing, 69 (1), 200 - 224, 2014.
[35] A. Frommer, ”Parallel nonlinear multisplitting methods”, Numerische Mathematik, 56, 269 - 282, 1989.
[36] A. Frommer, D. Szyld, ”On asynchronous iterations”, Journal of Computational and Applied Mathematics, 123 , 201-216, 2000.
[37] A. Frommer, DB Szyld, ”H -splittings and two-stage iterative methods”, Numerische Mathematik, 63, 345 - 356, 1992.
[38] T. Garcia, P. Spitéri, G. Khenniche, ”Behavior of parallel two-stage method for the simulation of steel solidification in continuous casting”, Advances in Engineering Software, Elsevier Science, to appear, in press.
[39] F. Jézéquel, R. Couturier, C. Denis, ”Solving large sparse linear systems in a grid environment: the GREMLINS code versus the PETSc library”, Journal of Supercomputing. 59 (3), 1517 - 1532, 2012.
[40] MT Jones, DB. Szyld, ”Two-stage multisplitting methods with overlapping blocks”, Numerical Linear Algebra with Applications, 3, 113 - 124, 1996.
[41] D.P. O’Leary, R.E. White, ”Multi-splittings of matrices and parallel solution of linear systems”, SIAM:Journal on Algebraic Discrete Methods, 6, 630 - 640, 1985.
[42] P. Spiteri, J.C. Miellou, D. El Baz, ”Parallel asynchronous Schwarz and multisplitting methods for a non linear diffusion problem”, Numerical Algorithms, 33, 461-474, 2003.
[43] D. Szyld, ”Different models of parallel asynchronous iterations with overlapping block”, Computational and Applied Mathematics, 17, 101-115, 1998.
[44] D.R. Wang, Z.Z. Bai, D.J. Evans, ”On the monotone convergence of multisplitting method for a class of system of weakly nonlinear equations”, Intern. J. Computer Math., 60, 229 - 242, 1996.
[45] R. E. White, ”Parallel algorithms for nonlinear problems”, SIAM J. Alg. Discrete Meth., 7, 137 - 149, 1986.
[46] Y.D. Han, J.H. Yun, ”Convergence of multisplitting methods with preweighting for an H-matrix”, Bull. Korean Math. Soc., 5, 997 - 1006, 2012.
[47] J.C. Miellou, D. El Baz, P. Spiteri, ”A new class of asynchronous iterative algorithms with order interval”, Mathematics of Computation, 67-221, 237-255, 1998.
[48] P. Spiteri, J.-C. Miellou, D. El Baz, ”Asynchronous Schwarz alternating methods with flexible communication for the obstacle problem”, Réseaux et systèmes répartis, 13 (1), 47-66, 2001.
[49] A. Frommer, D. Szyld, ”Asynchronous iterations with flexible communication for linear systems”, Calculateurs Parallèles, Réseaux et Systèmes Répartis, 10, 421-429, 1998.

[50] D. El Baz, A. Frommer, P. Spitéri, ”Asynchronous iterations with flexible communications : contracting operators”, Journal of Computational and Applied Mathematics, Elsevier, 176, 91-103, 2005.
[51] J.C. Miellou, D.J. Evans, ”Stopping criteria for parallel asynchronous algorithms”, Computer studies report 503, Loughborough university of technology, 1989.
[52] E. Chajakis, S.A. Zenios, ’Synchronous and asynchronous implementations of relaxation algorithms for nonlinear network optimization”, Parallel Computing, 17, 873-8941, 1991.
[53] D.P. Bertsekas, J. N. Tsitsiklis, ”Parallel and distributed iterative algorithms: a selective survey”, Automatica, 25, 3-21, 1991.
[54] E.W. Dijkstra, C.S. Scholten, ”Termination detection for diffusing computation”, Processing Letters, 11, 1-4, 1980.
[55] K.M. Chandy, L. Lamport, ” Distributed snapshots : determining global states of distributed systems”, ACM Trans. Comput. Syst., 3-1, 63-75, 1985.
[56] S.A. Savari, D.P. Bertsekas, ”Finite termination of asynchronous iterative algorithms”, Parallel Computing, 22(1), 39-56, 1996.
[57] J.M. Bahi, S. Contassot-Vivier, R. Couturier, ”Parallel iterative algorithms: from sequential to grid computing”, Chapman & Hall/CRC, 2007.
[58] J.C. Miellou, P. Spitéri, D. El Baz, ”Stopping criteria, forward and backward errors for perturbed asynchronous linear fixed point methods in finite precision”, IMA Journal of Numerical Analysis, 25, p. 429-442, 2005.
[59] F. Magoules, G. Gbikpi-Benissan, ”Distributed convergence detection based on global residual error under asynchronous iterations”, Parallel and Distributed Systems, IEEE Transactions on, 29(4), 819-829, 2018.
[60] F. Magoules, G. Gbikpi-Benissan, ”JACK2: An MPI-based communication library with non-blocking synchronization for asynchronous iterations”, Advances in Engineering Software, 11, 116-133, 2018.
[61] F. Robert, ”Blocs H-matrices et convergence des méthodes itératives classiques par blocs”, Linear Algebra and its Applications, 2, 223-265, 1969.
[62] J. Schroeder, ”Computing error bounds in solving linear systems”, University of Wisconsin, M.R.C. Technical Report 242, 1961.
[63] P. Spitéri, D. El Baz, J.C. Miellou, ”Perturbation of parallel asynchronous linear iterations by floating point errors”, Electronic Transactions on Numerical Analysis (ETNA), 13, 38-55, 2002.
[64] J.C. Miellou, P. Spitéri, D. El Baz, ”A new stopping criterion for linear perturbed asynchronous iterations”, Journal of Computational and Applied Mathematics, Elsevier, 219 (2), 471-483, 2008.
[65] P. Spitéri, ”Finite precision computation for linear fixed point methods of parallel asynchronous iterations”, in Techniques for parallel, distributed and cloud computing in engineering, B.H.V. Topping, P. Ivanyi (Eds.), Saxe-Coburg Publications, 163-196, 2015.
[66] J.C. Miellou, P. Spitéri, ”Stopping criteria for parallel asynchronous iterations for fixed point methods”, in Developments in parallel, distributed, grid and cloud computing for Engineering, B.H.V. Topping, P. Ivanyi (Eds.), Saxe-Coburg Publications, 277-308, 2013.
[67] J. Wilkinson, ”Rounding error in algebraic processes”, Notes on Applied sciences, Prentice-Hall series in automatic computation, vol. 32, 1963.
[68] J. Wilkinson, ”The algebraic eigenvalue problem”, Oxford university press, 1965.

[69] J. C. Miellou, P. Cortey-Dumont, M. Boulbrachêne, ”Perturbation of fixed point iterative methods”, Advances in Parallel Computing, I, 81–122, 1990.
[70] P. Spitéri, D. El Baz, J.C. Miellou, M. Jarraya, ”Mathematical study of perturbed asynchronous iterations designed for distributed termination”, International Mathematical Journal, 1 (5), 491-503, 2002.
[71] P. Spiteri, ”Simulation d’exécutions parallèles pour la résolution d’inéquations variationnelles stationaires”, Revue E.D.F. Informatique et Mathématique Appliquées, Série C, 1, 149-158, 1983.
[72] J.C. Miellou, P. Spiteri, G.R. Perrin, ”A cheap method of performance evaluation for subdomain decomposition parallel algorithms of three dimensional elliptic problems”, Journal of systems and software, 6 (1), 169 - 173, 1986.
[73] L. Giraud, P. Spiteri, ”Implementations of parallel solutions for nonlinear boundary value problems”, Parallel Computing’91, in Advances in Parallel Computing 4, eds. D.J. Evans, G.R. Joubert, and H. Liddel, Elsevier Science B.V., North Holland, Amsterdam, 203-211, 1992.
[74] R. Guivarch, P. Spitéri, ” Implantation de méthodes de sous-domaines asynchrones avec PVM et MPI sur IBM-SP2”, Calculateurs Parallèles, Editions Hermes Paris, 10 (4), 431-438, 1998.
[75] C. E. Ramamonjisoa, L. Ziane Khodja, D. Laiymani, A. Giersch, R. Couturier, ”Simulation of Asynchronous Iterative Algorithms Using SimGrid”, In 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS), 890-895, 2014.
[76] F. Magoules, G. Gbikpi-Benissan, ”JACK: An asynchronous communication kernel library for iterative algorithms”, The Journal of Supercomputing, 73(8), 3468-3487, 2017.
[77] J. Bahi, R. Couturier, L. Ziane Khodja. ”Parallel GMRES implementation for solving sparse linear systems on GPU clusters”, Proceedings of the 19th High Performance Computing Symposia. Society for Computer Simulation International, 2011.
[78] L. Ziane-Khodja, M. Chau, R. Couturier, J. Bahi, P. Spitéri, ”Parallel solution of American options derivatives on GPU clusters”, International Journal of Computers and Mathematics with Applications, Elsevier, 65 (11), 1830-1848, 2013.
[79] K. Ghidouche, R. Couturier, A. Sider, ”A parallel implementation of the Durand-Kerner algorithm for polynomial root-finding on GPU”, Advanced Networking Distributed Systems and Applications (INDS), 2014 International Conference on. IEEE, 2014.
[80] K. Ghidouche, A Sider, R Couturier, C Guyeux, ”Efficient high degree polynomial root finding using GPU”, Journal of Computational Science 18, 46-56, 2017.
[81] J. Bahi, E. Griepentrog, J.C. Miellou, ”Parallel treatment of a class of differential algebraic systems”, SIAM J. Numer. Anal., 23, 1969 - 1980, 1996.
[82] A. Frommer, P. Spitéri, ”On linear asynchronous iterations when the spectral radius of the modulus matrix is one”, Computing, 15, 91-104, 2001.
[83] D. El Baz, P. Spitéri, J.C. Miellou, D. Gazen, ”Asynchronous iterative algorithms with flexible communication for non linear network flow problems”, Journal of Parallel and Distributed Computing, 38, 1-15, 1996.
[84] D. El Baz, D. Gazen, P. Spitéri, J.C. Miellou, ” Mise en oeuvre de méthodes itératives asynchrones avec communication flexible : Application à la résolution d’une classe de problèmes d’optimisation”, Calculateurs Parallèles, 8 (4), 393-410, 1996.

[85] R. Couturier, P. Canalda, F. Spies. ”Iterative algorithms on heterogeneous network computing: Parallel polynomial root extracting”, International Conference on High-Performance Computing. Springer, Berlin, Heidelberg, 2002.
[86] F. Magoules, G. Gbikpi-Benissan, ”Asynchronous Parareal time discretization for partial differential equations”, SIAM J. Sci. Comput., 40 (6), 704-725 , 2018.
[87] F. Magoules, G. Gbikpi-Benissan, Q. Zou, ”Asynchronous iterations of Parareal algorithm for option pricing models”, Mathematics, 6 (4), 45, 1-18, 2018.
[88] M. Chau, R. Couturier, J. Bahi, P. Spitéri, ”Asynchronous grid computation for american options derivative”, Advances in Engineering Software, Elsevier Science, 60-61, 136-144, 2013.
[89] M. Chau, R. Couturier, J. Bahi, P. Spitéri, ”Parallel solution of the obstacle problem in Grid environment”, International Journal of High Performance Computing Applications, SAGE Publications, 25 (4), 488-495, 2011.
[90] P. Spitéri, A. Ouaoua, M. Chau, H. Boutabia, ”Parallel solution of the discretized and linearized G-heat equation”, International Journal of High Performance Computing and Networking, Inderscience Publishers, 11 (1), 66-82, 2018.
[91] R. Guivarch, P. Spitéri, H.C. Boisson, J.C. Miellou, ”Schwartz alternating parallel algorithm applied to incompressible flow computation in vorticity stream function formulation”, Parallel Algorithm and Application, 11, 205-225, 1997.
[92] M. Chau, P. Spitéri, H.C. Boisson, ”Parallel numerical simulation for the coupled problem of continuous flow electrophoresis”, International Journal for Numerical Methods in Fluids, Wiley Interscience Publications, 55, 945-963, 2007.
[93] M. Chau, P. Spitéri, R. Guivarch, H.C. Boisson, ”Parallel asynchronous iterations for the solution of a 3D continuous flow electrophoresis problem”, Computers and Fluids, Elsevier, 37, 1126-1137, 2008.
[94] M. Chau, T. Garcia, P. Spitéri, ”Asynchronous grid computing for the simulation of the 3D electrophoresis coupled problem”, Advances in Engineering Software, Elsevier Science, 60-61, 111-121, 2013.
[95] M. Chau, P. Tauber, P. Spitéri, ”Parallel Schwarz alternating methods for anisotropic diffusion of speckled medical images”, Numerical Algorithms, Springer-Verlag, 51, 85-114, 2009.
[96] P. Tauber, M. Chau, P. Spitéri, ”Parallel Methods for 3D+t Anisotropic Diffusion of Dynamic Positron Emission Tomography Images”, International Conference on Engineering Computational Technology, Naples, 02/09/2014-05/09/2014, B.H.V. Topping, P. Ivanyi (Eds.), Civil-Comp Proceedings, (online), Access : http://www.ctresources.info/ccp/paper.html?id=8432, 2014.
[97] H. Khochemane, H. Boutabia, M. Chau, P. Spitéri, ”Parallel solution of the porous medium equation”, Computers and Mathematics with Applications, Elsevier, 71 (4), 931-948, 2016.
[98] P. Spiteri, ”Parallel asynchronous algorithms for solving boundary value problems”, *Parallel Algorithms and Architectures,* eds. M. Cosnard et al., North-Holland, 73-84, 1986.
[99] F. Magoules, D.B. Szyld, C. Venet, ”Asynchronous optimized Schwarz methods with and without overlap”, Numerische Mathematik, 137(1), 199-227, 2017.
[100] J. Bahi, C. Guyeux, ”Hash Functions Using Chaotic Iterations ”, Journal of Algorithms and Computational Technology, 4, 167-181, 2010.

[101] J. Bahi, X. Fang, C. Guyeux, Q. Wang, " Evaluating Quality of Chaotic Pseudo-Random Generators. Application to Information Hiding ", International Journal On Advances in Security, 4, 118-130, 2011.
[102] C. Guyeux, J. Bahi, " A Topological Study of Chaotic Iterations. Application to Hash Functions ", Computational Intelligence for Privacy and Security, 394, 51-73, 2012.
[103] J. Bahi, X. Fang, C. Guyeux, L. Larger, " FPGA Design for Pseudorandom Number Generator Based on Chaotic Iteration used in Information Hiding Application ", Applied Mathematics and Information Sciences, 7, 2175-2188, 2013.
[104] Q. Wang, S. Yu, C. Guyeux, J. Bahi, X. Fang, " Theoretical design and circuit implementation of integer domain chaotic systems ", International Journal of Bifurcation and Chaos, 24, 118-128, 2014.
[105] X. Fang, Q. Wang, C. Guyeux, J. Bahi, " FPGA Acceleration of a Pseudorandom Number Generator based on Chaotic Iterations ", Journal of Information Security and Applications, 19, 78-87, 2014.
[106] J. Bahi, X. Fang, C. Guyeux, Q. Wang, " Suitability of chaotic iterations schemes using XORshift for security applications ", Journal of Network and Computer Applications, 37, 282-292, 2014.